목 차

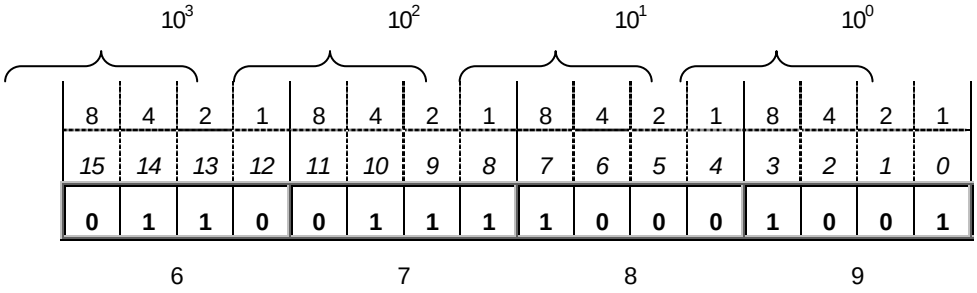
모듈 Ⅰ. GLOFA-GM 프로그래밍

제 1 장 데이터 형식 및 표현

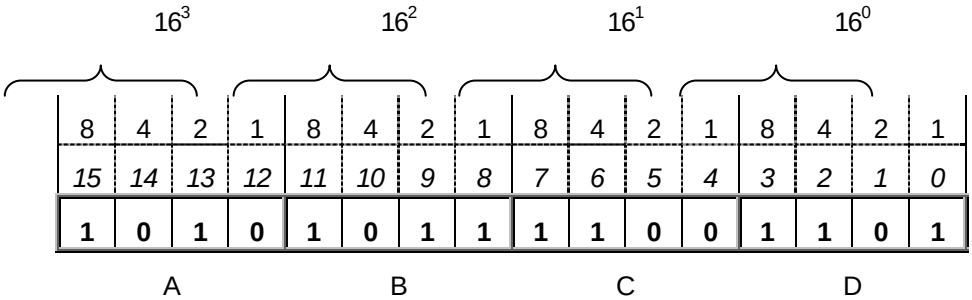
데이터 형식	형식	표현 범위		크기
Boolan	BOOL	0 ~ 1 (On/Off)		1bit
Byte	BYTE	BIN	16#0~16#FF	8bit
		BCD	16#0~16#99	
Word	WORD	BIN	16#0~16#FFFF	16bit
		BCD	16#0~16#9999	
Double Word	DWORD	BIN	16#0~16#FFFFFFFF	32bit
		BCD	16#0~16#99999999	
Long Word	LWORD	BIN	16#0~16#FFFFFFFFFFFFFFFF	64bit
		BCD	16#0~16#9999999999999999	
Short Integer	SINT	-128 ~ 127		8bit
Integer	INT	-32768 ~ 32767		16bit
Double Integer	DINT	-214783648 ~ 214783647		32bit
Long Integer	LINT	-4.611686018427e+18 ~ 4.611686018427e+18		64bit
Unsigned Short Integer	USINT	0 ~ 255		8bit
Unsigned Integer	UINT	0 ~ 65535		16bit
Unsigned Double Integer	UDINT	0 ~ 4294836225		32bit
Unsigned Long Integer	ULINT	0 ~ 1.844674406512e+19		64bit

각 비트위치에 있어서 비트열의 표현은 다음과 같습니다.

예1) Word 영역의 경우 BCD 비트열 표현



예2) Word 영역의 경우 BIN 비트열 표현



제 2 장 표준 라이브러리 및 프로그래밍

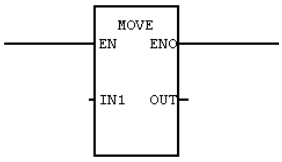
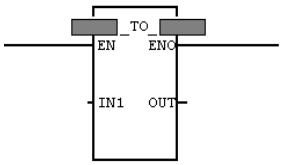
2.1 표준 펑션및 펑션블록

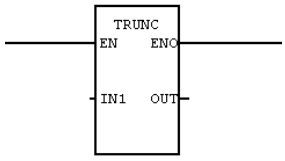
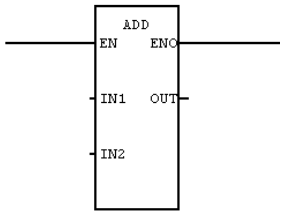
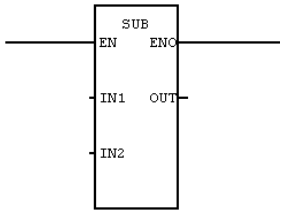
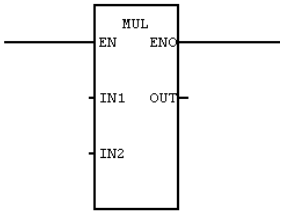
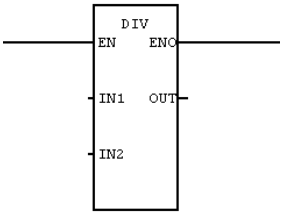
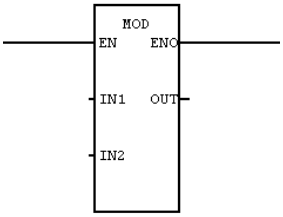
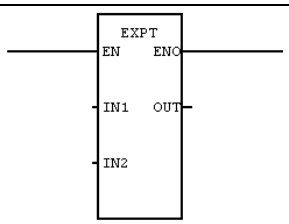
표준 펑션에는 전송 펑션, 변환 펑션, 비교 펑션, 산술연산펑션, 논리연산 펑션,
비트 시프트 펑션, 선택 펑션, 문자열 펑션, 날짜 시각 펑션, 시스템 제어 펑션 등이 있고
표준 펑션블록에는 카운터, 타이머, 에지검출, 바이스터블 펑션블록 등이있습니다.

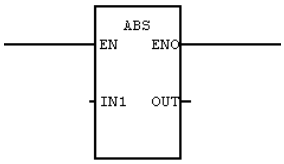
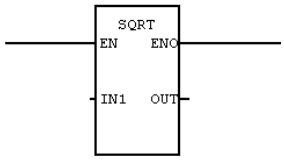
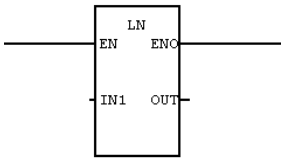
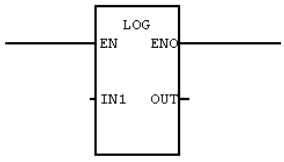
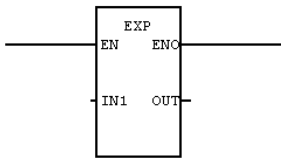
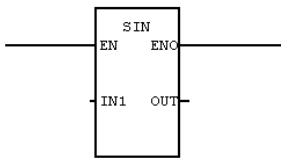
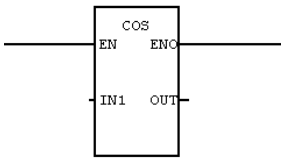
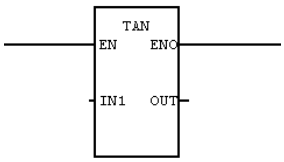
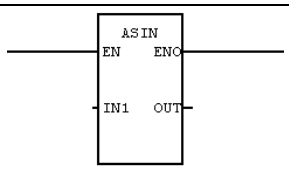
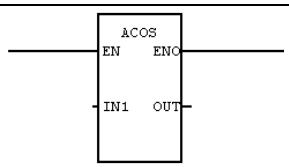
시퀀스 연산자

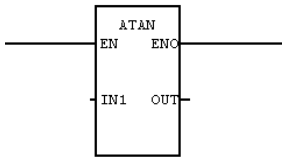
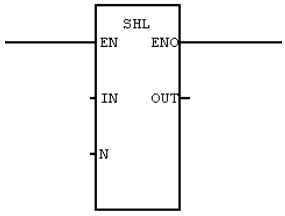
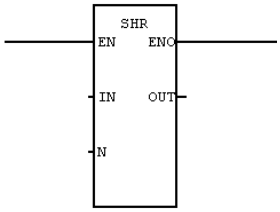
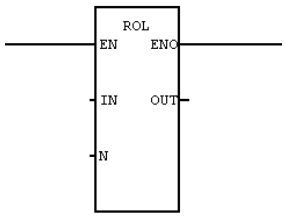
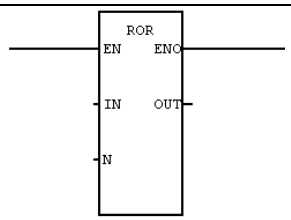
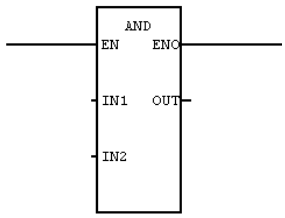
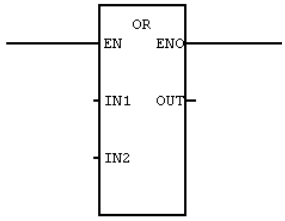
구분	명령어	기호	기능 설명	비 고
시퀀스 연산자	A 접점		A 접점 연산	
	B 접점		B 접점 연산	
	상승 검출 접점		상승 에지에서 1Scan On 접점	
	하강 검출 접점		하강 에지에서 1Scan On 접점	
	출력 코일		연산 결과 출력	
	반전 코일		연산 결과 반전 출력	
	출력 Set		연산 결과 세트 출력	
	출력 Reset		연산 결과 리셋 출력	
	상승 검출 출력		상승 에지에서 1Scan On 출력	
	하강 검출 출력		하강 에지에서 1Scan On 출력	
	점프		레이블 위치로 점프	
	프로그램 종료		현재 프로그램 종료	

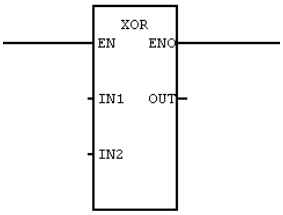
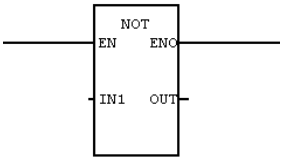
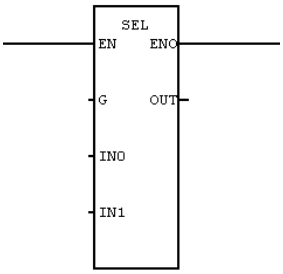
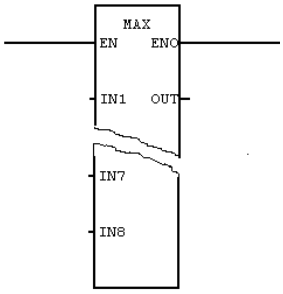
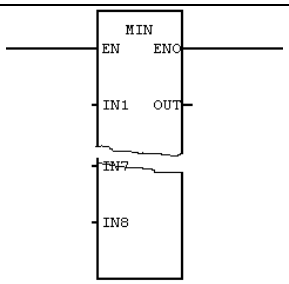
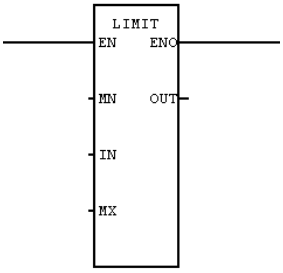
펑션 일람

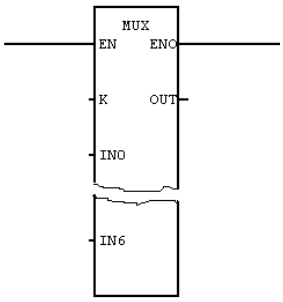
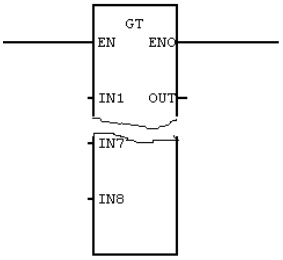
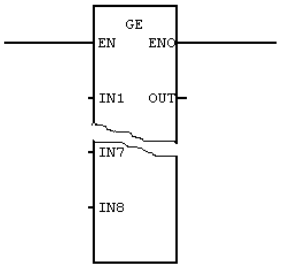
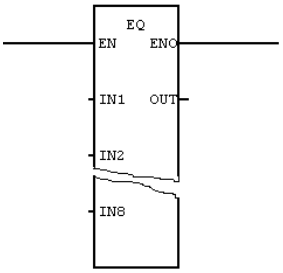
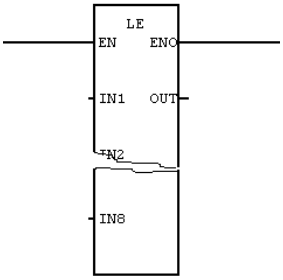
구분	명령어	기호	기능 설명	비 고
전송 펑션	MOVE		데이터 전송 IN1 : 전송원(모든 형식) OUT : 전송선(모든 형식)	
변환 펑션	****_TO_****		데이터 형식 변환 펑션 IN1 : 전송원 OUT : 전송선 변환 명령 펑션의 종류 SINT_TO_INT 외 14 종 INT_TO_SINT 외 14 종 DINT_TO_SINT 외 14 종 LINT_TO_SINT 외 14 종 USINT_TO_SINT 외 14 종 UINT_TO_SINT 외 14 종 UDINT_TO_SINT 외 14 종 ULINT_TO_SINT 외 14 종 BYTE_TO_SINT 외 14 종 WORD_TO_SINT 외 14 종 DWORD_TO_SINT 외 14 종 LWORD_TO_SINT 외 14 종 BCD_TO_SINT 외 7 종 REAL_TO_SINT 외 7 종 LREAL_TO_SINT 외 7 종 STRING_TO_SINT 외 18 종 NUM_TO_STRING TIME_TO_UDINT 외 2 종 DATE_TO_UINT 외 2 종 TOD_TO_UDINT 외 2 종 DT_TO_DATE 외 2 종	LINT ULINT LWORD REAL LREAL 관련 펑션은 GM1 과 GM2 만 가능

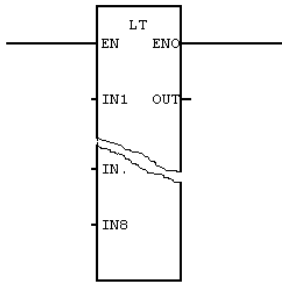
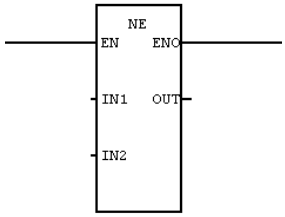
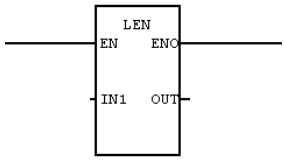
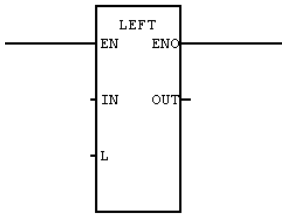
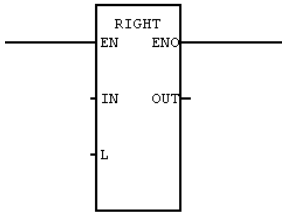
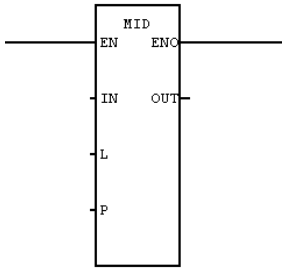
구분	명령어	기호	기능 설명	비 고
변환 평션	TRUNC		실수를 정수로 변환 IN1 : 전송원(REAL, LREAL) OUT : 전송선(DINT, LINT)	GM1, GM2 전용
수치 연산 평션	ADD		덧셈 평션 IN1 ~ IN8 : 더할 값(Any_INT) OUT : 결과값(Any_INT)	
	SUB		뺄셈 평션 IN1 : 연산원(Any_INT) IN2 : 뺄 값(Any_INT) OUT : 결과값(Any_INT)	
	MUL		곱셈 평션 IN1 ~ IN8 : 곱할 값(Any_INT) OUT : 결과값(Any_INT)	
	DIV		나눗셈(몫) IN1 : 연산원(Any_INT) IN2 : 나눌 값(Any_INT) OUT : 몫(Any_INT)	
	MOD		나눗셈(나머지) IN1 : 연산원(Any_INT) IN2 : 나눌 값(Any_INT) OUT : 나머지 값(Any_INT)	
	EXPT		지수 연산 IN1 : 정수(Any_REAL) IN2 : 지수(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용

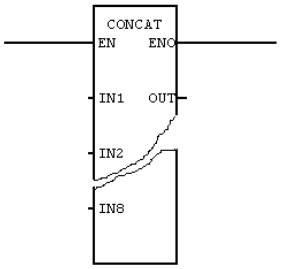
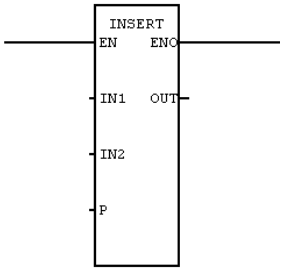
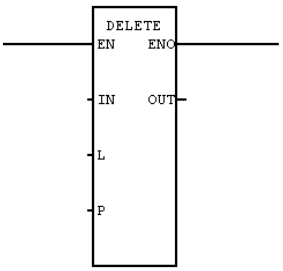
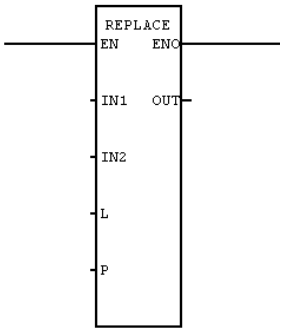
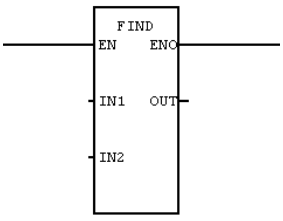
구분	명령어	기호	기능 설명	비 고
수치 연산 평션	ABS		절대값 IN1 : 정수(Any_INT) OUT : 결과값(Any_INT)	
	SQRT		제곱근 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	LN		자연 대수 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	LOG		상용 대수 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	EXP		자연 지수 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
삼각 평션	SIN		싸인 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	COS		코싸인 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	TAN		탄젠트 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	ASIN		아크 싸인 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
	ACOS		아크 코싸인 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용

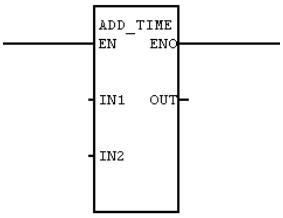
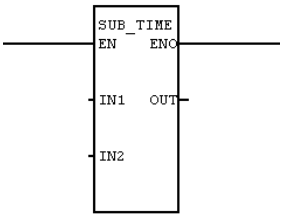
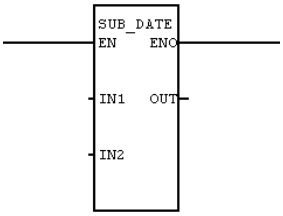
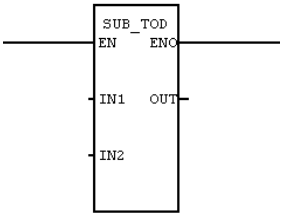
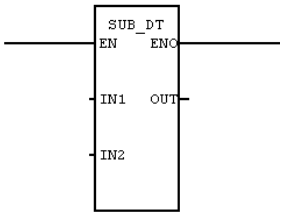
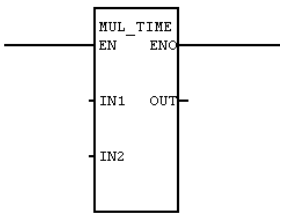
구분	명령어	기호	기능 설명	비 고
삼각평션	ATAN		아크 탄젠트 연산 IN1 : 연산원(Any_ REAL) OUT : 결과값(Any_ REAL)	GM1, GM2 전용
이동 평션	SHL		비트열 왼쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수(INT) OUT : 전송선(Any_BIT)	
	SHR		비트열 오른쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수(INT) OUT : 전송선(Any_BIT)	
회전 명령	ROL		비트열 왼쪽으로 회전 IN : 전송원(Any_BIT) N : 회전할 비트 수(INT) OUT : 전송선(Any_BIT)	
	ROR		비트열 오른쪽으로 회전 IN : 전송원(Any_BIT) N : 회전할 비트 수(INT) OUT : 전송선(Any_BIT)	
논리연산	AND		논리곱 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
	OR		논리합 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	

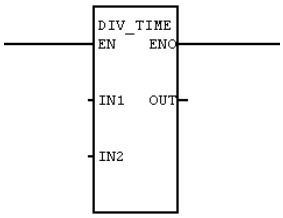
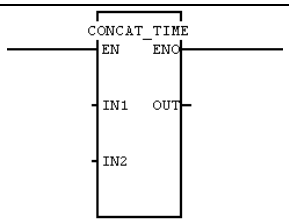
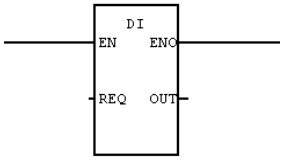
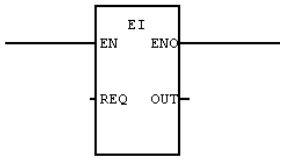
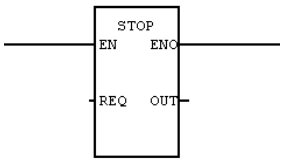
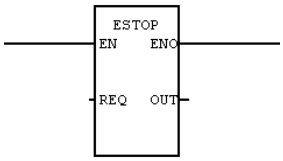
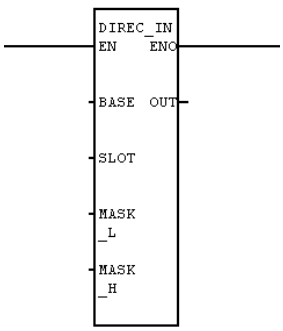
구분	명령어	기호	기능 설명	비 고
논리연산	XOR		배타적 논리합 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
	NOT		논리 반전 IN1, IN2 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
선택평션	SEL		2 개중 선택 G : 출력 선택(BOOL) IN0 : G 가 Off 일 경우 선택 값 (Any) IN1 : G 가 On 일 경우 선택 값 (Any) OUT : 출력값(Any)	
	MAX		최대값 구하기 IN1 ~ IN8 : 선택값(Any_INT) OUT : 최대 출력값(Any_INT)	
	MIN		최소값 구하기 IN1 ~ IN8 : 선택값(Any_INT) OUT : 최소 출력값(Any_INT)	
	LIMIT		상하한 제한값 MN : 하한값(Any_INT) IN : 전송원(Any_INT) MX : 상한값(Any_INT) OUT : 출력값(Any_INT)	

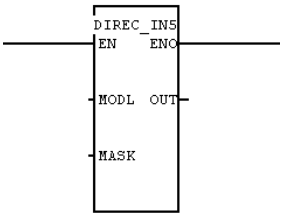
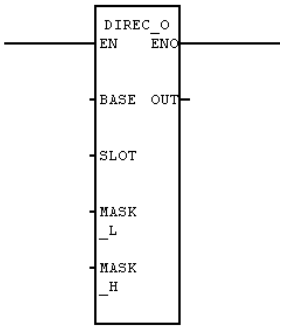
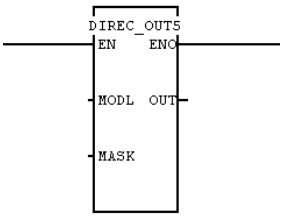
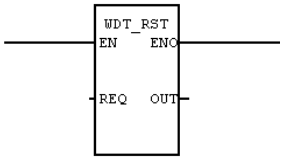
구분	명령어	기호	기능 설명	비 고
선택 평션	MUX		최대 7 개중 선택 K : 선택 입력 번호 IN0 : 전송원 0 번(Any) IN1 : 전송원 1 번(Any) IN2 : 전송원 2 번(Any) IN3 : 전송원 3 번(Any) IN4 : 전송원 4 번(Any) IN5 : 전송원 5 번(Any) IN6 : 전송원 6 번(Any) OUT : 출력값(Any)	
비교 평션	GT(>)		비교 평션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 > IN2 > .... > IN7 > IN8 의 조건 성립시 OUT 출력 On	
	GE(≥)		비교 평션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 ≥ IN2 ≥ .... ≥ IN7 ≥ IN8 의 조건 성립시 OUT 출력 On	
	EQ(=)		비교 평션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 = IN2 = ... = IN7 = IN8 의 조건 성립시 OUT 출력 On	
	LE(≤)		비교 평션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 ≤ IN2 ≤ .... ≤ IN7 ≤ IN8 의 조건 성립시 OUT 출력 On	

구분	명령어	기호	기능 설명	비 고
	LT(<)		비교 평션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 < IN2 < .... < IN7 < IN8 의 조건 성립시 OUT 출력 On	
	NE(≠)		비교 평션 IN1 , IN2 : 비교 데이터(Any) OUT : 출력(BOOL) IN1 ≠ IN2 의 조건 성립시 OUT 출력 On	
문자열 평션	LEN		문자열 길이 IN1 : 문자열 입력(STRING) OUT : 문자열 길이(INT)	
	LEFT		문자열 왼쪽 부분 전송 IN : 문자열 입력(STRING) L : 문자열 길이(INT) OUT : 문자열 출력(STRING)	
	RIGHT		문자열 오른쪽 부분 전송 IN : 문자열 입력(STRING) L : 문자열 길이(INT) OUT : 문자열 출력(STRING)	
	MID		문자열 중간 부분 전송 IN : 문자열 입력(STRING) L : 문자열 길이(INT) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	

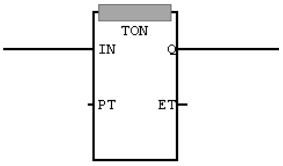
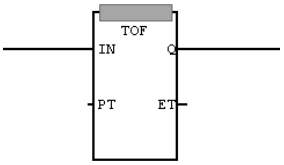
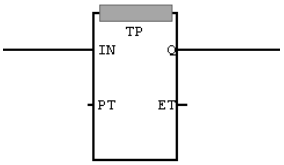
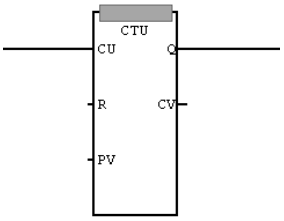
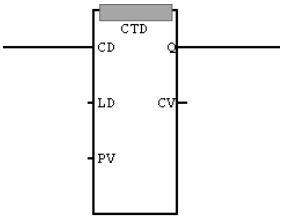
구분	명령어	기호	기능 설명	비 고
문자열 평션	CONCAT		문자열 연결 입력 문자열을 순서대로 연결 IN1 ~ IN8 : 문자열(STRING) OUT : 문자열 출력(STRING)	
	INSERT		문자열 삽입 IN1 : 문자열 입력(STRING) IN2 : 삽입할 문자열(STRING) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	DELETE		문자열 삭제 IN1 : 문자열 입력(STRING) L : 삭제할 문자열 길이(INT) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	REPLACE		문자열 대체 IN1 : 문자열 입력(STRING) IN2 : 대체할 문자열(STRING) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	FIND		문자열 찾기 IN1 : 문자열 입력(STRING) IN2 : 검색할 문자열(STRING) OUT : 문자열 선두 위치(INT)	

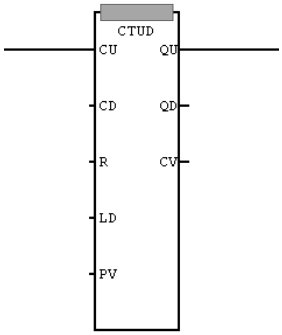
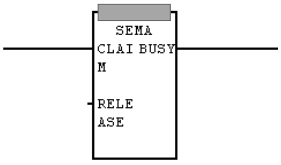
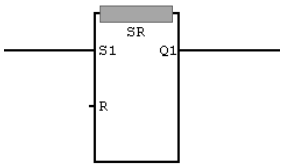
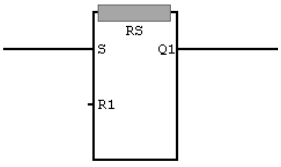
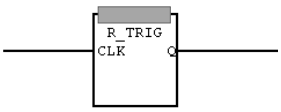
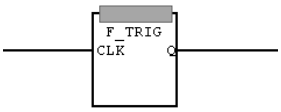
구분	명령어	기호	기능 설명	비 고
날짜 시간 평션	ADD_TIME		시간 더하기 IN1 : 시각 또는 시간 (TIME, TOD, TD) IN2 : 더할 시간(TIME) OUT : 결과 시각 또는 시간 (TIME, TOD, TD)	
	SUB_TIME		시간 빼기 IN1 : 시각 또는 시간 (TIME, TOD, TD) IN2 : 뺄 시간(TIME) OUT : 결과 시각 또는 시간 (TIME, TOD, TD)	
	SUB_DATE		날짜 빼기 IN1 : 날짜(DATE) IN2 : 뺄 날짜(DATE) OUT : 결과 시간(TIME)	
	SUB_TOD		시각 빼기 IN1 : 시각(TIME OF DAY) IN2 : 뺄 시각(TIME OF DAY) OUT : 결과 시간(TIME)	
	SUB_DT		날짜 시각 빼기 IN1 : 시각(DATE&TIME) IN2 : 뺄 시각(DATE&TIME) OUT : 결과 시간(TIME)	
	MUL_TIME		시간 곱하기 IN1 : 입력 시간(TIME) IN2 : 곱할 값(INT) OUT : 결과 시간(TIME)	

구분	명령어	기호	기능 설명	비 고
날짜 시간 평션	DIV_TIME		시간 나누기 IN1 : 입력 시간(TIME) IN2 : 나눌 값(INT) OUT : 결과 시간(TIME)	
	CONCAT_TIME		날짜와 시각 연결 IN1 : 입력 날짜(DATE) IN2 : 입력 시각(TOD) OUT : 결과 날짜 시각(DT)	
시스템 제어 평션	DI		인터럽트 금지 REQ : 금지 요구(BOOL) OUT : 금지 확인(BOOL)	
	EI		인터럽트 허가 REQ : 허가 요구(BOOL) OUT : 허가 확인(BOOL)	
	STOP		PLC 정지 요구 REQ : 정지 요구(BOOL) OUT : 정지 확인(BOOL)	
	ESTOP		PLC 비상 정지 요구 REQ : 정지 요구(BOOL) OUT : 정지 확인(BOOL)	
	DIREC_IN		입력 데이터 즉시 갱신 BASE : 베이스 모듈 번호 SLOT : 입력 모듈 슬롯 위치 MASK_L : 하위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) MASK_H : 상위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 제외

구분	명령어	기호	기능 설명	비 고
시스템 제어 평션	DIREC_IN5		입력 데이터 즉시 갱신 MODL : 입력 모듈 번호 MASK : 하위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 전용
	DIREC_O		출력 데이터 즉시 갱신 BASE : 베이스 모듈 번호 SLOT : 출력 모듈 슬롯 위치 MASK_L : 하위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) MASK_H : 상위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 제외
	DIREC_OUT5		출력 데이터 즉시 갱신 MODL : 출력 모듈 번호 MASK : 하위 32Bit 중 갱신 하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 전용
	WDT_RST		워치 독 타이머 리셋 REQ : 리셋 요구(BOOL) OUT : 실행 완료(BOOL)	

펑션 블록

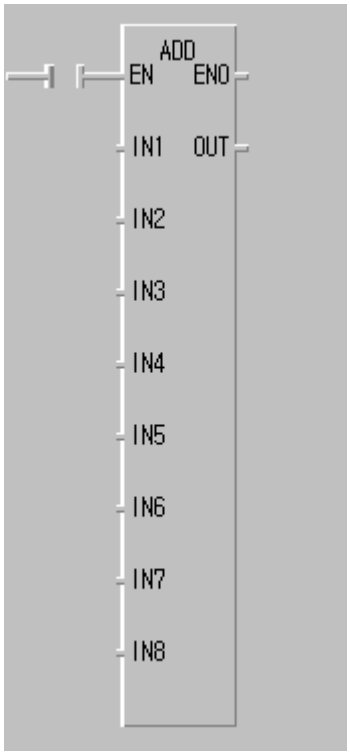
구분	명령어	기호	기능 설명	비 고
타이머 펑션 블록	TON		On 딜레이 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값	
	TOF		Off 딜레이 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값	
	TP		펄스 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값(TIME)	
카운터 펑션 블록	CTU		가산 카운터 CU : 펄스 입력(BOOL) R : 현재 값 리셋(BOOL) PV : 설정 값(INT) Q : 출력(BOOL) CV : 현재 값(INT)	
	CTD		감산 카운터 CD : 펄스 입력(BOOL) LD : 설정 값 Read(BOOL) PV : 설정 값(INT) Q : 출력(BOOL) CV : 현재 값(INT)	

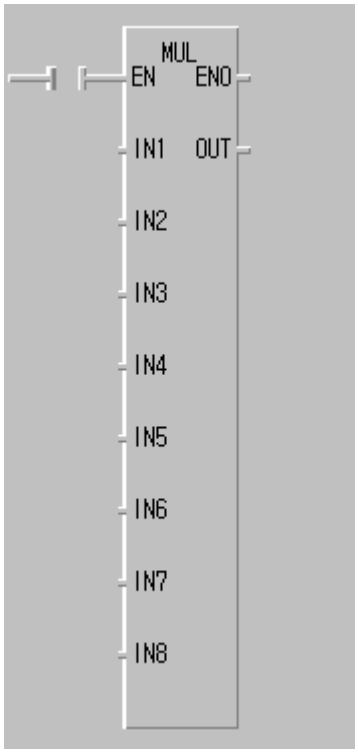
구분	명령어	기호	기능 설명	비 고
카운터 평션 블록	CTUD		가감산 카운터 CU : 가산 펄스 입력(BOOL) CD : 감산 펄스 입력(BOOL) R : 현재 값 리셋(BOOL) LD : 설정 값 Read(BOOL) PV : 설정 값(INT) QU : 가산 카운트 출력(BOOL) QD : 감산 카운트 출력(BOOL) CV : 현재 값(INT)	
평션 블록	SEMA		시스템 자원 제어(Semaphore) CLAIM : 자원 독점 요구 (BOOL) RELEASE : 자원 해방(BOOL) BUSY : 자원 취득 불가 (BOOL)	
	SR		Set 우선 쌍안정(Bistable) S1 : Set 신호(BOOL) R : Reset 신호(BOOL) Q1 : 출력(BOOL)	
	RS		Reset 우선 쌍안정(Bistable) S : Set 신호(BOOL) R1 : Reset 신호(BOOL) Q1 : 출력(BOOL)	
	R_TRIG		상승 에지 검출 CLK : 입력(BOOL) Q : 출력(BOOL)	
	F_TRIG		하강 에지 검출 CLK : 입력(BOOL) Q : 출력(BOOL)	

2.2 프로그램 작성

2.2.1 산술 연산 프로그램 작성 예

1) 펑션 명령 : 다중 입력 펑션(2 ~ 8 개 까지)





산술 펑션

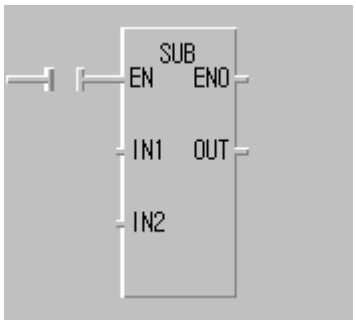
EN : 실행 요구 접점 (BOOL 형)

IN1~IN8 : 셈할 정수값 또는 Any INT 형 변수 입력

ENO : 실행후 EN 신호를 출력(BOOL 형)

OUT : 결과값을 저장할 변수 (입력과 동일한 형식 의 변수)

2 입력 펑션



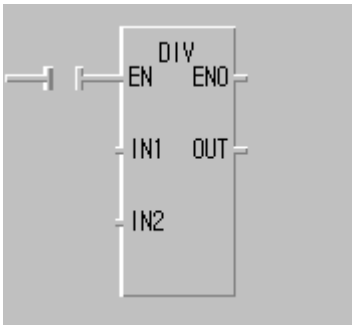
EN : 실행 요구 접점(BOOL 형)

IN1 : 피 감산 정수값 또는 Any INT 형 변수 입력

IN2 : 감산 정수값 또는 Any INT 형 변수 입력

ENO : 실행후 EN 신호를 출력(BOOL 형)

OUT : 결과값을 저장할 변수(입력과 동일한 형식 의 변수)



EN : 실행 요구 접점(BOOL 형)

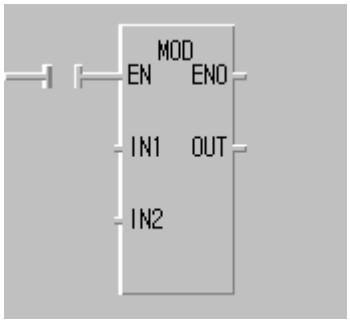
IN1 : 피 제산 정수값 또는 Any INT 형 변수 입력

IN2 : 제산 정수값 또는 Any INT 형 변수 입력

ENO : 실행후 EN 신호를 출력(BOOL 형)

OUT : 결과값을 저장할 변수(입력과 동일한 형식 의 변수)

주) 몫만 취하고 나머지는 버림



EN : 실행 요구 접점(BOOL 형)
IN1 : 피 제산 정수값 또는 Any INT 형 변수 입력
IN2 : 제산 정수값 또는 Any INT 형 변수 입력

ENO : 실행후 EN 신호를 출력(BOOL 형)
OUT : 결과값을 저장할 변수(입력과 동일한 형식 의 변수)
주) 나머지만 취하고 몫은 버림

2) 연산 에러 플레그

_ERR : 연산 에러가 발생하면 On 되고 에러 해재시 Off 됩니다.

_LER : 연산 에러가 발생하면 Set 되고 사용자 프로그램에 의해 Reset 할 때까지 On 됩니다.

3) 프로그램 편집

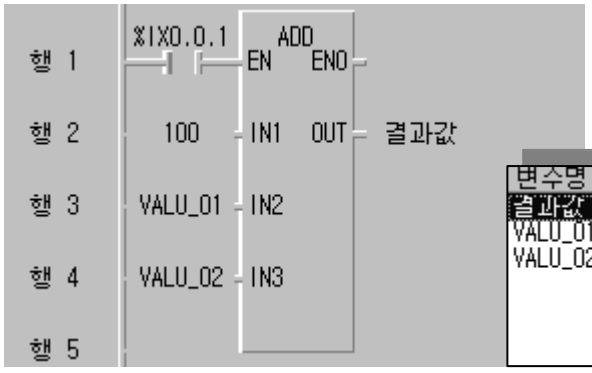
도구 상자에서 {F} 를 선택하여 원하는 위치에 클릭하여 펀션 목록 상자가 나타나면 ADD 펑션을 선택하고 입력 개수 를 설정(3 개)합니다

EN 입력 접점은 %IX0.0.1 으로 선언하여 실행 조건 접점으로 합니다.

IN1 은 상수 100 을 입력하고 IN2 및 IN3 는 각각 VALU_01, VALU_02 의 INT 형 변수를 선언합니다

OUT 에는 상수 100 과 변수 VALU_01 과 VALU_02 를 모두 더한값이 저장될 변수를 선언하며

이 변수의 형태는 반드시 입력측 변수와 같은 INT 형으로 해야 합니다.



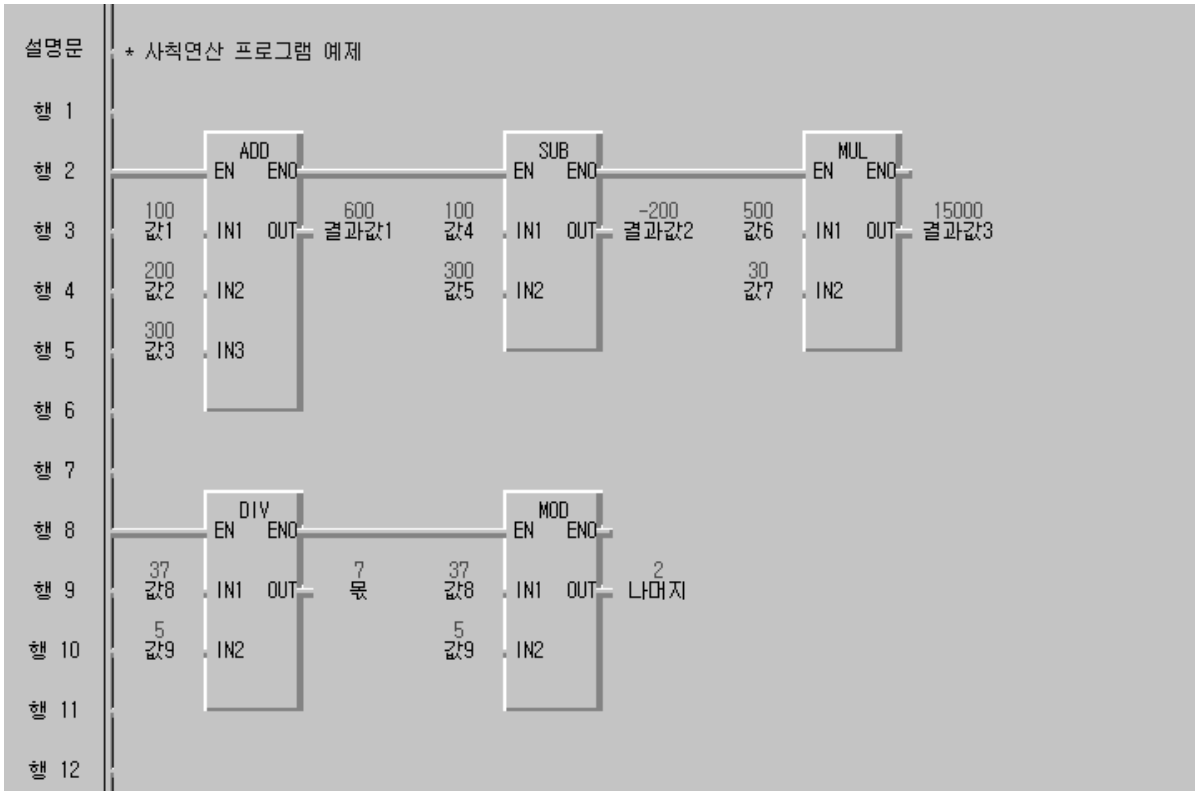
완성된 덧셈 펑션과 사용된 지역 변수

변수명	변수 종류	메모리할당	사용...	데이터 타입
결과값	VAR	<자동>	*	INT
VALU_01	VAR	<자동>	*	INT
VALU_02	VAR	<자동>	*	INT

사칙연산 '변수' List

변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
값1	INT	<자동>	100	VAR	*	
값2	INT	<자동>	200	VAR	*	
값3	INT	<자동>	300	VAR	*	
값4	INT	<자동>	100	VAR	*	
값5	INT	<자동>	300	VAR	*	
값6	INT	<자동>	500	VAR	*	
값7	INT	<자동>	30	VAR	*	
값8	INT	<자동>	37	VAR	*	
값9	INT	<자동>	5	VAR	*	
결과값1	INT	<자동>		VAR	*	
결과값2	INT	<자동>		VAR	*	
결과값3	INT	<자동>		VAR	*	
나머지	INT	<자동>		VAR	*	
몫	INT	<자동>		VAR	*	

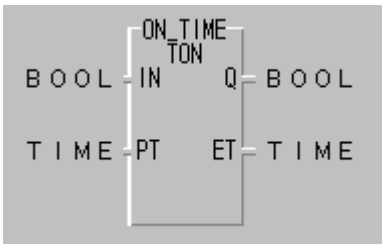
사칙연산 '프로그램' List 및 모니터 결과



2.2.2 펑션및 펑션블록 프로그램

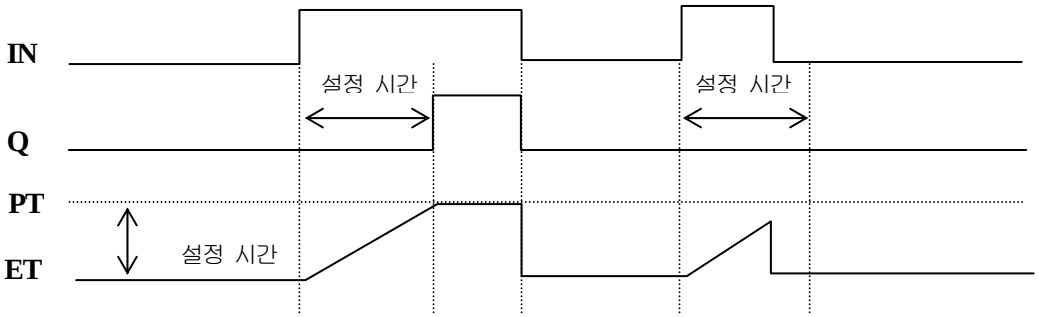
1) 타이머의 종류와 기능

On 딜레이 타이머

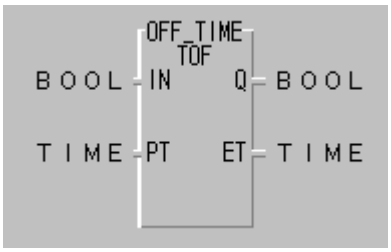


입력 IN : 타이머 기동 조건
PT : 설정 시간(Preset Time)
1 msec ~ 49 day
출력 Q : 타이머 출력
ET : 경과된 시간(Elapsed Time)

On 딜레이 타이머의 동작

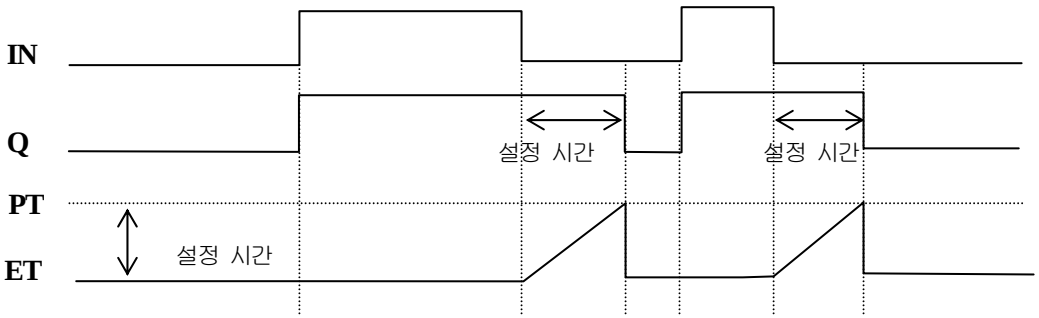


Off 딜레이 타이머

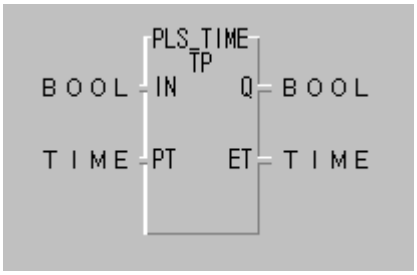


입력 IN : 타이머 기동 조건
PT : 설정 시간(Preset Time)
1 msec ~ 49 day
출력 Q : 타이머 출력
ET : 경과된 시간(Elapsed Time)

Off 딜레이 타이머의 동작

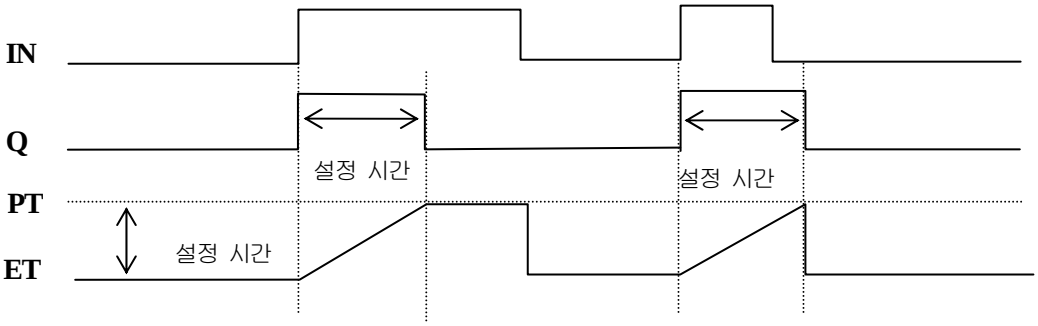


펄스 타이머

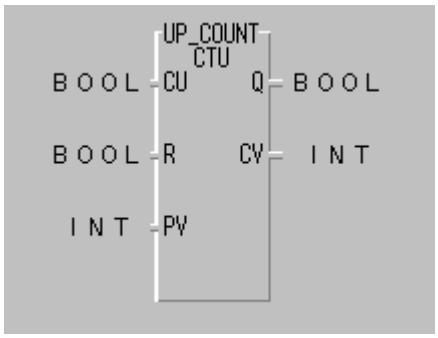


입력 IN : 타이머 기동 조건
PT : 설정 시간(Preset Time)
1 msec ~ 49 day
출력 Q : 타이머 출력
ET : 경과된 시간(Elapsed Time)

펄스 타이머의 동작

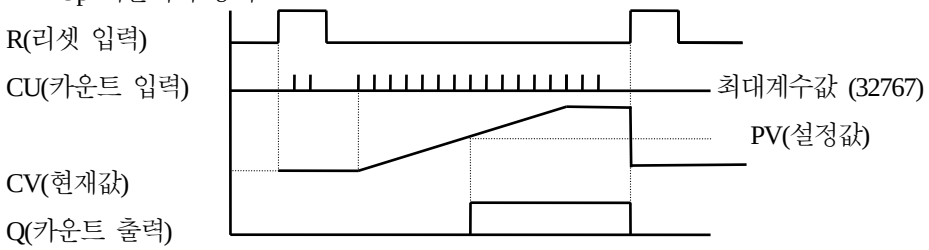


2) 카운터 평션블록의 기능



입력 CU: 업_카운트(Up_Count) 펄스입력
R : 리셋 입력(Reset)
PV: 설정값 (Preset Value)
출력 Q : 업_카운트(Up_Count) 출력
CV : 현재값(Current Value)

- Up 카운터의 동작

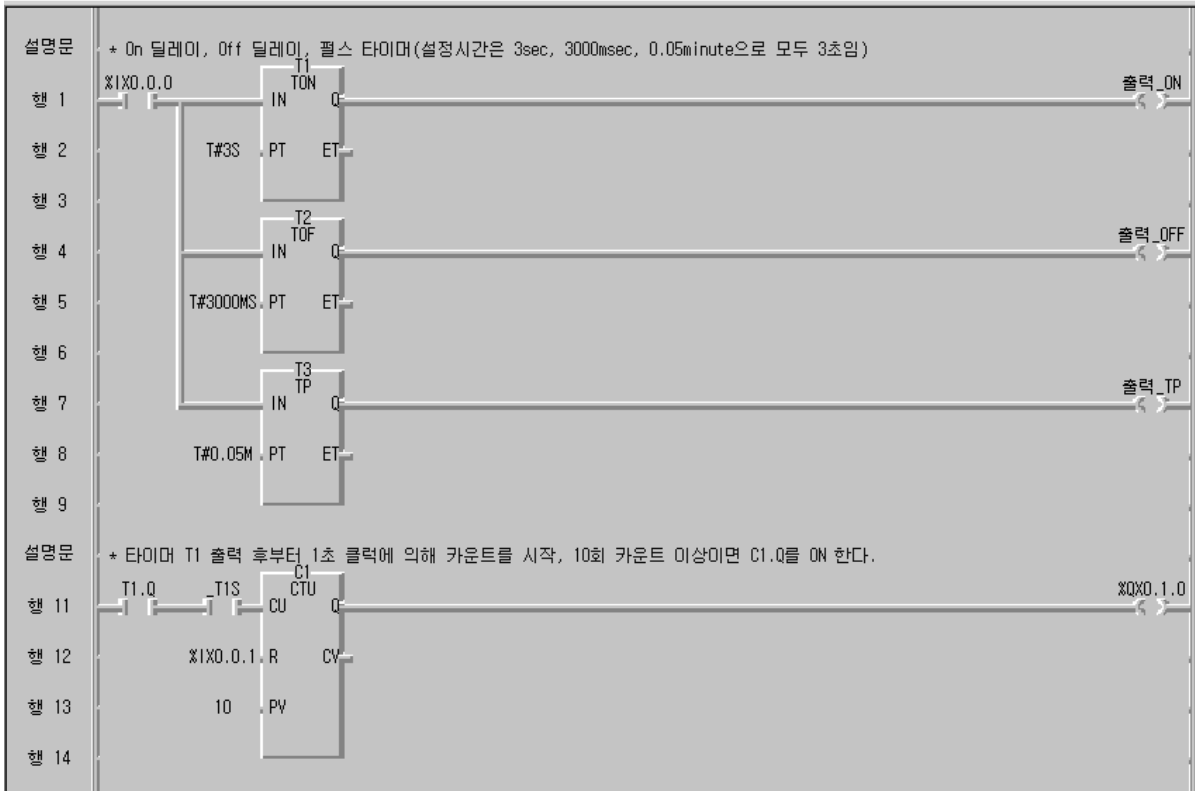


CV 가 1NT 의 최대값인 32767 미만일때만 증가하고 32767 이 되면 더이상 증가하지 않습니다.
리셋 입력 R 이 On 이되면 현재값 CV 는 0 으로 클리어(Clear)됩니다.
출력 Q 는 CV 가 PV 이상이 될때만 On 이 됩니다.
PV 값은 CTU 평션블록을 수행시 설정값을 새롭게 가져와 연산합니다.

3) 프로그램 작성 예

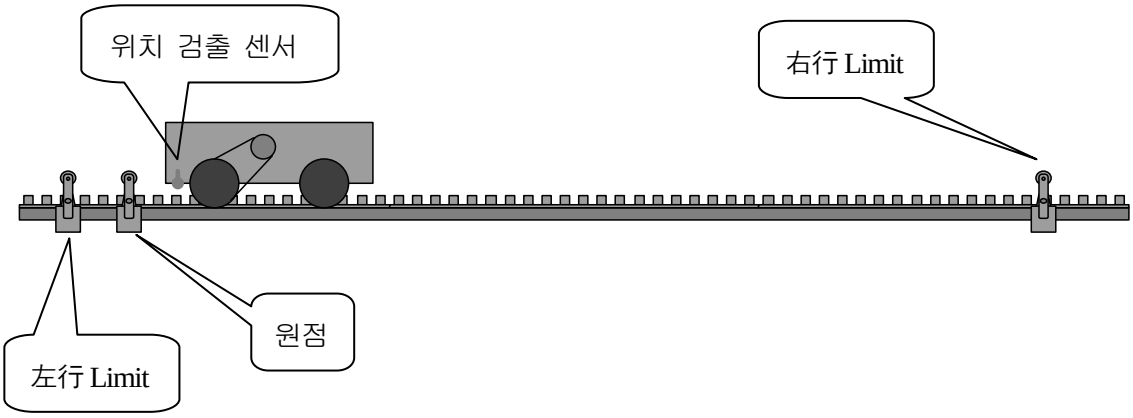
- 타이머와 카운터의 테스트 프로그램

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	출력_OFF	BOOL	<자동>		VAR	*	
2	출력_ON	BOOL	<자동>		VAR	*	
3	출력_TP	BOOL	<자동>		VAR	*	
4	C1	FB Instance	<자동>		VAR	*	
5	T1	FB Instance	<자동>		VAR	*	
6	T2	FB Instance	<자동>		VAR	*	
7	T3	FB Instance	<자동>		VAR	*	



- 비교 펑션 응용 프로그램 예

시스템 구성(대차 제어 시스템)



시스템 개요

본 시스템은 목표위치를 정하고 기동지령을 하면 현 위치 값과 비교하여 그 대소를 판단하여 좌 또는 우 방향을 결정하여 대차가 이동되면 위치검출 센서가 레일 옆에 설치한 요철을 카운트하여 목표위치 값과 현 위치 값이 일치하면 정지하고 또 다시 다음 목표 위치 값을 설정한 후 기동지령을 하여 다음 위치로 이동하는 프로그램을 작성합니다.

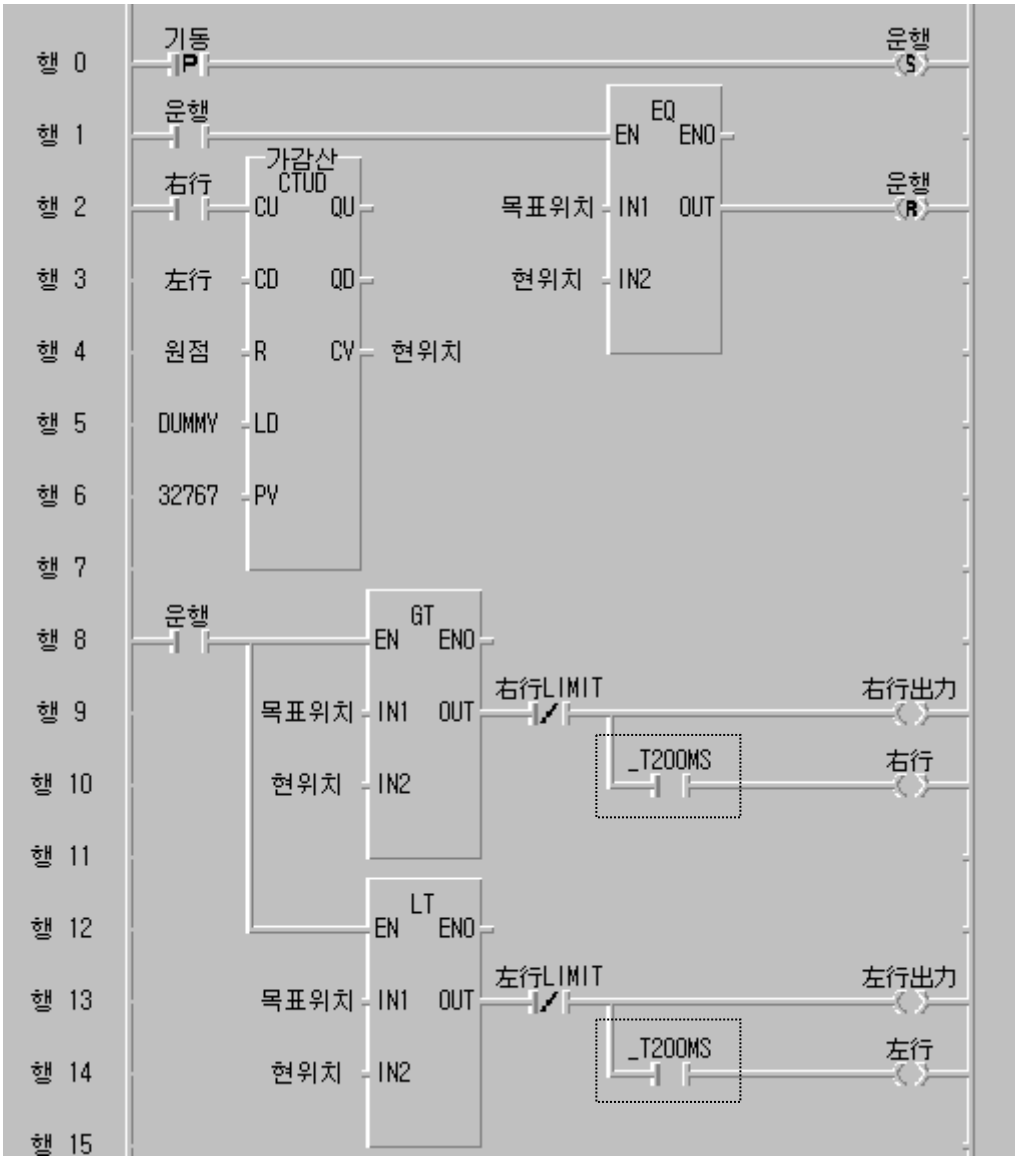
프로그램 작성

- 프로그램 테스트를 위하여 실제 프로그램에서 다음의 일부 내용을 변경하기로 합니다
  다음의 프로그램 리스트의 행 번호 10 과 14 행의 플래그 "_T200MS"는 위 그림의 위치 검출 센서의 외부 펄스 입력 대체하여 편집한 것으로 200msec 주기의 입력 펄스로 간주하여 테스트 하기로 하였습니다

- 조작 방법
  원점 스위치 조작으로 카운터를 초기화하고 목표위치 변수를 모니터링중 더블 클릭한 후 임의의 목표위치 값을 써 넣은 후 기동 지령 접점 신호를 On 합니다.

  左行出力 및 右行出力의 상태와 현위치 값의 변화를 확인하며 운전상태를 모니터링 합니다

프로그램 List

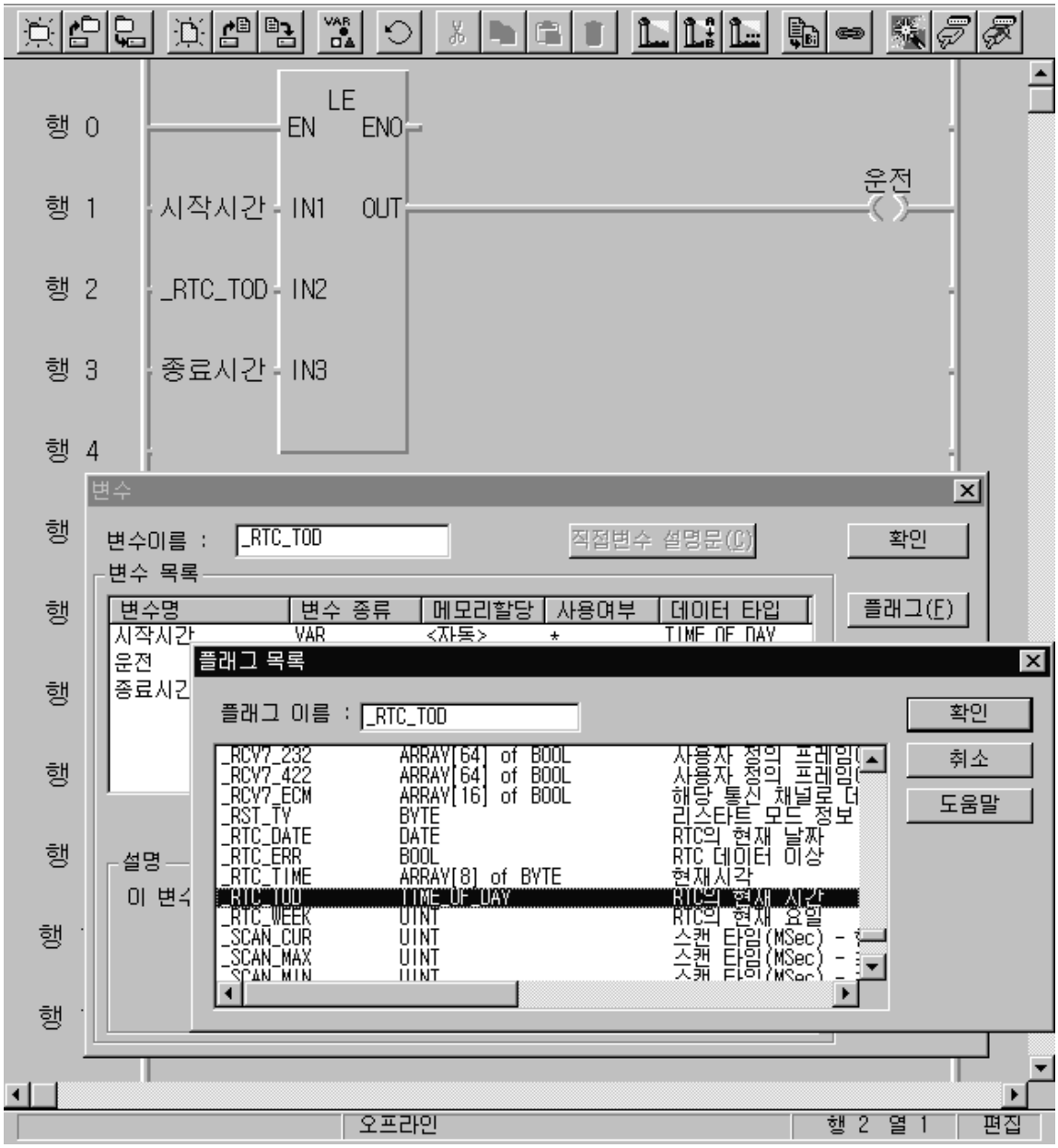


주) 실제의 시스템에서는 위 프로그램의 [ ] 부분을 시스템 구성도의 위치 검출 센서의 입력 신호로 대치하여 실제의 레일의 요철을 검출하여 시운전을 합니다.

비교 펑션과 특수 플레그의 활용 프로그램

프로그램 개요

시스템의 RTC 기능을 이용하여 지정한 시각에서 가동을 시작하여 지정한 시각에서 가동을 중지하는 프로그램의 예제 입니다.



주) 변수명 시작시간, 종료시간은 플래그 _RTC_TIME 과 같은 Time_Of_Day 로 설정 합니다.
테스트를 위하여 변수명 시작시간, 종료시간의 입력값은 TOD#09:20:00.0 및 TOD#17:30:00.0 으로 입력 합니다.(오전 9 시 20 분, 오후 5 시 30 분)
실행 모니터링 화면의 해당 변수를 마우스로 더블 클릭하면 데이터 입력 대화상자가 나오며 이곳의 입력란에 원하는 데이터를 타입에 맞추어 입력합니다.

2.3 배열(Array)변수에 대하여

배열(Array)이란 동일한 이름과 동일한 데이터 타입(WORD, BOOL, INT 등)으로 된 데이터가 순서대로 나열된 형태의 변수를 말합니다

예를 들어, 배열변수의 이름을 "LED 열"로 정의하고 배열 원소 개수를 5 개로 지정한 경우, 이 변수는 LED 열[0], LED 열[1], LED 열[2], LED 열[3], LED 열[4]인 5 개의 원소로 구성 되며, [ ]안의 숫자는 배열변수에 할당된 원소 번호로, INT 형 변수를 사용하면 간접 어드레스 지정을 할 수 있습니다.

2.3.1 배열변수 활용 프로그램 예

(1) 변수 선언

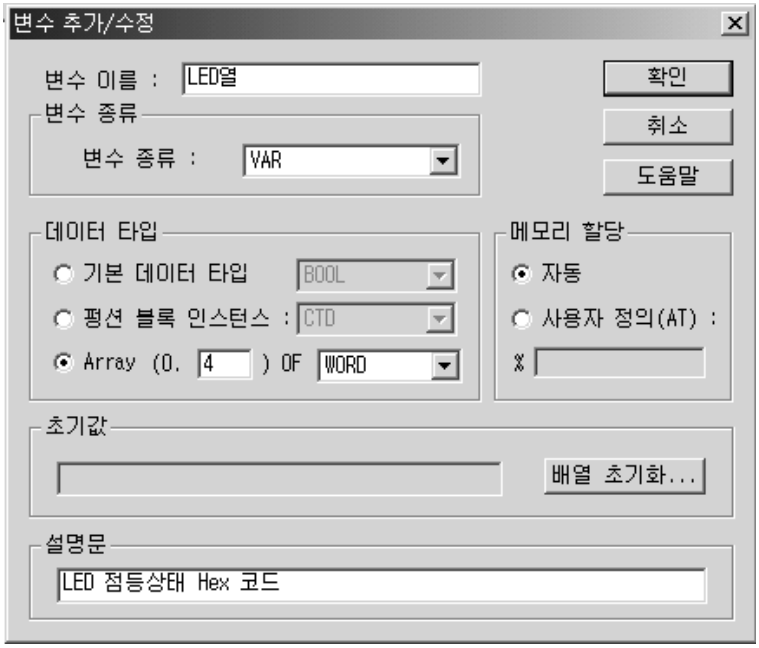
배열변수명	LED 열	변수명	A
원소 개수	5 개	데이터 타입	INT
데이터 타입	WORD		

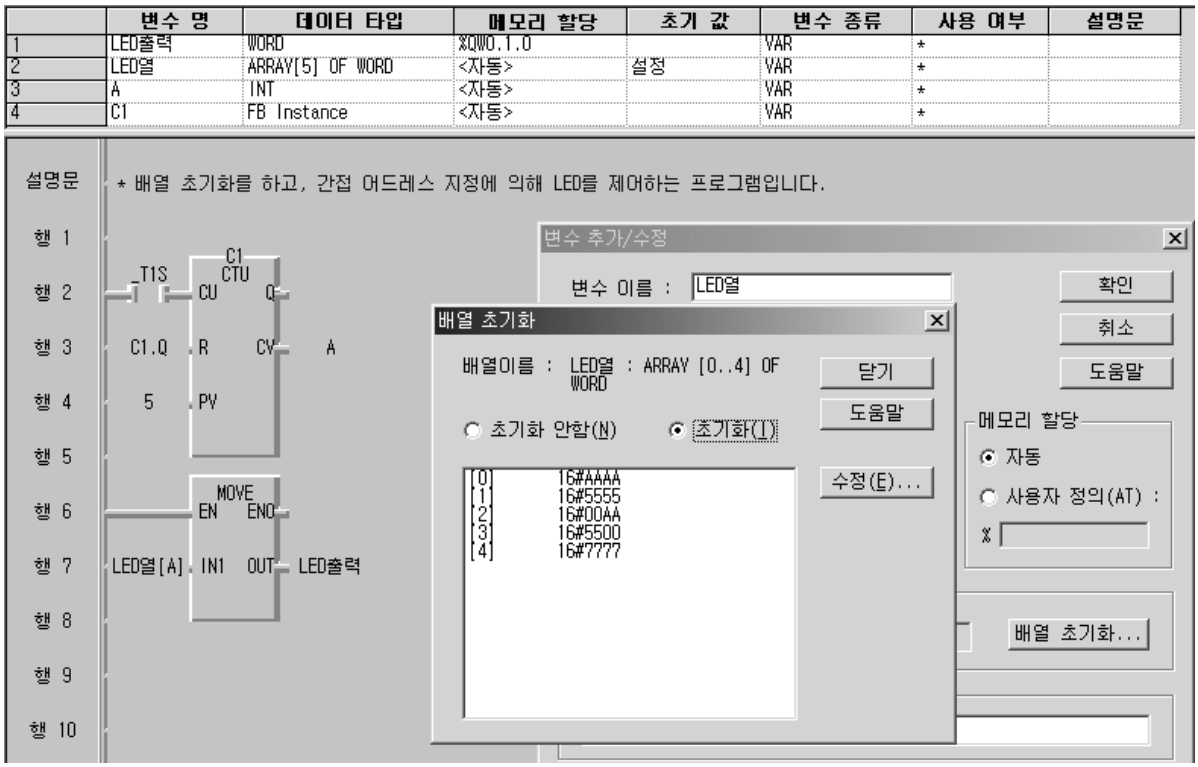
(2) 프로그램 편집

배열 변수 "LED 열"의 배열원소에 미리 입력해둔 초기값 데이터를 1 초 간격으로 원소 번호순으로 변수 "LED 출력"으로 전송하여 %QW0.1.0 의 출력 모듈로 출력하는 프로그램 예입니다.

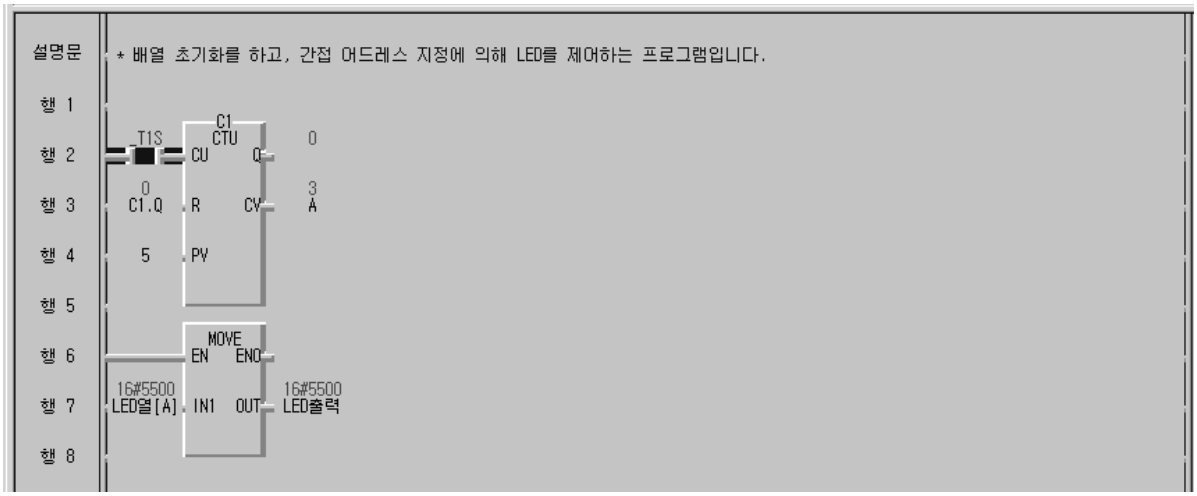
<배열변수 선언 및 초기값 입력>

변수명 "LED 열"을 WORD 형 배열 변수로 선언합니다.





프로그램 모니터링 결과



INT 형 변수 "A"의 현재값이 3 이므로 "LED 열[A]"의 원소 번호가 3 이 되어, 결국 "LED 열[3]"에 들어 있는 데이터값 16#5500 이 LED 출력으로 전송됩니다.

제 3 장 응용 라이브러리의 활용

3.1 응용 펑션 라이브러리

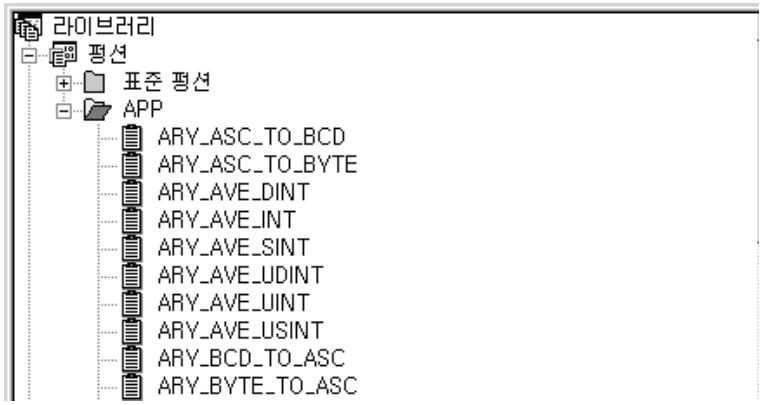
기존의 MASTER-K 사용자 편의를 위해 제공되는 라이브러리 파일입니다. 응용 펑션 라이브러리 파일(APP.*fu)을 사용할 때는 반드시 프로젝트에 추가로 등록해야 합니다. 단, 종래의 확장 라이브러리 파일(mkstdlib.*fu) 내의 펑션은 응용 펑션 라이브러리 파일(APP.*fu)에 포함되어 있습니다.

No.	펑션 이름	기 능
1	ENCO_BYTE,WORD,DWORD	On 된 비트 위치를 숫자로 출력
2	DECO_BYTE,WORD,DWORD	숫자에 대응되는 비트 위치를 On
3	BSUM_BYTE,WORD,DWORD	On 된 비트 개수를 숫자로 출력
4	BMOV_BYTE,WORD,DWORD	비트 스트링의 일부분을 복사 이동

3.1.1 응용 펑션 프로그램 예

(1) 프로젝트 항목 추가

응용 펑션을 사용하려면 프로젝트 항목 추가를 통해 APP.4fu 를 프로젝트에 추가하여야 합니다.



(2) 프로그램 작성 예

- 10 진 숫자 입력 프로그램 예(ENCO_WORD 펑션 활용)

개요

10 진 Key(Ten Key)의 조작에 의하여 지정한 변수에 원하는 숫자를 써넣는 프로그램입니다.

입출력 구성

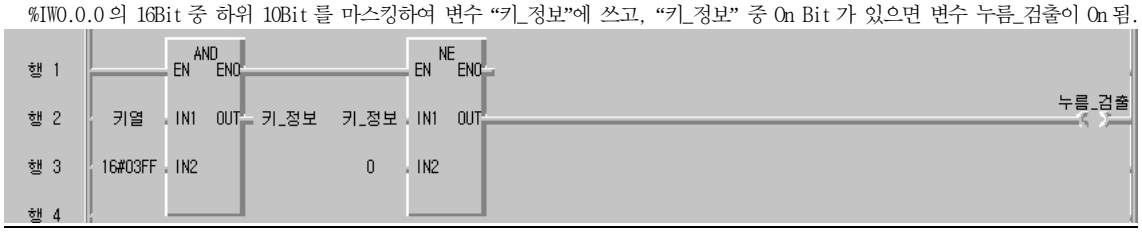
숫자 0 : %IX0.0.0
숫자 1 : %IX0.0.1
숫자 2 : %IX0.0.2
숫자 3 : %IX0.0.3
숫자 4 : %IX0.0.4
숫자 5 : %IX0.0.5
숫자 6 : %IX0.0.6
숫자 7 : %IX0.0.7
숫자 8 : %IX0.0.8
숫자 9 : %IX0.0.9
정정 : %IX0.0.14
확인 : %IX0.0.15

프로그램 작성

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	키열	WORD	%IW0.0.0		VAR	*	
2	정정	BOOL	%IX0.0.14		VAR	*	
3	확인	BOOL	%IX0.0.15		VAR	*	
4	표시기	WORD	%QW0.1.1		VAR	*	
5	누름_검출	BOOL	<자동>		VAR	*	
6	버퍼	INT	<자동>		VAR	*	
7	정수값	INT	<자동>		VAR	*	
8	키_정보	WORD	<자동>		VAR	*	
9	BCD값	WORD	<자동>		VAR	*	

행 1 의 AND 및 NE 평션

%IW0.0.0 의 16Bit 중 하위 10Bit 를 마스킹하여 변수 "키_정보"에 쓰고, "키_정보" 중 On Bit 가 있으면 변수 누름_검출이 On 됨.

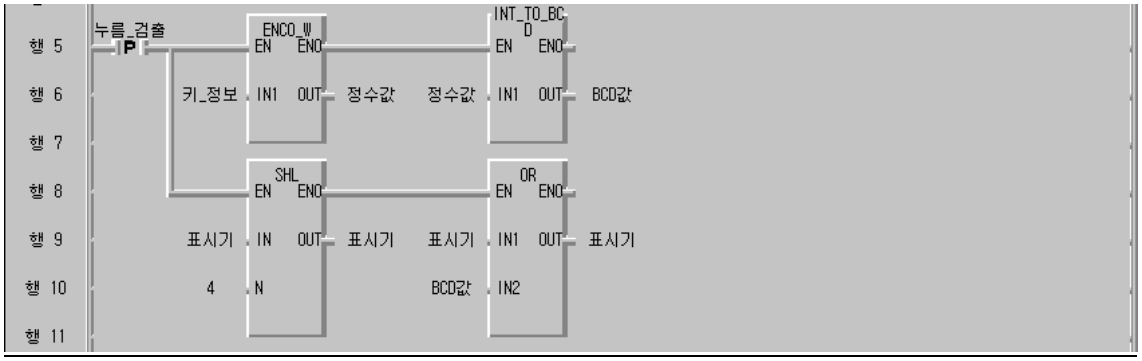


행 5 의 ENCO_W(또는 ENCO_WORD) 및 INT_TO_BCD 평션

"키_정보" 중 On Bit 의 수치값를 구하여 변수 "정수값"에 저장하고, 이를 BCD 값으로 변환해 변수 "BCD 값"에 저장.

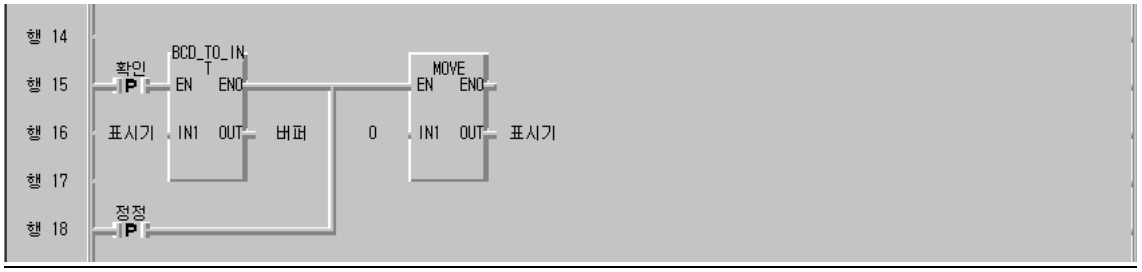
행 8 의 SHL 및 OR 평션

"표시기"의 Bit 열을 4 Bit 만큼 왼쪽으로 이동함. "표시기" + "BCD 값" = "표시기"



행 15 의 BCD_TO_INT 및 MOVE

"표시기"를 INT 형으로 변환하여 "버퍼"에 저장하고, 다음 수치 입력을 위해 "표시기"를 클리어함.



제 4 장 사용자 정의 라이브러리 작성

사용자가 직접 만든 펑션 블록(또는 펑션)을 저장하는 라이브러리 파일을 말합니다.

메뉴의 “프로젝트/ 라이브러리/ 프로젝트 항목 추가/ 라이브러리”에서 해당 파일을 추가하여 사용합니다. 사용자 정의 라이브러리 파일에는 펑션(**.?fu) 및 펑션 블록(**.?fb)이 있습니다.

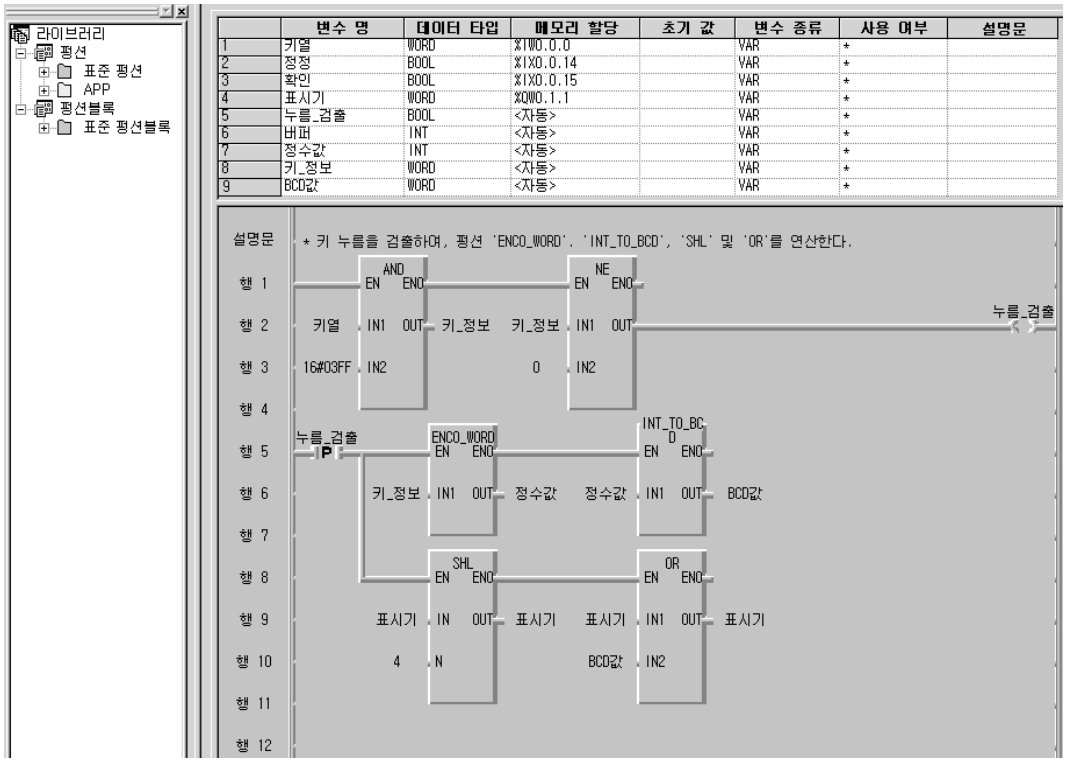
(?는 기종에 따라 1, 2, 3, 4, 6,7 로 구분됨)

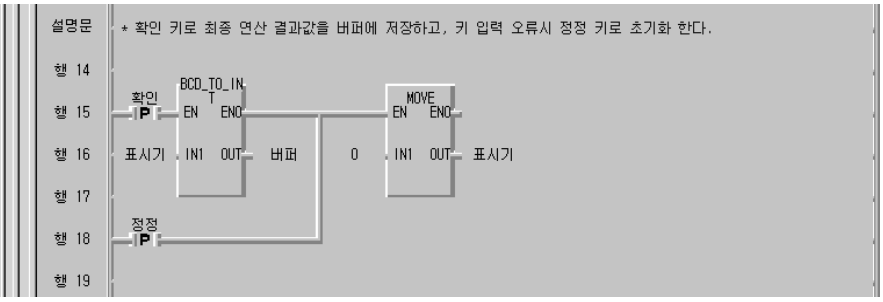
자주 사용하는 프로그램들을 표준화하여 펑션 블록(또는 펑션)으로 만들어 사용자 라이브러리 파일에 등록해 두고, 다른 프로젝트 수행 시 재 활용함으로써 프로그램의 작성을 편리하고 신속하게 수행할 수 있는 기능입니다.

4.1 사용자 정의 펑션블록 작성 예

제 3 장에서 작성한 ‘10 진 숫자 입력 프로그램(텐키 입력 프로그램)’을 **사용자 정의 펑션 블록**으로 만들어 활용하는 예입니다.

(1) 기 작성한 ‘10 진 숫자 입력 프로그램’ 열기

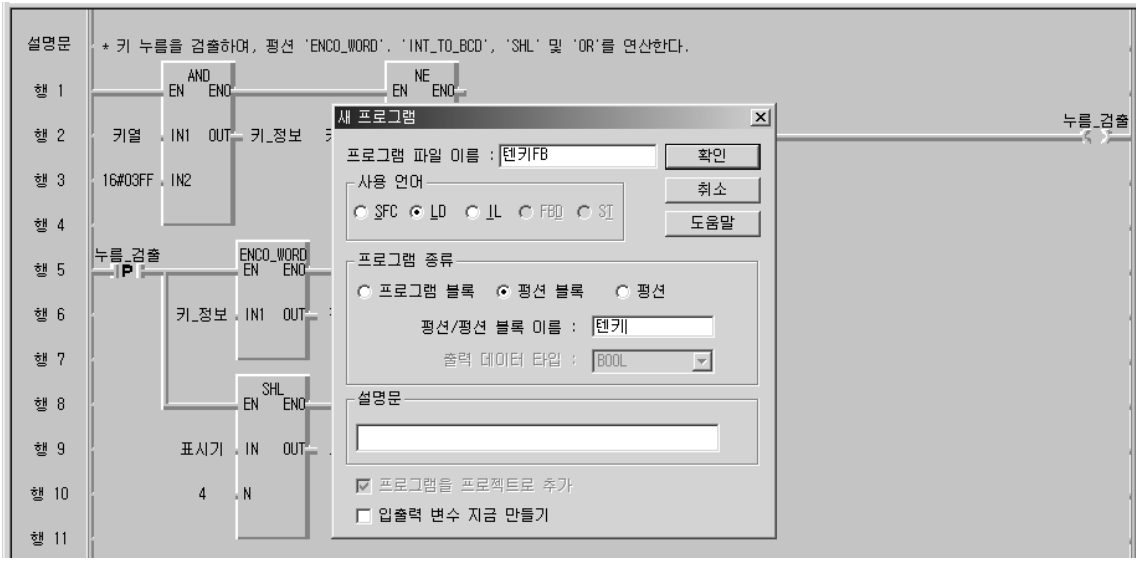




주) ‘10 진 숫자 입력 프로그램에서 응용 펑션 라이브러리 파일(APP.4fu)의 ENCO_WORD 펑션이 사용되므로, 라이브러리에 이 응용 펑션 라이브러리 파일이 추가되어야 합니다.

(2) 새 프로그램을 열어 프로그램 종류를 '펑션 블록'으로 변경하기

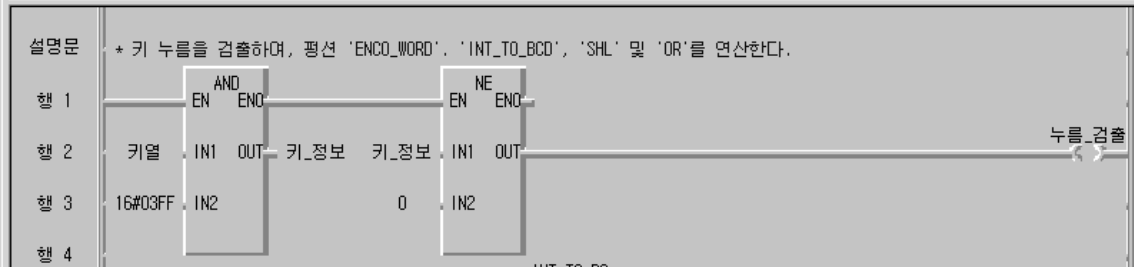
'10 진 숫자 입력 프로그램'이 열린 상태에서 새 프로그램을 열어 프로그램 파일 이름을 '텐키 FB'로 명명하고, 프로그램 종류를 '프로그램 블록'에서 '펑션 블록'으로 변경한다. 또한, 펑션 블록 이름 '텐키'를 부여한다.

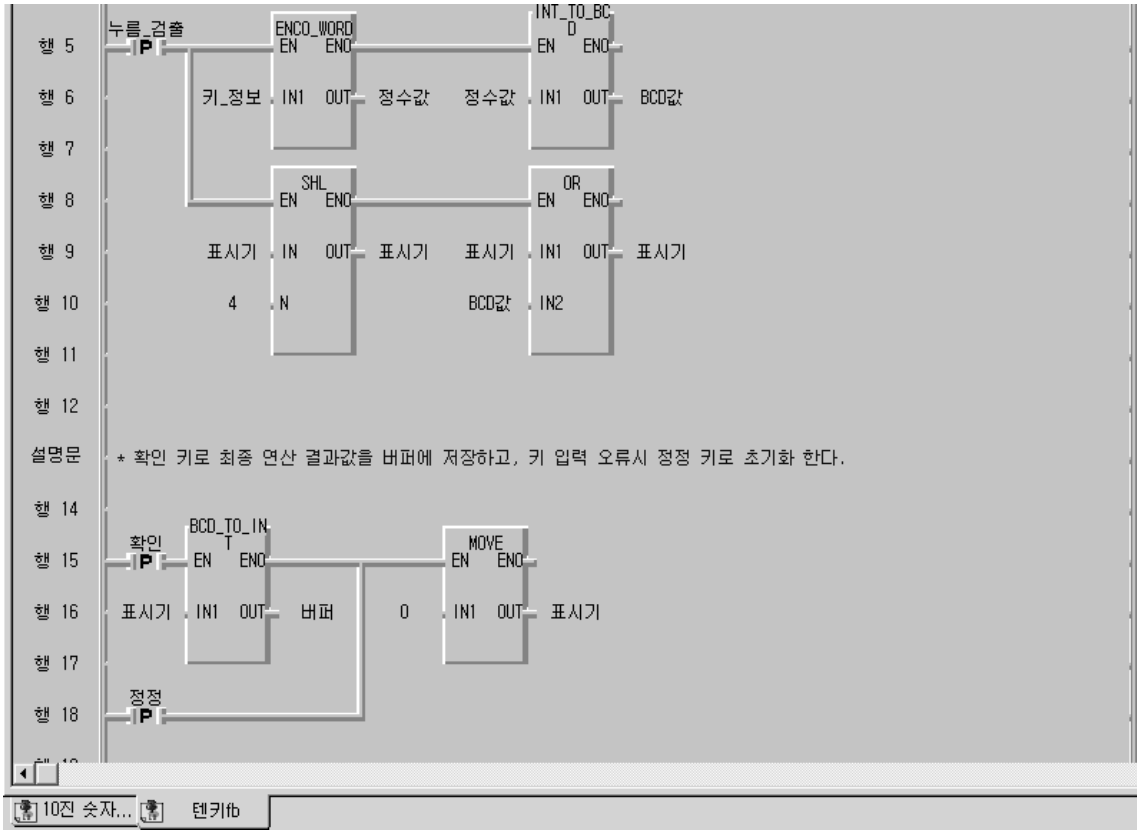


(3) 프로그램 블록을 복사하여 펑션 블록에 붙여넣기

프로그램 블록 '10 진 숫자 입력 프로그램'을 복사하여 펑션 블록 '텐키 FB'에 붙여넣고, '텐키 FB'내 지역 변수의 메모리 할당을 모두 '자동'으로 수정한다.

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	키열	WORD	<자동>		VAR		
2	정정	BOOL	<자동>		VAR		
3	확인	BOOL	<자동>		VAR		
4	표시기	WORD	<자동>		VAR		
5	누름_검출	BOOL	<자동>		VAR		
6	버퍼	INT	<자동>		VAR		
7	정수값	INT	<자동>		VAR		
8	키_정보	WORD	<자동>		VAR		
9	BCD값	WORD	<자동>		VAR		

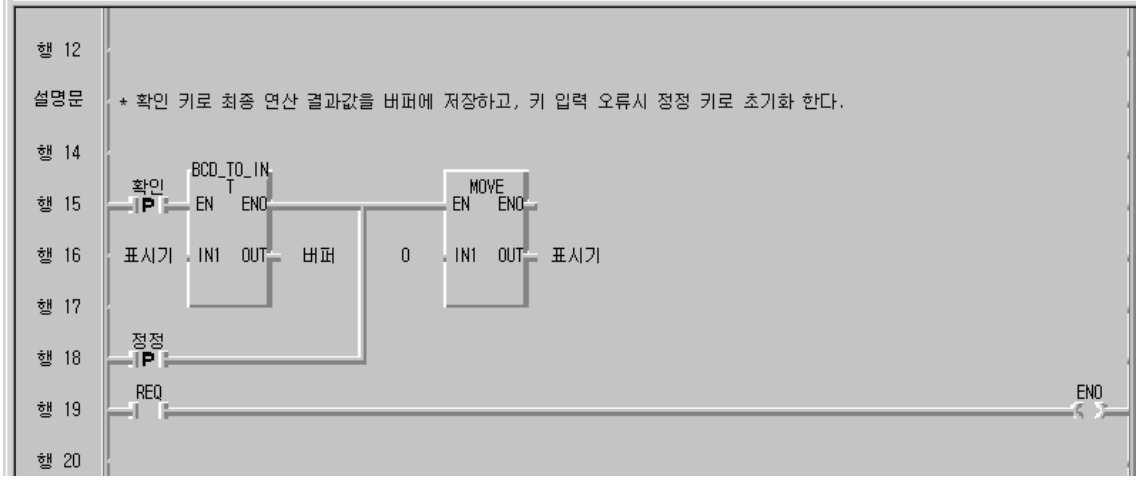




(4) 펑션 블록 '텐키 FB'의 끝 행에 입력 및 출력 BOOL 변수 추가 편집

펑션 블록에는 1 개 이상의 입출력 BOOL 변수가 있어야 하므로, '텐키 FB'의 끝 행에 입출력 BOOL 변수를 추가로 편집한다.

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	누름_검출	BOOL	<자동>		VAR		
2	버퍼	INT	<자동>		VAR		
3	정수값	INT	<자동>		VAR		
4	정정	BOOL	<자동>		VAR		
5	키_정보	WORD	<자동>		VAR		
6	키열	WORD	<자동>		VAR		
7	표시기	WORD	<자동>		VAR		
8	확인	BOOL	<자동>		VAR		
9	BCD값	WORD	<자동>		VAR		
10	ENO	BOOL	<자동>		VAR		
11	REQ	BOOL	<자동>		VAR		



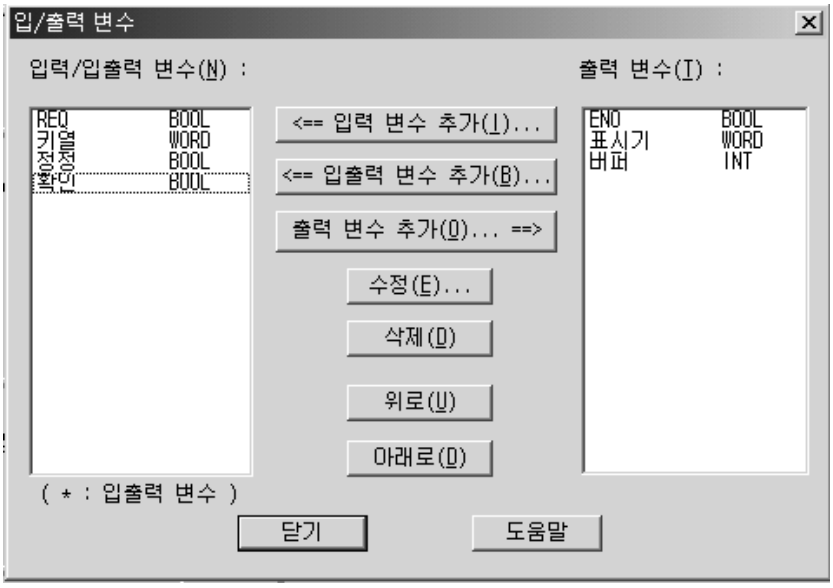
(5) 펑션 블록 '텐키 FB'의 지역 변수 목록에서 입출력 변수로 정의할 변수 삭제

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	누름_검출	BOOL	<자동>		VAR		
2	버퍼	INT	<자동>		VAR		
3	정수값	INT	<자동>		VAR		
4	정정	BOOL	<자동>		VAR		
5	키_정보	WORD	<자동>		VAR		
6	키열	WORD	<자동>		VAR		
7	표시기	WORD	<자동>		VAR		
8	확인	BOOL	<자동>		VAR		
9	BCD값	WORD	<자동>		VAR		
10	ENO	BOOL	<자동>		VAR		
11	REQ	BOOL	<자동>		VAR		

↓

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	누름_검출	BOOL	<자동>		VAR		
2	정수값	INT	<자동>		VAR		
3	키_정보	WORD	<자동>		VAR		
4	BCD값	WORD	<자동>		VAR		

(6) 입출력 변수 정의

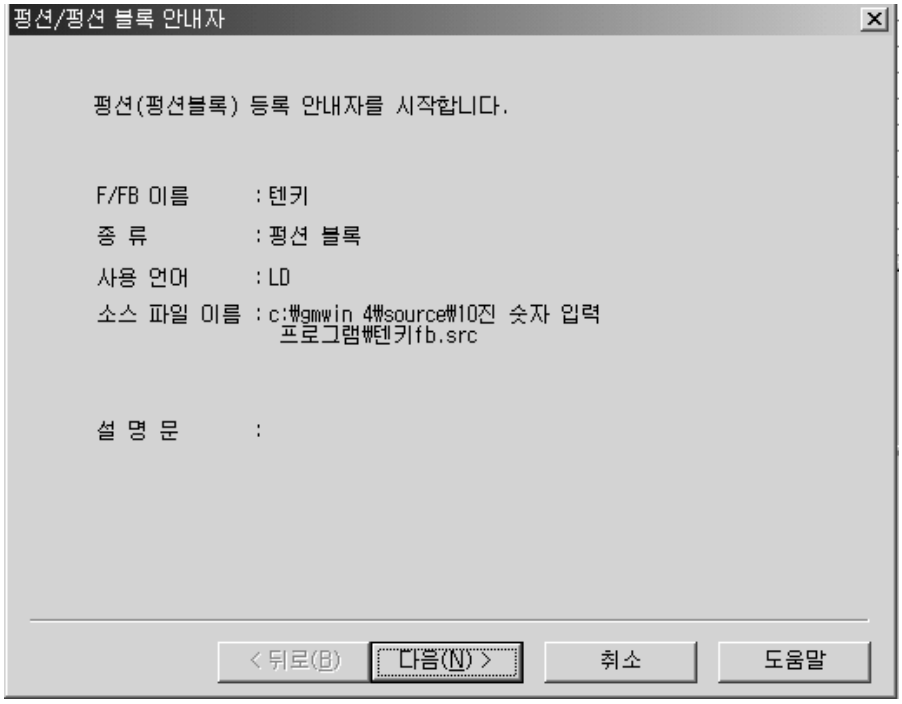


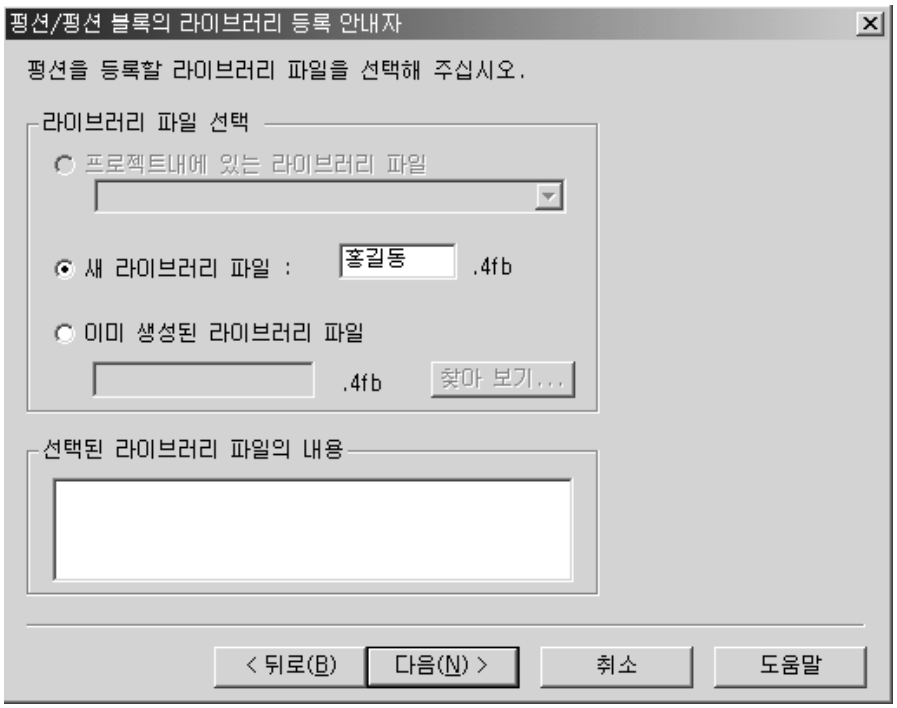
주) 펑션 블록은 맨 위의 첫 번째 입출력 변수는 BOOL 형이어야 합니다.

만약, BOOL 형 입출력이 프로그램에 필요하지 않을 경우에도 Dummy 변수로 선언해야 합니다.

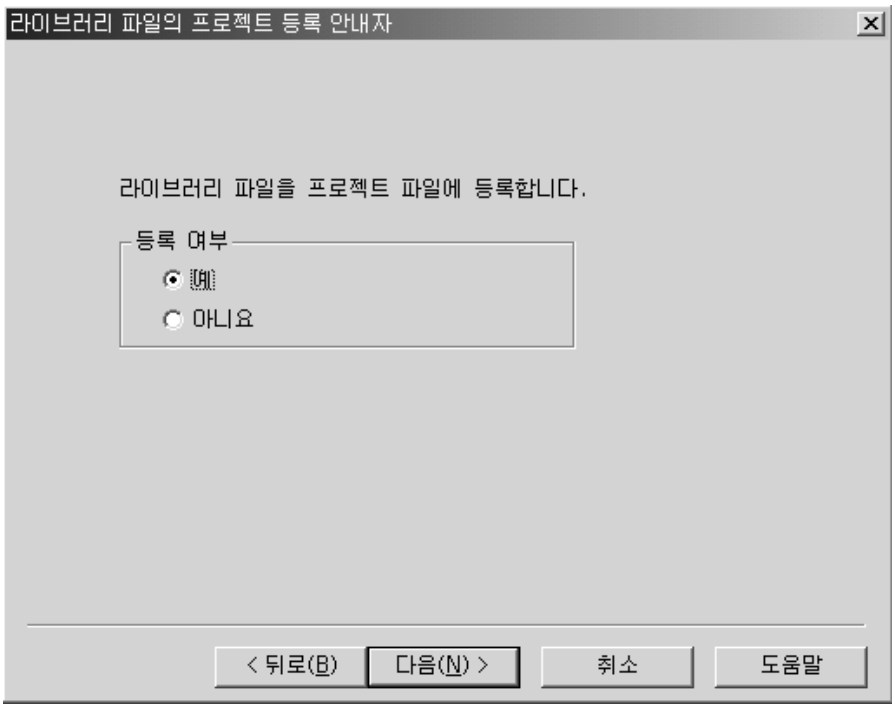
(7) 컴파일을 수행한 후 확인 단추를 누르면 '펑션 블록 안내자'가 나타납니다. 다음 단추를 눌러 나타난 '펑션 블록의 라이브러리 등록 안내자'에 새 라이브러리 파일 이름을 임의로 부여합니다.

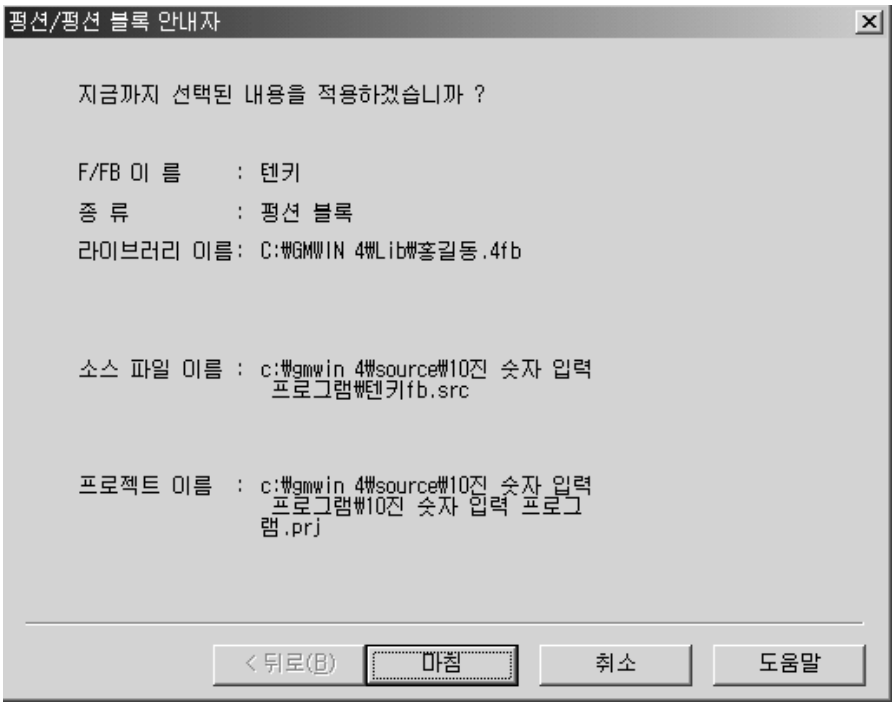
(홍길동.4fb)



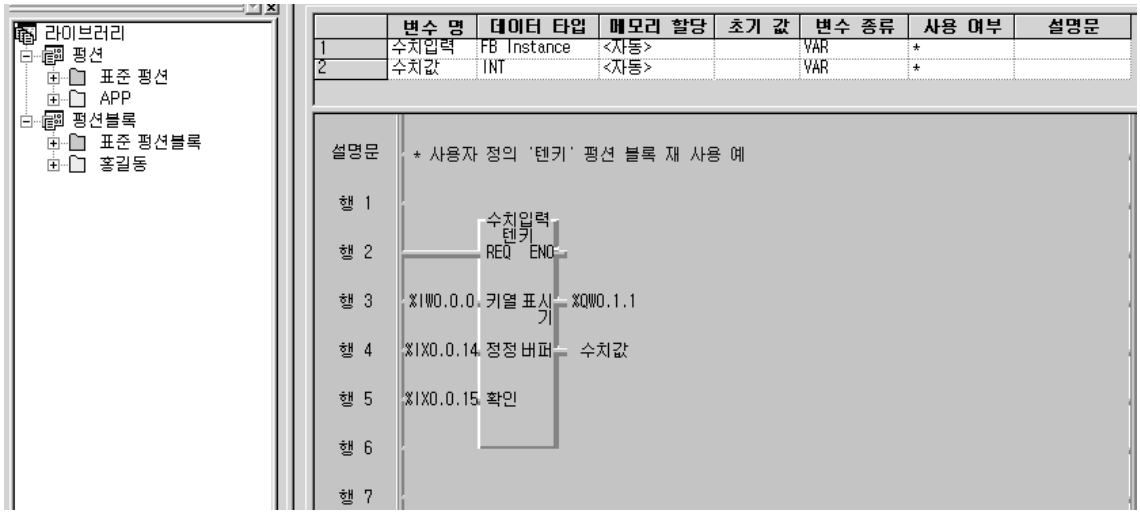


(8) '라이브러리 파일의 프로젝트 등록 안내자'를 통하여 라이브러리 파일을 프로젝트 파일에 등록하고 관련 내용을 적용합니다.





(9) 작성한 '텐키' 사용자 정의 펑션 블록을 재 사용합니다.



제 5 장 SFC(Sequencial Function Chart)프로그램 작성

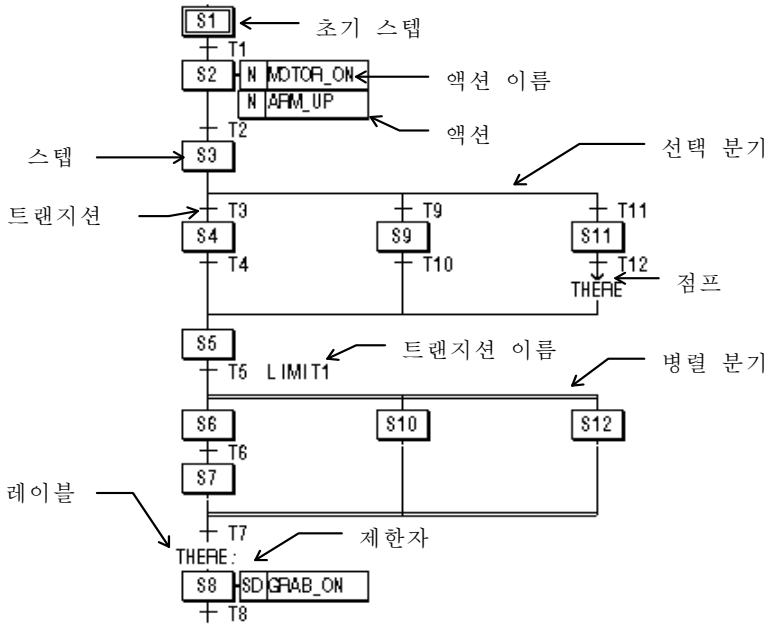
5.1 SFC 개요

▷ SFC 는 종래의 PLC 언어를 이용하여 응용 프로그램을 실행처리 순서에 따라 플로 차트 (Flow Chart) 형식으로 전개하는 구조화 표현 방식 언어입니다.

▷ SFC 는 응용 프로그램을 스텝과 트랜지션으로 분할하여 서로 연결하는 방법을 제공하며각 스텝은 액션으로, 각 트랜지션은 트랜지션 조건과 연관됩니다.

▷ SFC 는 상태 정보를 가지고 있어야 하기 때문에 프로그램 종류 중 프로그램 블록과 사용자 펑션블록만 SFC 로 작성할 수 있습니다.(사용자 펑션은 작성 불가)

▷ 형태



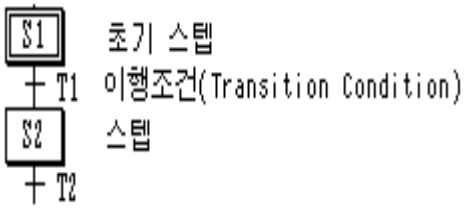
5.2 SFC 구조

5.2.1 스텝

▷ 스텝은 액션이 연결됨으로써 시퀀스 제어의 단위를 나타냅니다.

▷ 스텝이 활성화 상태이면 부착되어 있는 액션의 내용이 실행됩니다.

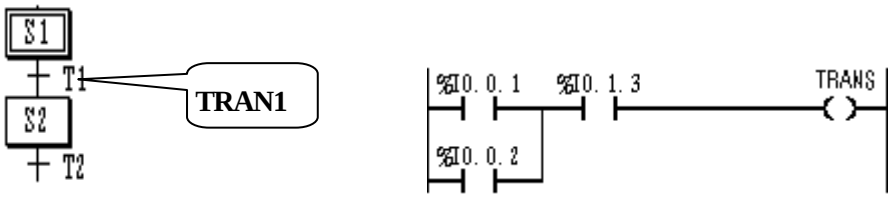
▷ 초기 스텝은 최초로 활성화되는 스텝입니다.



▷ 최초의 활성화 상태인 초기 스텝(S1)의 다음 이행 조건(Transition Condition)이 성립되면 현재 활성화 상태인 스텝(S1)은 비활성화 상태로 되고 다음에 연결된 스텝(S2)이 활성화 상태로 됩니다.

5.2.2 트랜지션

▷ 트랜지션은 스텝간의 실행 처리 이행 조건을 나타냅니다.
▷ 이행 조건은 PLC 언어인 IL 또는 LD 로 표현되어야 합니다.
▷ 이행 조건의 결과는 항상 BOOL 로 되어야 하며그 변수의 이름은 어느 트랜지션이나 TRANS 가 됩니다.
▷ 이행 조건의 결과가 1 일 경우 현재 스텝은 비 활성화되고 다음 스텝이 활성화됩니다.
▷ 스텝과 스텝 사이에는 반드시 트랜지션이 있어야 합니다.



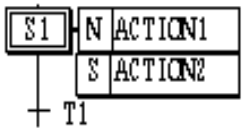
TRAN1 의 내용

TRANS 가 On 되면 S1 이 비활성화되고 S2 가 활성화 상태가 됩니다.
TRANS 변수는 내부적으로 선언된 변수입니다.
모든 트랜지션에서 이행 조건을 TRANS 변수로 출력시켜야 합니다.

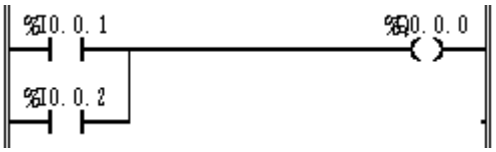
5.2.3 액션

▷ 각 스텝에는 액션을 2 개까지 연결할 수 있습니다.
▷ 액션이 없는 스텝은 대기 액션으로 여겨지며 다음의 이행 조건이 1 이 될 때까지 대기상태가 됩니다.
▷ 액션은 PLC 언어인 IL 또는 LD 로 구성되고 스텝이 활성화될 동안 액션의 내용이 실행됩니다.
▷ 액션 제한자가 액션을 제어하는 데 사용됩니다.
▷ 액션이 활성화되었다가 비활성화 상태로 될 때 액션에서 실행된 접점 출력은 0 으로 됩니다.
단, Set 출력, 펑션, 펑션블록 출력은 비 활성화되기 전의 상태가 유지됩니다.

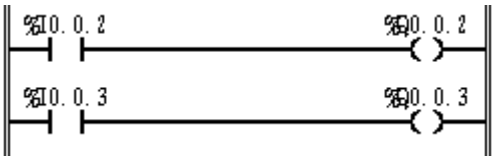
STEP 의 형태



ACTION1 의 내용



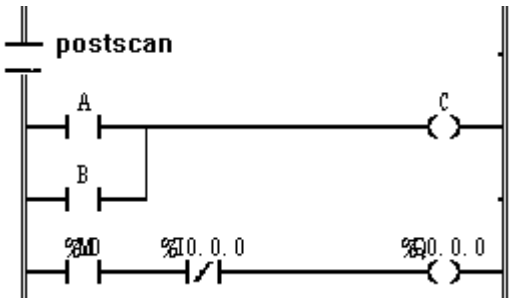
ACTION2 의 내용



- ACTION1 은 S1 이 활성화된 경우에만 실행됩니다.
- ACTION2 은 S1 이 활성화된 후 R 제한자를 만날 때까지 실행됩니다.
  S1 이 비 활성화되어도 계속 실행합니다.
- 액션이 비 활성화되는 순간 이 액션을 포스트 스캔(Post Scan)한 후 다음 스텝으로 넘어갑니다.

포스트 스캔

활성화된 액션이 비 활성화되는 순간 해당 액션은 ON 된 코일 출력을 모두 OFF 합니다.
단 펑션, 펑션블록, Set 출력 등은 해당되지 않습니다.



그림에서 postscan 접점이 0 이므로 C 와 %Q0.0.0 은 0 이 됩니다.

5.2.4 액션 제한자(Action Qualifier)

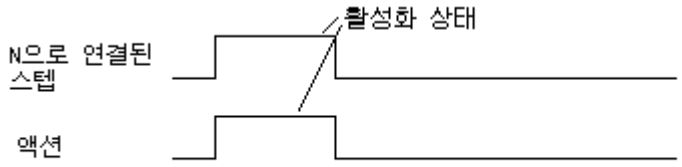
▷ 액션이 사용될 때마다 액션 제한자가 사용됩니다.

▷ 스텝에 연관된 액션은 지정된 제한자에 따라 실행 시점과 시간이 정의됩니다.

액션 제한자의 종류는 다음과 같습니다.

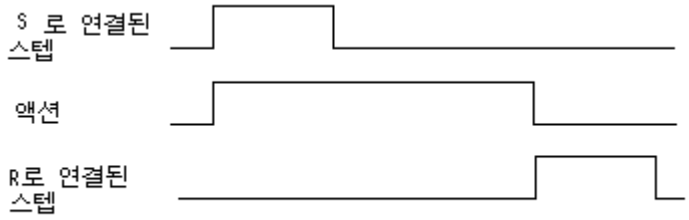
1) N(Non-Stored)

스텝이 활성화된 동안만 액션이 실행됩니다.



2) S(Set)

스텝이 활성화되면 R 제한자가 실행될 때까지 액션이 실행됩니다.

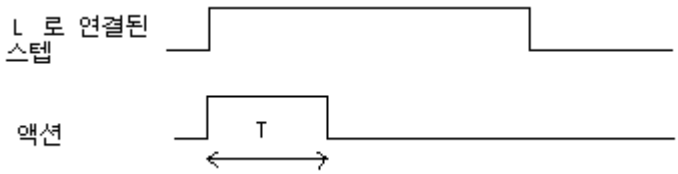


3) R(Overriding Reset)

이전에 S, SD, DS, SL 제한자로 실행된 액션의 실행을 중지시킵니다.

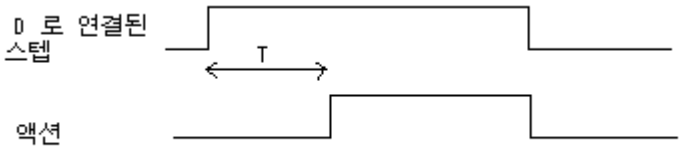
4) L(Time Limited)

스텝이 활성화된 후 지정된 시간 또는 스텝이 비 활성화될 때까지 액션이 실행됩니다.



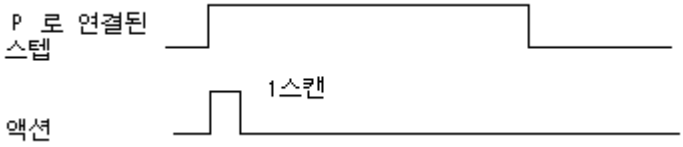
5) D(Time Delayed)

스텝이 활성화된 후 지정된 시간이 경과한 후부터 비활성화될 때까지 액션이 실행됩니다.



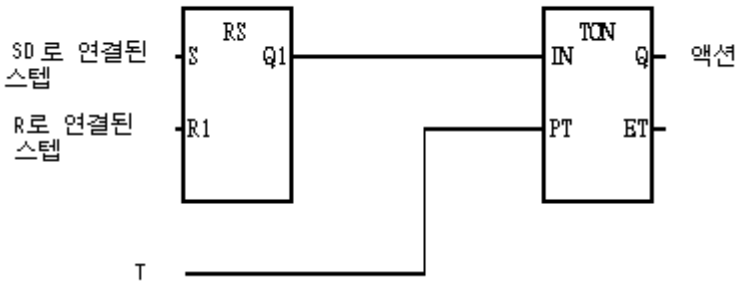
6) P(Pulse)

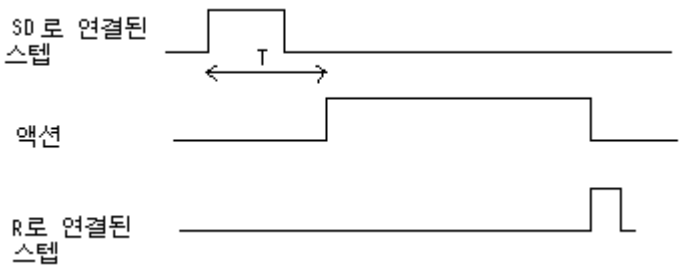
스텝이 활성화된 순간에만 액션이 실행됩니다.



7) SD(Stored & Time Delayed)

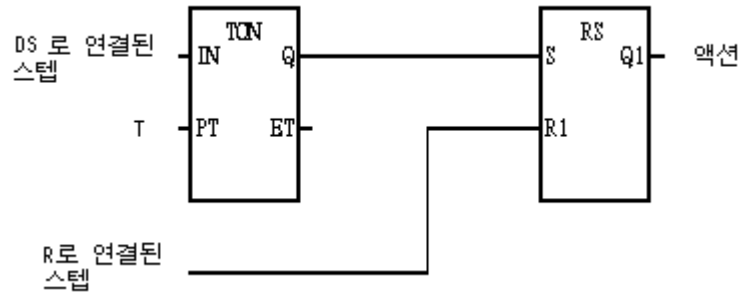
스텝이 활성화된후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 R 제한자가 실행되면 액션은 실행되지 않습니다.

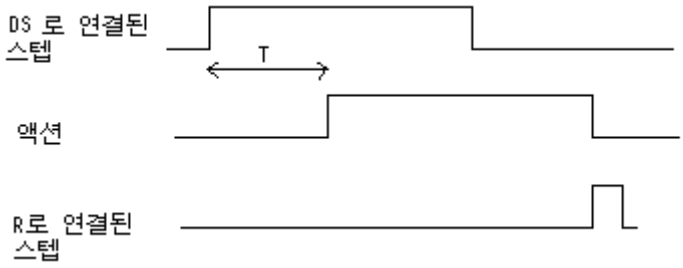




8) DS(Delayed & Stored)

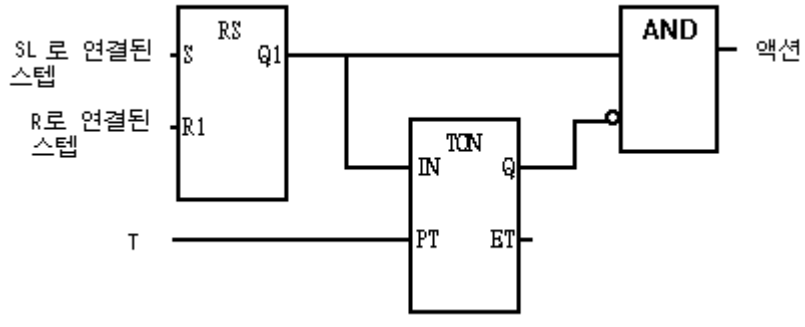
스텝이 활성화된 후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 스텝이 비 활성화되거나 R 제한자가 실행되면 액션은 실행되지 않습니다.

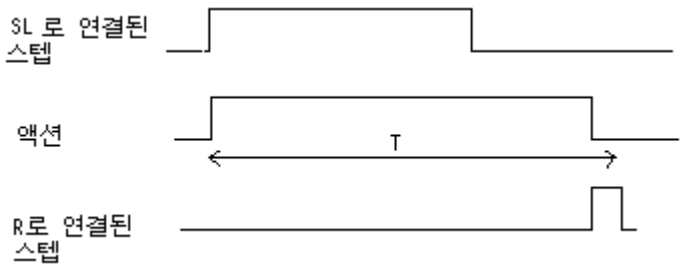




9) SL(Stored & Timed Limited)

스텝이 활성화된 후 지정된 시간 또는 R 제한자가 실행될 때까지 액션이 실행됩니다.





5.3 SFC 전개 규칙

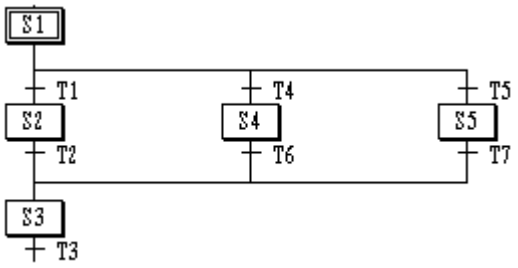
5.3.1 직렬 연결

▷ 2 개의 스텝은 직접 연결되지 않고 항상 트랜지션에 의해 분리됩니다.
▷ 2 개의 트랜지션은 직접 연결되지 않고 항상 스텝에 의해 분리됩니다.
▷ 직렬로 연결되어 있는 스텝간의 이행은 상위 스텝이 활성화된 상태에서 다음에 연결된 트랜지션의 이행조건이 1 로 되면 하위스텝이 활성화 상태가 됩니다.

5.3.2 선택 분기

▷ 선택 분기로 연결되어 있으면 상위 스텝이 활성화된 상태에서 다음에 연결된 2 개 이상의 트랜지션 중 이행조건이 1 로된 곳의 다음스텝이 활성화됩니다.
그 다음은 직렬 연결과 동일합니다.
▷ 선택 분기는 여러 트랜지션 중 먼저 1 로 된 곳의 해당 액션만 실행되고 나머지 액션은 실행되지 않습니다.
▷ JUMP 의 삽입은 선택 분기의 끝에만 가능합니다.

예



- T1 의 이행 조건이 1 이 되었을 경우
  S1 -> S2 -> S3 순으로 활성화 상태가 됩니다.
- T4 의 이행 조건이 1 이 되었을 경우
  S1 -> S4 -> S3 순으로 활성화 상태가 됩니다.
- T5 의 이행 조건이 1 이 되었을 경우
  S1 -> S5 -> S3 순으로 활성화 상태가 됩니다.

이행 조건이 동시에 1 이 되었을 경우에는 가장 왼쪽에 있는 트랜지션 쪽으로 이행됩니다.

- T1,T4 의 이행 조건이 동시에 1 이 되었을 경우
  S1 -> S2 -> S3 순으로 활성화 상태가 됩니다.
- T4,T5 의 이행 조건이 동시에 1 이 되었을 경우
  S1 -> S4 -> S3 순으로 활성화 상태가 됩니다.

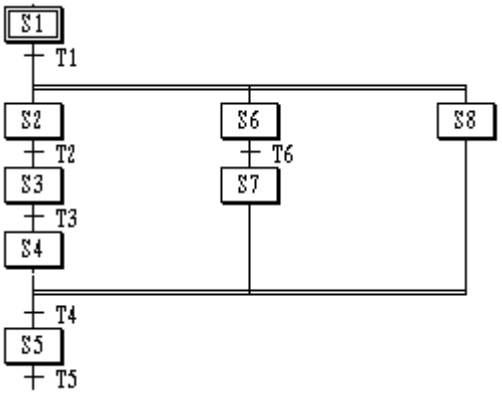
5.3.3 병렬 분기

▷ 병렬 분기로 연결되어 있으면 상위 스텝이 활성화 상태에서 다음에 연결되어있는 트랜지션의 이행 조건이 1 로 되면 이 트랜지션 밑에 연결된 모든 스텝이 활성화 상태로 됩니다. 각 분기의 전개는 직렬연결과 동일합니다. 이때 활성화 상태인 스텝은 분기의 수 만큼 존재하게 됩니다.

▷ 병렬 분기에서 합쳐질 경우 각 분기의 마지막 스텝이 모두 활성화 상태에서 트랜지션의 이행조건이 1 로 되면 다음에 연결된 스텝이 활성화 상태로 됩니다.

▷ 병렬 분기는 두가지 동작을 동시에 실행합니다.

예



- S1 이 활성화된 상태에서 이행 조건 T1 이 1 이면 S2, S6, S8 이 활성화 상태로 되고 S1 이 비활성화 상태로 됩니다.
- S4, S7, S8 이 활성화된 상태에서 이행 조건 T4 가 1 이면 S5 가 활성화 상태로 되고 S4, S7, S8 은 비활성화 상태로 됩니다.

● Active 순서

S1-+->S2--->S3--->S4--+->S5

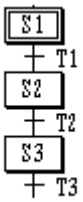
+->S6--->S7----------+

+->S8----------+

5.3.4 점프

▷ SFC 마지막 스텝이 활성화 상태로 된 후 다음에 연결되어 있는 트랜지션의 이행 조건이 1 로 되면 SFC 초기 스텝(Initial Step)이 활성화 상태로 됩니다.

예

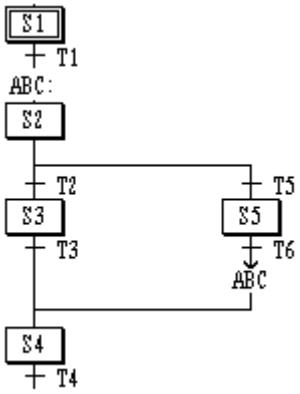


▷ 점프를 사용하면 원하는 곳으로 전개를 이어 나갈 수 있습니다.

▷ 점프는 SFC 프로그램 끝 또는 선택 분기 끝에만 올 수 있습니다.
병렬 분기 안으로 또는 밖으로는 점프할 수 없습니다.
병렬 분기 안에서의 점프는 가능합니다.

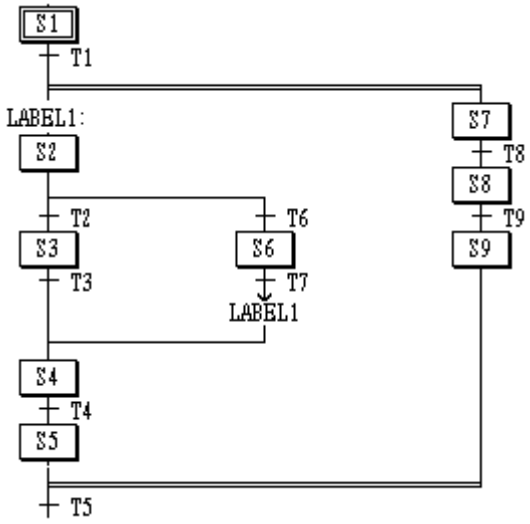
예

1) 선택분기 끝에서의 점프

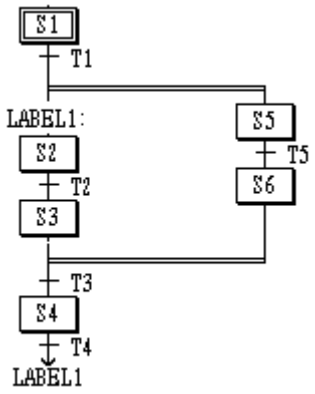


- S5 다음에는 S2 이 활성화됩니다.

2) 병렬 분기 안에서의 점프



3) 병렬 분기 안으로는 점프할 수 없습니다.



5.4 SFC 프로그램 예

5.4.1 직렬 연결 프로그램 예

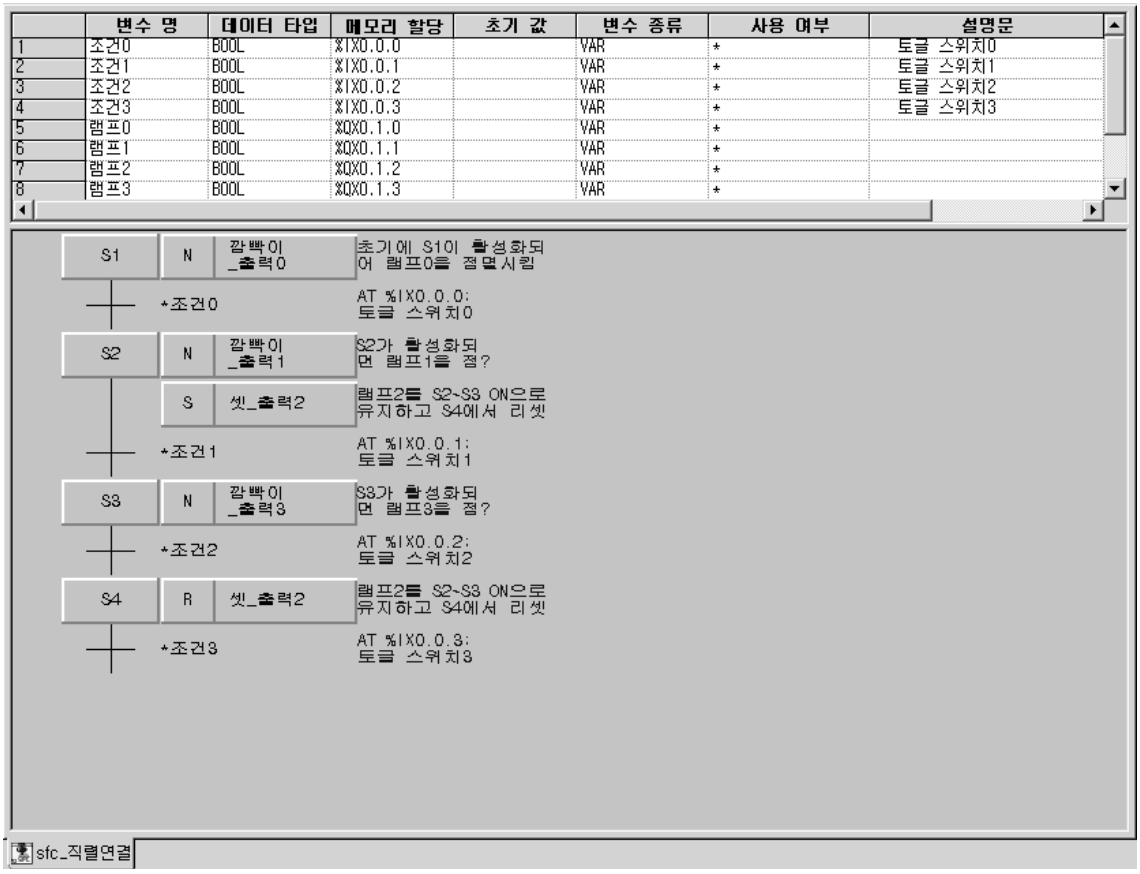
(1) 프로그램 편집

S1 의 (N)‘깜빡이_출력 0’ : 초기에 S1 이 활성화되어 램프 0 을 점멸 시킴.

S2 의 (N)‘깜빡이_출력 1’ : S2 가 활성화되면 램프 1 을 점멸 시킴.
S2 의 (S)‘셋_출력 2’ : 램프 2 를 S2~S3 에서 ON 으로 유지시킴.

S3 의 (N)‘깜빡이_출력 3’ : S3 가 활성화되면 램프 3 을 점멸 시킴.

S4 의 (R)‘셋_출력 2’ : 램프 2 를 S4 에서 리셋시킴.



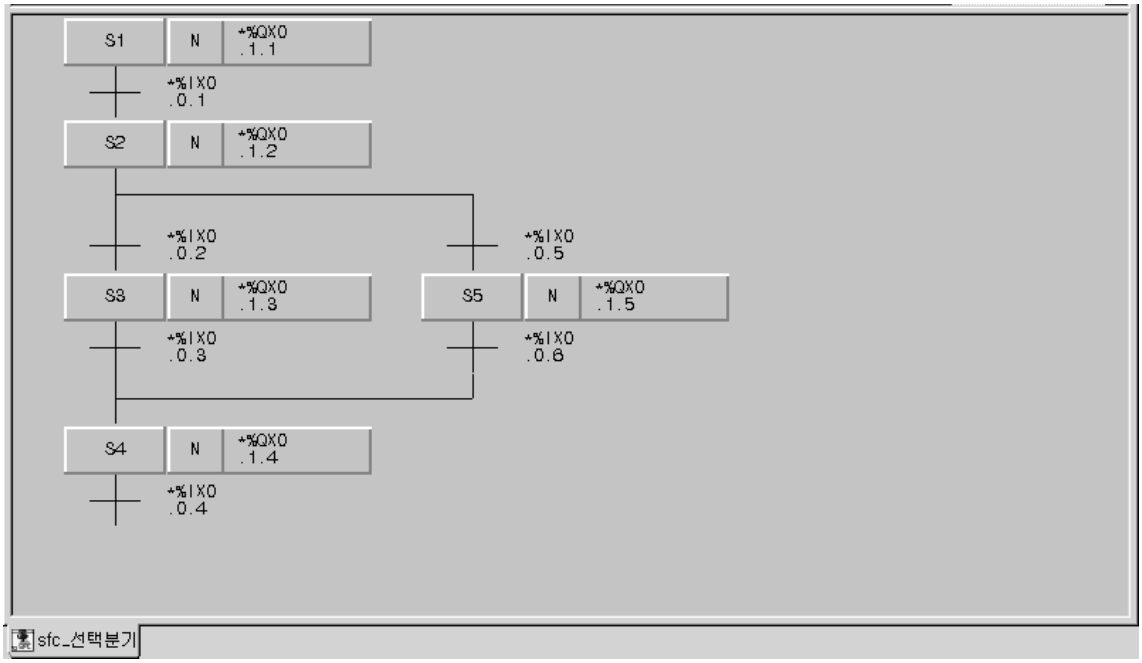

c:₩gmwin 4₩source₩sfc직렬₩sfc_직렬연결.src[액션:깜빡이_...
설명문
* 액션: 깜빡이_출력0
행 1
_T1S
램프0
행 2
행 3
행 4
행 5
c:₩gmwin 4₩source₩sfc직렬₩sfc_직렬연결.src[액션:셋_출력2]
* 액션: 셋_출력2
_ON
램프2
c:₩gmwin 4₩source₩sfc직렬₩sfc_직렬연결.src[액션:깜빡이_...
* 액션: 깜빡이_출력1
램프1
* 액션: 깜빡이_출력3
램프3

(2) 프로그램 모니터링


c:₩gmwin 4₩source₩sfc직렬₩sfc_직렬...
S1
N
깜빡이
_출력0
+조건0
S2
N
깜빡이
_출력1
S
셋_출력2
+조건1
S3
N
깜빡이
_출력3
+조건2
S4
R
셋_출력2
+조건3
설명문
* 액션: 깜빡이_출력1
행 1
_T1S
램프1
행 2
행 3
행 4
행 5
행 6
행 7
행 8
행 9
행 10
행 11
* 액션: 셋_출력2
_ON
램프2

5.4.2 선택 분기 프로그램 예

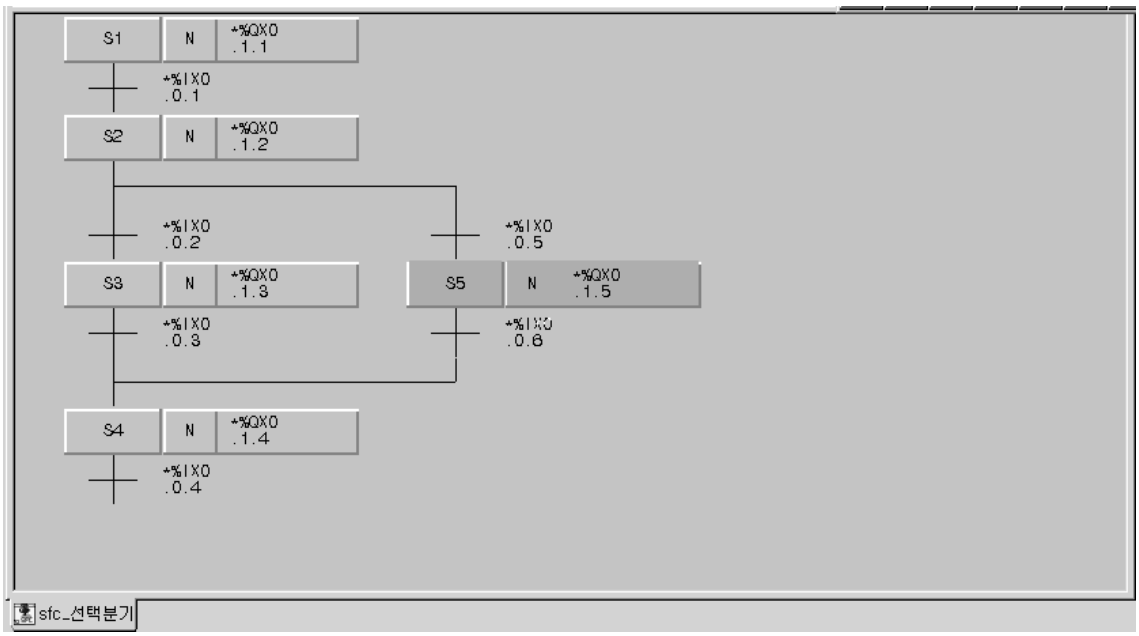
(1) 프로그램 편집



(2) 프로그램 모니터링

☞ 스텝 S2 가 활성화되면 램프 %Q0.1.2 가 점등 됩니다.

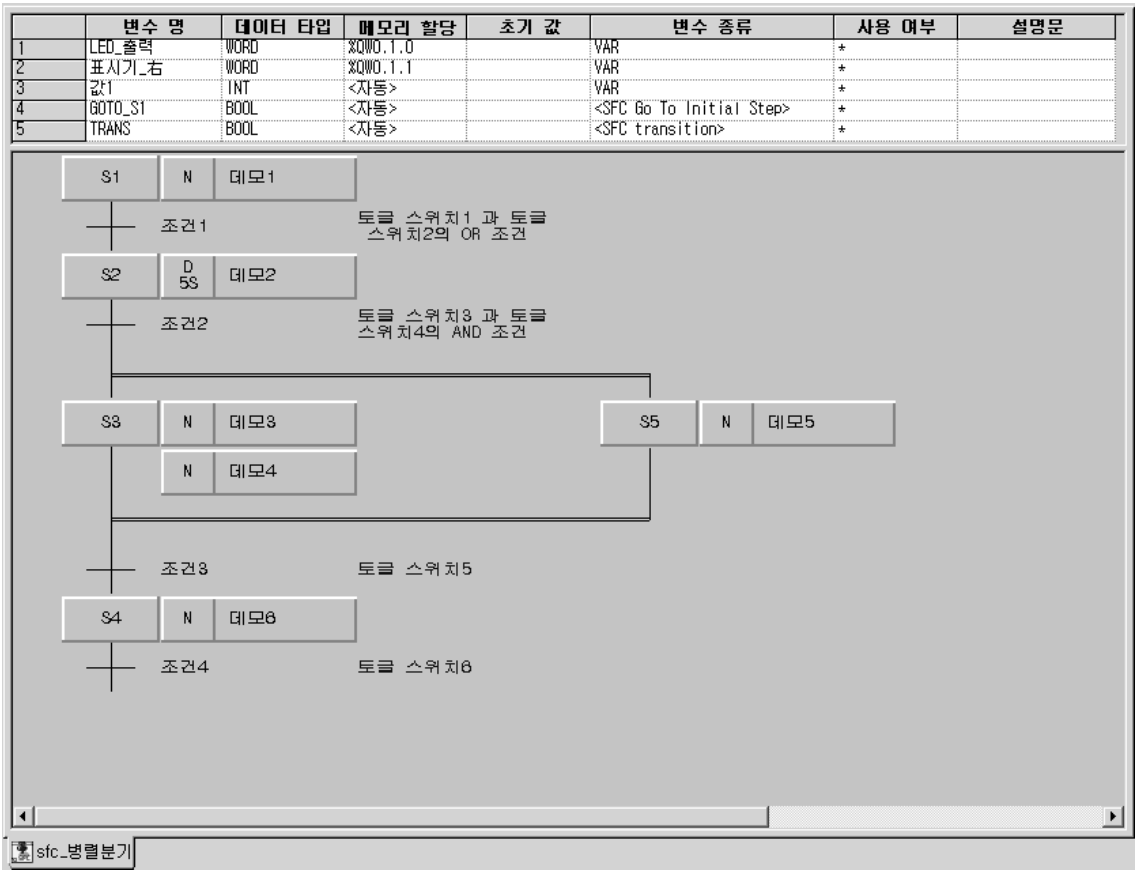
☞ 분기된 T2 나 T5 중 먼저 ON 된 해당 스텝이 활성화 됩니다.

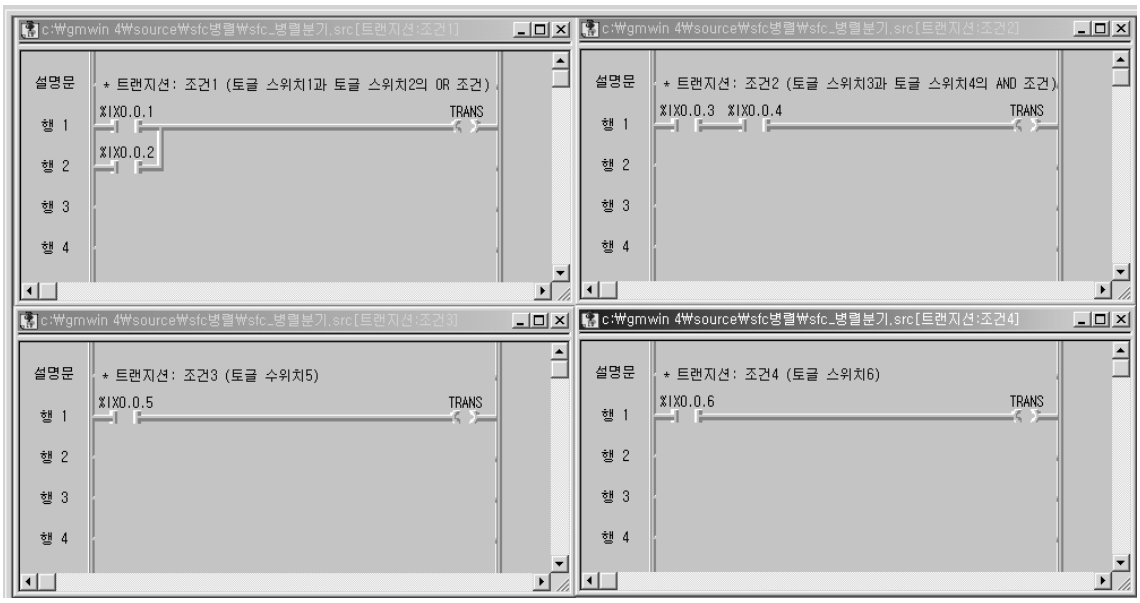


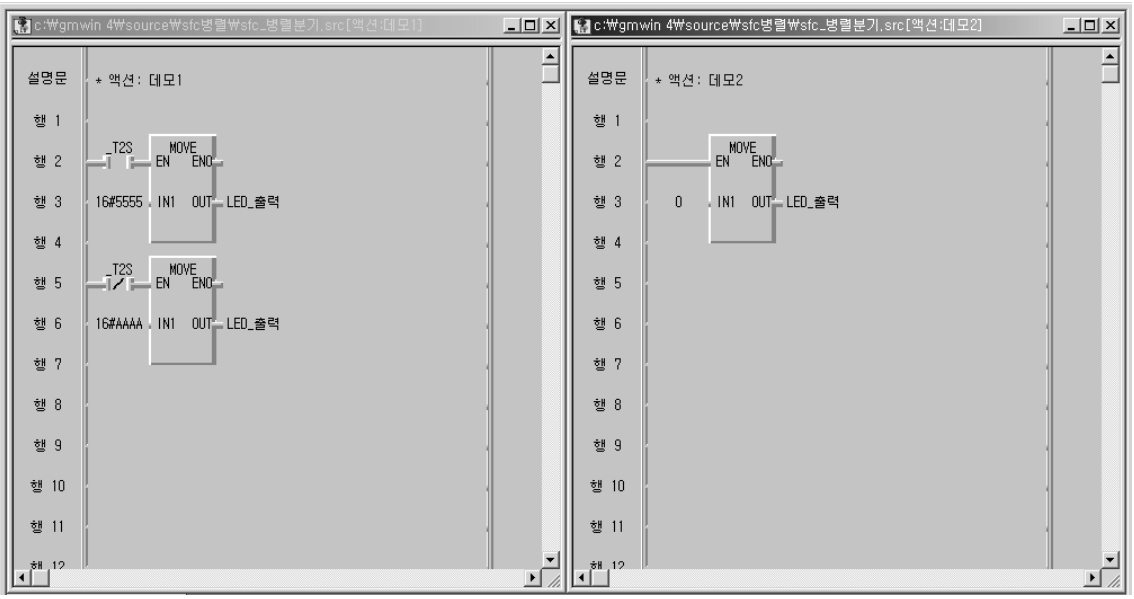
5.4.3 병렬 분기 프로그램 예

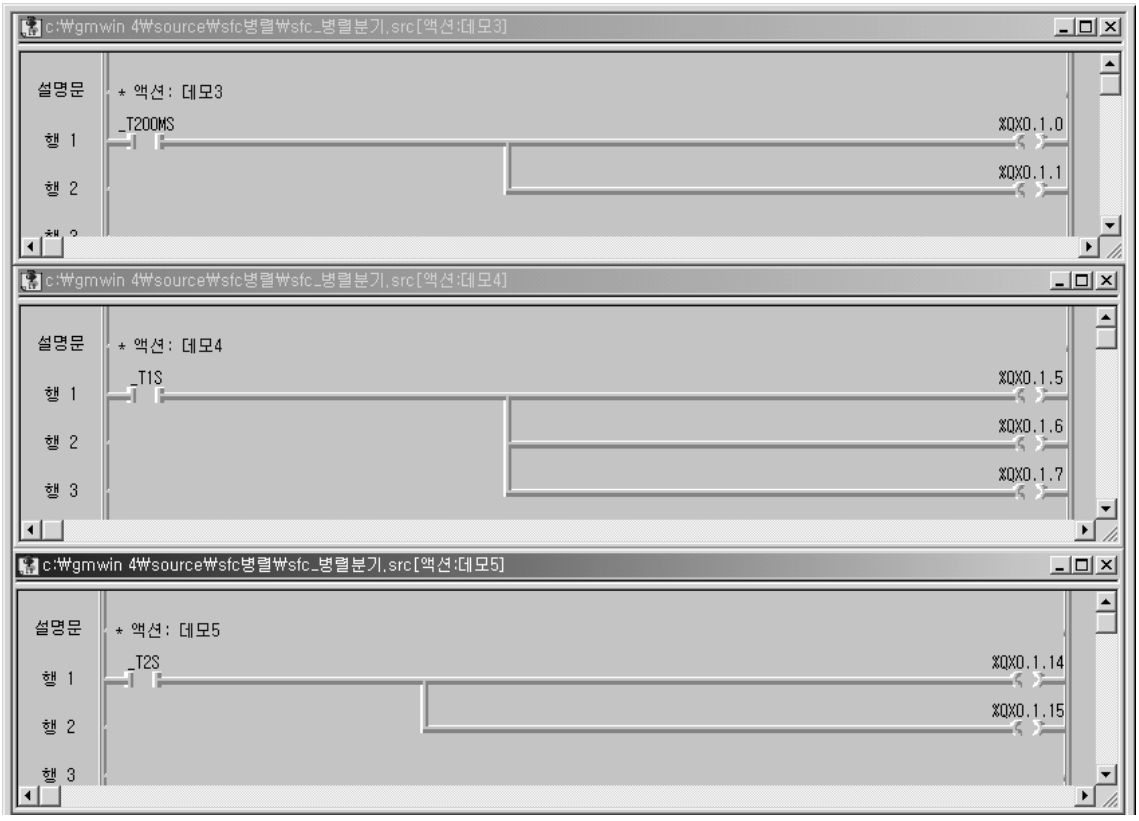
(1) 프로그램 편집

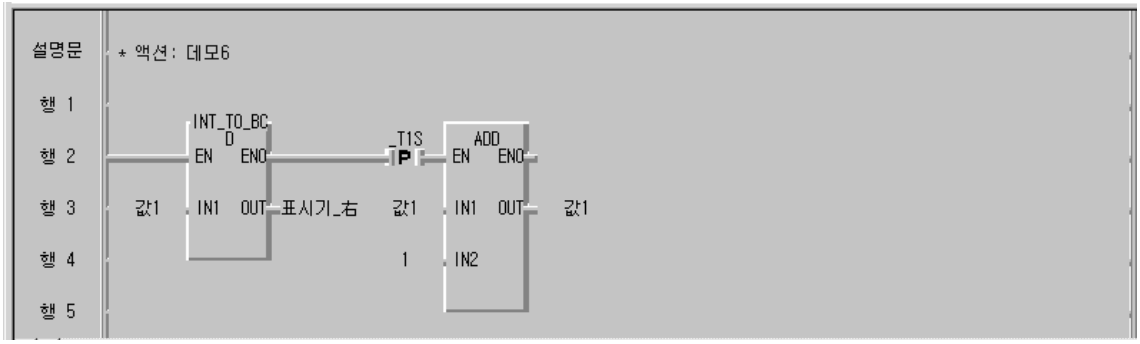
	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	LED_출력	WORD	%QW0.1.0		VAR	*	
2	표시기_右	WORD	%QW0.1.1		VAR	*	
3	값1	INT	<자동>		VAR	*	
4	GOTO_S1	BOOL	<자동>		<SFC Go To Initial Step>	*	
5	TRANS	BOOL	<자동>		<SFC transition>	*	





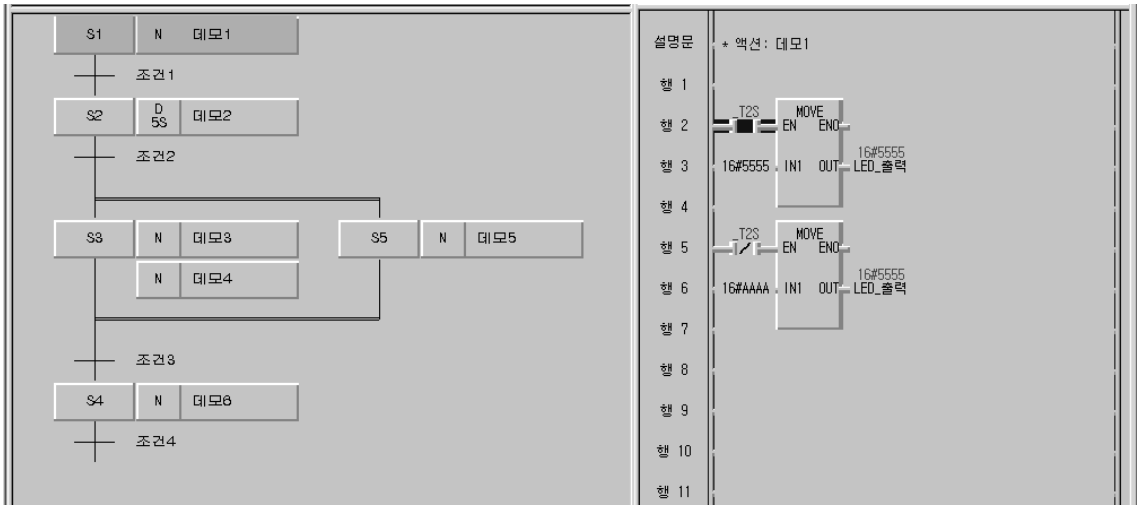

c:₩gmwin 4₩source₩sfc병렬₩sfc_병렬분기.src[액션:데모1]
설명문
* 액션: 데모1
행 1
행 2
_T2S
MOVE
EN ENO
행 3
16#5555
IN1 OUT
LED_출력
행 4
행 5
_T2S
MOVE
EN ENO
행 6
16#AAAA
IN1 OUT
LED_출력
행 7
행 8
행 9
행 10
행 11
c:₩gmwin 4₩source₩sfc병렬₩sfc_병렬분기.src[액션:데모2]
설명문
* 액션: 데모2
행 1
행 2
MOVE
EN ENO
행 3
0
IN1 OUT
LED_출력
행 4
행 5
행 6
행 7
행 8
행 9
행 10
행 11


c:₩gmwin 4₩source₩sfc병렬₩sfc_병렬분기.src[액션:데모3]
설명문
* 액션: 데모3
행 1
_T200MS
%QX0.1.0
행 2
%QX0.1.1
c:₩gmwin 4₩source₩sfc병렬₩sfc_병렬분기.src[액션:데모4]
설명문
* 액션: 데모4
행 1
_T1S
%QX0.1.5
행 2
%QX0.1.6
행 3
%QX0.1.7
c:₩gmwin 4₩source₩sfc병렬₩sfc_병렬분기.src[액션:데모5]
설명문
* 액션: 데모5
행 1
_T2S
%QX0.1.14
행 2
%QX0.1.15
행 3

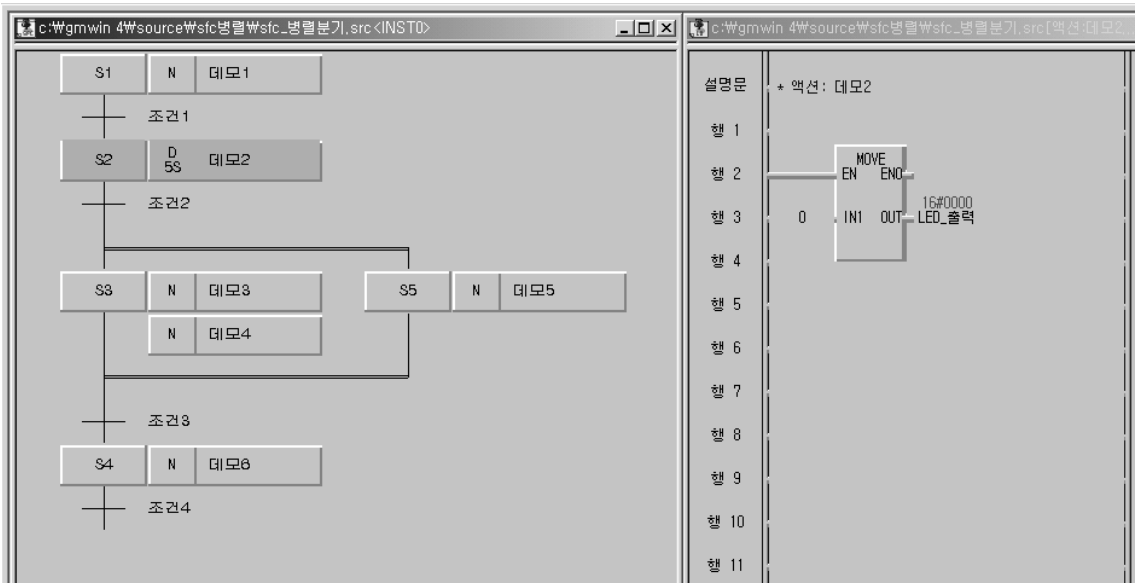


(2) 프로그램 모니터링

☞ 초기에 스텝1이 활성화되어 액션: 데모1 프로그램이 실행 됩니다.

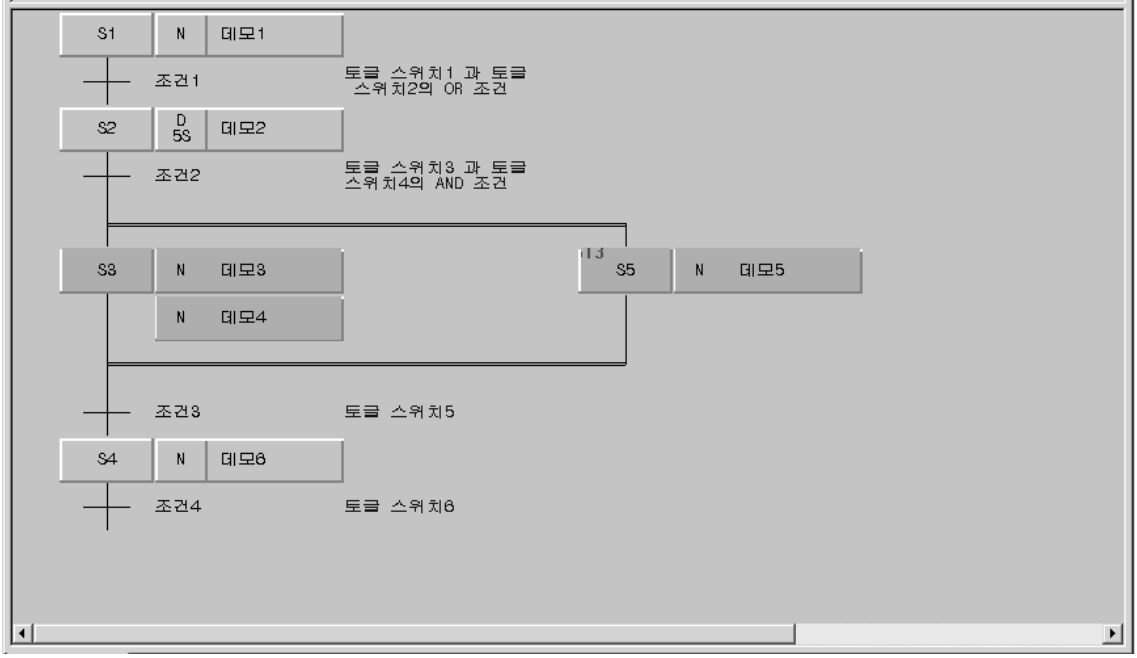


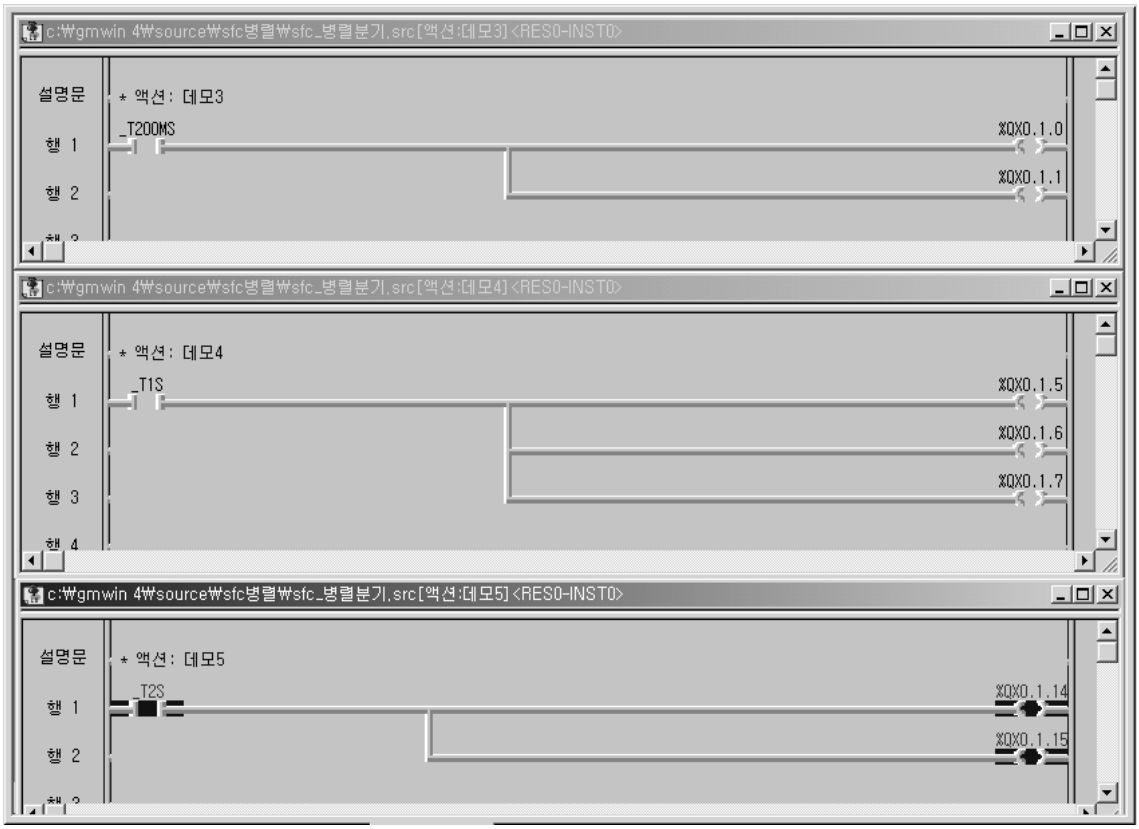
☞ 액션 제한자 D(Time Delyed)에 의해 S2 가 활성화된 후 5 초 후부터 액션 데모 2 가 실행됩니다.



☞ S3 가 활성화되면 액션 데모 3, 액션 데모 4 및 액션: 데모 5 가 동시에 실행됩니다.

	변수 명	변수 값	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	LED_출력	16#0000	WORD	%QW0.1.0		VAR	*	
2	표시기_右	16#0000	WORD	%QW0.1.1		VAR	*	
3	값1	0	INT	<자동>		VAR	*	
4	GOTO_S1	0	BOOL	<자동>		<SFC Go To Initial Step>	*	
5	TRANS	0	BOOL	<자동>		<SFC transition>	*	

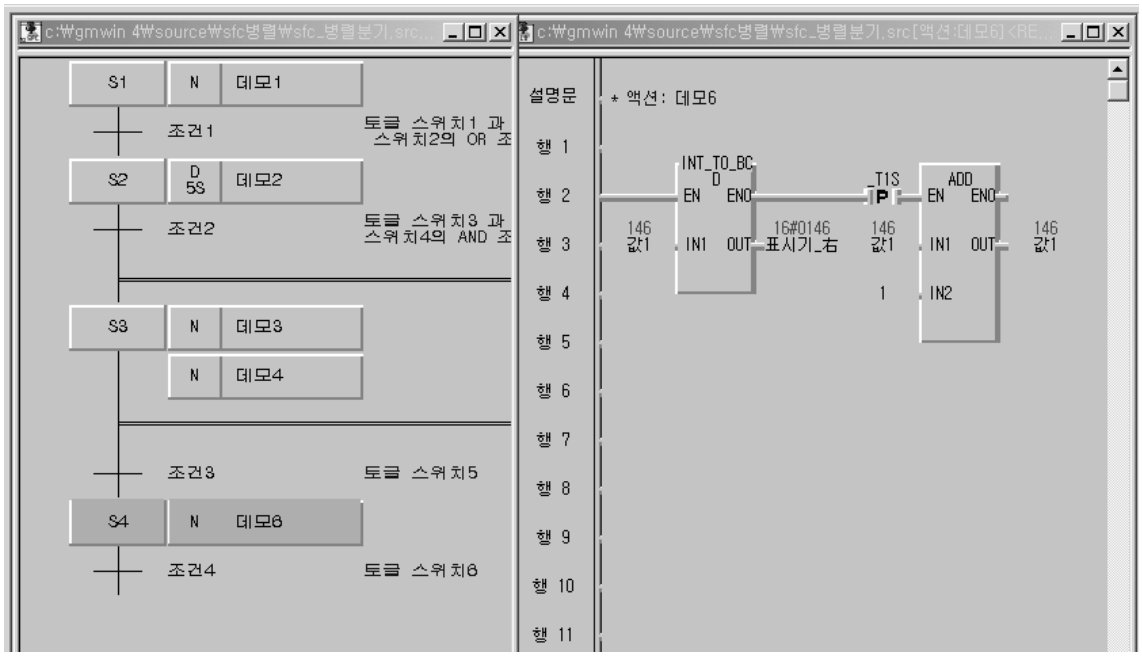




☞ S4 가 활성화되면 액션 “데모 6”이 실행 우측 표시기의 수치를 1 초마다 1 만큼 증가

☞ S4 가 비 활성화되면 액션 “데모 6”은 멈추며 S1 이 다시 활성화되어 액션 “데모 1”이 재실행.

☞ S4 가 재 활성화되면 액션: 데모 6 이 재 실행되고 우측 표시기의 수치는 이전 값부터 다시 증가.



모듈Ⅱ. 아날로그 입력(AD)

- 적 용 모 델 -

제1장 성능규격 및 변환특성

본 제품은 GLOFA PLC GMR/1/2/3/4/6시리즈및 MASTER-K200S/300S/1000S시리즈의 CPU와 조합하여 사용하는 아날로그/디지털 변환 모듈.(이하 AD 변환 모듈)입니다

A/D 변환 모듈은 PLC외부기기로부터의 아날로그 신호(전압 또는 전류입력)를 부호가 있는 14또는12비트 바이너리 데이터의 디지털 값으로 변환하는 모듈입니다.

1.1 성능 규격

항 목		규 격		
		G3F-AD4A	G3F-AD4B	G3F-AD3A
적용 PLC		GM3/MK1000S		
입출력 점유점수		16점		
아날로그 입 력	전 압	DC -5～5V (입력 저항 560kΩ) DC-10～10V(입력 저항 560kΩ)	DC 1～5V (입력 저항 600kΩ)	DC 1～5V(입력 저항 600kΩ) DC 0～10V (입력 저항 600kΩ)
	전 류	DC-20～20㎃(입력저항 250Ω)	DC 4～20㎃ (입력 저항 250Ω)	DC 4～20㎃ (입력 저항 250Ω)
	전압/전류 선택	▸ 제품 옆면의 입력 전환 선택 스위치로 채널마다 선택 (On :전류, Off :전압)하고 ▸ 전류 입력 사용시 제품 전면의 입력범위 선택 스위치를 V1/I 쪽으로 선택합니다.	▸ 제품 옆면의 입력 전환 선택 스위치로 채널마다 선택 (On :전류, Off :전압) 하고 ▸ 프로그램 초기화 펑션블록 (입력변수: IN_SEL)으로지정 (0:전류, 1:전압)	▸ 제품 옆면의 입력 전환 선택 스위치로 채널마다 선택 (On : 전류, Off : 전압) ▸ 전압 종류 선택은 프로그램으로 설정함
디지털 출력		▸ 16비트 바이너리 값(데이터 : 14비트) ▸ 출력 데이터[DATATYPE] 지정에 따라 채널마다 설정 가능 "0"설정 : -192 ～ 16191 , "1"설정 : -8192 ～ 8191(G3F-AD4A) "0"설정 : 0 ～ 16000 , "1"설정 : -8000 ～ 8000(G3F-AD4B)		▸ 16비트 바이너리 값 (데이터 : 12비트) ▸ 출력 데이터 : 0～4000)
최대 분해능	DC 1～5V		0.25㎷ (1/16000)	1㎷ (1/4000)
	DC -5～5V	0.625㎷ (1/16000)		
	DC -10～10V	1.25㎷ (1/16000)		
	DC 0～10V			2.5㎷ (1/4000)
	DC 4～20㎃		1.0㎂ (1/16000)	4㎂ (1/4000)
	DC -20～20㎃	2.5㎂ (1/16000)		
정 밀 도		±0.5% [풀 스케일 (Full Scale)] (주위 온도가 25℃일때는 ±0.3%)		
최대 변환 속도		3.0㎳/채널	3.0㎳/채널	5.0㎳/채널
절대 최대 입력		전압 : ±12V, 전류 : ±25㎃		
아날로그입력점수		16채널/1모듈	16채널/1모듈	8채널/1모듈
절 연 방 식		입력단자와 PLC전원간 포토 커플러 절연(채널간 비절연)		
접 속 단 자		38점 단자대	38점 단자대	20점 단자대
내부 소비 전류		0.7A	0.54A	0.5A
중 량		630g	560g	310g

주의

A/D 변환 모듈은 공장 출하 시 아날로그입력이 전류로 설정되어 있으며 이에 대해 오프셋/게인 값이 조정되어져 있습니다.

항 목		규 격		
		G4F-AD3A	G4F-AD2A	G6F-AD2A
적용 PLC		GM4/MK300S		GM6/MK200S
입출력 점유점수		16점		
아날로그 입 력	전 압	DC 1～5V(입력 저항 600kΩ) DC 0～10V (입력 저항 600kΩ)	DC -5～5V (입력 저항 560kΩ) DC-10～10V(입력 저항 560kΩ)	DC 1～5V (입력저항1MΩ이상) DC 0 ~ 10V (입력저항1MΩ이상) DC -10～10V (입력저항1MΩ이상)
	전 류	DC 4～20mA(입력 저항 250Ω)	DC-20～20mA(입력저항 250Ω)	DC-20～20mA(입력저항 250Ω)
	전압/전류 선택	► 제품 옆면의 입력 전환 선택 스위치로 채널마다 선택 (On: 전류, Off: 전압) ► 전압 종류 선택은 프로그램으로 설정함	► 입력 단자에 의해 선택 (전류 입력 사용시 단자대 V단자와I단자를 연결합니다.) ► 전압 종류 선택은 제품 옆면의 스위치로 선택	► 입력 단자에 의해 선택 (전류 입력 사용시 단자대 V단자와I단자를연결합니다.) ► 전압 종류 선택은 제품 옆면의 스위치로 선택
디지털 출력		► 16비트 바이너리 값 (데이터 : 12비트) ► 출력 데이터 : 0～4000)	► 16비트 바이너리 값 (데이터 : 14비트) ► 출력 데이터[DATATYPE]지정에 따라 채널마다 설정 가능 "0"설정 : -192 ～ 16191 , "1"설정 :-8192～ 8191	► 16비트 바이너리 값 (데이터 : 12비트) ► 출력 데이터[DATATYPE]지정에 따라 채널마다 설정 가능 "0"설정 : -48 ～ 4047 , "1"설정 :-2048～ 2047
최대 분해능	DC 1～5V	1mV (1/4000)		1mV (1/4000)
	DC -5～5V		0.625mV (1/16000)	
	DC -10～10V		1.25mV (1/16000)	5mV (1/4000)
	DC 0～10V	2.5mV (1/4000)		2.5mV (1/4000)
	DC 4～20mA	4μA (1/4000)		4μA (1/4000)
	DC -20～20mA		2.5μA (1/16000)	
정 밀 도		±0.5% [풀 스케일 (Full Scale)] (주위 온도가 25℃일때는 ±0.3%)		
최대 변환 속도		5.0ms/채널	3.0ms/채널	5.0ms/채널
절대 최대 입력		전압 : ±15V, 전류 : ±25mA		
아날로그입력점수		8채널/1모듈	16채널/1모듈	4채널/1모듈
절 연 방 식		입력단자와 PLC전원간 포토 커플러 절연(채널간 비절연)		
접 속 단 자		20점 단자대	38점 단자대	18점 단자대
내부 소비 전류		0.5A	0.54A	DC5V:40mA ,DC15V:50mA ,DC-15V:20mA
중 량		280g	560g	200g

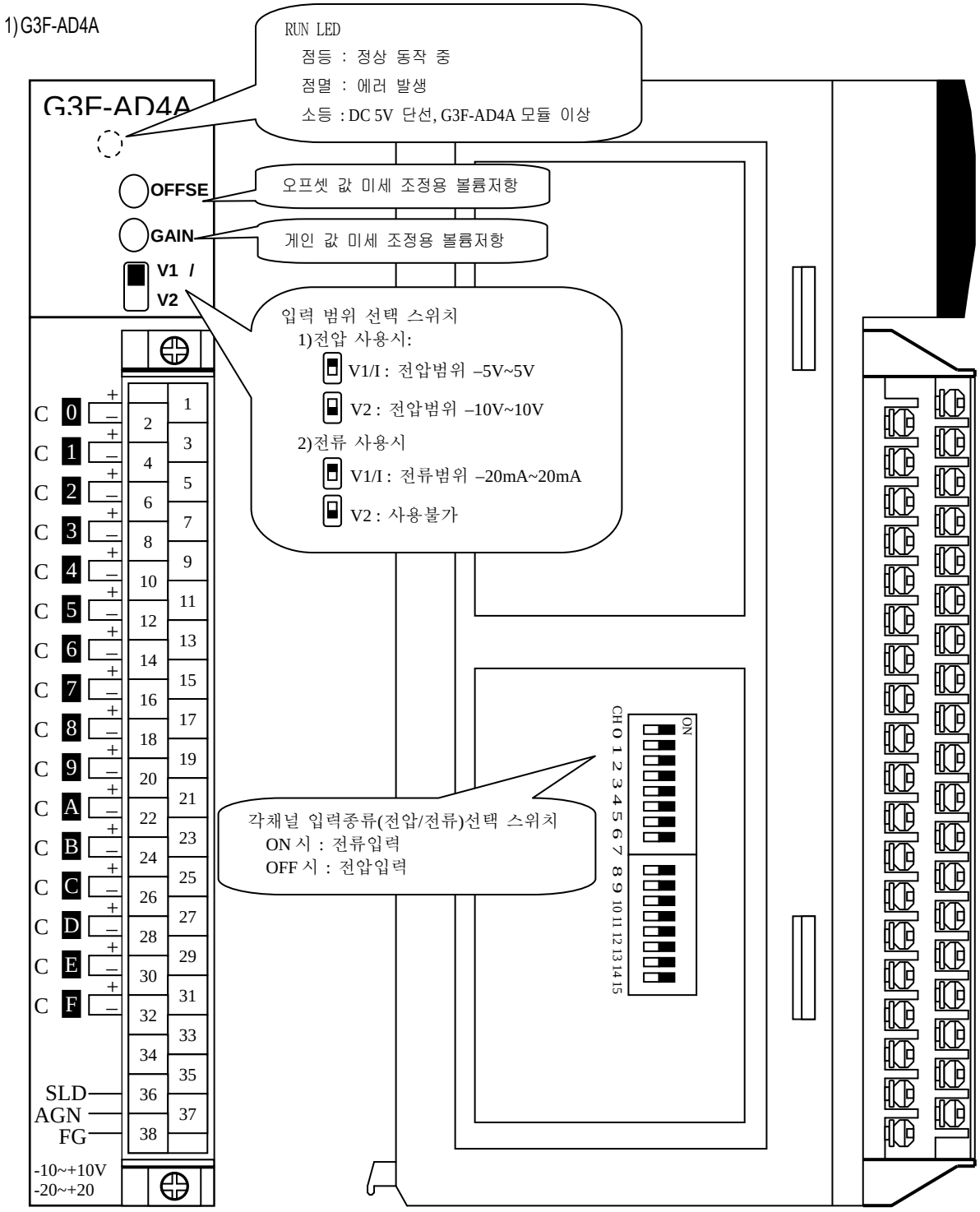
주의

* A/D 변환 모듈은 공장 출하 시 아날로그입력이 전류로 설정되어 있으며 이에 대해 오프셋/게인 값이 조정되어져 있습니다.
* G6F-AD2A를 사용하는 경우 전원 모듈은 반드시 GM6-PAFB를 사용해야 합니다.

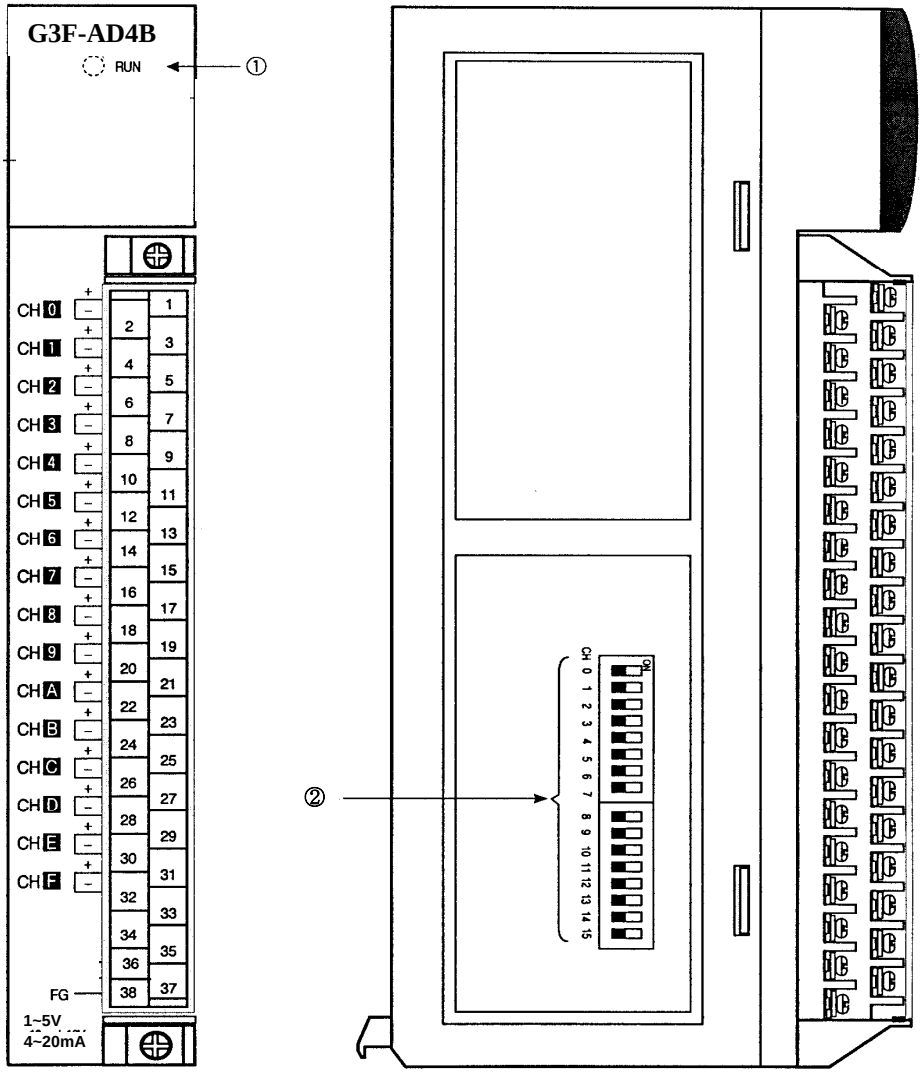
1.2 각 부의 명칭과 역할

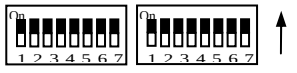
각 부분의 명칭에 대해서 설명합니다.

1) G3F-AD4A

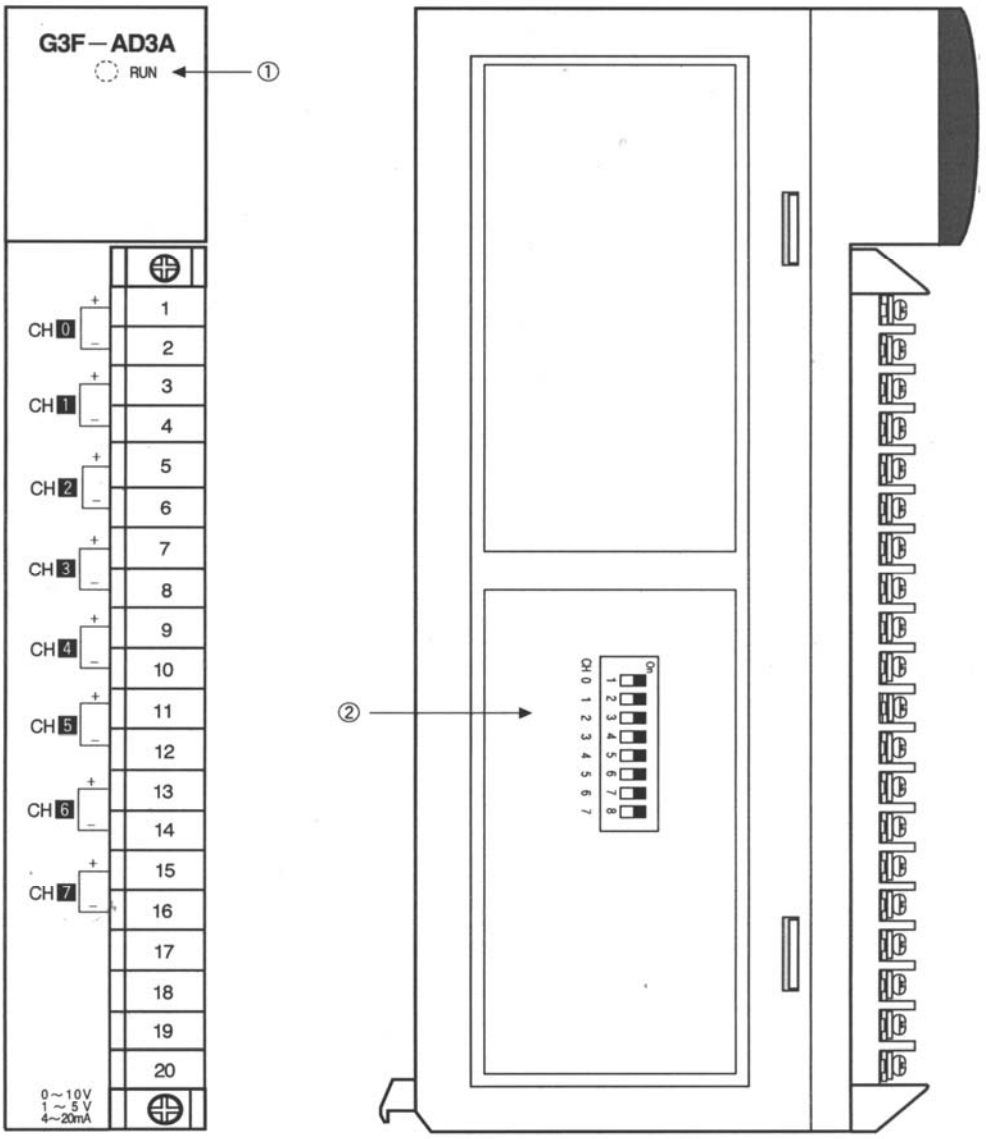


2) G3F-AD4B



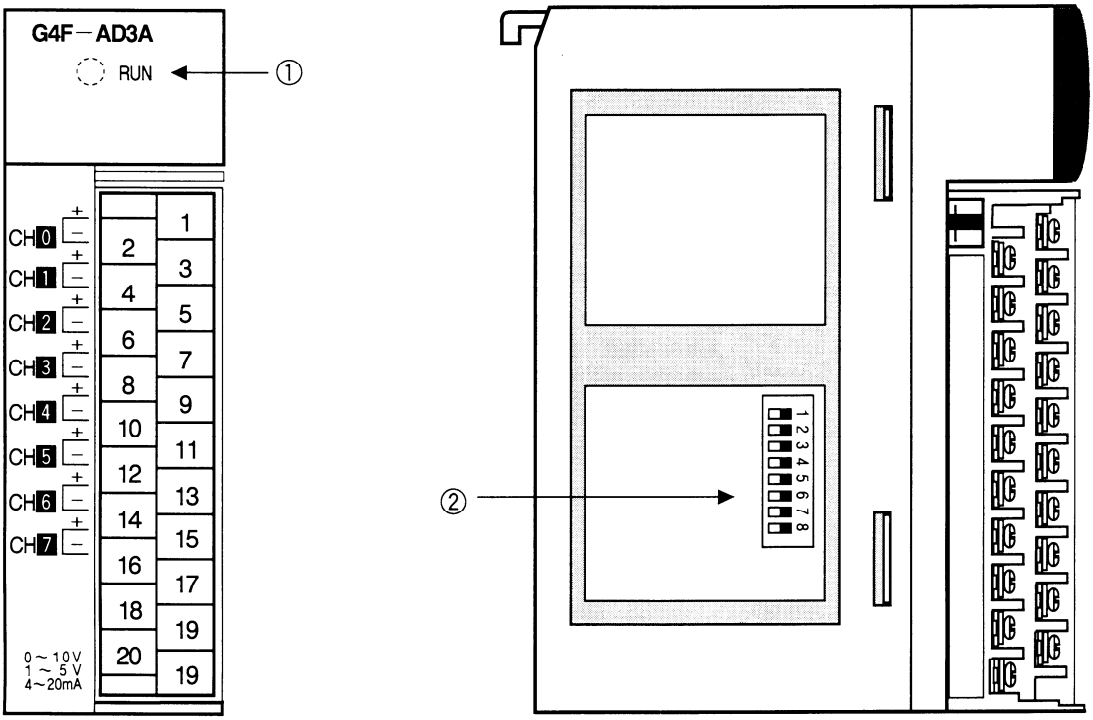
	내용
①	RUN LED G3F-AD4B의 동작 상태를 표시
②	전압/전류 선택 스위치 1)전압 선택 시 스위치의 위치  스위치는 Off에 위치함. 2)전류 선택 시 스위치의 위치  스위치는 On에 위치함

G3F-AD3A



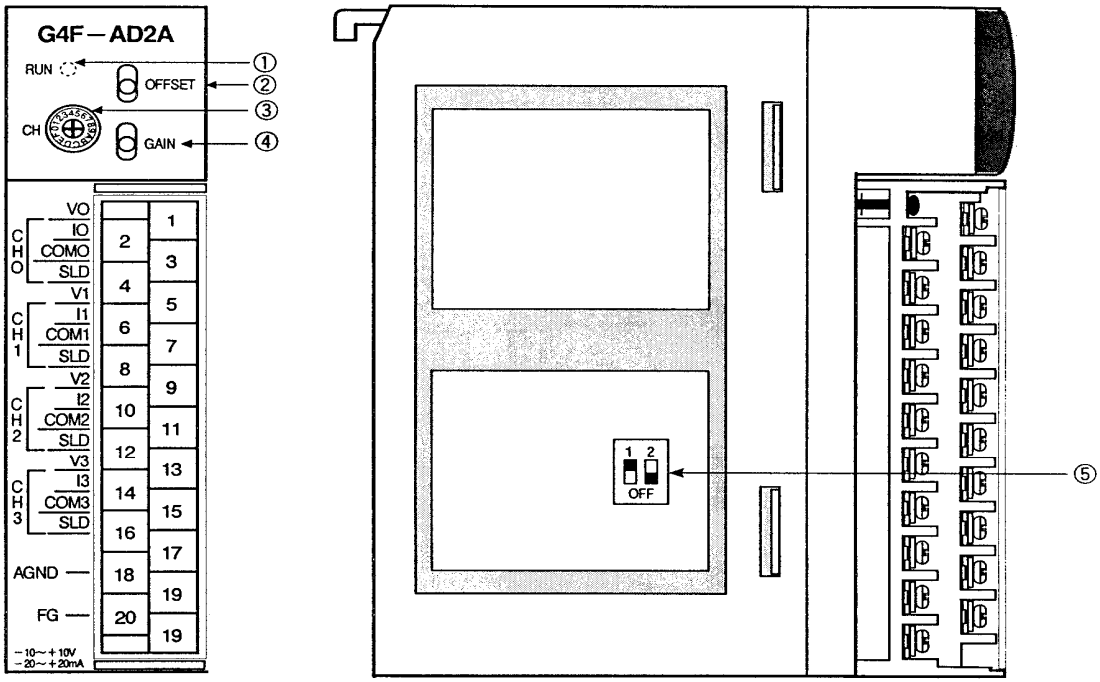
No	내용
①	RUN LED G3F-AD3A의 동작 상태를 표시
②	전압/전류 선택 스위치 1)전압 선택 시 스위치의 위치  스위치는 Off에 위치함. 2)전류 선택 시 스위치의 위치  스위치는 On에 위치함

4) G4F-AD3A



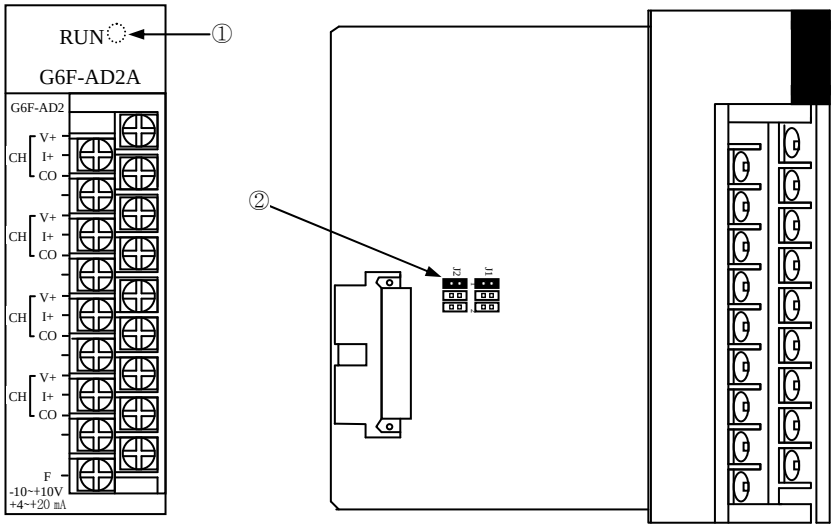
No	내용
①	RUN LED G4F-AD3A의 동작 상태를 표시
②	전압/전류 선택 스위치 1)전압 선택 시 스위치의 위치 스위치는 Off에 위치함. 2)전류 선택 시 스위치의 위치 스위치는 On에 위치함

5) G4F-AD2A

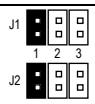


번호	내 용	번호	내 용
①	RUN LED ▶G4F-AD2A의 동작 상태를 표시 ▶노멀(Normal)모드:채널 선택 스위치가 "4~F"의 위치일 때 -점등: 정상 동작 중 -점멸: 에러 발생 -소등: DC 5V단선, G4F-AD2A 모듈 이상 ▶테스트(Test)모드:채널 선택 스위치가 "0~3"의 위치일 때 -점등: 오프셋/게인 스위치가 세트(스위치를 위쪽으로 조작)일 때 -점멸(1.0초): 오프셋/게인 스위치를 조작하지 않을 때 -점멸(0.2초): 오프셋 및 게인 설정 에러 일 때	④	게인 스위치 ▶스위치를 위쪽 방향으로 조정하면 해당 채널의 아날로그 입력 값을 게인 값으로 기억
②	오프셋 스위치 ▶스위치를 위쪽 방향으로 조정하면 해당 채널의 아날로그 입력 값을 오프셋 값으로 기억	⑤	입력 범위 선택 스위치 ▶공장 출하시는 전류 입력 상태로 설정되어져 있습니다. 아날로그 입력 / 스위치의 위치 전압 DC -10~10V 전압 DC -5~5V 전류 DC -20~20 ㎃ ▶스위치의 위치를 [1 2 OFF] 또는 [1 2 OFF] 상태로 위치하지 말아 주십시오. 오동작의 원인이 됩니다.
③	테스트 모드용 채널 선택 스위치 ▶오프셋 값 및 게인 값 조정시 해당 채널을 선택한다. (채널 스위치가 "0~3"위치일 때만 유효함)		

6) G6F-AD2A



No	내용
①	RUN LED G6F-AD2A의 동작 상태를 표시
②	입력범위 선택 스위치

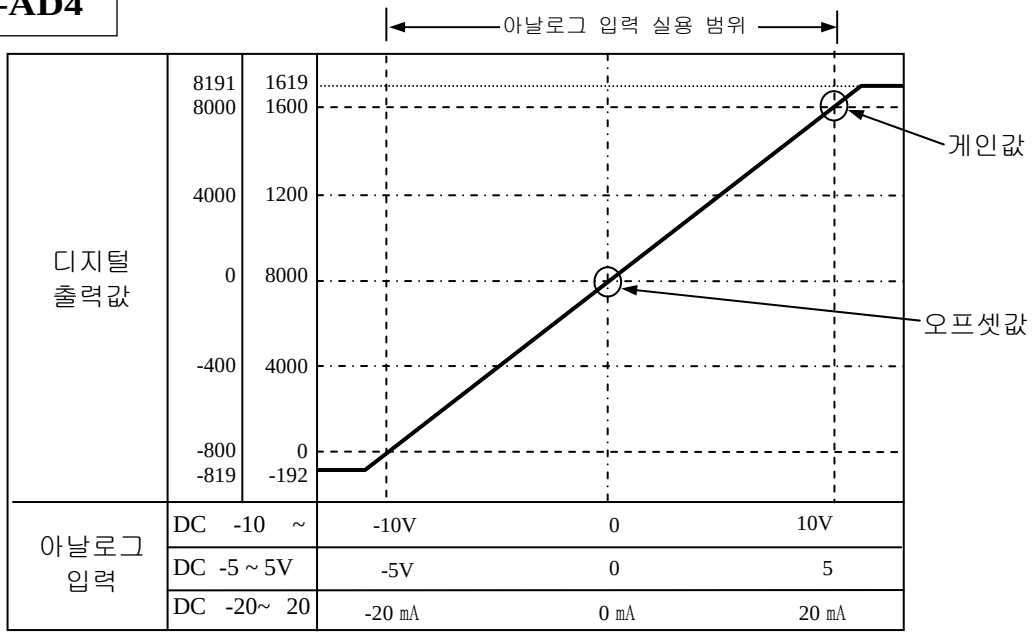
아날로그 입력		스위치의 위치
전 압	DC 1~5V	
	DC 0~10V	
	DC-10~10V	
전 류	DC 4~20mA	

1.3 입출력 변환 특성

▶입출력 변환 특성은 PLC 외부기기로부터의 아날로그 신호(전압 또는 전류 입력)를 디지털 값으로 변환 할 때의 오프셋과 게인 값을 직선으로 연결한 기울기입니다.

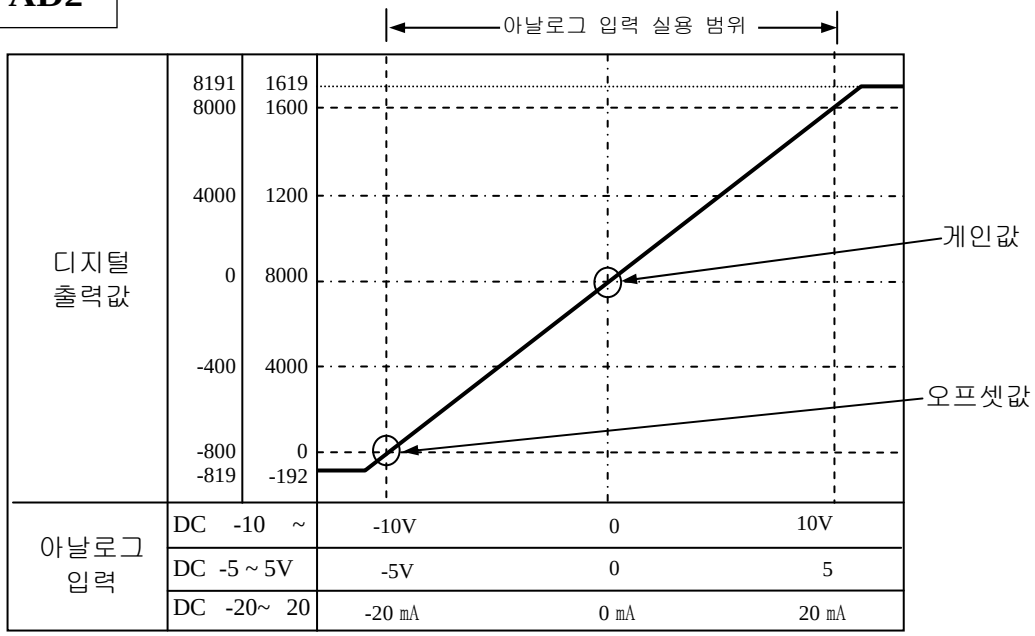
▶A/D 변환 모듈의 입출력 변환 특성에 대하여 설명합니다.

G3F-AD4



디지털 출력값				
아날로그 입력	DC -10 ~	-10V	0	10V
	DC -5 ~ 5V	-5V	0	5
	DC -20~ 20	-20 ㎃	0 ㎃	20 ㎃

G4F-AD2



디지털 출력값				
아날로그 입력	DC -10 ~	-10V	0	10V
	DC -5 ~ 5V	-5V	0	5
	DC -20~ 20	-20 ㎃	0 ㎃	20 ㎃

1.3.1 G3F-AD4A의 입출력 특성

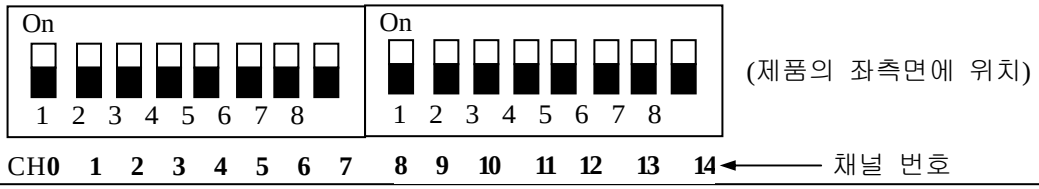
G3F-AD4A는 채널마다 아날로그 입력 전환 선택 스위치로 전압/전류 선택이 가능하고 오프셋/게인 설정은 16채널 일괄로 실행합니다.

1) 전압 입력 특성

전압 입력은 입력 전환 선택 스위치의 위치를 각 채널별로 Off에 위치하여 사용합니다.

입력 전환 선택 스위치의 조작

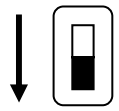
▶전압 선택시 Off 에 위치



가) DC -10~10V 일 때

▶아날로그 입력 선택 스위치를 아래쪽(V2)으로 설정해 주십시오.

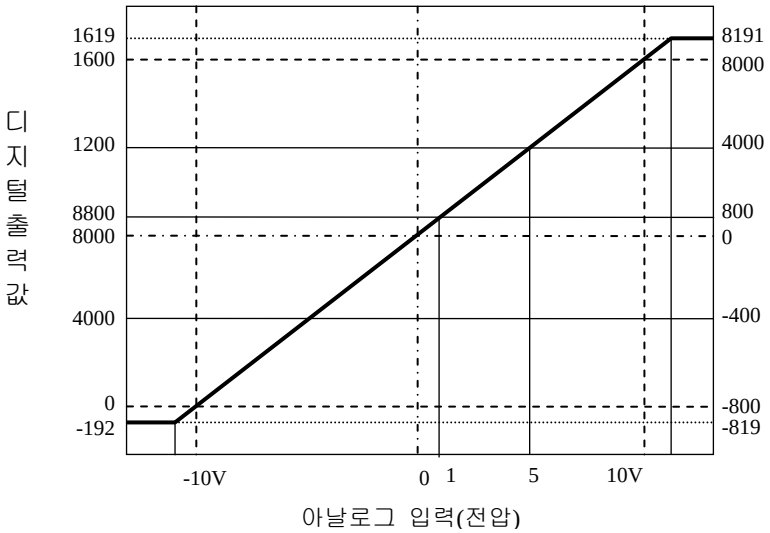
입력 선택 스위치의 조작



V1/I (DC -5~5V 선택)

V2 (DC -10~10V 선택)

<제품의 전면부에 위치>



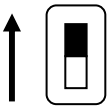
▶전압 입력 특성에 대한 디지털 출력 값은 다음과 같습니다.

디지털 출력 범위	아날로그 입력 전압(V)						
	-10.24	-10	-5	0	5	10	10.24
-192 ~ 16191	-192	0	4000	8000	12000	16000	16191
-8192 ~ 8191	-8192	-8000	-4000	0	4000	8000	8191

나) DC -5~5V 일 때

▶아날로그 입력 선택 스위치를 위쪽(V1/I)으로 설정해 주십시오.

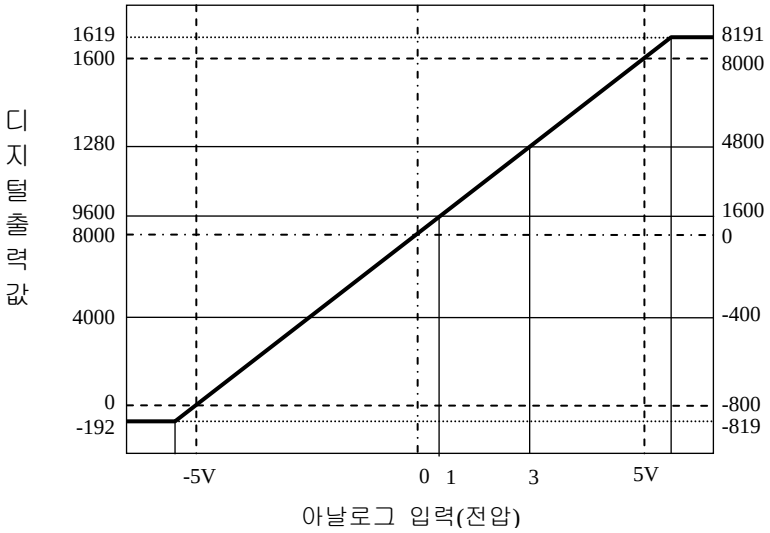
입력 선택 스위치의 조작



V1/I (DC -5~5V 선택)

V2 (DC -10~10V 선택)

<제품의 전면부에 위치>



▶전압 입력 특성에 대한 디지털 출력 값은 다음과 같습니다.

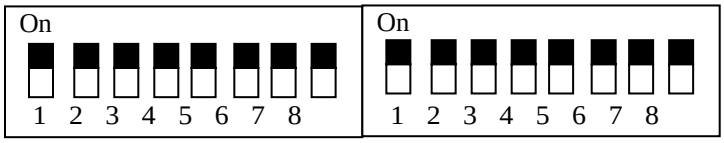
디지털 출력 범위	아날로그 입력 전압(V)						
	-5.12	-5	-2.5	0	2.5	5	5.12
-192 ~ 16191	-192	0	4000	8000	12000	16000	16191
-8192 ~ 8191	-8192	-8000	-4000	0	4000	8000	8191

2) 전류 입력 특성

▶전류 입력은 입력 전환 선택 스위치의 위치를 각 채널별로 On에 설정해 주십시오

입력 전환 선택 스위치의 조작

▶전압 선택시 Off 에 위치

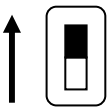


(제품의 좌측면에 위치)

CH**0 1 2 3 4 5 6 7 8 9 10 11 12 13 14** ← 채널 번호

▶아날로그 입력 선택 스위치를 위쪽(V1/I)으로 설정해 주십시오.

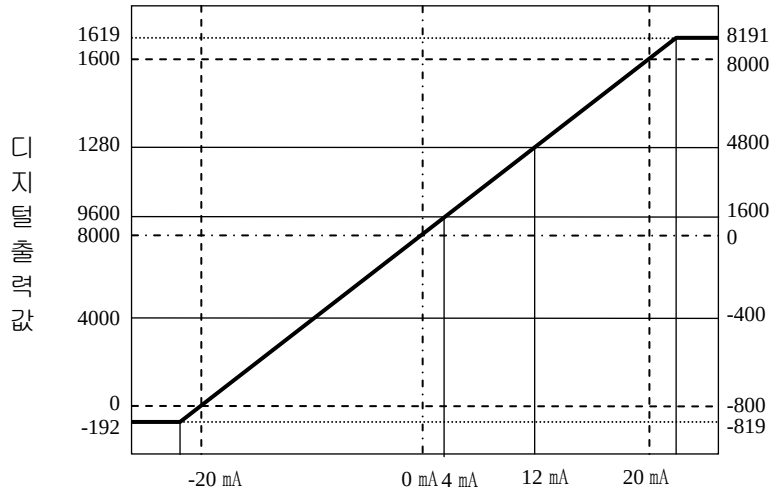
입력 선택 스위치의 조작



V1/I (DC -20~20 ㎃ 선택)

V2 (전류 사용 불가)

<제품의 전면부에 위치>



▶전류 입력 특성에 대한 디지털 출력 값은 다음과 같습니다.

디지털 출력 범위	아날로그 입력 전류(㎃)						
	-20.48	-20	0	4	12	20	20.24
-192 ~ 16191	-192	0	8000	9600	12800	16000	16191
-8192 ~ 8191	-8192	-8000	0	1600	4800	8000	8191

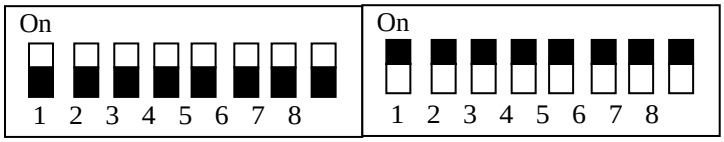
3) 전압, 전류를 동시에 사용할 때의 입출력 특성

▶전압과 전류를 동시에 사용할 때는 입력 전환 선택 스위치를 해당 채널별 전압과 전류로 설정해 주십시오

예) 전압 사용 채널 : 0 ~ 7　　전류 사용 채널 : 8 ~ 15

입력 전환 선택 스위치의 위치

▶전압 선택시 Off 에 위치

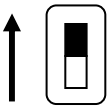


(제품의 좌측면에 위치)

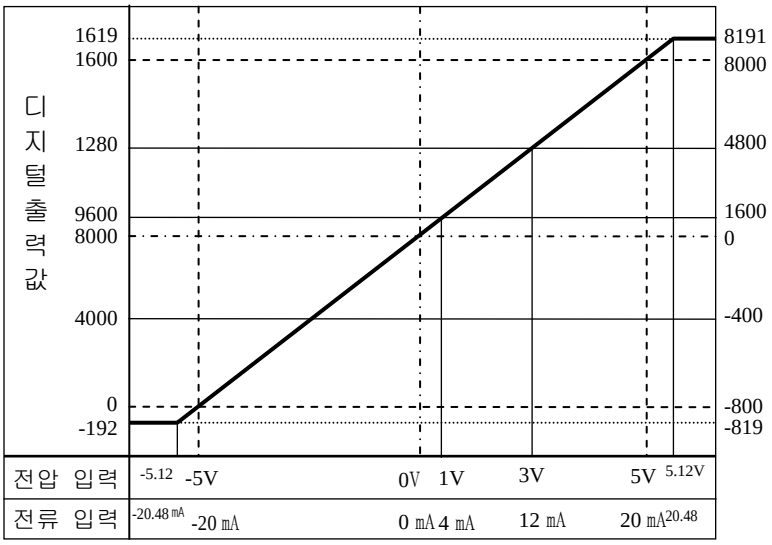
CH0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ◄——— 채널 번호

▶아날로그 입력 선택 스위치를 위쪽(V1/I)으로 설정해 주십시오.

입력 선택 스위치의 조작



V1/I (전압 선택 : DC -5 ~ 5V)
(전류 선택 : DC –20 ~ 20 ㎃)
<제품의 전면부에 위치>



아날로그 입력

▶전압과 전류를 동시에 사용할 때의 입력 특성에 대한 디지털 출력 값은 다음과 같습니다.

디지털 출력 범위	아날로그 입력 전류(㎃)						
	-5.12V	-5V	0V	1V	3V	5V	5.12V
	-20.48㎃	-20㎃	0㎃	4㎃	12㎃	20㎃	20.24㎃
-192 ~ 16191	-192	0	8000	9600	12800	16000	16191
-8192 ~ 8191	-8192	-8000	0	1600	4800	8000	8191

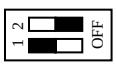
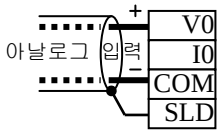
1.3.2 G4F-AD2A의 입출력 특성

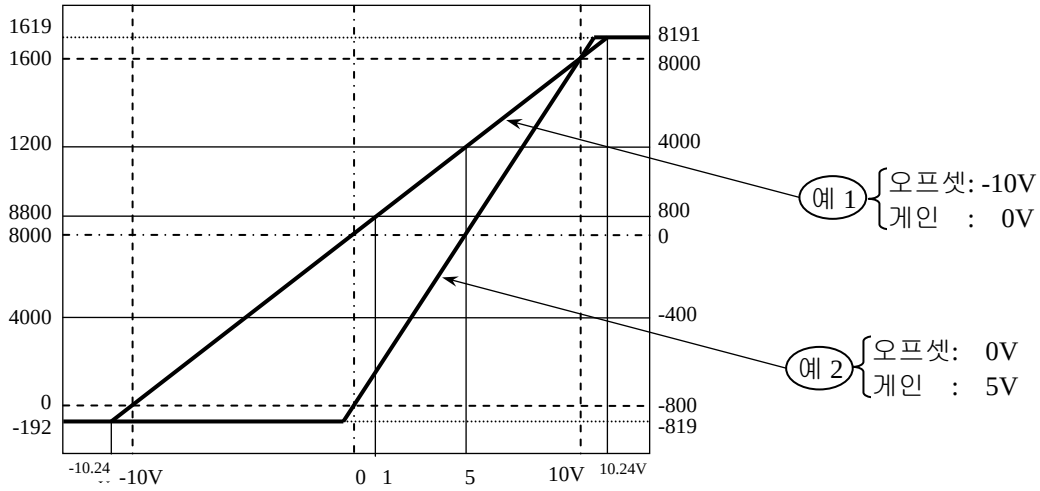
1) 전압 입력 특성

▶G4F-AD2A는 채널마다 전압/전류 선택이 가능하고 오프셋/게인을 조절할 수 있습니다.

▶전압 입력으로 사용할 때는 입력 범위 선택 스위치의 조작에 의해 DC -5~5V 또는 DC -10~10V를 선택할 수 있습니다.

가) DC -10~10V 일 때

입력 범위 선택 스위치	단자대 결선
 1 번 On, 2 번 Off 선택 (제품의 좌측면에 위치)	

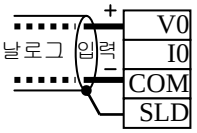


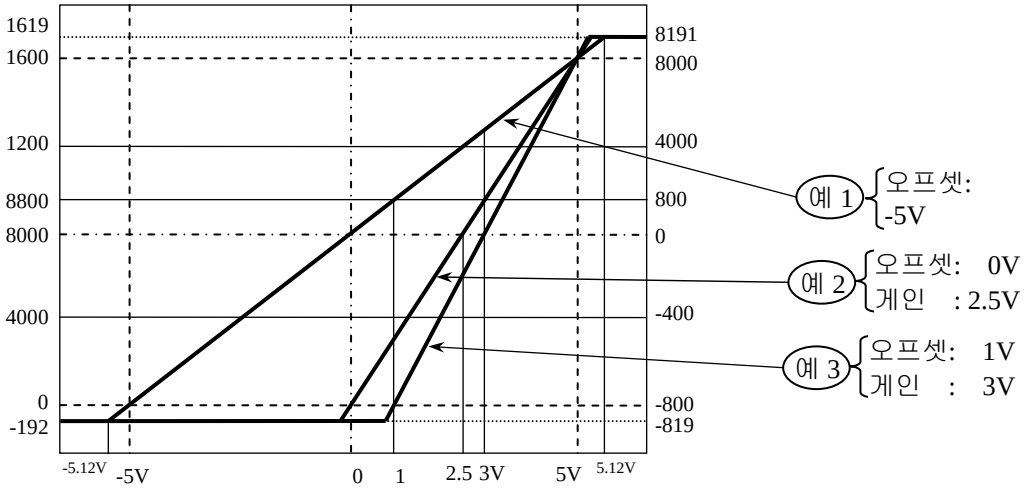
아날로그 입력(전압)

▶오프셋/게인 값 설정을 변경한 경우 전압 입력 특성에 대한 디지털 출력은 다음과 같습니다.

구 분	디지털 출력범위	오프셋값	게인값	아날로그 입력 전류(V)					
				-10	-5	0	3	5	10
예 1	-192 ~ 16191	-10V	0V	0	4000	8000	10400	12000	16000
예 2		0V	5V	-192	-192	0	4800	8000	16000
예 1	-8192 ~ 8191	-10V	0V	-8000	-4000	0	2400	4000	8000
예 2		0V	5V	-8192	-8192	-8000	-3200	0	8000

나) DC -5~5V 일 때

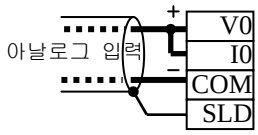
입력 범위 선택 스위치	단자대 결선
 1 번 Off, 2 번 On 선택 (제품의 좌측면에 위치)	

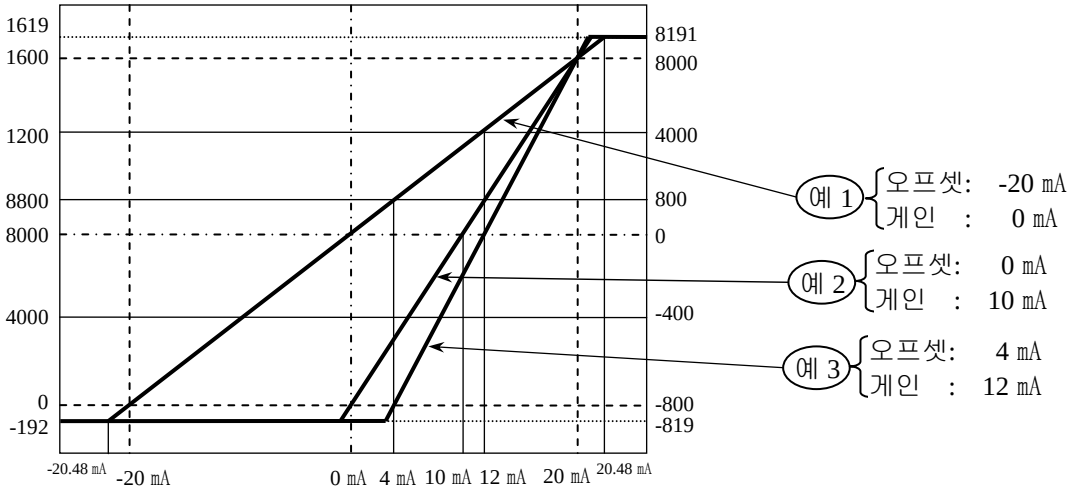


▶오프셋/게인 값 설정을 변경한 경우 전압 입력 특성에 대한 디지털 출력은 다음과 같습니다.

구 분	디지털 출력범위	오프셋값	게인값	아날로그 입력 전류(V)				
				-5	0	1	3	5
(예 1)	-192 ~ 16191	-5V	0V	0	8000	9600	12800	16000
(예 2)		0V	2.5V	-192	0	3200	9600	16000
(예 3)		1V	3V	-192	-192	0	8000	16000
(예 1)	-8192 ~ 8191	-5V	0V	-8000	0	1600	4800	8000
(예 2)		0V	2.5V	-8192	-8000	-4800	1600	8000
(예 3)		1V	3V	-8192	-8192	-8000	0	8000

2) **전류 입력 특성(DC -20 ~ 20mA)**

입력 범위 선택 스위치	단자대 결선
 1 번 Off, 2 번 On 선택 (제품의 좌측면에 위치)	



아날로그 입력(전류)

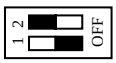
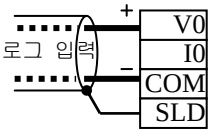
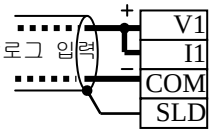
▶오프셋/게인 값 설정을 변경한 경우 전류 입력 특성에 대한 디지털 출력은 다음과 같습니다.

구 분	디지털 출력범위	오프셋값	게인값	아날로그 입력 전류(㎃)					
				-20	0	4	10	12	20
(예 1)	-192 ~ 16191	-20mA	0mA	0	8000	9600	12000	12800	16000
(예 2)		0mA	10mA	-192	0	3200	8000	9600	16000
(예 3)		4mA	12mA	-192	-192	0	6000	8000	16000
(예 1)	-8192 ~ 8191	-20mA	0mA	-8000	0	1600	4000	4800	8000
(예 2)		0mA	10mA	-8192	-8000	-4800	0	1600	8000
(예 3)		4mA	12mA	-8192	-8192	-8000	-2000	0	8000

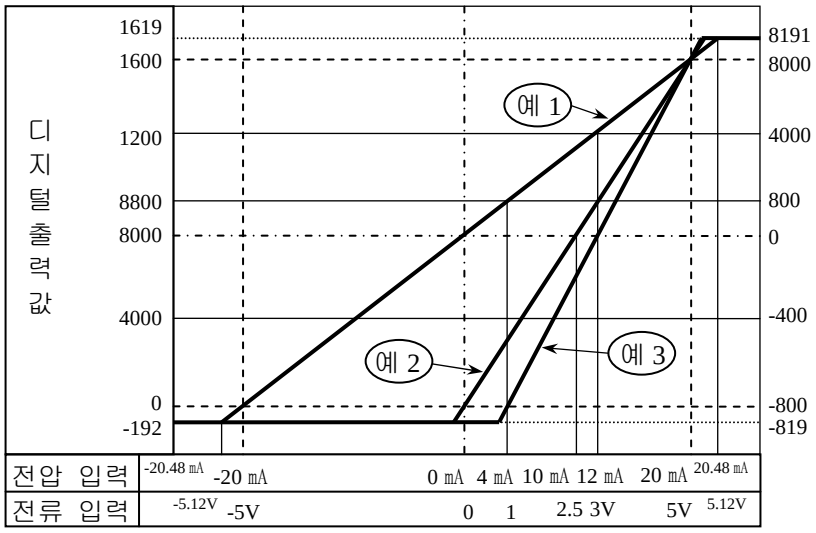
3) **전압, 전류를 동시에 사용할 때의 입출력 특성**

▶전압과 전류를 동시에 사용할 때는 입력 범위 선택 스위치를 DC -5~5V 범위로 선택(1번은 Off, 2번은 On)해 주십시오.

예) 전압 사용 채널 : 0　　전류 사용 채널 : 1

입력 범위 선택 스위치	단자대 결선	
	전압 사용(0 채널)	전류 사용(1 채널)
 1 번 Off, 2 번 On 선택 (제품의 좌측면에 위치)		

▶전압과 전류를 동시에 사용할 때는 전압 입력 범위는 DC -5~5V만 사용할 수 있습니다.



아날로그 입력

▶오프셋/게인 값 설정을 변경한 경우 전압/전류 입력 특성에 대한 디지털 출력은 다음과 같습니다.

구 분	디지털 출력범위	오프셋값		게인값		아날로그 입력 전압/전류					
		전압 (v)	전류 (㎃)	전압 (v)	전류 (㎃)	-5V	0V	1V	2.5V	3V	5V
						-20㎃	0㎃	4㎃	10㎃	12㎃	20㎃
예 1	-192 ~ 16191	-5	-20	0	0	0	8000	9600	12000	12800	16000
예 2		0	0	2.5	10	-192	0	3200	8000	9600	16000
예 3		1	4	3	12	-192	-192	0	6000	8000	16000
예 1	-8192 ~ 8191	-5	-20	0	0	-8000	0	1600	4000	4800	8000
예 2		0	0	2.5	10	-8192	-8000	-4800	0	1600	8000
예 3		1	4	3	12	-8192	-8192	-8000	-2000	0	8000

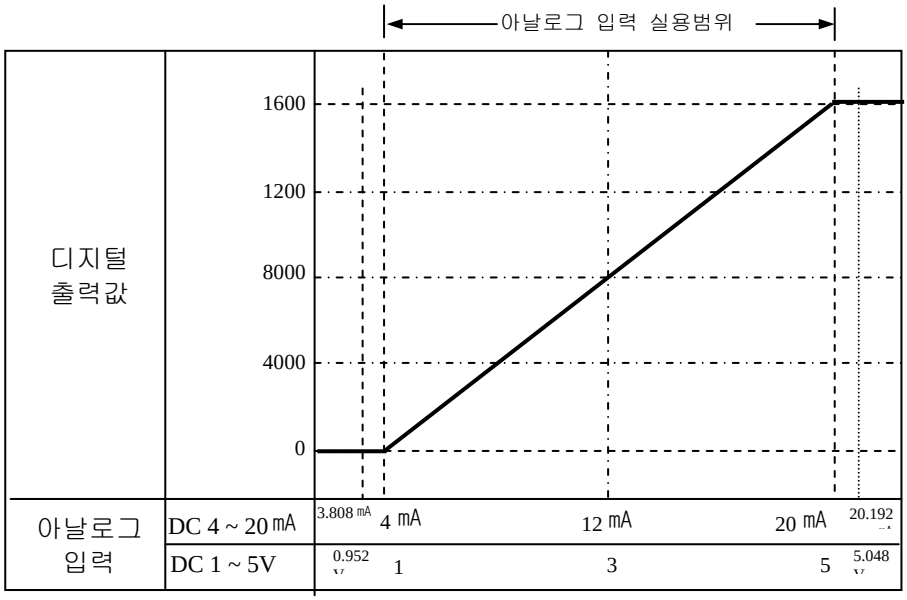
주의

1. 디지털 출력 범위가 -192 ~ 16191로 지정된 경우 디지털 출력 값이 16191또는 -192를 초과하는 아날로그 값이 입력 되어도 디지털 출력 값은 16191 또는 -192로 고정됩니다.
   디지털 출력 범위가 -8192 ~ 8191로 지정된 경우 디지털 출력 값이 8191또는 -8192를 초과하는아날로그 값이 입력 되어도 디지털 출력 값은 8191 또는 -8192로 고정됩니다.
2. 전압은 ±15V, 전류는 ±25㎃ 이상을 입력하지 말아 주십시오.
   열 상승에 의해 불량의 원인이 됩니다.
3. G4F-AD2A에서 오프셋/게인 설정은 필히 게인 값이 오프셋 값보다 크게 설정하여 주십시오.
   그렇지 않으면 정확한 디지털 출력을 얻을 수 없게 됩니다.

1.3.3 G3F-AD4B의 입출력 특성

▶입출력 변환 특성은 PLC 외부기기로부터의 아날로그 신호(전압 또는 전류 입력)를 디지털 값으로 변환 할 때의 기울기로 아래 그림과 같습니다.

▶A/D 변환 모듈은 채널 마다 아날로그 입력 범위 선택 스위치로 전압/전류 선택이 가능하고 오프셋/게인 설정은 고정되어 있으므로 변경할 수 없습니다.



입출력 변환 특성

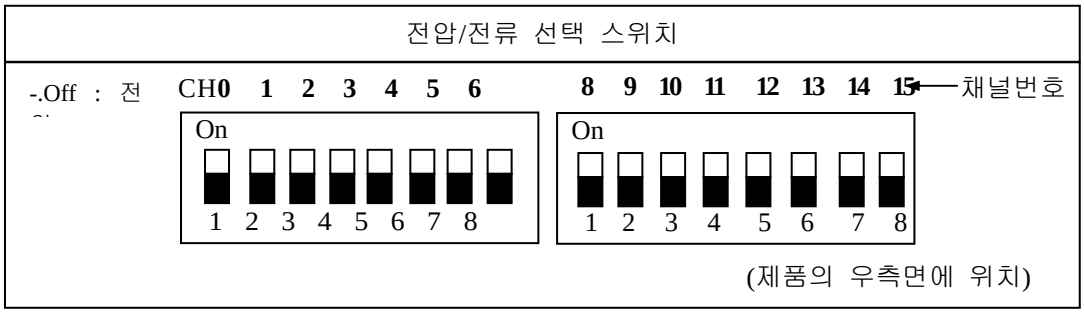
주의

1. 디지털 출력 값이 16000 또는 0을 초과하는 아날로그 값이 입력되어도 디지털 출력 값은 16000 또는 0로 고정됩니다.
2. 전압은 +15V, 전류는 +25㎃ 이상을 입력하지 마십시오.
   열 상승에 의해 불량의 원인이 됩니다.

1) 전압 입력(DC 1~5V) 특성

▶전압선택시 초기화 펑션블록의 입력 변수 IN_SEL에서 채널마다 "1"로 설정하여야 합니다..

▶또한, 입력 범위 선택 스위치의 위치를 각 채널별로 Off에 위치하여 사용합니다.



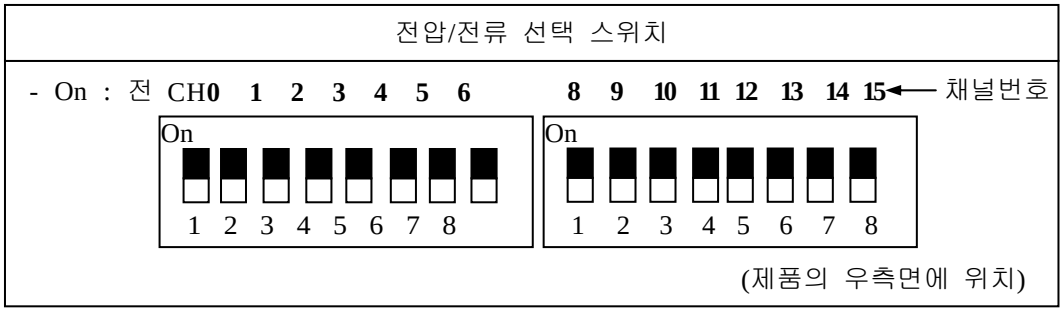
▶전압 입력 특성에 대한 디지털 출력 값은 다음과 같습니다.

구 분	아날로그 입력 전압(V)						
	1이하	1	2	3	4	5	5이상
디지털 출력값	0	0	4000	8000	12000	16000	16000

2) 전류 입력 특성

▶전류선택시 초기화 펑션블록의 입력 변수 IN_SEL에서 채널마다 "0"으로 설정하여야 합니다..

▶전류 입력은 전압/전류 선택 스위치의 위치를 각 채널별로 On에 위치하여 사용합니다.



▶전류 입력 특성에 대한 디지털 출력값은 다음과 같습니다.

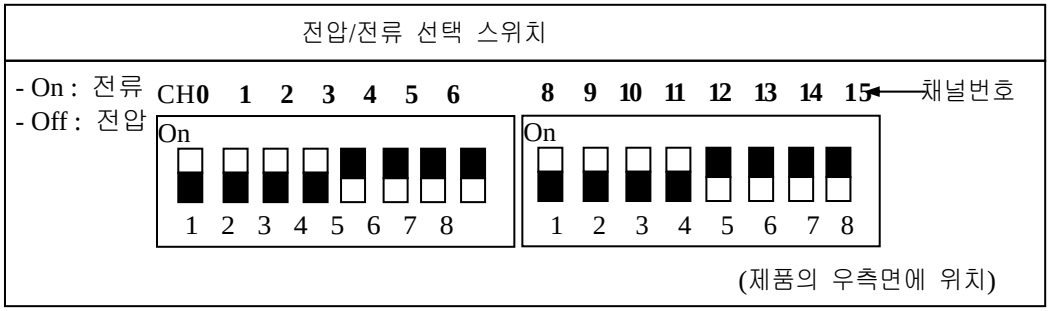
구 분	아날로그 입력 전류(mA)						
	4이하	4	8	12	16	20	20이상
디지털 출력값	0	0	4000	8000	12000	16000	16000

3) 전압, 전류를 동시에 사용할 때의 입력 특성

▶아날로그 입력을 **전압** 입력으로 사용하는 채널은 해당채널마다 초기화 펑션블록의 입력 변수 IN_SEL에서 "1"로 설정하고, 전압/전류 선택 스위치의 위치를 Off로 선택해야하며

▶ 아날로그 입력을 **전류** 입력으로 사용하는 채널은 해당채널마다 초기화 펑션블록의 입력 변수 IN_SEL에서 "0"으로 설정하고, 전압/전류 선택 스위치의 위치를 On으로 선택해야 합니다.

예) 전압 사용 채널 : 채널0 ~ 3, 8 ~ 11, 전류 사용 채널 : 채널4 ~7, 12 ~15

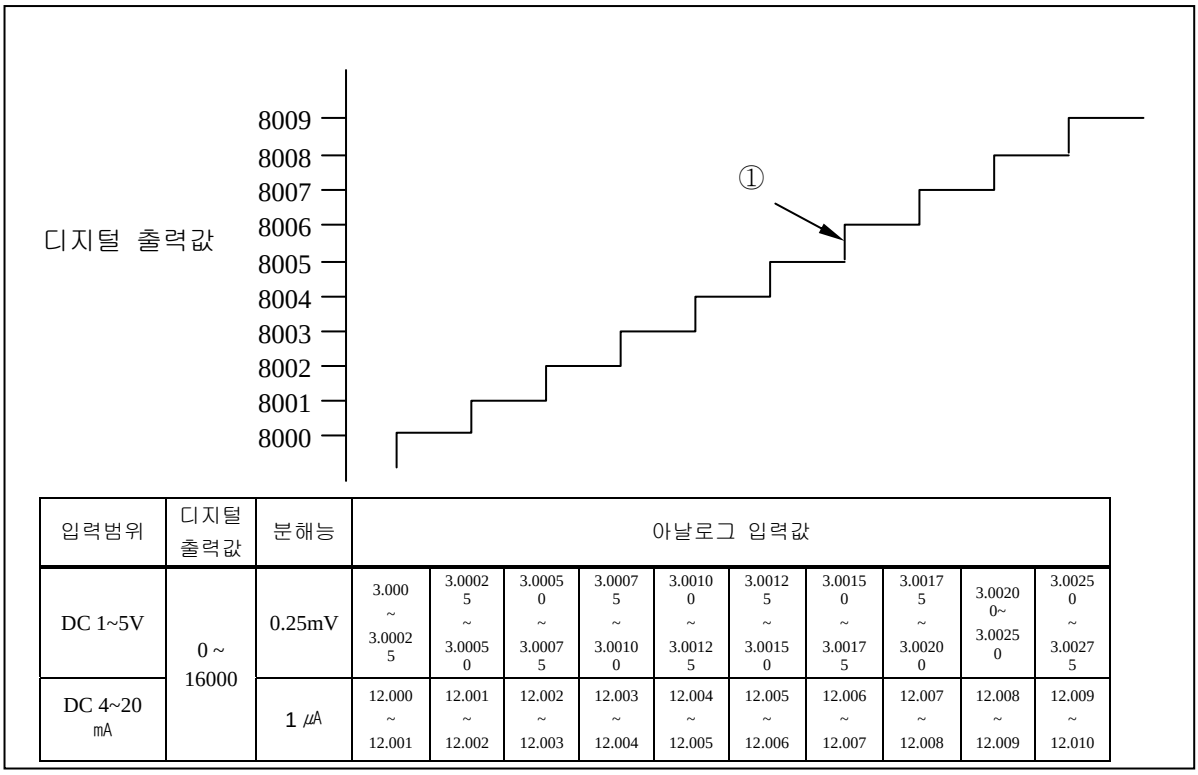


▶전압/전류 입력 특성에 대한 디지털 출력값은 다음과 같습니다.

구 분		아날로그 입력						
입력 종류	1~ 5V	1이하	1	2	3	4	5	5이상
	4~20㎃	4이하	4	8	12	16	20	20이상
디지털 출력값		0	0	4000	8000	12000	16000	16000

4) 아날로그 입력과 디지털 출력의 관계

아날로그 입력과 디지털 출력의 관계는 다음과 같습니다.



입력범위	디지털 출력값	분해능	아날로그 입력값									
DC 1~5V	0 ~ 16000	0.25mV	3.000 ~ 3.0002 5	3.0002 5 ~ 3.0005 0	3.0005 0 ~ 3.0007 5	3.0007 5 ~ 3.0010 0	3.0010 0 ~ 3.0012 5	3.0012 5 ~ 3.0015 0	3.0015 0 ~ 3.0017 5	3.0017 5 ~ 3.0020 0	3.0020 0~ 3.0025 0	3.0025 0 ~ 3.0027 5
DC 4~20 ㎃		1 ㎂	12.000 ~ 12.001	12.001 ~ 12.002	12.002 ~ 12.003	12.003 ~ 12.004	12.004 ~ 12.005	12.005 ~ 12.006	12.006 ~ 12.007	12.007 ~ 12.008	12.008 ~ 12.009	12.009 ~ 12.010

아날로그 입력과 디지털 출력

1.3.4 오프셋/게인 설정과 디지털 출력의 관계 (G4F-AD2A만 해당됨)

1) **분해능**

분해능은 다음식에 의해 결정됩니다.

(1) 전압 입력의 경우

$$\text{분해능} = \frac{\text{게인 값} - \text{오프셋 값}}{8000} \times 1000(\text{㎷})$$

예) 게인 값: 0V 오프셋 값: -10V

$$\text{분해능} = \frac{0-(-10)}{8000} \times 1000(\text{㎷}) = 1.25(\text{㎷})$$

(2) 전류 입력의 경우

$$\text{분해능} = \frac{\text{게인 값} - \text{오프셋 값}}{8000} \times 1000(\mu\text{A})$$

예) 게인 값: 0㎃ 오프셋 값: -20㎃

$$\text{분해능} = \frac{0-(-20)}{8000} \times 1000(\mu\text{A}) = 2.5(\mu\text{A})$$

2) **최대 분해능과 디지털 출력 값과의 관계**

오프셋/게인 설정에 의해 다음의 계산 식과 같이 된 경우에는 디지털 출력 값이 1씩 증감되지 않습니다.

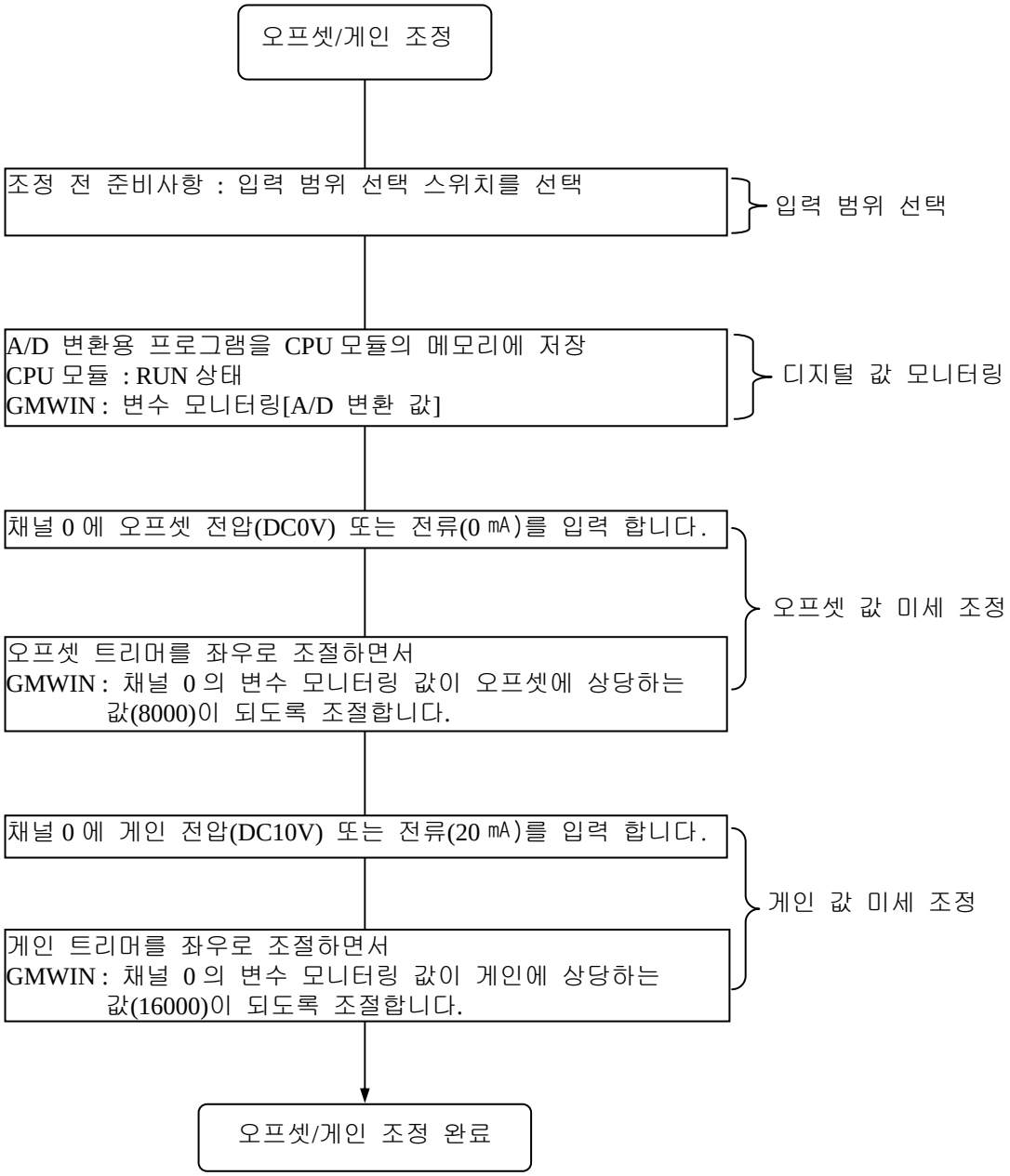
$$\frac{\text{게인 값} - \text{오프셋 값}}{8000} < \text{분해능}$$

3) 오프셋/게인 설정

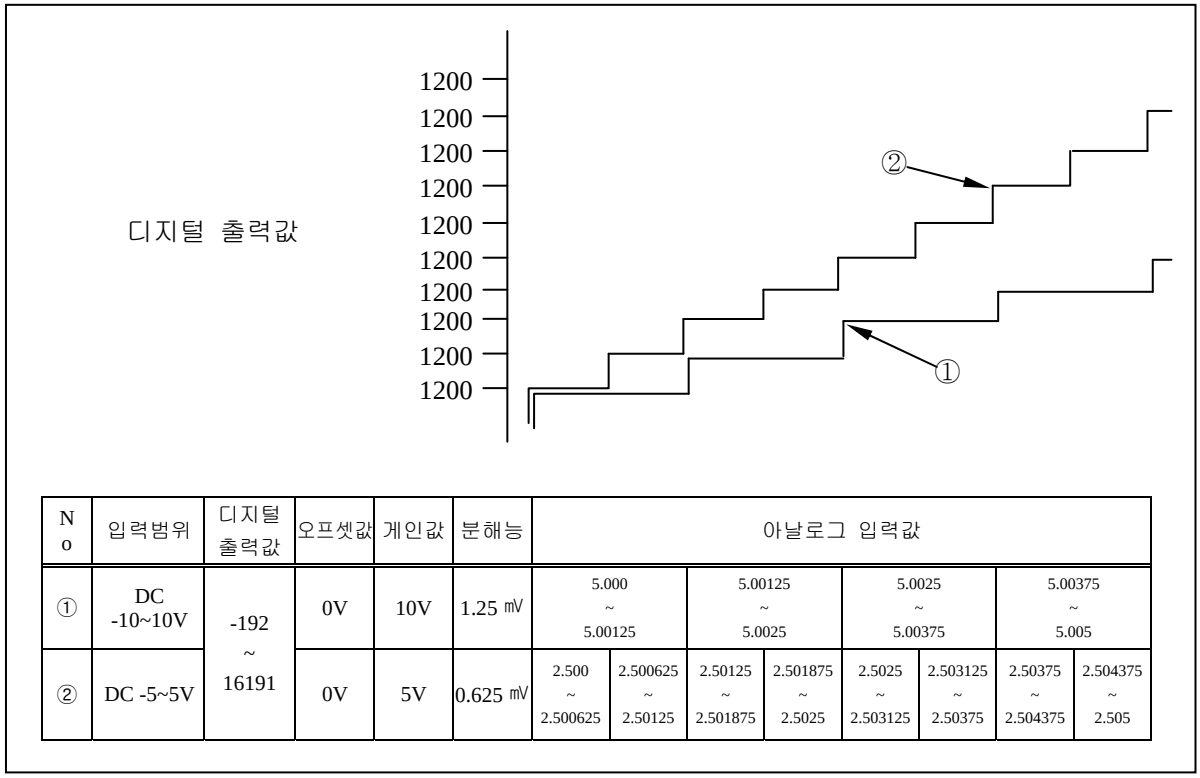
(1) G3F-AD4A의 오프셋/게인 설정

가) 오프셋/게인 설정 순서

오프셋/게인 값은 16채널 일괄로 조정합니다.

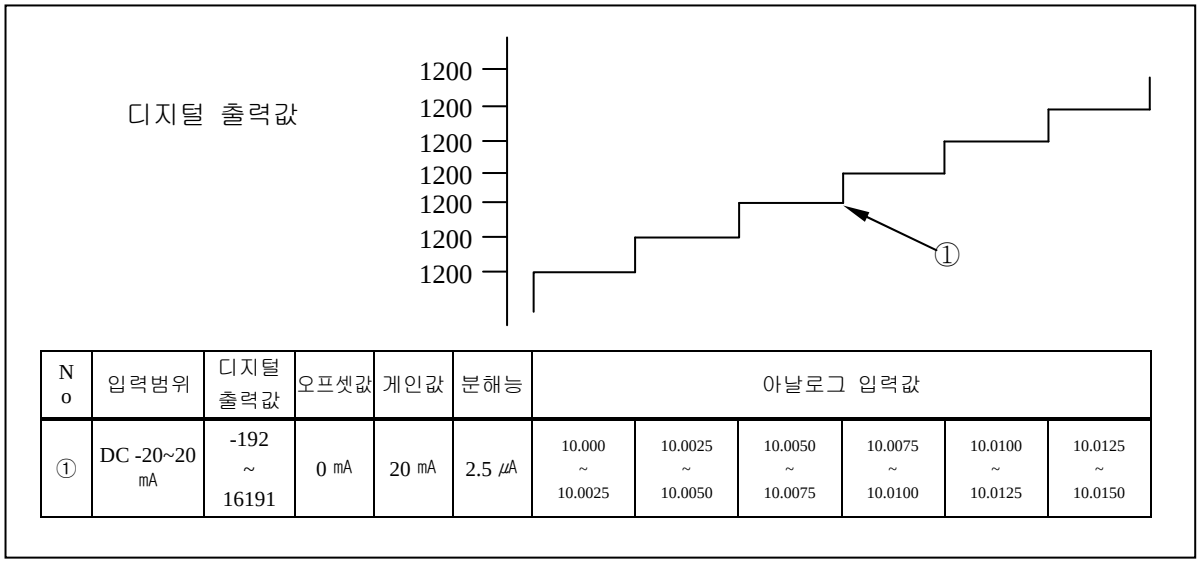


나) 오프셋/게인 설정에 따른 입출력 특성



No	입력범위	디지털 출력값	오프셋값	게인값	분해능	아날로그 입력값							
①	DC -10~10V	-192 ~ 16191	0V	10V	1.25 ㎷	5.000 ~ 5.00125		5.00125 ~ 5.0025		5.0025 ~ 5.00375		5.00375 ~ 5.005	
②	DC -5~5V		0V	5V	0.625 ㎷	2.500 ~ 2.500625	2.500625 ~ 2.50125	2.50125 ~ 2.501875	2.501875 ~ 2.5025	2.5025 ~ 2.503125	2.503125 ~ 2.50375	2.50375 ~ 2.504375	2.504375 ~ 2.505

전압 입력과 디지털 출력



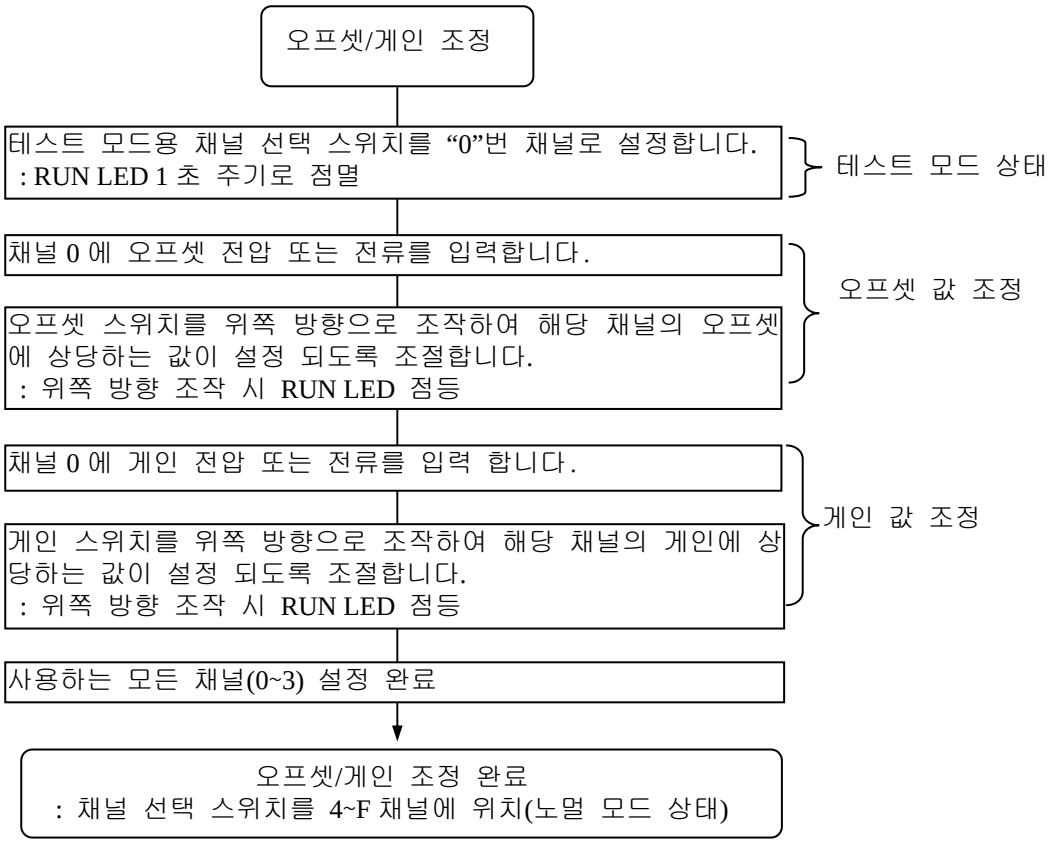
No	입력범위	디지털 출력값	오프셋값	게인값	분해능	아날로그 입력값					
①	DC -20~20 ㎃	-192 ~ 16191	0 ㎃	20 ㎃	2.5 ㎂	10.000 ~ 10.0025	10.0025 ~ 10.0050	10.0050 ~ 10.0075	10.0075 ~ 10.0100	10.0100 ~ 10.0125	10.0125 ~ 10.0150

전류 입력과 디지털 출력

(2) G4F-AD2A의 오프셋/게인 설정

가) 오프셋/게인 설정 순서

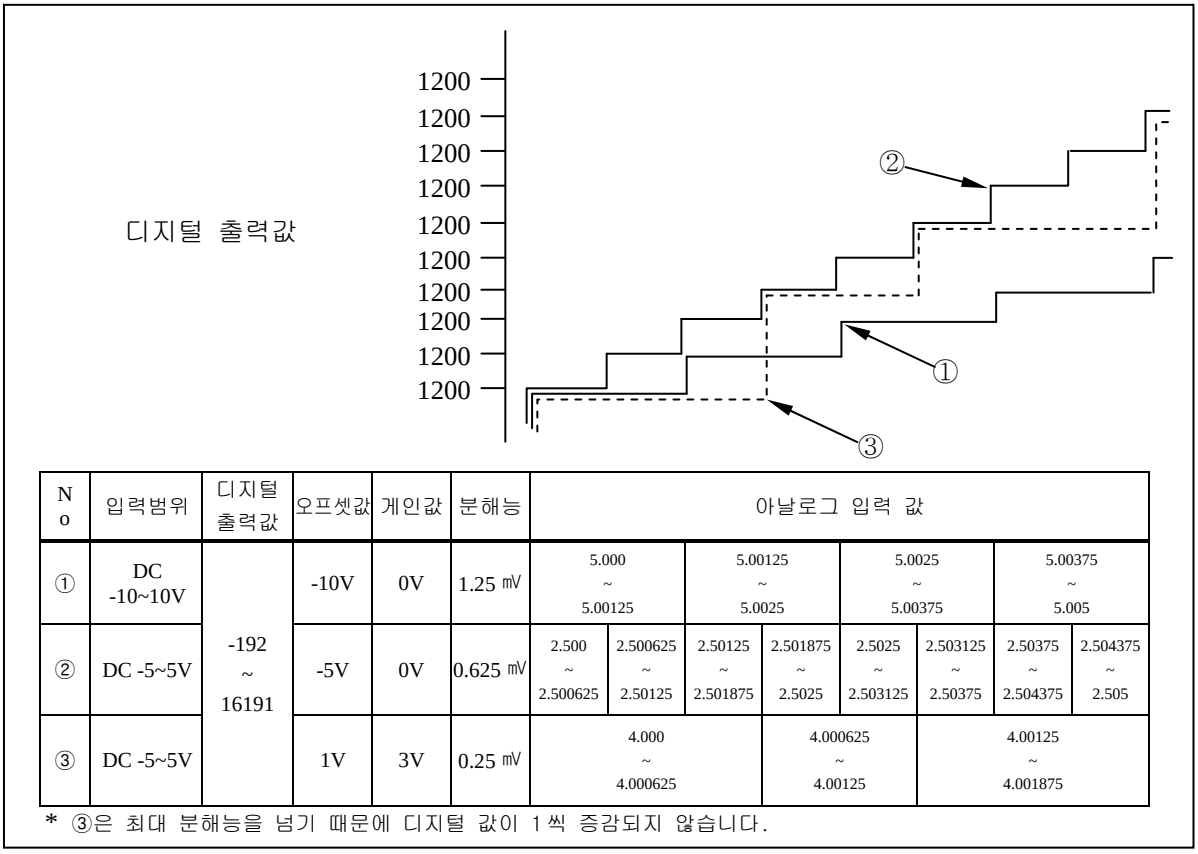
오프셋/게인 값은 각각의 채널에 대해서 조정합니다.



주의

1. 오프셋 값 및 게인 값은 실 사용 상태로 설정해 주십시오.
2. 오프셋 값 및 게인 값은 G4F-AD2A 내에 기억되어 전원이 Off 되어도 지워지지 않습니다.
3. 테스트(Test) 모드로 하면 모든 채널의 A/D변환이 중지됩니다.
4. 오프셋/게인 설정은 DC –10 ~ 0 ~ 10V 또는 DC –20 ~ 0 ~ 20㎃의 범위로 해 주십시오.
5. 이 범위를 초과해서 설정한 경우 최대 분해능/정밀도가 성능 규격 안에 들 수 없습니다.
6. 3.2.2항의 *5에 표시된 장소의 접지를 변경(미실시→실시 또는 실시→떼어냄)할 경우에는 반드시 오프셋/게인 설정을 처음부터 다시 해 주십시오.

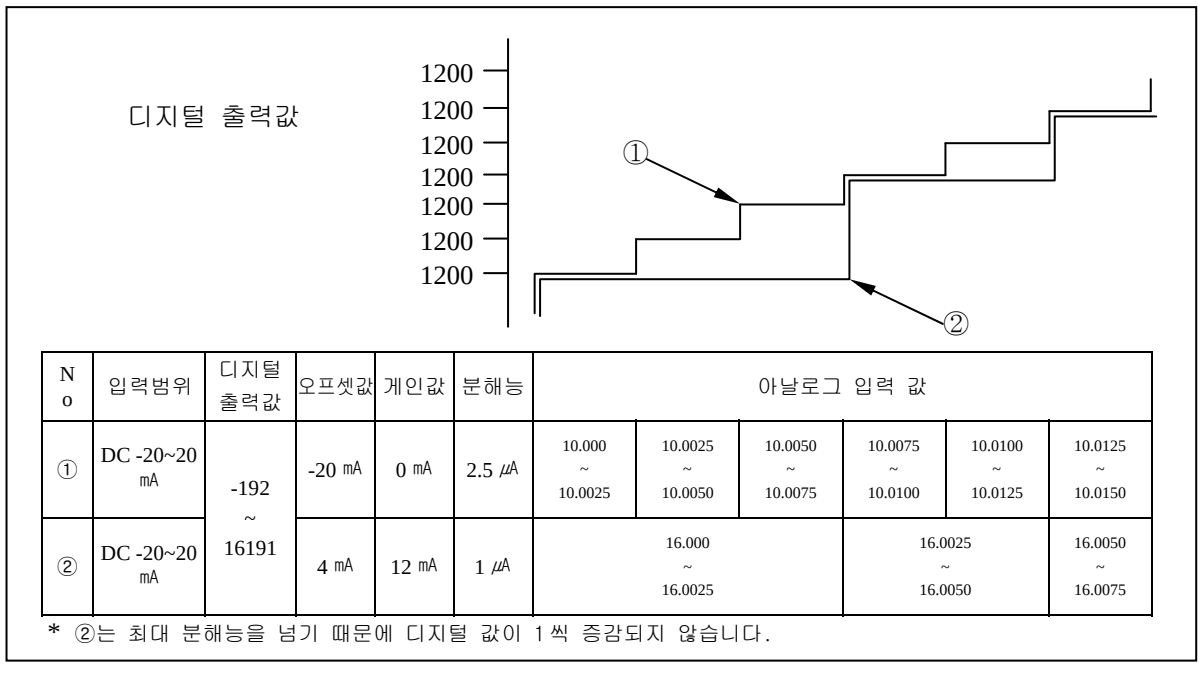
나) 오프셋/게인 설정에 따른 입출력 특성



No	입력범위	디지털 출력값	오프셋값	게인값	분해능	아날로그 입력 값							
①	DC -10~10V	-192 ~ 16191	-10V	0V	1.25 ㎷	5.000 ~ 5.00125		5.00125 ~ 5.0025		5.0025 ~ 5.00375		5.00375 ~ 5.005	
②	DC -5~5V		-5V	0V	0.625 ㎷	2.500 ~ 2.500625	2.500625 ~ 2.50125	2.50125 ~ 2.501875	2.501875 ~ 2.5025	2.5025 ~ 2.503125	2.503125 ~ 2.50375	2.50375 ~ 2.504375	2.504375 ~ 2.505
③	DC -5~5V		1V	3V	0.25 ㎷	4.000 ~ 4.000625			4.000625 ~ 4.00125			4.00125 ~ 4.001875	

* ③은 최대 분해능을 넘기 때문에 디지털 값이 1씩 증감되지 않습니다.

전압 입력과 디지털 출력



No	입력범위	디지털 출력값	오프셋값	게인값	분해능	아날로그 입력 값					
①	DC -20~20 ㎃	-192 ~ 16191	-20 ㎃	0 ㎃	2.5 ㎂	10.000 ~ 10.0025	10.0025 ~ 10.0050	10.0050 ~ 10.0075	10.0075 ~ 10.0100	10.0100 ~ 10.0125	10.0125 ~ 10.0150
②	DC -20~20 ㎃		4 ㎃	12 ㎃	1 ㎂	16.000 ~ 16.0025			16.0025 ~ 16.0050		16.0050 ~ 16.0075

* ②는 최대 분해능을 넘기 때문에 디지털 값이 1씩 증감되지 않습니다.

전류 입력과 디지털 출력

1.4 A/D 변환 특성

A/D 변환 방
- 필터 처리
- 샘플링 (순시 값) 처리
- 평균 처리
  - 횟수에 의한 평균 처리
  - 시간에 의한 평균 처리

1.4.1 필터 처리

필터는 노이즈 또는 입력 값의 급격한 변동을 필터(지연) 처리함으로써 아날로그 값에 대한 안정된 디지털 입력 값으로 사용할 수 있습니다.

$$PVfn = (1-\alpha) \times PVn + \alpha \times PVfn_{-1}$$

여기서
- $PVfn$:현재의 필터 출력 값
- PVn :현재의 A/D 변환 값
- $PVfn_{-1}$:이전의 필터 출력 값
- α :필터 상수(0.01 ~0.99)

사용되는 필터 상수 값은 1 ~99 입니다.

예1) G3F-AD4A/G4F-AD2A의 경우

●오프셋 : DC-10V, 게인 : DC0V일 때

(입력 전압 범위 : DC-10~ 10V, 디지털 출력 범위 : -192 ~ 16191)

●아날로그 입력이 –10V →–5V →0V →5V 순으로 변하면, α값에 따라 필터 출력 값은 다음과 같습니다.

α값	필터 출력 값				비 고
0.01	0	3960	7960	11960	이전 값에 1% 치우친 값
0.5	0	2000	5000	8500	이전 값에 50% 치우친 값
0.99	0	40	120	239	이전 값에 99% 치우친 값

예2) G3F-AD4B의 경우

●입력 전압 범위 : DC1~ 5V, 디지털 출력 범위 : 0 ~ 16000

●아날로그 입력이 1V →2V →3V →4V 순으로 변하면, α값에 따라 필터 출력 값은 다음과 같습니다.

α값	필터 출력 값				비 고
0.01	0	3960	7960	11960	이전 값에 1% 치우친 값
0.5	0	2000	5000	8500	이전 값에 50% 치우친 값
0.99	0	40	120	239	이전 값에 99% 치우친 값

▶즉, 필터를 금지하면 현재의 A/D 변환 값을 그대로 출력하게 되고, 필터를 사용하면 현재의 A/D 변환 값과 이전 값 사이에서 필터 상수 값에 의해 어느쪽에 비중을 많이 두느냐에 따라 A/D 변환 값이 결정 됩니다.

1.4.2 샘플링(순시값) 처리

일반적인 A/D변환 처리입니다.

즉 입력된 아날로그 값을 평균처리 없이 바로 디지털 값으로 변환됩니다.

샘플링 처리된 디지털 값이 메모리에 저장되는 시간은 사용하는 채널 수에 따라 달라집니다.

(처리 시간) = (사용 채널 수) X (변환 속도)

예) 사용 채널 수가 3인 경우

- G3F-AD4A(G3F-AD4B) : 3(사용 채널 수) X 3(변환 속도) = 9(ms)
- G4F-AD2A : 3(사용 채널 수) X 5(변환 속도) = 15(ms)

샘플링이란 평균 처리를 사용하지 않았을 때, 아날로그 입력이 디지털 값으로 바로 변환하는 것을 의미합니다.

1.4.3 평균 처리

1) 평균 처리 사용 이유

노이즈나 비정상적인 아날로그 입력을 평균 처리함으로써 시스템의 제어를 원활하게 하기 위해서 사용합니다

2) 평균 처리 종류

평균 처리의 종류는 시간 평균과 횟수 평균이 있습니다.

(1) 시간 평균 처리

가) 설정 범위

- G3F-AD4A(G3F-AD4B) : 96 ~ 12,000(ms)
- G4F-AD2A : 40 ~ 20,000(ms)

나) 시간 평균 처리를 설정하였을 경우 사용 채널 수에 따라 설정 시간 내의 평균 처리 횟수가 정해집니다.

$$\text{설정 시간 내의 평균 처리 횟수} = \frac{\text{설정 시간}}{\text{사용 채널 수 X 변환 속도}}$$

예) 사용 채널 수 4, 설정 시간 120ms인 경우 평균 처리 횟수

- G3F-AD4A(G3F-AD4B) : 120 ÷ (4 X 3) = 10회
- G4F-AD2A : 120 ÷ (4 X 5) = 6회

다) 설정 시간을 (사용 채널 수 X 변환 속도)로 나누었을 때 나머지가 발생 하였을 경우
처리 횟수는 올림하여 평균 처리 횟수를 (사용 채널 수 X 변환 속도로 나눈 값의 몫 + 1회)로 평균 처리합니다.

예) 사용 채널 수 4, 설정 시간 150㎳인 경우
- G3F-AD4A(G3F-AD4B) : 150 ÷(4 X 3) = 12회 + 나머지 6 → 13회
- G4F-AD2A : 150 ÷(4 X 5) = 7회 + 나머지 10 → 8회

(2) 횟수 평균 처리

가) 설정 범위
- G3F-AD4A(G3F-AD4B, G4F-AD2A) : 2 ~ 4,000(회)

나) 횟수 평균에 의한 평균 값이 메모리에 저장되는 시간은 사용 채널 수에 따라 달라집니다.

처리 시간 = 설정 횟수 X 사용 채널 수 X 변환 속도

예) 사용 채널 수 4, 평균 처리 횟수가 50회 인경우
- G3F-AD4A(G3F-AD4B) : 50 X 4 X 3 = 600㎳
- G4F-AD2A : 50 X 4 X 5 = 1000㎳

1.5 입력 배선 예

1.5.1 배선시의 주의사항

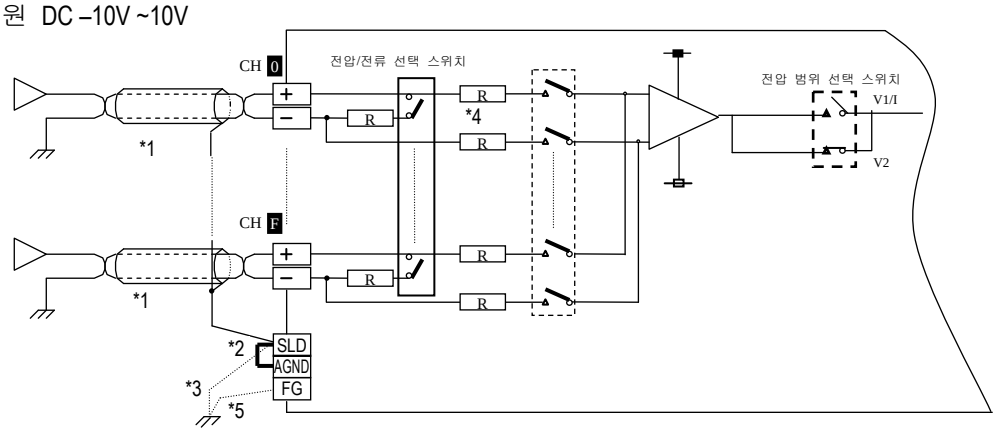
1) 교류와 A/D변환 모듈의 외부입력신호를 별도의 케이블을 사용하여 교류측에서 발생하는 서지 또는 유도 노이즈의 영향을 받지 않도록 하여 주십시오
2) 전선은 주위온도, 허용하는 전류를 고려해서 선정되어야 하며, 전선의 최대사이즈 AWG22(0.3 ㎟) 이상이 좋습니다.
3) 배선할 경우에 고온이 발생하는 기기나 물질에 너무 가까이 있거나, 기름등에 배선이 장시간 직접 접촉하게 되면 합선의 원인이 되어 파손이나 오동작을 발생할 수 있습니다.
4) 단자대에 아날로그 입력을 인가하기 전에 극성을 확인해야 합니다.
5) 배선을 고압선이나 동력선과 함께 배선하는 경우에는 유도 장애를 일으켜 오동작이나 고장의 원인이 될 수 있습니다.

1.5.2 배선 예

1) G3F-AD4A

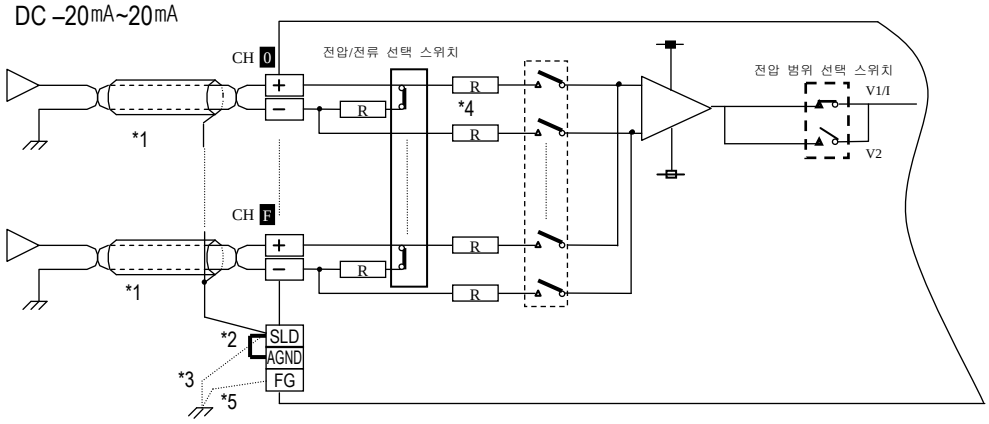
(1) 전압 입력의 경우

신호원 DC –10V ~10V



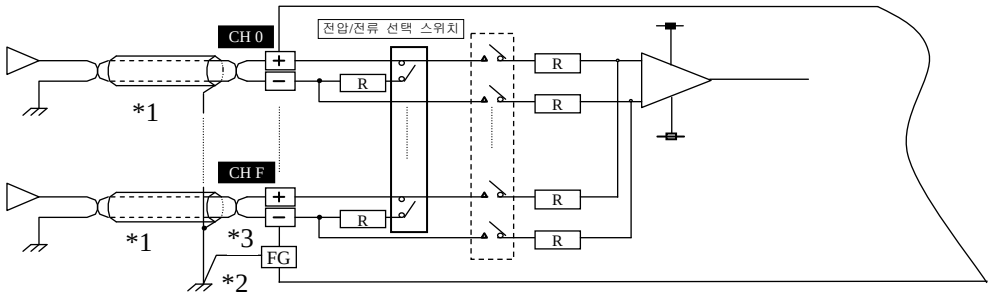
(2) 전류 입력의 경우

신호원 DC –20㎃~20㎃

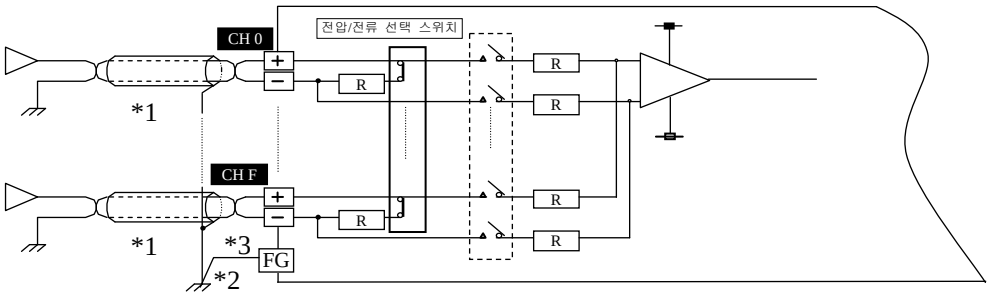


2) G3F-AD4B

(1)전압 입력의 경우

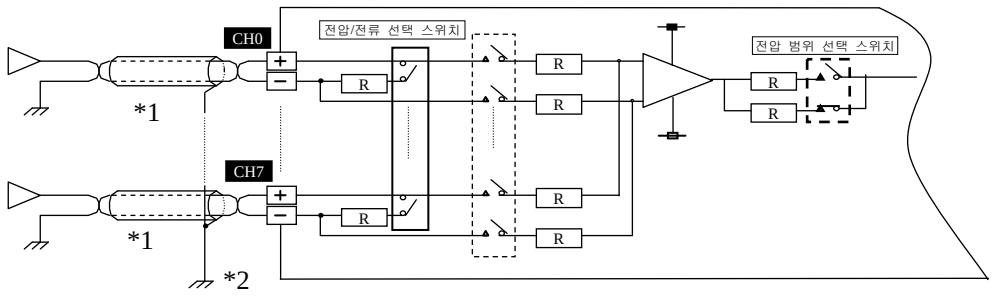


(2)전류 입력의 경우

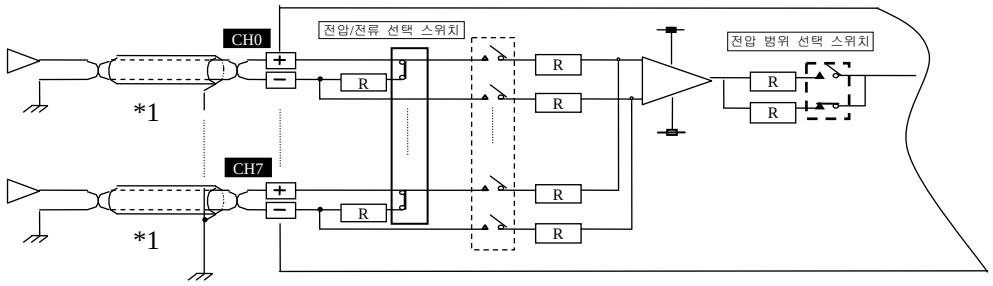


3) G3F-AD3A, G4F-AD3A

(1)전압 입력의 경우



(2)전류 입력의 경우



*1 : 전선은 2심 트위스트 실드선을 사용하여 주십시오.

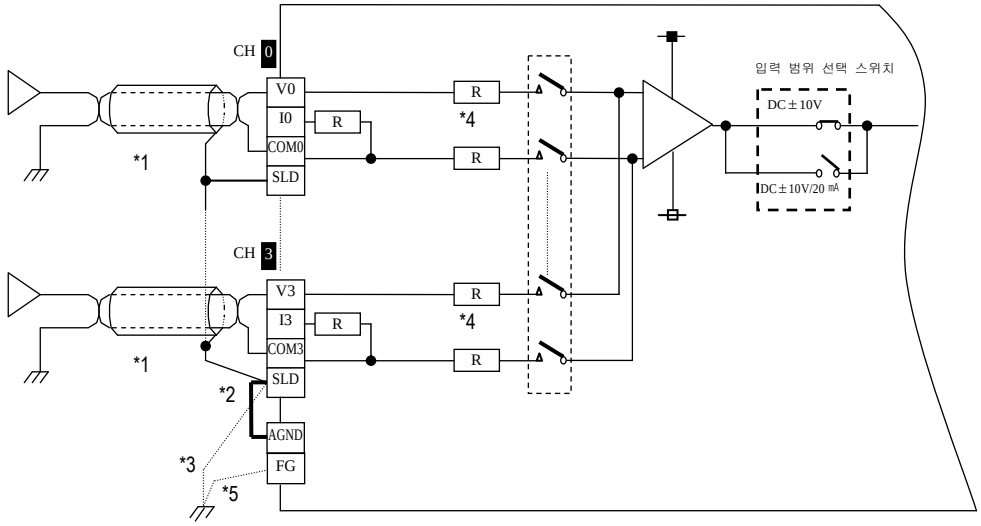
*2 : 노이즈가 많은 경우 접지해 주십시오

*3 : 노이즈가 많은 경우 SLD단자를 FG와 함께 접지해 주십시오.

4) G4F-AD2A

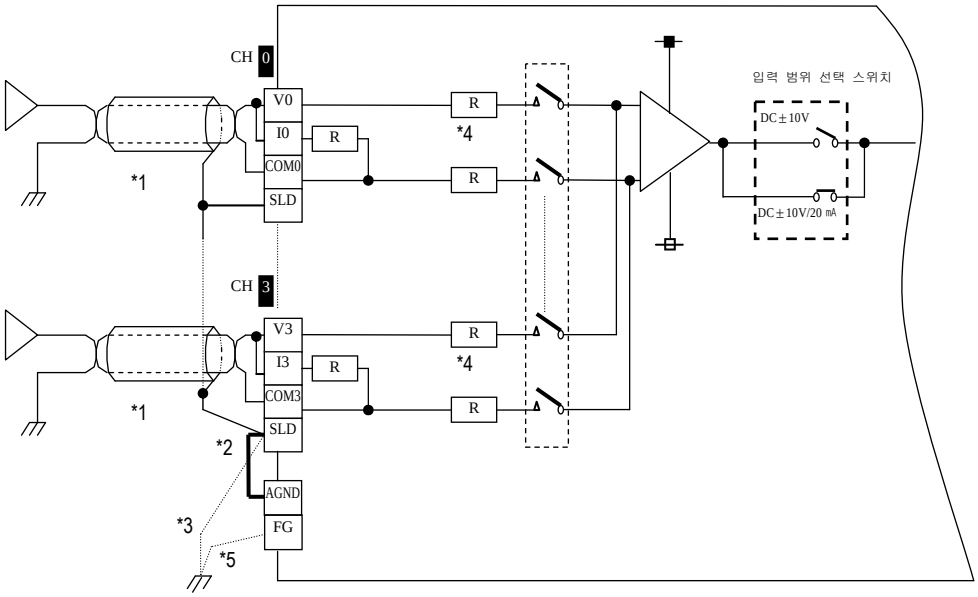
(1)전압 입력의 경우

신호원 DC –10V ~10V



(2)전류 입력의 경우

신호원 DC –20㎃~20㎃



*1 : 전선은 2심 트위스트 실드 선을 사용하여 주십시오.

*2 : SLD단자와 AGND단자를 반드시 접속하십시오.

*3 : 노이즈가 많은 경우 SLD단자를 FG와 함께 접지해 주십시오.

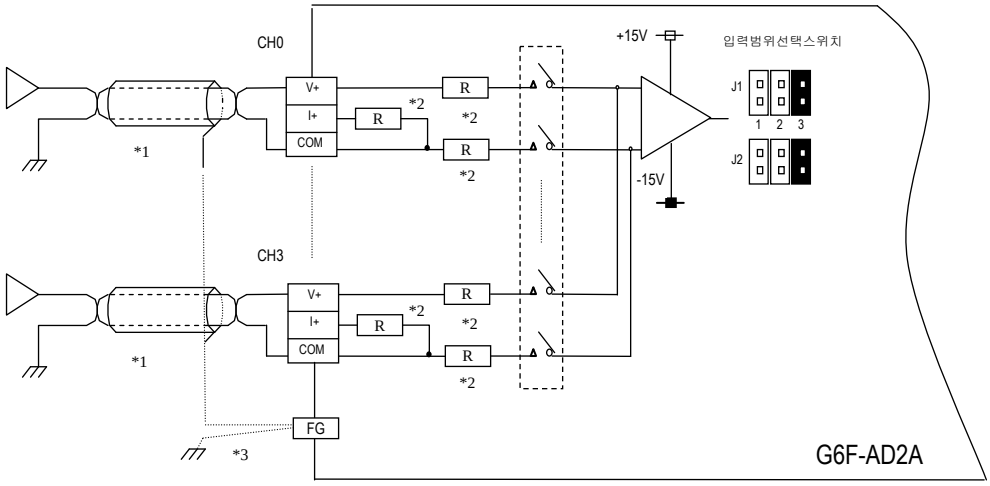
*4 : 입력 저항을 나타냅니다.

*5 : 노이즈가 많은 경우 접지해 주십시오.
전원 모듈의 FG와 접지하는 것이 좋은 경우가 있습니다.

5) G6F-AD2A

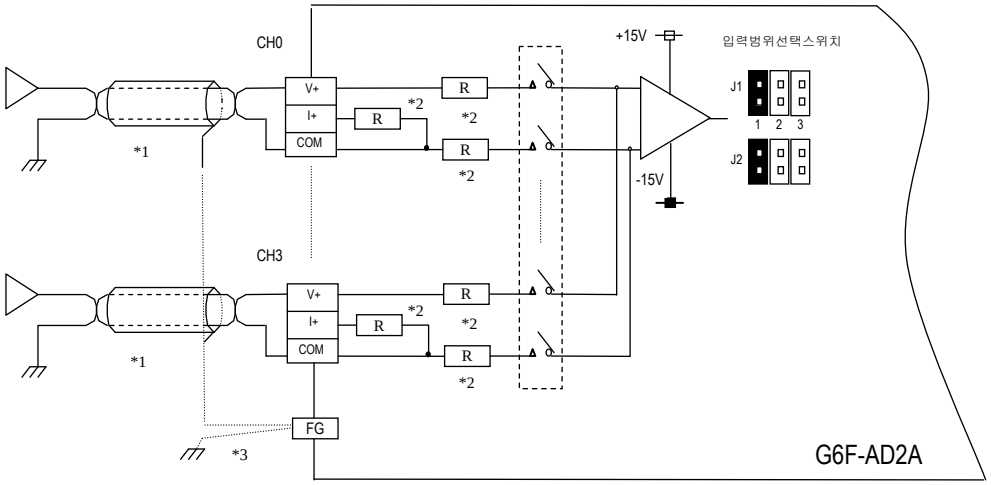
(1)전압 입력의 경우

신호원 DC –10~10V



(2)전류 입력의 경우

신호원 DC 4~20mA



*1 : 전선은 2심 트위스트 실드선을 사용하여 주십시오.
*2 : 입력 저항을 나타냅니다.
*3 : 노이즈가 많은 경우 접지해 주십시오. 전원 모듈의 FG와 접지하는 것이 좋은 경우가 있습니다.

제2장 GLOFA-GM 프로그래밍

A/D 변환 모듈을 사용하기 위해서는 A/D변환 모듈의 내부 메모리와 PLC CPU간의 인터페이스를 위한 펑션블록을 사용해야 합니다.

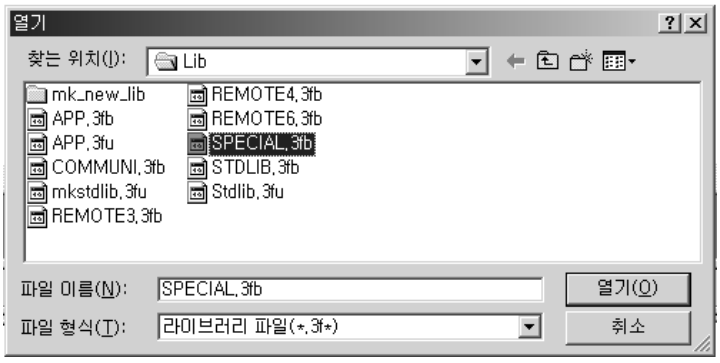
2.1 GMWIN에서 A/D 변환 모듈의 펑션블록 등록 절차

D/A 변환모듈용 펑션블록을 사용하기 위해서는 먼저 라이브러리의 등록이 필요합니다.

- Project창에서 Library창을 선택합니다.
- 마우스 오른쪽 버튼을 눌러 팝업 메뉴를 부릅니다.
- 팝업 메뉴에서 [프로젝트 항목 추가]-[라이브러리]선택.

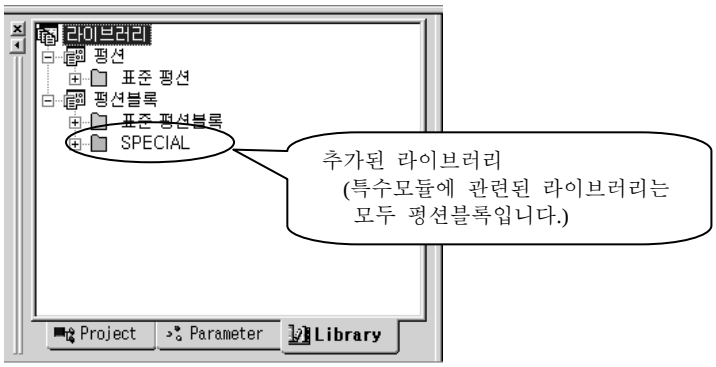
- 열기창에서 SPECIAL.□fb 또는 REMOTE*.□fb 파일을 선택하고 **열기**를 클릭합니다.

.(□ : GMR의 경우 "R" , GM1/2의 경우 "1", GM3의 경우 "3", GM4의 경우 "4", GM6의 경우 "6" 에 해당됩니다.)



(GM3의 경우 예)

- Library창에서 SPECIAL 라이브러리가 추가된 것을 확인합니다.



(SPECIAL 라이브러리를 추가한 경우 예)

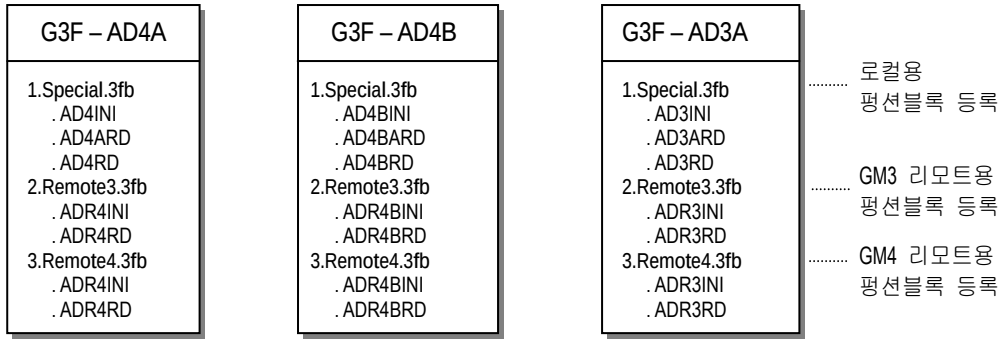
2.2 펑션블록 의 종류

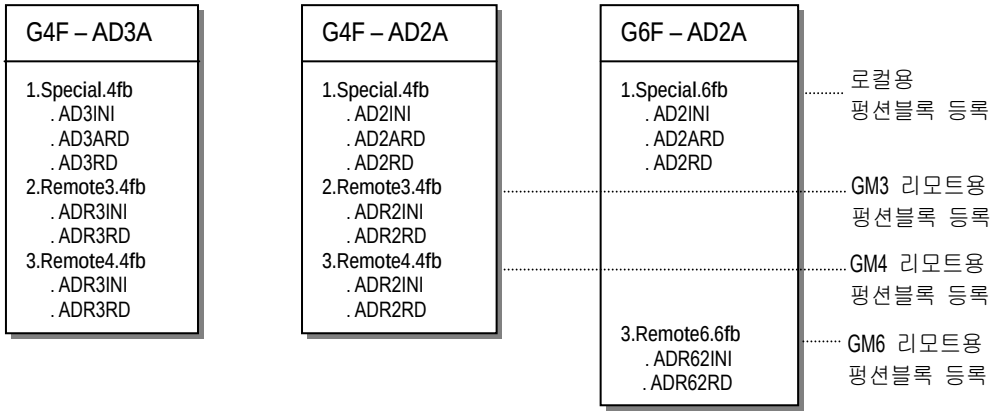
GMWIN에서 사용되는 A/D변환 모듈용 펑션블록에 대해서 설명합니다.

펑션블록의 종류는 다음과 같습니다.

No	G3F-AD4A		G3F-AD4B		G3F-AD3A		비 고
	로컬	리모트	로컬	리모트	로컬	리모트	
1	AD4INI	ADR4INI	AD4BINI	ADR4BINI	AD3INI	ADR3INI	모듈 초기화
2	AD4ARD	ADR4RD	AD4BARD	ADR4BRD	AD3ARD	ADR3RD	A/D 변환 값 읽기(복수형)
3	AD4RD	-	AD4BRD	-	AD3RD	-	A/D 변환 값 읽기(단일형)

No	G4F-AD3A		G4F-AD2A		G6F-AD2A		비 고
	로컬	리모트	로컬	리모트	로컬	리모트	
1	AD3INI	ADR3INI	AD2INI	ADR2INI	AD2INI	ADR62INI	모듈 초기화
2	AD3ARD	ADR3RD	AD2ARD	ADR2RD	AD2ARD	ADR62RD	A/D 변환 값 읽기(복수형)
3	AD3RD	-	AD2RD	-	AD2RD	-	A/D 변환 값 읽기(단일형)





2.3 펑션블록의 공통사항

아래의 입출력 변수명의 기능과 사용법은 2.4항에서 설명되는 모든 펑션블록에 공통되는 내용으로 동일합니다.

구분	변수명	Datatype	내용
입력	REQ	BOOL	펑션블록 실행 요구 영역 ● 이 영역은 초기화 펑션블록의 실행을 요구하는 영역입니다. ● 프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 모듈 초기화 펑션블록이 실행됩니다.
	BASE	USINT	베이스 위치 번호 ● A/D 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. ● 설정 범위 : GM1시리즈(0 ~ 31), GM2시리즈(0 ~ 7), GM3/4시리즈(0 ~ 3) GM6시리즈(0,1)
	SLOT	USINT	슬롯의 위치 번호 ●위치 결정 모듈이 장착된 슬롯의 번호를 설정하는 영역입니다. ●설정 범위 : 0 ~ 7
	CH	BOOL [Array]*1	사용 채널 지정 영역 ● "0"이면 사용하지 않는 채널 ● "1"이면 사용하는 채널
출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역 ● 초기화 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며, 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
	STAT	USINT	에러 상태 표시 영역 ●펑션블록 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다.
	ACT	BOOL [Array]*1	운전 채널 표시 영역 ● 초기화 펑션블록이 에러 없이 실행된 후 지정된 채널의 설정 조건이 정상이면 "1"이 출력되고, 비정상이면 "0"이 출력됩니다. ● 운전 지정이 안된 채널은 "0"이 출력됩니다.
	DATA	INT [Array]*1	A/D 변환 값 출력 영역 G3F-AD4A, G4F-AD2A : -192 ~ 16191 또는 –8192 ~ 8191 G4F-AD4B : 0 ~ 16000 또는 –8000 ~ 8000 G3F-AD3A, G4F-AD3A : -48 ~ 4047 또는 –2048 ~ 2047 G6F-AD2A : -48 ~ 4047 또는 –2048 ~ 2047

알아두기

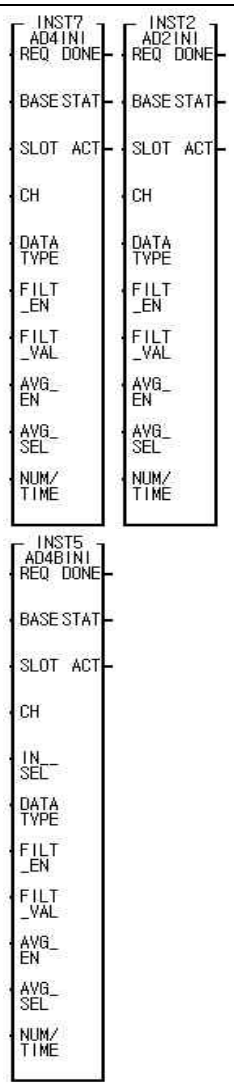
※1 : Array 수는 G3F-AD4A :16, G3F-AD4B: 16, G3F-AD3A :8, G4F-AD3A :8, G4F-AD2A : 4, G6F-AD2A :4 이며, 원소번호가 채널을 의미합니다.

2.4 로컬용 펑션블록

2.4.1 모듈 초기화 펑션블록

(1) G3F-AD4A : AD4INI , G4F-AD2A : AD2INI, G3F-AD4B : AD4BINI

모듈 초기화 펑션블록은 A/D 변환 모듈의 베이스 위치, 슬롯 장착 위치, 사용 채널 지정, A/D변환용 Data Type, 필터 처리 정보 및 평균 처리 정보를 설정하여 프로그램에 이용합니다.

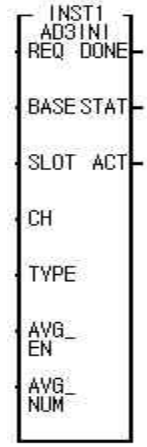
펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	펑션블록 실행 요구 영역
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL [Array]	사용 채널 지정 영역
		IN_ SEL*3	BOOL [Array] *3	아날로그 입력 종류(전류/전압) 지정 ● "0"이면 전류 입력 선택 ● "1"이면 전압 입력 선택
		DATA TYPE	BOOL [Array]*1	출력 데이터 타입 지정 영역 ● 각 채널의 출력 데이터 타입의 종류를 설정하는 영역입니다. ● "0"이면 디지털 출력 범위가 -192 ~ 16191 (G3F-AD4B : 0 ~ 16000) ● "1"이면 디지털 출력 범위가 -8192 ~ 8191 (G3F-AD4B : -8000 ~ 8000)
		FILT _EN	BOOL [Array]*2	필터 처리의 사용 허가/금지 설정 영역 ● "0"이면 필터 처리 사용 금지 ● "1"이면 필터 처리 사용 허가
		FILT _VAL	USINT [Array]*2	필터 상수 값 설정 영역 ● 설정 범위 : 1~99
		AVG_ EN	BOOL [Array]*2	평균 처리의 사용 허가/금지 설정 영역 ● "0"이면 평균 처리의 사용 금지 ● "1"이면 평균 처리의 사용 허가
		AVG_ SEL	BOOL [Array]*2	평균 처리 방식 지정 영역 ● "0"이면 횟수 평균 ● "1"이면 시간 평균
		NUM/ TIME	USINT [Array]*2	평균 횟수 또는 평균 시간 지정 영역 ● AVG_SEL에서 지정한 평균 처리 방식에 따라 횟수(회) 또는 시간(ms)을 지정 ● 평균 횟수 : 2~4000 ● 평균 시간 : G3F-AD4A/G3F-AD4B [96~12,000(ms)] G4F-AD2A [40~20,000(ms)]
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL [Array]	운전 채널 표시 영역

알아두기

※1 : Array 수는 G3F-AD4A :16, G3F-AD4B: 16, G4F-AD2A : 4 이며, 원소번호가 채널을 의미합니다.
※2 : Array 수는 G3F-AD4A :4, G3F-AD4B: 16, G4F-AD2A : 4 입니다.
G3F-AD4A 에서는 원소번호 [0]이 채널 0, 1, 2, 3 을 일괄로 지정하며,
원소번호 [1]이 채널 4, 5, 6, 7 을 일괄로 지정하며,
원소번호 [2]이 채널 8, 9, 10, 11 을 일괄로 지정하며,
원소번호 [3]이 채널 12, 13, 14, 15 를 일괄로 지정합니다.
G4F-AD2A 에서는 원소번호가 채널을 의미합니다.
※3 : G3F-AD4B 에서만 사용하며, :Array 수는 16 이며, 원소번호가 채널을 의미합니다.

(2) G3F-AD3A/G4F-AD3A : AD3INI

모듈 초기화 펑션블록은 A/D 변환 모듈의 베이스 위치, 슬롯 장착 위치, 사용 채널 지정, 사용 전압 종류 지정, 횟수 평균 처리 정보를 설정하여 프로그램에 이용합니다.

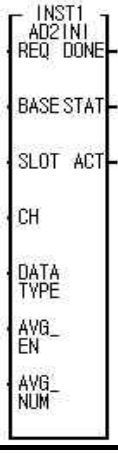
펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	펑션블록 실행 요구 영역
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL[8]	사용 채널 지정 영역
		TYPE	BOOL[8]	아날로그 입력 전압 종류를 지정하는 영역 ● "0"이면 전압 DC1~5, 전류 DC4~20㎃ ● "1"이면 전압 DC0~10 ▣ "1"로 선택하면 전류를 아날로그 입력으로 사용할 수 없습니다.
		AVG_ EN	BOOL[8]	횟수 평균 처리의 사용허가/금지 설정 영역 ● "0"이면 샘플링 처리 ● "1"이면 횟수 평균 처리
		AVG_ NUM	USINT[8]	횟수 평균 상수 값 설정 영역 ● 설정 범위 : 2~255
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL[8]	운전 채널 표시 영역 .

알아두기

● Datatype 중 BOOL[8]과 USINT[8]은 원소의 수가 8 임을 나타내며 이는 채널수와 번호를 의미합니다.

(3) G6F-AD2A : AD2INI

모듈 초기화 펑션블록은 A/D 변환 모듈의 베이스 위치, 슬롯 장착 위치, 사용 채널 지정, 사용 전압 종류 지정, 횟수 평균 처리 정보를 설정하여 프로그램에 이용합니다

펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	펑션블록 실행 요구 영역
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL[4]	사용 채널 지정 영역
		DATA TYPE	BOOL[4]	디지털 출력 데이터 종류를 지정하는 영역 ● "0"이면: -48 ~ 4047 ● "1"이면: -2048 ~ 2047
		AVG_ EN	BOOL[4]	횟수 평균 처리의 사용허가/금지 설정 영역 ● "0"이면 샘플링 처리 ● "1"이면 횟수 평균 처리
		AVG_ NUM	USINT[4]	횟수 평균 상수 값 설정 영역 ● 설정 범위 : 2~255
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL[4]	운전 채널 표시 영역

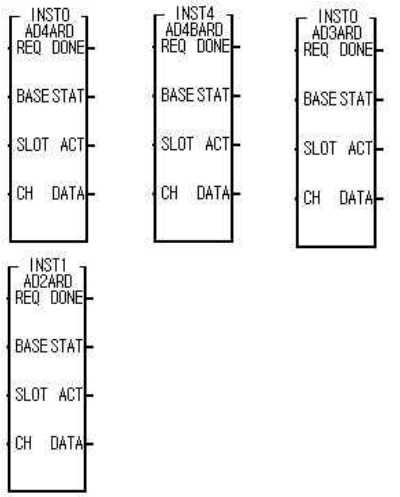
알아두기

● Datatype 중 BOOL[4]과 USINT[4]은 원소의 수가 4 임을 나타내며 이는 채널수와 번호를 의미합니다.

2.4.2 모듈 읽기_Array형 펑션블록

(1) G3F-AD4A : AD4ARD, G3F-AD4B : AD4BARD, G3F-AD3A/G4F-AD3A : AD3ARD, G4F-AD2A/G6F-AD2A : AD2ARD

복수형 모듈 읽기 펑션블록은 A/D 변환 모듈의 채널을 일괄로 처리하며 사용 채널로 지정되면 A/D 변환한 디지털 값을 읽어서 출력변수 DATA에 나타내어 프로그램에 이용합니다.

펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	펑션블록 실행 요구 영역
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL [Array]*1	사용 채널 지정 영역
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL [Array]*1	운전 채널 표시 영역
		DATA	INT [Array]*1	A/D 변환 값 출력 영역

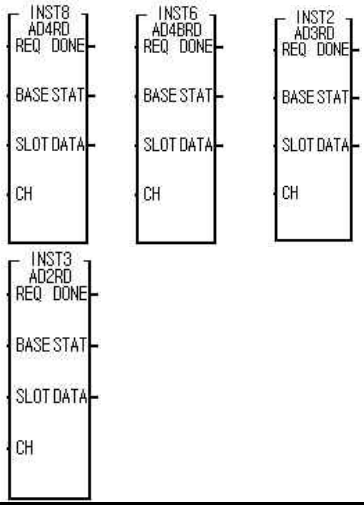
알아두기

※1 : Array 수는 G3F-AD4A :16, G3F-AD4B: 16, G4F-AD2A : 4 이며, 원소번호가 채널을 의미합니다.

2.4.3 모듈 읽기_단일형 펑션블록

(1) G3F-AD4A : AD4RD, G3F-AD4B : AD4BRD, G3F-AD3A/G4F-AD3A : AD3RD, G4F-AD2A/G6F-AD2A : AD2RD

단일형 모듈 읽기 펑션블록은 A/D 변환 모듈의 한 채널만 처리하며 A/D 변환한 디지털 값을 읽어서 출력변수 DATA에 나타내어 프로그램에 이용합니다.

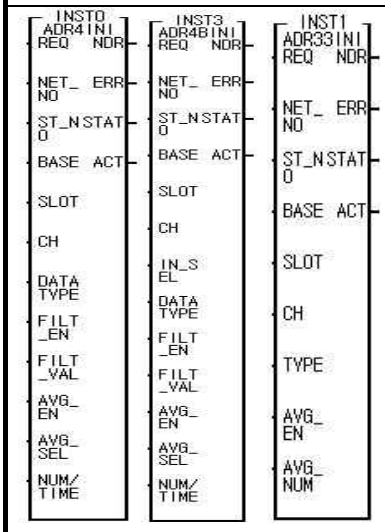
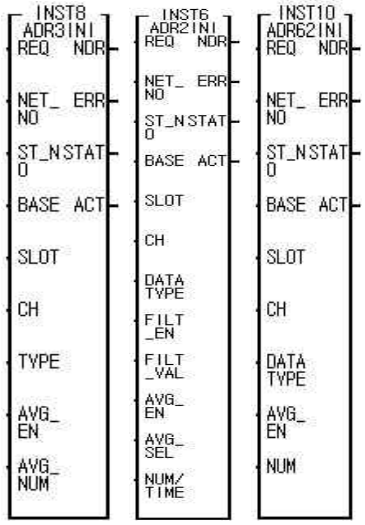
펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	펑션블록 실행 요구 영역
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	USINT	사용 채널 지정 영역
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역
		STAT	USINT	에러 상태 표시 영역
		DATA	INT	A/D 변환 값 출력 영역

2.5 리모트용 펑션블록

2.5.1 모듈 초기화 펑션블록

(1) G3F-AD4A : ADR4INI, G3F-AD4B : ADR4BINI ,G3F-AD3A : ADR33INI, G4F-AD3A : ADR3INI
G4F-AD2A : ADR2INI, G6F-AD2A : ADR62INI)

모듈 초기화 펑션블록은 A/D 변환 모듈의 자국 통신 모듈의 슬롯 위치 번호, 리모트 I/O국에 장착된 통신 모듈의 국번, 베이스 위치, 슬롯 장착 위치, 사용 채널 지정, A/D변환용 Data Type, 필터 처리 정보 및 평균 처리 정보를 설정하여 프로그램에 이용합니다

펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	상승 Edge에서 펑션블록 실행 요구 영역
		NET_NO	USINT	펑션블록이 전송될 자국의 통신 모듈이 장착된 슬롯 위치 번호 ● 설정 범위 : 0 ~ 7
		ST_NO	USINT	리모트 I/O국에 장착된 통신 모듈의 국번 ● 설정 범위 : 0 ~ 63
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL [Array]*1	사용 채널 지정 영역
		IN_SEL*3	BOOL [Array]	아날로그 입력 종류(전류/전압) 지정 ● "0"이면 전류 입력 선택 ● "1"이면 전압 입력 선택
		DATA TYPE	BOOL [Array]*1	출력 데이터 타입 지정 영역
		FILT_EN	BOOL [Array]*2	필터 처리의 사용 허가/금지 설정 영역
		FILT_VAL	USINT [Array]*2	필터 상수 값 설정 영역
		AVG_EN	BOOL [Array]*2	평균 처리의 사용 허가/금지 설정 영역
		AVG_SEL	BOOL [Array]*2	평균 처리 방식 지정 영역
		NUM/TIME	USINT [Array]*2	평균 횟수 또는 평균 시간 지정 영역
	출력	NDR	BOOL	펑션블록 실행이 에러 없이 종료된 경우 "1"이 되며, 실행 조건이 성립된 스켄동안 "1"을 유지하고 다음 스켄에서 "0"이 됩니다.
		ERR	BOOL	에러 정보 표시 영역 ● 초기화 펑션블록 실행 중 에러가 발생되면 "1"이 출력되어 운전 정지 상태가 되고, 실행 조건이 성립된 스켄 동안 "1"을 유지하며, 다음 스켄에서 "0"이 됩니다.
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL [Array]*1	운전 채널 표시 영역 .

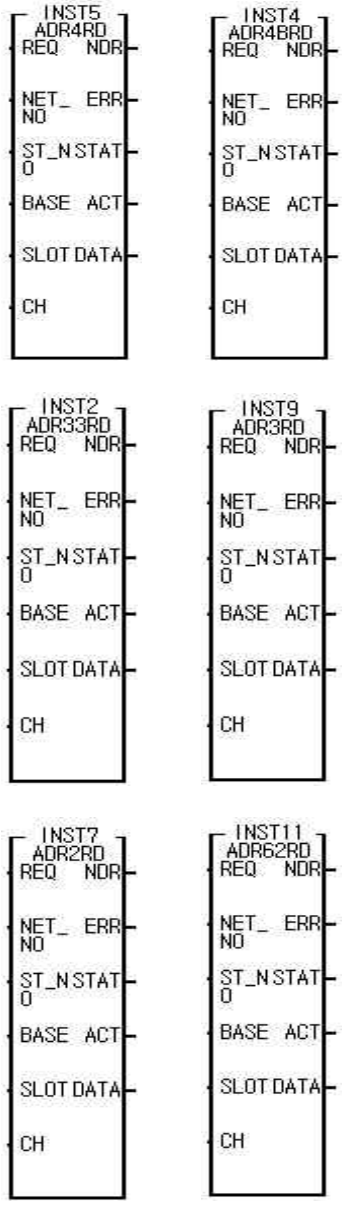
알아두기

※1 : Array 수는 G3F-AD4A :16, G3F-AD4B: 16, G4F-AD2A : 4 이며, 원소번호가 채널을 의미합니다.
※2 : Array 수는 G3F-AD4A :4, G3F-AD4B: 16, G4F-AD2A : 4 입니다.
G3F-AD4A 에서는 원소번호 [0]이 채널 0, 1, 2, 3 을 일괄로 지정하며,
원소번호 [1]이 채널 4, 5, 6, 7 을 일괄로 지정하며,
원소번호 [2]이 채널 8, 9, 10, 11 을 일괄로 지정하며,
원소번호 [3]이 채널 12, 13, 14, 15 를 일괄로 지정합니다.
G4F-AD2A 에서는 원소번호가 채널을 의미합니다.
※3 : G3F-AD4B 에서만 사용하며, :Array 수는 16 이며, 원소번호가 채널을 의미합니다.

2.5.2 모듈 읽기 펑션블록

(1) G3F-AD4A : ADR4RD, G3F-AD4B : ADR4BRD, G3F-AD3A : ADR33RD, G4F-AD3A : ADR3RD, G4F-AD2A: ADR2RD, G6F-AD2A : ADR62RD

리모트에서의 모듈 읽기 펑션블록은 A/D 변환 모듈의 전 채널을 일괄로 처리하며 사용 채널로 지정되면 A/D 변환한 디지털 값을 읽어서 출력변수 DATA에 나타내어 프로그램에 이용합니다

펑션블록형태	구분	변수명	Datatype	내용
	입력	REQ	BOOL	상승 Edge에서 펑션블록 실행 요구 영역
		NET_NO	USINT	펑션블록이 전송될 자국의 통신 모듈이 장착된 슬롯 위치 번호 ● 설정 범위 : 0 ~ 7
		ST_NO	USINT	리모트 I/O국에 장착된 통신 모듈의 국번 ● 설정 범위 : 0 ~ 63
		BASE	USINT	베이스 위치 번호
		SLOT	USINT	슬롯의 위치 번호
		CH	BOOL [Array]	사용 채널 지정 영역
	출력	NDR	BOOL	펑션블록 실행이 에러 없이 종료된 경우 "1"이 되며, 실행 조건이 성립된 스켄동안 "1"을 유지하고 다음 스켄에서 "0"이 됩니다.
		ERR	BOOL	에러 정보 표시 영역 ● 초기화 펑션블록 실행 중 에러가 발생되면 "1"이 출력되어 운전 정지 상태가 되고, 실행 조건이 성립된 스켄 동안 "1"을 유지하며, 다음 스켄에서 "0"이 됩니다.
		STAT	USINT	에러 상태 표시 영역
		ACT	BOOL [Array]	운전 채널 표시 영역
		DATA	INT [Array]	A/D 변환 값 출력 영역

2.6 펑션블록 상의 에러 코드

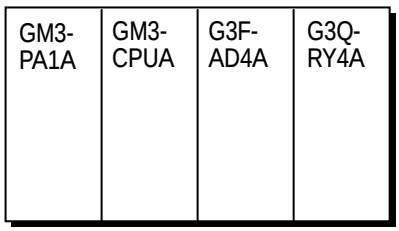
출력 변수 STAT에 나타내는 에러 종류 및 조치 방법에 대해서 설명합니다.

STAT 번호	구분	내 용	펑션블록			조 치 방 법
			초기화	읽기		
				Array형	단일형	
0	로컬	정상 동작중	O	O	O	-
1		베이스의 위치가 설정 범위 초과	O	O	O	설정 범위 내로 수정(4.2항 참조)
2		해당 베이스의 H/W 에러	O	O	O	A/S 의뢰
3		슬롯의 위치 번호가 설정 범위 초과	O	O	O	A/D 변환 모듈이 장착된 올바른 슬롯 번호 지정
4		지정한 슬롯에 A/D 변환 모듈이 비어 있음	O	O	O	지정된 슬롯에 A/D 변환 모듈을 장착
5		A/D 변환 모듈이 아닌 다른 모듈이 장착되어 있음	O	O	O	지정된 슬롯에 A/D 변환 모듈을 장착
6		채널 번호가 설정 범위 초과	-	-	O	사용 채널 지정을 바르게 설정
7		A/D 변환 모듈의 H/W 에러	O	O	O	A/S 의뢰
8		A/D 변환 모듈의 공용 메모리 에러	O	O	O	A/S 의뢰
9		초기화 펑션블록에서 사용채널 미지정	-	O	O	초기화 펑션블록에서 사용채널을 바르게 지정
10		테스트(Test) 모드		O	O	테스트 모드에서 노멀 모드로 전환(G4F-AD2A만 해당)
11		필터 값 설정 초과	O			설정 범위 1 ~ 99 내로 수정
17		평균 횟수/시간 평균 값 설정 범위 초과	○	-	-	설정범위[횟수:2~4000회, 시간:G3F-AD4A, G3F-AD4B (96~12,000ms)/G4F-AD2A(40~20,000ms)]로 수정
128	리모트	리모트용 통신 모듈의 H/W 에러	O	O	-	리모트 통신 모듈 참조
129		베이스의 위치가 설정 범위 초과	O	O		설정 범위 내로 수정(4.2항 참조)
131		슬롯의 위치 번호가 설정 범위 초과	O	O		A/D 변환 모듈이 장착된 올바른 슬롯 번호 지정
133		A/D 변환 모듈이 아닌 다른 모듈이 장착되어 있음	O	O		지정된 슬롯에 A/D 변환 모듈을 장착
135		A/D 변환 모듈의 H/W 에러	O	O		A/S 의뢰
136		A/D 변환 모듈의 공용 메모리 에러	O	O		A/S 의뢰
137		초기화 펑션블록에서 사용채널 미지정	-	O		초기화 펑션블록에서 사용채널을 바르게 지정
138		테스트(Test) 모드		O		테스트 모드에서 노멀 모드로 전환(G4F-AD2A만 해당)
144		필터 값 설정 초과	O			설정 범위 1 ~ 99 내로 수정
145		평균 횟수 값 설정 범위 초과	○	-		설정범위[횟수:2~4000회, 시간:G3F-AD4A, G3F-AD4B (96~12,000ms)/G4F-AD2A(40~20,000ms)]로 수정

제 3 장 프로그램 예제(GLOFA-GM 용)

3.1 A/D 변환값의 대소 구분 프로그램

1) 시스템 구성



2) 초기 설정 내용

(1) 사용 채널 : 채널 0, 채널 2, 채널 4
(2) 출력 Data Type 지정 : -192 ~ 16191(채널 0, 2, 4)
(3) 필터 사용 채널 : 채널 0
(4) 필터 상수 설정 : 채널 0 = 50
(5) 평균 처리 지정 : 채널 2, 4
(6) 횟수 평균 지정 및 설정 값 : 채널 2 = 100 회
(7) 시간 평균 지정 및 설정 값 : 채널 4 = 200 ㎳
(8) 아날로그 입력 : 전류 입력

3) 프로그램 설명

(1) 채널 0 의 디지털 값이 12,000 보다 작을 때 %Q0.1.0 을 On
(2) 채널 2 의 디지털 값이 13,600 보다 클 때 %Q0.1.1 을 On
(3) 채널 4 의 디지털 값이 12,000 보다 크거나 같고 13,600 보다 작거나 같을 때 %Q0.1.2 을 On
(4) 채널 4 의 디지털 값이 12,800 과 같을 때 %Q0.1.3 을 On

4) 프로그램

(1)G4F-AD4A 의 경우

가) 프로그램상 주의 사항

알아두기

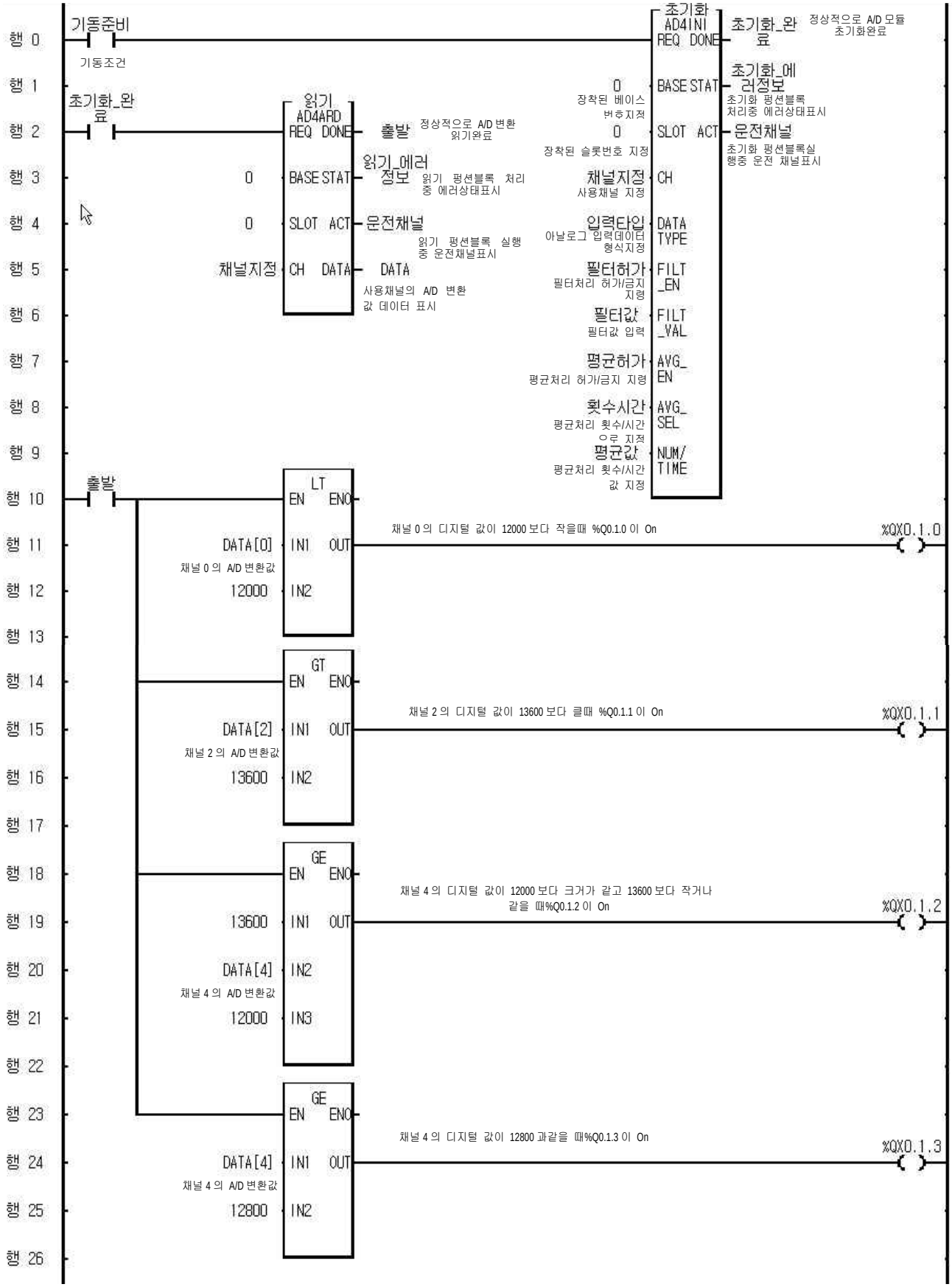
G3F-AD4A 를 사용할 때 프로그램의 초기 설정 내용 중

1) (3)항에서 채널 0 을 필터 사용으로 선택하면 채널 0 ~ 채널 3 모두가 필터 사용으로 선택됨.
2) (4)항에서 채널 0 을 필터 상수 값으로 설정하면 채널 0 ~ 채널 3 모두가 필터 상수 값을 50 으로 설정됨.
3) (5)항에서 채널 2 와 채널 4 를 평균 처리 지정하면 채널 0 ~ 채널 3, 채널 4 ~ 채널 7 모두가 평균 처리로 지정됨.
4) (6)항에서 채널 2 를 횟수 평균 값 100 회로 지정하면 채널 0 ~ 채널 3 모두가 횟수 평균 값 100 회로 설정됨.
5) (7)항에서 채널 4 를 시간 평균 값 200 ㎳로 지정하면 채널 4 ~ 채널 7 모두가 횟수 평균 값 200 ㎳로 설정됨.

나) 프로그램에 사용된 입출력 변수

Variable Name	Var_Kind	Data Type	(AT Address) (Initial Value)
채널지정	: VAR	: ARRAY [0..15] OF BOOL	: = { 1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0 }
초기화 I	: VAR	: FB instance	
읽기	: VAR	: FB instance	
평균허가	: VAR	: ARRAY [0..3] OF BOOL	: = { 1,1,0,0 }
횟수시간	: VAR	: ARRAY [0..3] OF BOOL	: = { 0,1,0,0 }
DATA	: VAR	: ARRAY [0..15] OF INT	
입력타입	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
필터허가	: VAR	: ARRAY [0..3] OF BOOL	: = { 1,0,0,0 }
필터값	: VAR	: ARRAY [0..3] OF USINT	: = { 50,0,0,0 }
운전채널	: VAR	: ARRAY [0..15] OF BOOL	
초기화_에러정보	: VAR	: USINT	
평균값	: VAR	: ARRAY [0..3] OF UNIT	: = { 100,200,0,0 }
기동준비	: VAR	: BOOL	
출발	: VAR	: BOOL	

다) 프로그램예


행 0
기동준비
기동조건
초기화
AD4INI
REQ DONE
초기화_완료
정상적으로 A/D 모듈 초기화완료
초기화_에러정보
초기화 평션블록 처리중 에러상태표시
0
장착된 베이스 번호지정
BASE STAT
0
장착된 슬롯번호 지정
SLOT ACT
운전채널
초기화 평션블록실행중 운전 채널표시
채널지정
사용채널 지정
CH
입력타입
아날로그 입력데이터 형식지정
DATA TYPE
필터허가
필터처리 허가/금지 지령
FILT_EN
필터값
필터값 입력
FILT_VAL
평균허가
평균처리 허가/금지 지령
AVG_EN
횟수시간
평균처리 횟수/시간으로 지정
AVG_SEL
평균값
평균처리 횟수/시간 값 지정
NUM/TIME
행 1
행 2
초기화_완료
읽기
AD4ARD
REQ DONE
출발
정상적으로 A/D변환 읽기완료
행 3
0
BASE STAT
읽기_에러정보
읽기 평션블록 처리중 에러상태표시
행 4
0
SLOT ACT
운전채널
읽기 평션블록 실행중 운전채널표시
행 5
채널지정
CH DATA
DATA
사용채널의 A/D 변환값 데이터 표시
행 6
행 7
행 8
행 9
행 10
출발
LT
EN ENO
행 11
DATA[0]
IN1 OUT
채널 0 의 A/D 변환값
채널 0 의 디지털 값이 12000 보다 작을때 %Q0.1.0 이 On
%QX0.1.0
행 12
12000
IN2
행 13
행 14
GT
EN ENO
행 15
DATA[2]
IN1 OUT
채널 2 의 A/D 변환값
채널 2 의 디지털 값이 13600 보다 클때 %Q0.1.1 이 On
%QX0.1.1
행 16
13600
IN2
행 17
행 18
GE
EN ENO
행 19
13600
IN1 OUT
채널 4 의 디지털 값이 12000 보다 크거나 같고 13600 보다 작거나 같을 때%Q0.1.2 이 On
%QX0.1.2
행 20
DATA[4]
IN2
채널 4 의 A/D 변환값
행 21
12000
IN3
행 22
행 23
GE
EN ENO
행 24
DATA[4]
IN1 OUT
채널 4 의 A/D 변환값
채널 4 의 디지털 값이 12800 과같을 때%Q0.1.3 이 On
%QX0.1.3
행 25
12800
IN2
행 26

(2)G4F-AD4B 의 경우

가) 프로그램에서 사용된 입출력 변수

Variable Name		Var_Kind	Data Type	(AT Address) (Initial Value)
기동준비		: VAR	: BOOL	
운전채널		: VAR	: ARRAY [0..15] OF BOOL	
읽기		: VAR	: FB instance	
읽기_에러정보		: VAR	: USINT	
입력종류		: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
입력타입	※1	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
채널지정		: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
초기화		: VAR	: FB instance	
초기화_에러정보		: VAR	: USINT	
출발		: VAR	: BOOL	
평균값	※2	: VAR	: ARRAY [0..15] OF USINT	: = { 0,0,100,0,200,0,0,0,0,0,0,0,0,0,0,0 }
평균허가	※3	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0 }
필터값	※4	: VAR	: ARRAY [0..15] OF USINT	: = { 50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
필터허가	※5	: VAR	: ARRAY [0..15] OF BOOL	: = { 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
횟수시간	※6	: VAR	: ARRAY [0..15] OF BOOL	: = { 1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0 }
DATA		: VAR	: ARRAY [0..15] OF INT	

나) G3F-AD4A 와 G3F-AD4B 의 차이점

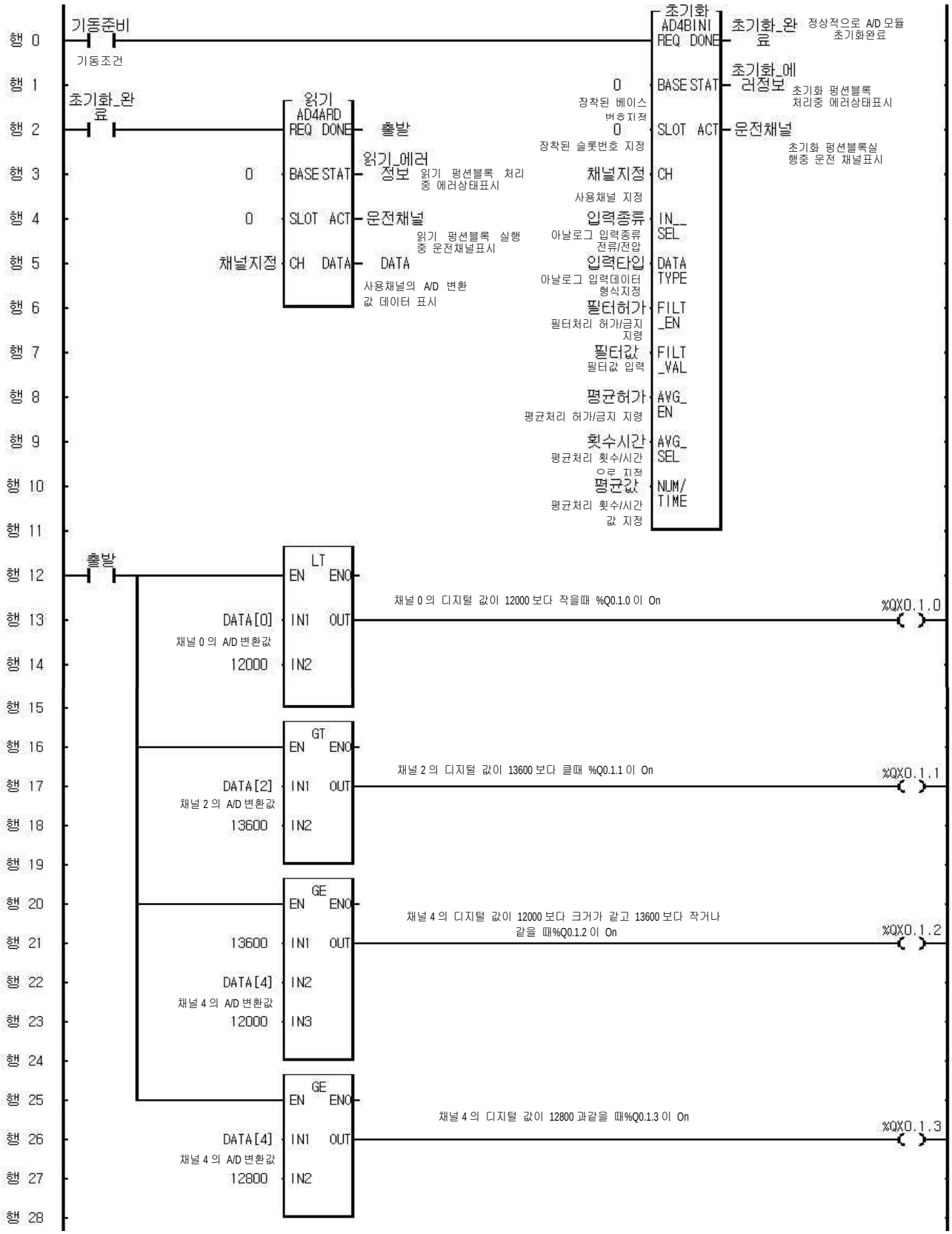
▶초기화 펑션블록의 입력 변수만 차이가 있습니다.

▶※1 은 G3F-AD4B 에서만 사용합니다.

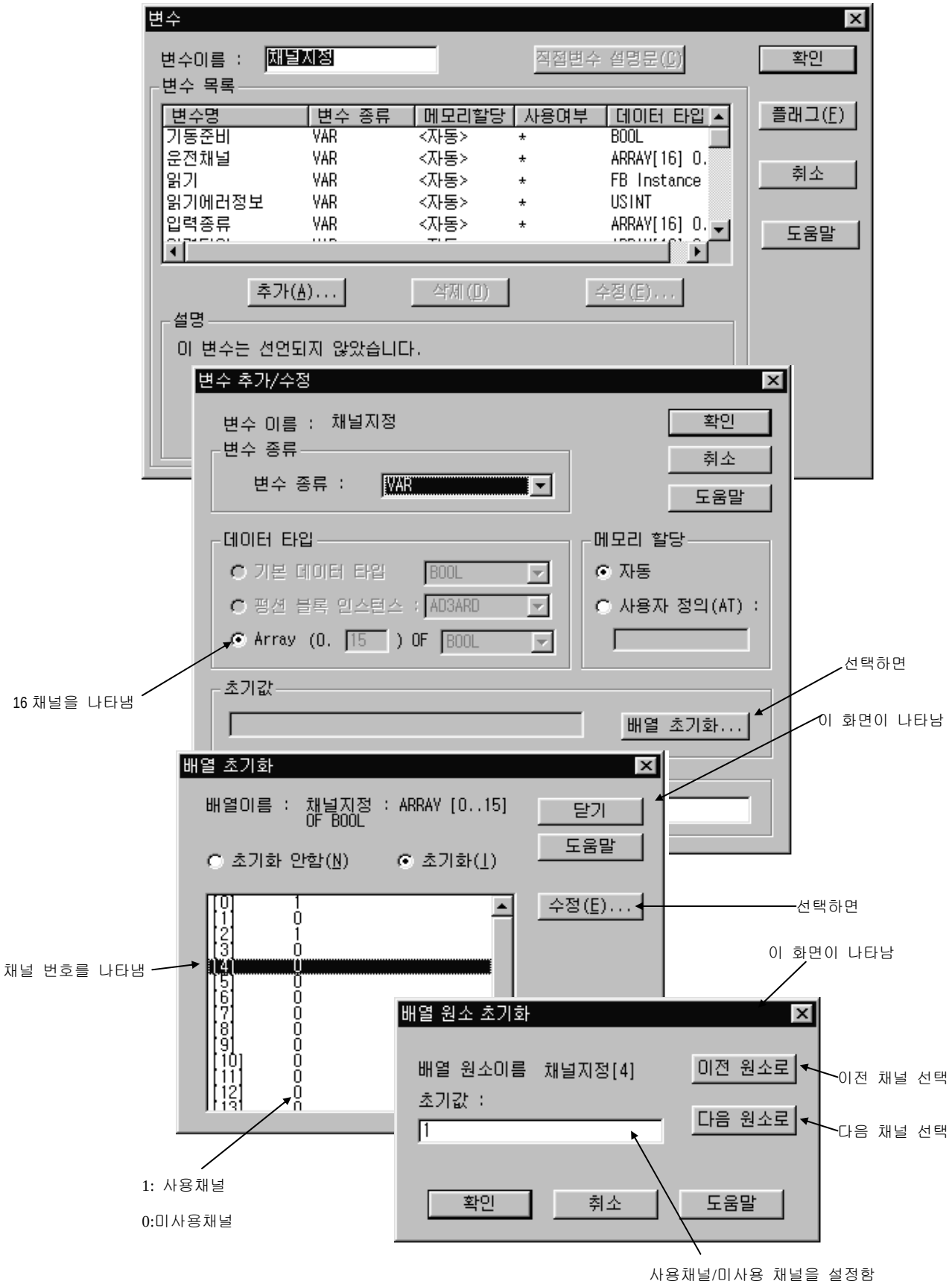
▶※2, ※3, ※4, ※5, ※6 은 필터와 평균 처리 기능으로
G3F-AD4A 에서는 4 채널을 묶어서 필터와 평균 기능을 처리하고
G3F-AD4B 에서는 채널마다 필터와 평균 기능을 처리합니다.

구분	F/B 입력 변수명	G3F-AD4A		G3F-AD4B		비고
		변수명	Data Type	변수명	Data Type	
※1	IN_SEL	-	-	입력타입	ARRAY [0..15] OF BOOL	G3F-AD4B 만
※2	FILT_EN	필터허가	ARRAY [0..3] OF BOOL	필터허가	ARRAY [0..15] OF BOOL	
※3	FILT_VAL	필터값	ARRAY [0..3] OF USINT	필터값	ARRAY [0..15] OF USINT	
※4	AVG_EN	평균허가	ARRAY [0..3] OF BOOL	평균허가	ARRAY [0..15] OF BOOL	
※5	AVG_SEL	횟수시간	ARRAY [0..3] OF BOOL	횟수시간	ARRAY [0..15] OF BOOL	
※6	NUM/TIME	평균값	ARRAY [0..3] OF USINT	평균값	ARRAY [0..15] OF USINT	

다) 프로그램예

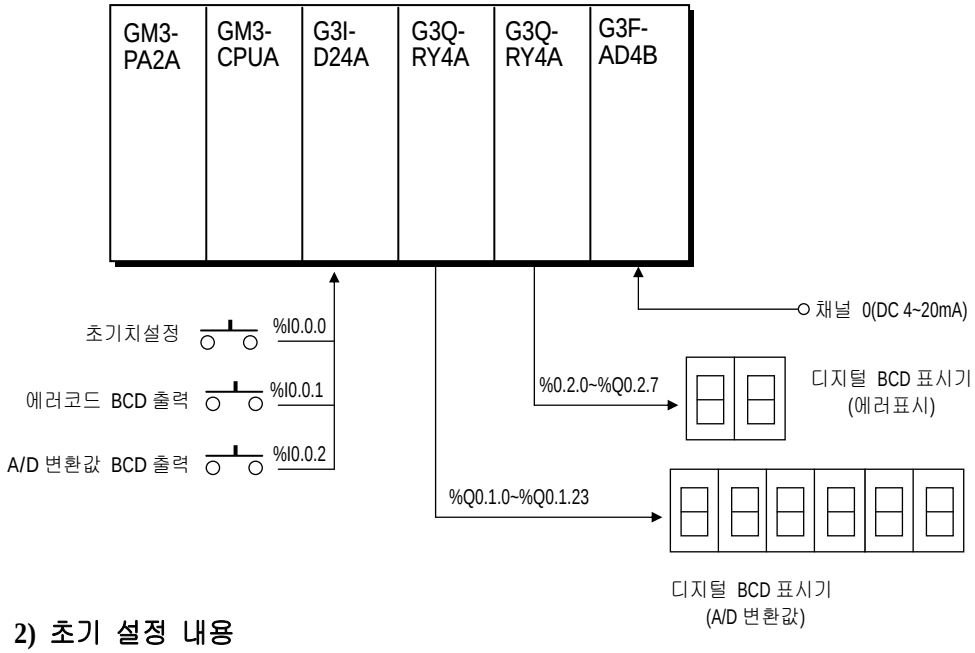

기동준비
기동조건
초기화_완료
읽기
AD4ARD
REQ DONE
출발
0
BASE STAT
읽기_에러정보
읽기 펑션블록 처리중 에러상태표시
0
SLOT ACT
운전채널
읽기 펑션블록 실행중 운전채널표시
채널지정
CH DATA
DATA
사용채널의 A/D 변환값 데이터 표시
초기화
AD4BINI
REQ DONE
초기화_완료
정상적으로 A/D모듈 초기화완료
0
장착된 베이스 번호지정
BASE STAT
초기화_에러정보
초기화 펑션블록 처리중 에러상태표시
0
장착된 슬롯번호 지정
SLOT ACT
운전채널
초기화 펑션블록실 행중 운전 채널표시
채널지정
사용채널 지정
CH
입력종류
아날로그 입력종류 전류/전압
IN_SEL
입력타입
아날로그 입력데이터 형식지정
DATA TYPE
필터허가
필터처리 허가/금지 지령
FILT_EN
필터값
필터값 입력
FILT_VAL
평균허가
평균처리 허가/금지 지령
AVG_EN
횟수시간
평균처리 횟수/시간으로 지정
AVG_SEL
평균값
평균처리 횟수/시간 값 지정
NUM/TIME
출발
LT
EN ENO
DATA[0]
채널 0 의 A/D 변환값
IN1 OUT
12000
IN2
채널 0 의 디지털 값이 12000 보다 작을때 %Q0.1.0 이 On
%QX0.1.0
GT
EN ENO
DATA[2]
채널 2 의 A/D 변환값
IN1 OUT
13600
IN2
채널 2 의 디지털 값이 13600 보다 클때 %Q0.1.1 이 On
%QX0.1.1
GE
EN ENO
13600
IN1 OUT
DATA[4]
채널 4 의 A/D 변환값
IN2
12000
IN3
채널 4 의 디지털 값이 12000 보다 크거나 같고 13600 보다 작거나 같을 때%Q0.1.2 이 On
%QX0.1.2
GE
EN ENO
DATA[4]
채널 4 의 A/D 변환값
IN1 OUT
12800
IN2
채널 4 의 디지털 값이 12800 과같을 때%Q0.1.3 이 On
%QX0.1.3

라) 입출력 변수 초기값 지정 방법 (채널 지정)



3.2 A/D 변환값 및 에러코드를 BCD 표시기로 출력하는 프로그램

1) 시스템 구성



2) 초기 설정 내용

(1) 사용채널 : 채널 0

(2) 아날로그 입력 : 전류 입력

(3) 필터 처리 지정 : 50

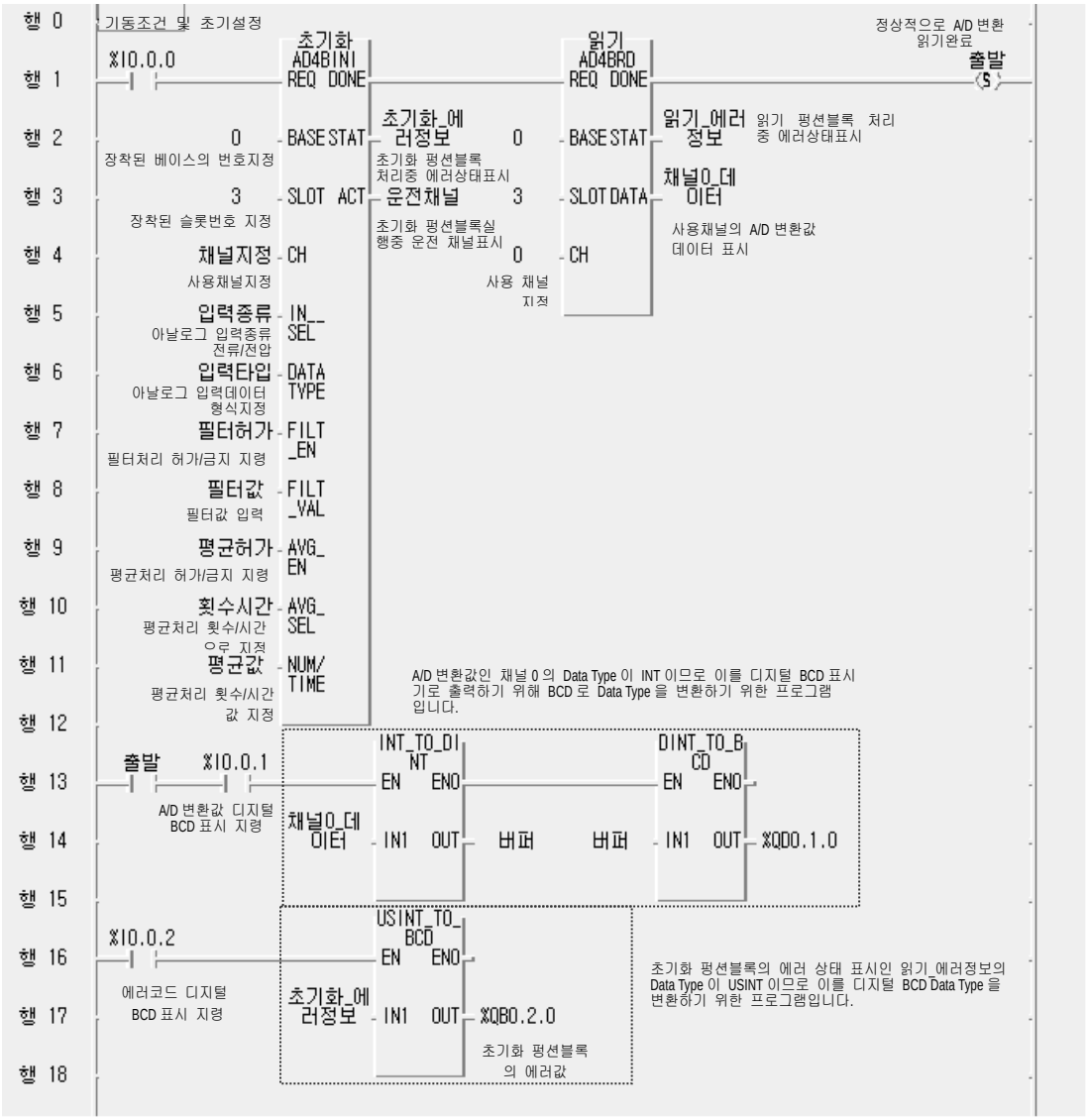
3) 프로그램 설명

(1) %I0.0.0 이 On 되면 A/D 변환 초기 설정을 한다.

(2) %I0.0.1 이 On 되면 A/D 변환값을 디지털 BCD 표시기에 출력한다.(%Q0.1.0~%Q0.1.23)

(3) %I0.0.2 가 On 되면 읽기 펑션블록의 에러코드를 디저털 BCD 표시기에 출력한다.
(%Q0.2.0~%Q0.2.7)

4) 프로그램

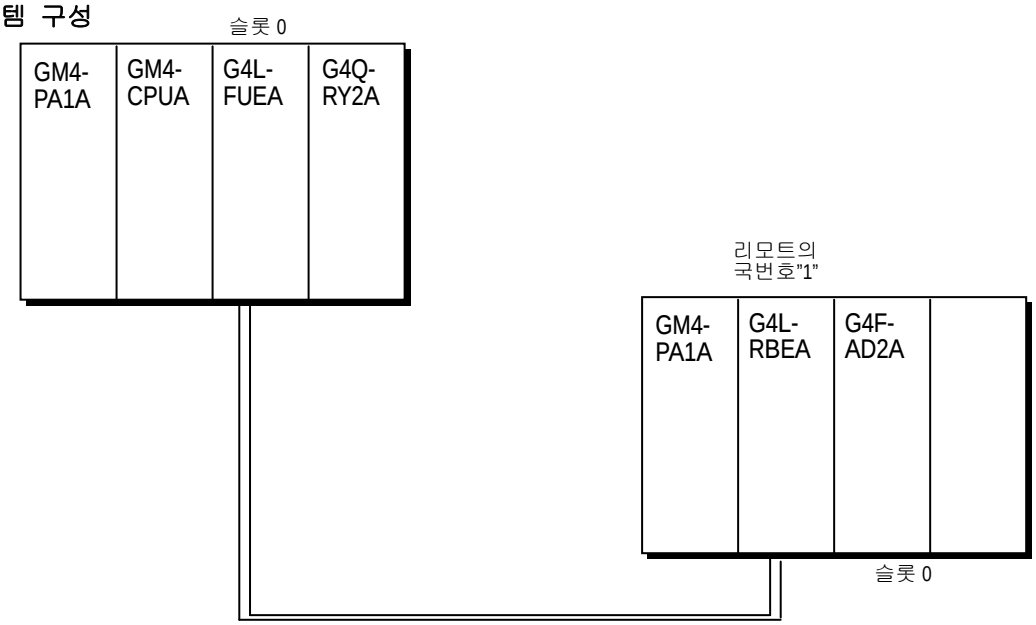


5) 프로그램에서 사용된 입출력 변수

Variable Name	Var_Kind	Data Type	(AT Address) (Initial Value)
버퍼	: VAR	: DINT	
운전채널	: VAR	: ARRAY [0..15] OF BOOL	
읽기	: VAR	: FB instance	
읽기_에러정보	: VAR	: USINT	
입력종류	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
입력타입	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
채널 0_데이터	: VAR	: INT	
채널지정	: VAR	: ARRAY [0..15] OF BOOL	: = { 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
초기화	: VAR	: FB instance	
초기화_에러정보	: VAR	: USINT	
출발	: VAR	: BOOL	
평균값	: VAR	: ARRAY [0..15] OF USINT	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
평균허가	: VAR	: ARRAY [0..15] OF BOOL	: = { 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
필터값	: VAR	: ARRAY [0..15] OF USINT	: = { 50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
필터허가	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
횟수시간	: VAR	: ARRAY [0..15] OF BOOL	: = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }

3.3 리모트 I/O 국에 A/D 변환 모듈을 장착할 때의 프로그램

1) 시스템 구성



2) 초기 설정 내용

(1) A/D 변환 허가 채널 :채널 0
(2) 변환 데이터의 범위 : -192 ~16191
(3) 시간에 의한 평균 처리 채널 : 채널 0 (설정 값: 100 ㎳)

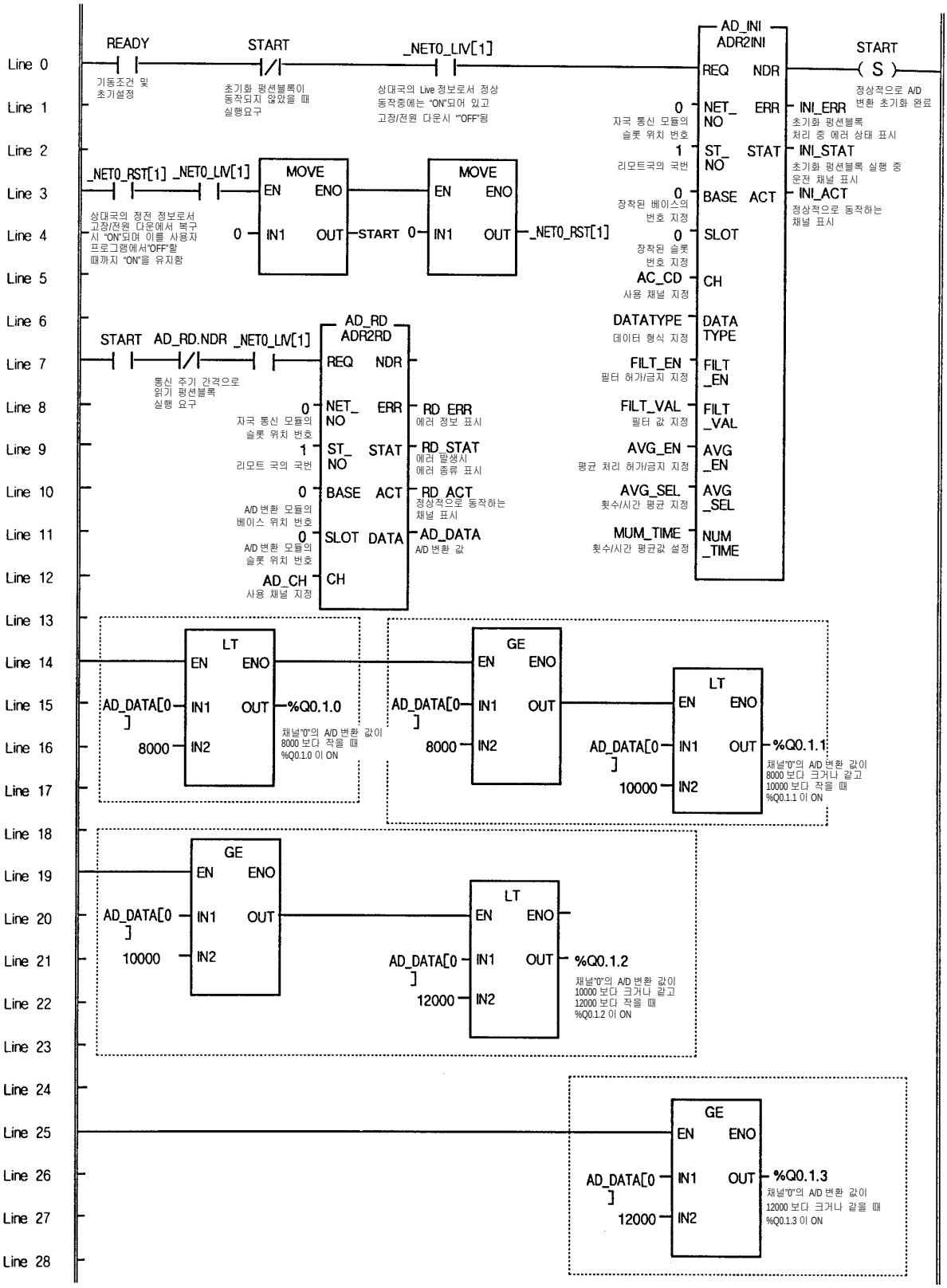
3) 프로그램 설명

(1) 채널 0 의 A/D 변환된 디지털 값이 8,000 보다 작을 때 %Q0.1.0 이 On
(2) 채널 0 의 A/D 변환된 디지털 갓이 8,000 보다 크거나 같고 10,000 보다 작을 때 %Q0.1.1 이 On
(3) 채널 0 의 A/D 변환된 디지털 값이 10,000 보다 크거나 같고 12,000 보다 작을 때 %Q0.1.2 이 On
(4) 채널 0 의 A/D 변환된 디지털 값이 12,000 보다 크거나 같을 때 %Q0.1.3 이 On

4) 프로그램에 사용된 입출력 변수

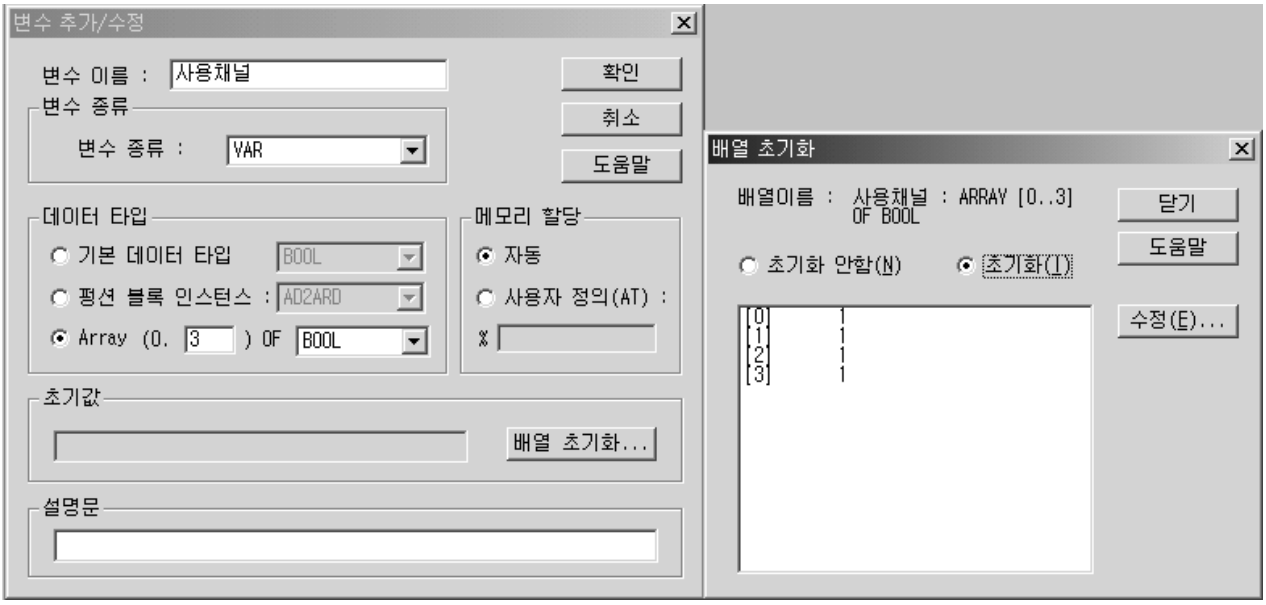
Variable Name	Var Kind	Data Type	(AT Address) (Initial Value)
AD_CH	: VAR	: ARRAY [0..3] OF BOOL	: = {0,0,0,0}
AD_DATA	: VAR	: ARRAY [0..3] OF INT	
AD_INI	: VAR	: FB Instance	
AD_RD	: VAR	: FB Instance	
AVG_EN	: VAR	: ARRAY [0..3] OF BOOL	: = {1,0,0,0}
AVG_SEL	: VAR	: ARRAY [0..3] OF BOOL	: = {1,0,0,0}
DATATYPE	: VAR	:ARRAY [0..3] OF BOOL	: = {0,0,0,0}
FILT_EN	: VAR	: ARRAY [0..3] OF BOOL	: = {0,0,0,0}
FILT_VAL	: VAR	: ARRAY [0..3] OF USINT	: = {0,0,0,0}
INI_ACT	: VAR	: ARRAY [0..3] OF BOOL	
INI_ERR	: VAR	: BOOL	
INI_STAT	: VAR	: USINT	
NUM_TIME	: VAR	: ARRAY [0..3] OF UINT	: = {1000,0,0,0}
RD_STAT	: VAR	: ARRAY [0..3] OF BOOL	
RD_ERR	: VAR	: BOOL	
RD_STAT	: VAR	: USINT	
READY	: VAR	: BOOL	

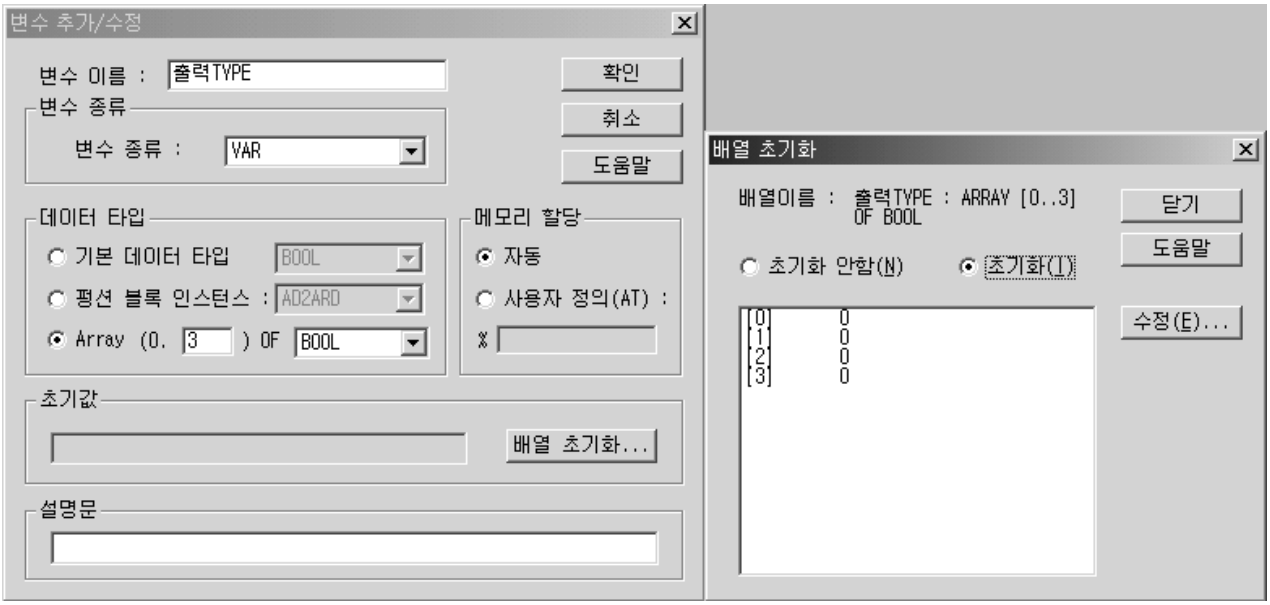
5) 프로그램


Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
Line 12
Line 13
Line 14
Line 15
Line 16
Line 17
Line 18
Line 19
Line 20
Line 21
Line 22
Line 23
Line 24
Line 25
Line 26
Line 27
Line 28
READY
기동조건 및 초기설정
START
초기화 평션블록이 동작되지 않았을 때 실행요구
_NET0_LIV[1]
상대국의 Live 정보로서 정상 동작중에는 "ON"되어 있고 고장/전원 다운시 "OFF"됨
AD_INI
ADR2INI
REQ
NDR
START
(S)
정상적으로 A/D 변환 초기화 완료
0
자국 통신 모듈의 슬롯 위치 번호
NET_ NO
ERR
INI_ERR
초기화 평션블록 처리 중 에러 상태 표시
1
리모트국의 국번
ST_ NO
STAT
INI_STAT
초기화 평션블록 실행 중 운전 채널 표시
0
장착된 베이스의 번호 지정
BASE
ACT
INI_ACT
정상적으로 동작하는 채널 표시
0
장착된 슬롯 번호 지정
SLOT
AC_CD
사용 채널 지정
CH
DATATYPE
데이터 형식 지정
DATA TYPE
FILT_EN
필터 허가/금지 지정
FILT _EN
FILT_VAL
필터 값 지정
FILT _VAL
AVG_EN
평균 처리 허가/금지 지정
AVG _EN
AVG_SEL
횟수/시간 평균 지정
AVG _SEL
MUM_TIME
횟수/시간 평균값 설정
NUM _TIME
_NET0_RST[1]
_NET0_LIV[1]
MOVE
EN
ENO
IN1
OUT
0
START
상대국의 정전 정보로서 고장/전원 다운에서 복구시 "ON"되며 이를 사용자 프로그램에서"OFF"할 때까지 "ON"을 유지함
AD_RD
ADR2RD
START
AD_RD.NDR
_NET0_LIV[1]
통신 주기 간격으로 읽기 평션블록 실행 요구
RD_ERR
에러 정보 표시
RD_STAT
에러 발생시 에러 종류 표시
RD_ACT
정상적으로 동작하는 채널 표시
AD_DATA
A/D 변환 값
AD_CH
사용 채널 지정
A/D 변환 모듈의 베이스 위치 번호
A/D 변환 모듈의 슬롯 위치 번호
LT
GE
AD_DATA[0]
8000
10000
12000
%Q0.1.0
%Q0.1.1
%Q0.1.2
%Q0.1.3
채널"0"의 A/D 변환 값이 8000 보다 작을 때 %Q0.1.0 이 ON
채널"0"의 A/D 변환 값이 8000 보다 크거나 같고 10000 보다 작을 때 %Q0.1.1 이 ON
채널"0"의 A/D 변환 값이 10000 보다 크거나 같고 12000 보다 작을 때 %Q0.1.2 이 ON
채널"0"의 A/D 변환 값이 12000 보다 크거나 같을 때 %Q0.1.3 이 ON

3.4 예제 프로그램 실습

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	사용채널	ARRAY[4] OF BOOL	<자동>	설정	VAR	*	
2	출력TYPE	ARRAY[4] OF BOOL	<자동>	설정	VAR	*	
3	평균방법	ARRAY[4] OF BOOL	<자동>	설정	VAR	*	
4	평균선택	ARRAY[4] OF BOOL	<자동>	설정	VAR	*	
5	필터상수	ARRAY[4] OF USINT	<자동>	설정	VAR	*	
6	필터선택	ARRAY[4] OF BOOL	<자동>	설정	VAR	*	
7	횟수시간	ARRAY[4] OF UINT	<자동>	설정	VAR	*	
8	표시기	WORD	%QW0.1.1		VAR	*	
9	개별읽기	FB Instance	<자동>		VAR	*	
10	변환값	ARRAY[4] OF INT	<자동>		VAR	*	
11	전체읽기	FB Instance	<자동>		VAR	*	
12	AD초기화	FB Instance	<자동>		VAR	*	





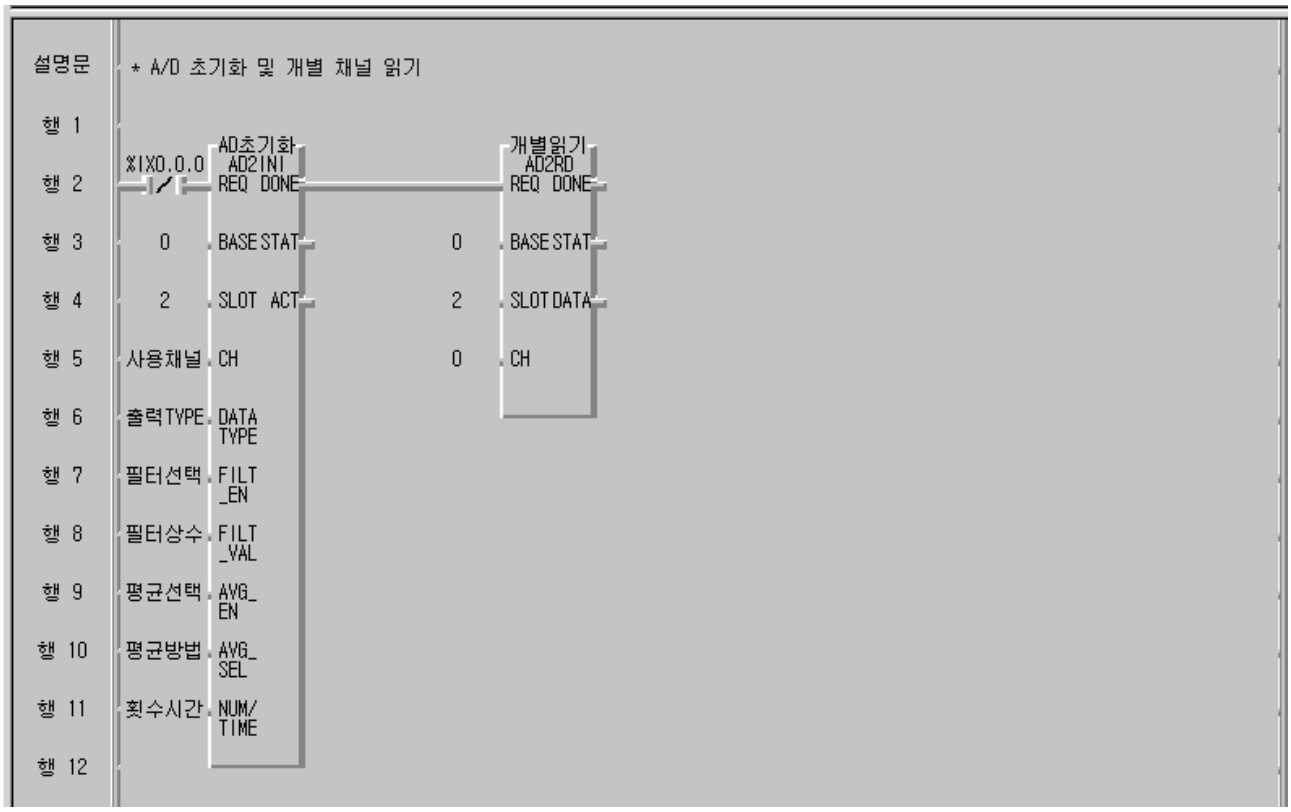

변수 추가/수정
변수 이름 : 평균방법
확인
취소
도움말
변수 종류
변수 종류 : VAR
데이터 타입
기본 데이터 타입 BOOL
펑션 블록 인스턴스 : AD2ARD
Array (0. 3) OF BOOL
메모리 할당
자동
사용자 정의(AT) :
%
초기값
배열 초기화...
설명문
배열 초기화
배열이름 : 평균방법 : ARRAY [0..3] OF BOOL
초기화 안함(N)
초기화(I)
[0] 0
[1] 0
[2] 0
[3] 0
닫기
도움말
수정(E)...

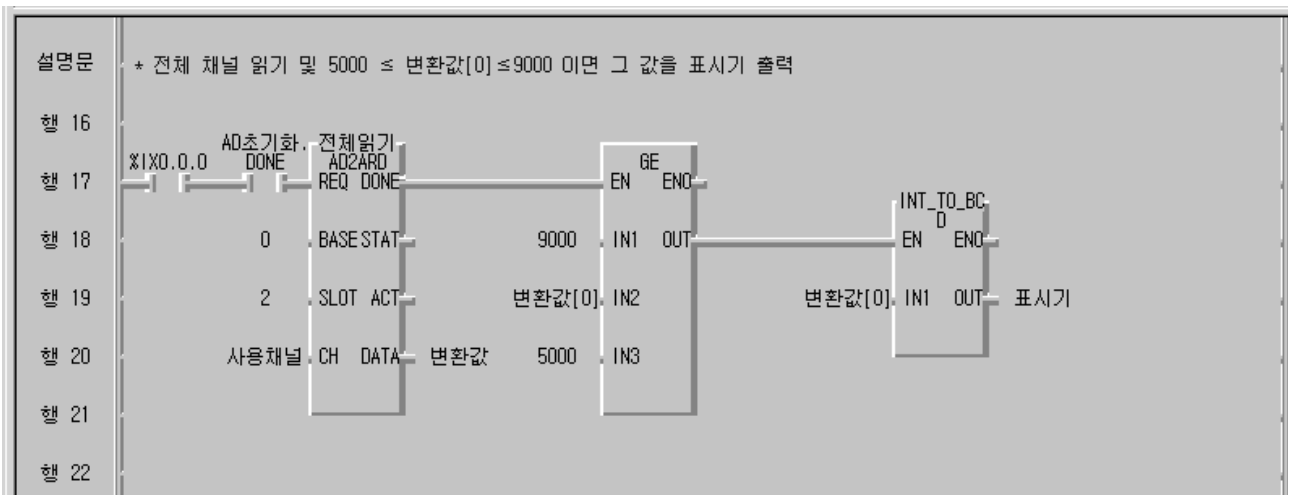

변수 추가/수정
변수 이름 : 평균선택
확인
취소
도움말
변수 종류
변수 종류 : VAR
데이터 타입
기본 데이터 타입 BOOL
펑션 블록 인스턴스 : AD2ARD
Array (0. 3) OF BOOL
메모리 할당
자동
사용자 정의(AT) :
%
초기값
배열 초기화...
설명문
배열 초기화
배열이름 : 평균선택 : ARRAY [0..3] OF BOOL
초기화 안함(N)
초기화(I)
[0] 1
[1] 1
[2] 1
[3] 1
닫기
도움말
수정(E)...


변수 추가/수정
변수 이름 : 필터상수
확인
취소
도움말
변수 종류
변수 종류 : VAR
데이터 타입
기본 데이터 타입 BOOL
펑션 블록 인스턴스 : AD2ARD
Array (0. 3) OF USINT
메모리 할당
자동
사용자 정의(AT) :
%
초기값
배열 초기화...
설명문
배열 초기화
배열이름 : 필터상수 : ARRAY [0..3] OF USINT
초기화 안함(N)
초기화(I)
[0] 10
[1] 10
[2] 10
[3] 10
닫기
도움말
수정(E)...


변수 추가/수정
변수 이름 : 필터선택
확인
취소
도움말
변수 종류
변수 종류 : VAR
데이터 타입
기본 데이터 타입 BOOL
평션 블록 인스턴스 : AD2ARD
Array (0. 3) OF BOOL
메모리 할당
자동
사용자 정의(AT) :
%
초기값
배열 초기화...
설명문
배열 초기화
배열이름 : 필터선택 : ARRAY [0..3] OF BOOL
초기화 안함(N)
초기화(I)
닫기
도움말
수정(E)...
[0] 1
[1] 1
[2] 1
[3] 1


변수 추가/수정
변수 이름 : 횟수시간
확인
취소
도움말
변수 종류
변수 종류 : VAR
데이터 타입
기본 데이터 타입 BOOL
평션 블록 인스턴스 : AD2ARD
Array (0. 3) OF UINT
메모리 할당
자동
사용자 정의(AT) :
%
초기값
배열 초기화...
설명문
배열 초기화
배열이름 : 횟수시간 : ARRAY [0..3] OF UINT
초기화 안함(N)
초기화(I)
닫기
도움말
수정(E)...
[0] 100
[1] 100
[2] 100
[3] 100


설명문
* A/D 초기화 및 개별 채널 읽기
행 1
행 2
행 3
행 4
행 5
행 6
행 7
행 8
행 9
행 10
행 11
행 12
%IX0.0.0
AD초기화
AD2INI
REQ DONE
BASE STAT
SLOT ACT
CH
DATA TYPE
FILT _EN
FILT _VAL
AVG_ EN
AVG_ SEL
NUM/ TIME
사용채널
출력TYPE
필터선택
필터상수
평균선택
평균방법
횟수시간
개별읽기
AD2RD
REQ DONE
BASE STAT
SLOT DATA
CH
0
2
0
0


설명문
* 전체 채널 읽기 및 5000 ≤ 변환값[0]≤9000 이면 그 값을 표시기 출력
행 16
행 17
행 18
행 19
행 20
행 21
행 22
%IX0.0.0
AD초기화
DONE
전체읽기
AD2ARD
REQ DONE
BASE STAT
SLOT ACT
CH DATA
사용채널
변환값
GE
EN ENO
IN1 OUT
IN2
IN3
9000
변환값[0]
5000
INT_TO_BCD
EN ENO
IN1 OUT
변환값[0]
표시기
0
2

모듈Ⅲ. 아날로그 출력(DA)

- 적 용 모 델 -

제 1 장 성능규격 및 변환특성

본 제품은 GLOFA PLC GMR/1/23/4/6시리즈 및 MASTER-K200S/300S/1000S의 CPU와 조합하여 사용하는 디지털/아날로그 변환 모듈로 전압 출력형, 전류 출력형, 전압전류 공통 출력형 (이하 D/A 변환 모듈)이 있습니다.

1.1 성능 규격

1) G3F-DA4V/I, G4F-DA1A

항 목		규 격		
		G3F–DA4I	G3F–DA4V	G4F–DA1A
적용 PLC		GM3/MK1000S		GM4/MK300S
입출력 점유점수		16점		
디 지 털 입 력		• 부호있는 16비트 바이너리값 (데이터:14비트) • 입력 데이터 지정에 따라 채널마다 설정가능 (“0” : –192 ~ 16191, “1” : – 8192 ~ – 8191)		
아 날 로 그 출력		DC 4 ~ 20㎃ (외부 부하저항 510Ω이하)	DC –5 ~ 5V (외부 부하저항 2KΩ~1MΩ) DC –10 ~ 10V (외부 부하저항 2KΩ~1MΩ) *전압 출력 종류는 제품 옆면의 스위치로 선택	DC –5 ~ 5V (외부 부하저항 2KΩ~1MΩ) DC 4 ~ 20mA (외부 부하저항 550Ω이하) *전압/전류 구분은 단자대에 의해 선택됩니다.
최 대 분 해 능		1 μA (1/16000)	DC –5 ~ 5V : 0.625mV(1/16000) DC –10 ~ 10V : 1.25mV(1/16000)	DC –10 ~ 10V : 1.25mV(1/16000) DC 4 ~ 20mA : 1.0μA(1/16000)
정 밀 도		±0.3% [풀 스케일(Full Scale)]		
최 대 변 환 속 도		15ms / 16채널		3ms / 2채널
절 대 최 대 출 력		DC+24mA	DC+15V	전압 : DC+15V 전류 : DC+24mA
아날로그 출력 점수		16채널/1모듈		2채널/1모듈
절 연 방 식		출력 단자와 PLC전원간 포토 – 커플러 절연 (채널간 비절연)		
접 속 단 자		38점 단자대		20점 단자대
내 부 소 비 전 류		0.25A		0.45A
외부 공급 전원	전압	DC+15V / DC–15V		–
	전류	DC+15V:0.5A DC–15V:0.1A	DC+15V:0.5A DC–15V:0.3A	
중 량		610g	630g	370g

2) G3F-DA3V/I, G4F-DA4V/I, G4F-DA2V/I

항 목		규 격					
		전압출력형			전류출력형		
		G3F-DA3V	G4F-DA3V	G4F-DA2V	G3F-DA3I	G4F-DA3I	G4F-DA2I
적용 PLC		GM3/ MK1000S	GM4/MK300S		GM3/ MK1000S	GM4/MK300S	
입출력 점유점수		16점					
디지털 입력		• 부호있는 16비트 바이너리 값(데이터 : 12비트)					
아날로그 출력		DC0 ~ DC10V (외부부하저항 2kΩ ~ 1MΩ)	:DC-10V ~ 10V (외부 부하저항 2kΩ ~ 1MΩ)		DC 4mA ~ 12mA (외부 부하저항 510Ω이하)		
최대분해능		2.5mV(1/4000)	5mV(1/4000)		4μA(1/4000)		
정밀도		±0.5%[풀 스케일(Full Scale)]					
최대변환속도		15ms/8채널		10ms/4채널	15ms/8채널		10ms/4채널
절대최대출력		DC +15V			DC +24mA		
아날로그 출력 점수		8채널		4채널	8채널		4채널
절연방식		입력단자와 PLC전원간 포토 커플러 절연 (채널간 비절연)					
접속단자		20점 단자대					
내부소비전류(DC5V)		550mA	700mA	400mA	60mA		680mA
외부공급전원	전압				DC21.6~26.4V		
	소비전류				230mA		
중 량		390g	280g	260g	410g	280g	260g

알아두기

GM3-PA1A와 GM3-PA2A의 출력용량은 DC+5V : 7A, DC+24V : 1.5A입니다.
GM4-PA1A와 GM4-PA2A의 출력용량은 DC5V:4A, DC24V:0.7A입니다.
GM4-PA1B와 GM4-PA2B의 출력용량은 DC5V:3A, DC24V:0.5A입니다.

3) G6F-DA2V/I

항 목		규 격	
		G6F-DA2V	G6F-DA2I
적용 PLC		GM6/MK200S	
입출력 점유점수		16점	
디지털 입력		• 부호있는 16비트 바이너리 값(데이터 : 12비트)	
아날로그 출력		전압출력 : DC -10V ~ 10V (외부 부하저항 2kΩ ~ 1MΩ)	전류출력 : DC 4mA ~ 20mA (외부 부하저항 510Ω이하)
최대분해능		5mV(1/4000)	4μA(1/4000)
정밀도		±0.5%[풀 스케일(Full Scale)]	
최대변환속도		10ms/4채널	10ms/4채널
절대최대출력		DC +15V	DC +24mA
아날로그 출력 점수		4채널	4채널
절연방식		입력단자와 PLC전원간 포토 커플러 절연 (채널간 비절연)	
접속단자		18점 단자대	
내부소비전류 *1	DC+5V	40mA	40mA
	DC+15V	80mA	120mA
	DC-15V	60mA	25mA
중 량		200g	200g

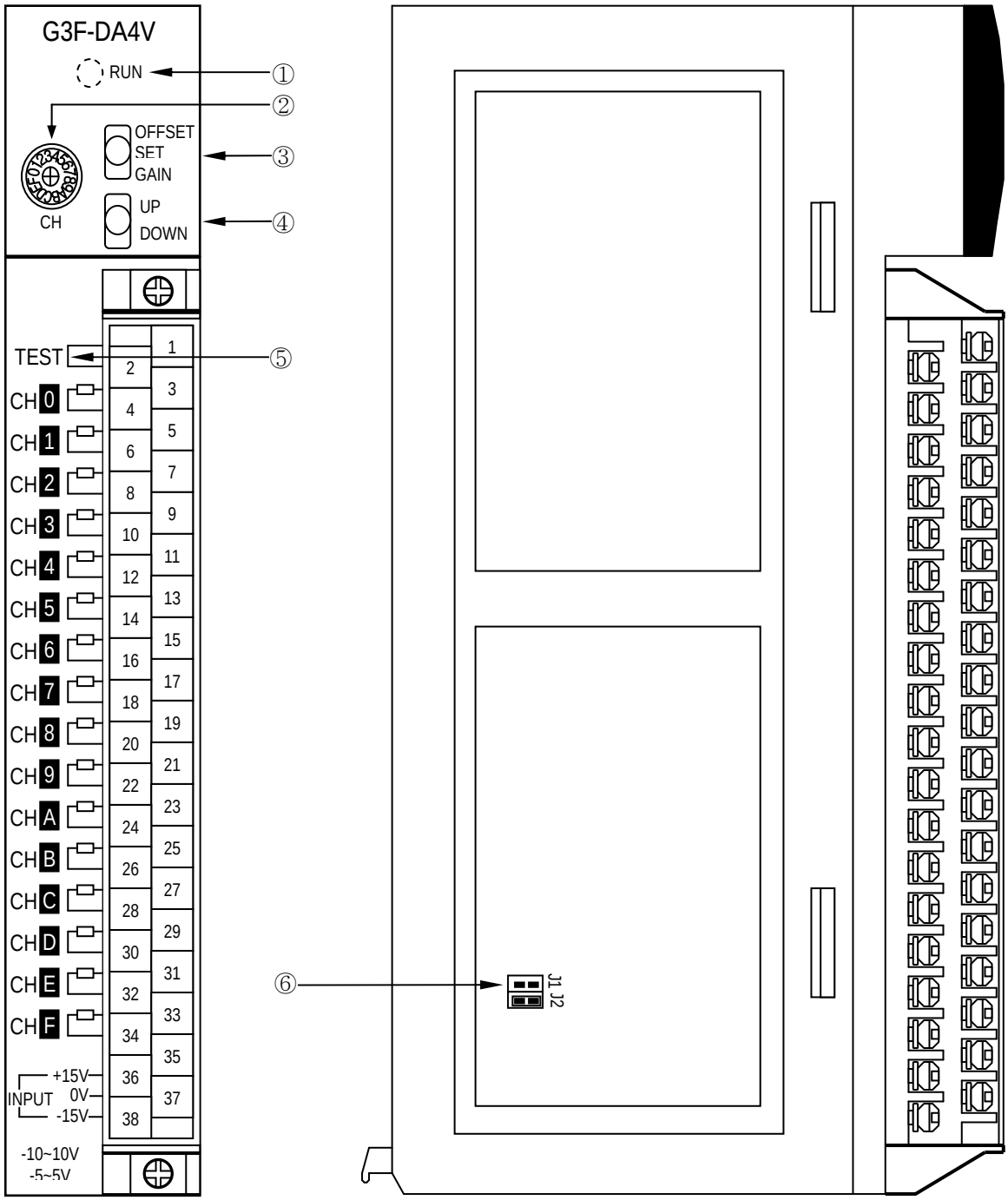
알아두기

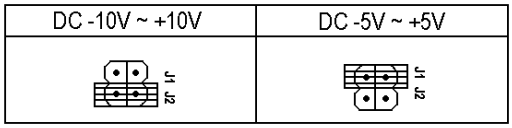
*1 GM6또는 K200S에서 아날로그 입출력모듈을 사용시 전원 모듈은 DC±15V출력이 있는 GM6-PAFB를 사용하십시오.
GM6-PAFB 는 DC+15V:0.5A, DC-15V:0.2A 입니다.
따라서 D/A변환 모듈을 동시에 여러대 사용할 경우 각 제품의 소비전류를 고려하여 소비전류가 규격치를 만족하도록 설정해 주십시오.

1.2 각 부의 명칭과 역할

1) G3F–DA4I / G3F–DA4V

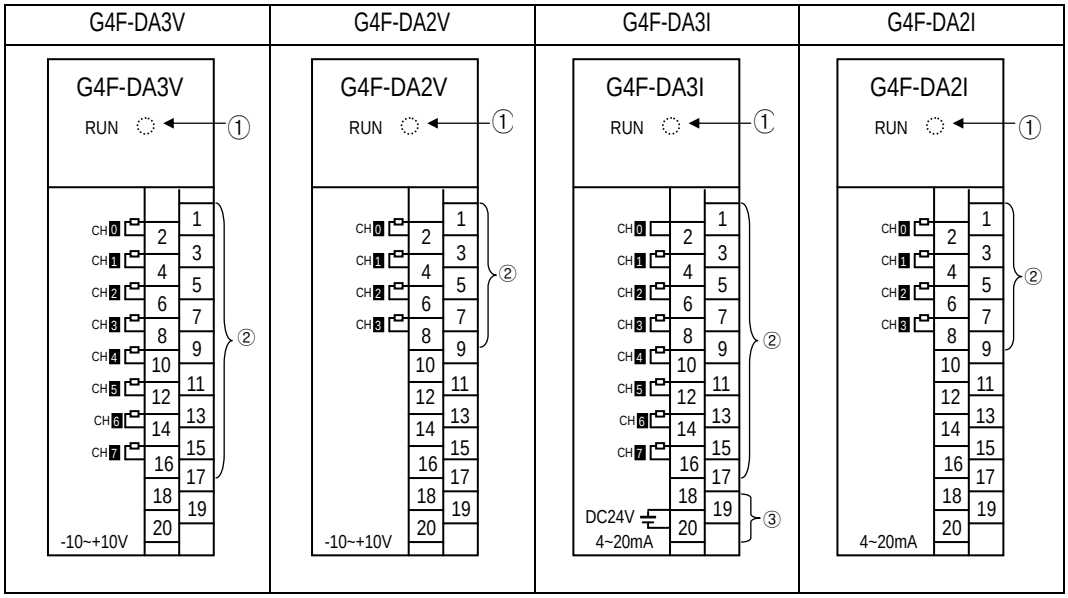
각 부분의 명칭에 대해서 설명합니다

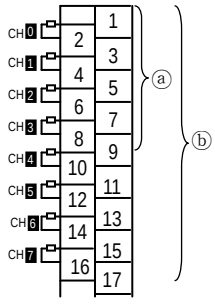
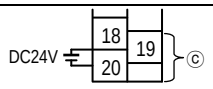


번호	내 용
①	RUN LED D/A 변환 모듈의 동작 상태를 표시 • 점 등 : 정상 동작 중 • 점 멸 : 에러 발생 (자세한 사항은 6.1항 참조) • 소 등 : DC 5V 단선, D/A 변환 모듈 이상
②	채널 선택 스위치 테스트 모드에서 오프셋/게인을 조정하고자 할 때 채널을 선택하는 스위치 설정 범위 : 0 ~ F (0을 선택하면 0번 채널의 오프셋/게인이 조정됨)
③	오프셋/게인 선택 스위치 오프셋/세트/게인을 선택하는 스위치 • 오프셋 위치 : 오프셋 값 조정 모드 • 게인 위치 : 게인 값 조정 모드 • 세트 위치 : 오프셋/게인 값 기억 모드 (오프셋/게인의 위치에서 세트 위치로 전환할 때 오프셋/게인 값이 D/A 변환 모듈의 내부 메모리에 기억됩니다.)
④	업/다운 스위치 – 채널 선택 스위치와 오프셋/게인 선택 스위치로 지정된 채널의 오프셋/게인 값을 변경하는 스위치 – 업/다운 위치에 따라 아날로그 출력이 다음과 같이 변경됩니다. • 2초 미만 업/다운 위치 : G3F–DA4I → 1회에 1μA 증감 G3F–DA4V → 1회에 1.25mV 증감 • 2초 이상 업/다운 위치 : G3F–DA4I → 0.2초마다 10μA 증감 G3F–DA4V → 0.2초마다 12.5mV 증감
⑤	테스트 단자 단자대 1,2를 연결하면 테스트 모드 단자대 1,2를 연결하지 않으면 노멀 모드
⑥	출력 범위 선택 스위치 G3F–DA4V만 해당됩니다. DC -10V ~ +10V / DC -5V ~ +5V  * 공장 출하시는 DC –10 ~ +10V로 선택되어져 있습니다.

2) G4F-DA3V/G4F-DA2V/G4F-DA3I/G4F-DA2I

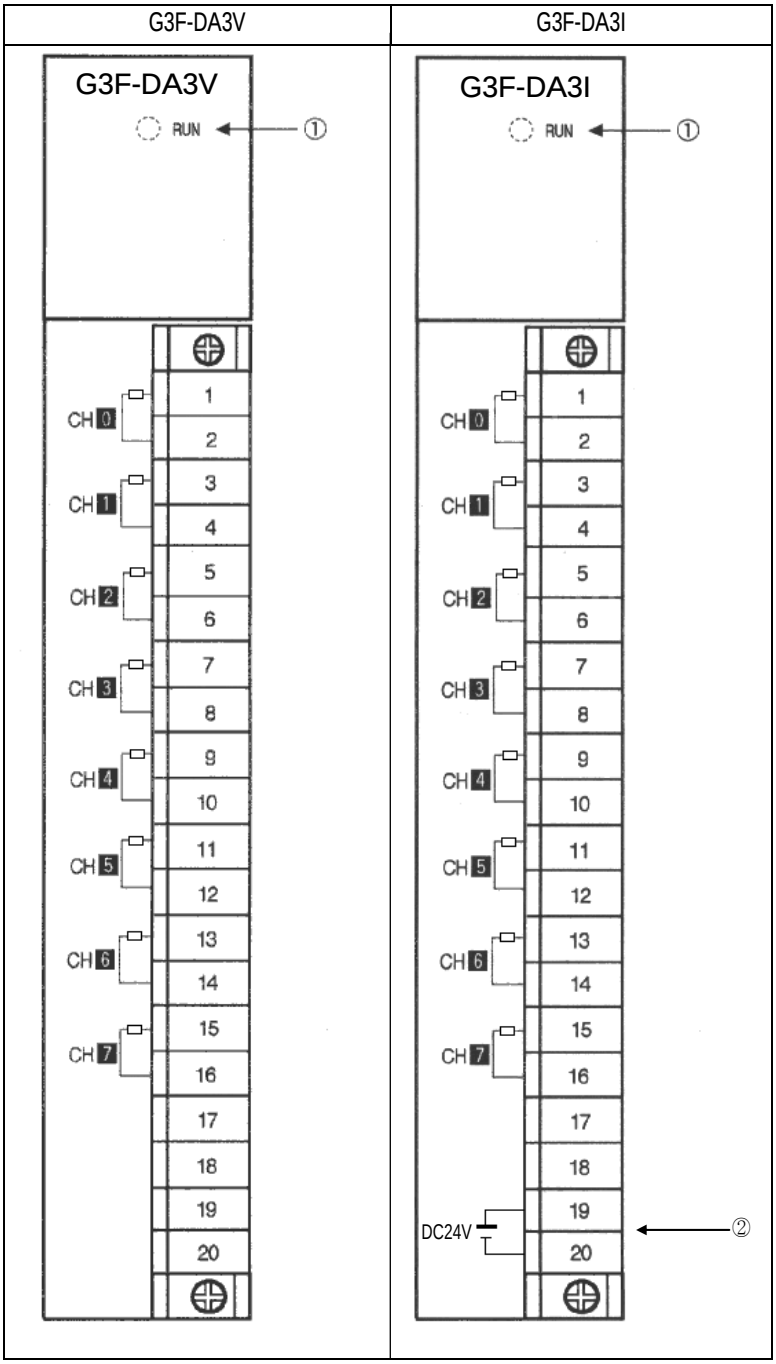
각 부분의 명칭에 대해서 설명합니다.



번호	내 용	
①	RUN LED	D/A 변환 모듈의 동작 상태를 표시 점등 : 정상 동작 중 소등 : DC5V 단선
②	아날로그 출력 단자대 	디지털 값을 아날로그값으로 변환하여 각 채널마다 외부와 연결할 수 있도록 되어 있는 단자대 ▶ ⓐ부 : G4F-DA2V 와 G4F-DA2I의 4채널(번호1 ~ 8) ▶ ⓑ부 : G4F-DA3V 와 G4F-DA3I의 8채널(번호1 ~ 16)
③	외부 입력 단자대 	G4F-DA3I만 해당 ▶ ⓒ부 : 외부 전원 DC24V 공급단자 (번호19 ~ 20)

3) G3F-DA3V/G3F-DA3I

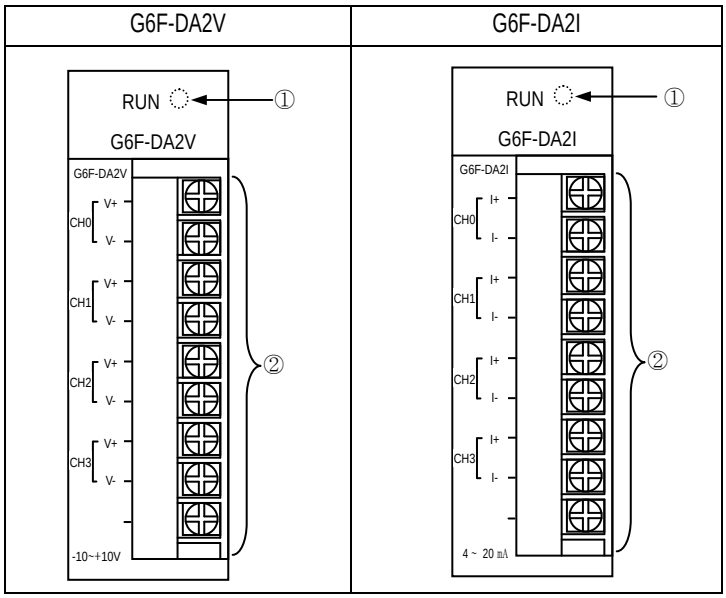
각 부분의 명칭에 대해서 설명합니다

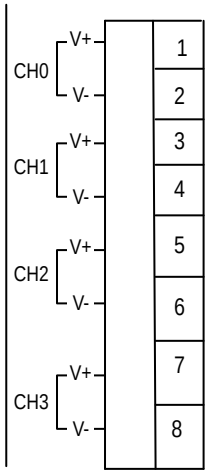


번호	내 용	
①	RUN LED	D/A 변환 모듈의 동작 상태를 표시 점등 : 정상 동작 중 소등 : DC5V 단선
②	외부 입력 단자대	G3F-DA3I만 해당 외부 전원 DC24V 공급단자 (번호19 ~ 20)

4) G6F-DA2V/G6F-DA2I

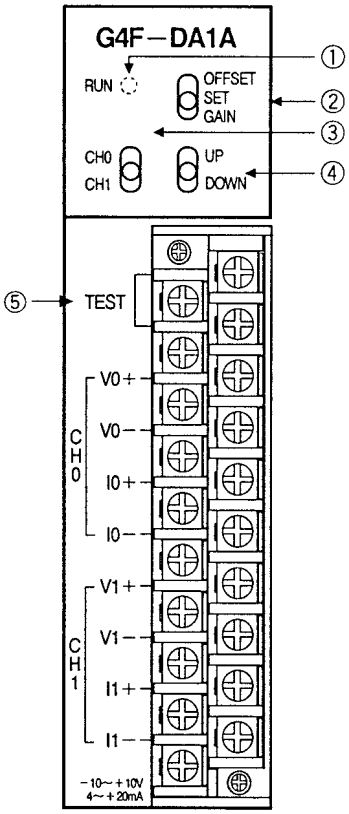
각 부분의 명칭에 대해서 설명합니다.



No	내 용	
①	RUN LED	D/A 변환 모듈의 동작 상태를 표시 점등:정상 동작중 소등:DC5V 단선
②	아날로그 출력 단자대 	디지털값을 아날로그 값으로 변환하여 각 채널마다 외부와 연결하는 단자대

5) G4F–DA1A

각 부분의 명칭에 대해서 설명합니다.

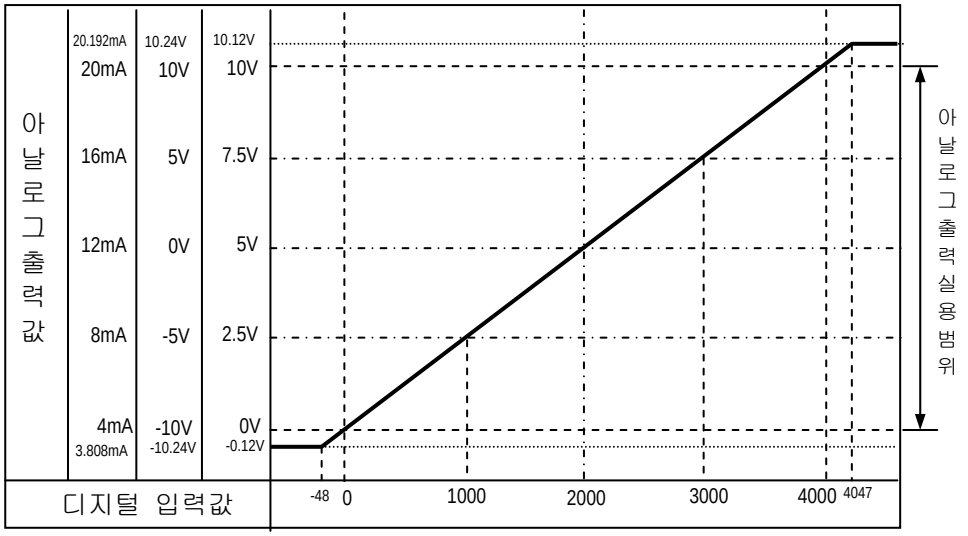


번호	내 용
①	RUN LED 운전 상태를 표시합니다. • 노멀 모드 – 점등 : 정상 동작중 – 점멸 : 에러 발생 – 소등 : DC 5V 단선 또는 G4F–DA1A 이상 • 테스트(Test)모드 – 점멸 : 오프셋/게인 선택 스위치가 오프셋 또는 게인의 위치일 때 1.0초 간격으로 점멸 – 소등 : 오프셋/게인 선택 스위치가 세트의 위치 일 때 소등
②	오프셋/게인 선택 스위치 • 오프셋 위치 : 오프셋 값 조정 모드 • 게인 위치 : 게인 값 조정 모드 • 세트 위치 : 오프셋/게인 값 기억 모드 (오프셋/게인의 위치에서 세트 위치로 전환할 때 오프셋/게인값이 G4F-DA1A의 내부 메모리에 기억됩니다.)
③	채널 선택 스위치 테스트 모드에서 오프셋/게인값 조정 채널 지정 • CHO 위치 : CHO 오프셋/게인 설정 가능 • 중립 : 오프셋/게인 설정 불가 • CH1 위치 : CH1 오프셋/게인 설정 가능
④	업/다운 스위치 – 지정 채널의 오프셋/게인 값을 변경하는 스위치 – 업/다운 위치에 따라 아날로그 출력이 다음과 같이 변경됩니다. (1) 2초 미만 업/다운 위치 : 1회에 전압 1.25mA 증감 전류 1.0μA 증감 (2) 2초 이상 업/다운 위치 : 0.2초마다 전압 12.5mA 증감 전류 10μA 증감
⑤	테스트 단자 단자대 1,3를 연결하면 테스트 모드 단자대 1,3를 연결하지 않을 때는 노멀 모드

제 2 장 입출력 변환특성

2.1 G3F-DA3V/I, G4F-DA3V/I, G4F-DA2V/I, G6F-DA2V/I

▶입출력 변환 특성은 PLC에서 설정된 디지털 신호를 아날로그 신호(전압 또는 전류)로 변환할 때의 기울기로 그림 2.1로 나타냅니다.



[그림 2.1] 입출력 변환 특성 예

▶ D/A 변환 모듈의 오프셋/게인 설정은 고정되어 있으므로 변경할 수 없습니다.

▶ G3F-DA3V 전압 출력일 경우 디지털 입력 "1"에 대한 아날로그 출력은 2.5mV에 해당됩니다.

▶ G4F-DA3V/DA2V 전압 출력일 경우 디지털 입력 "1"에 대한 아날로그 출력은 5mV에 해당됩니다.

▶ 전류 출력일 경우 디지털 입력 "1"에 대한 아날로그 출력은 4㎂입니다.

2.2 G3F-DA4V/I, G4F-DA1A

1) 오프셋값과 게인값은 다음과 같습니다.

a) 오프셋값

데이터 타입이 –8192 ~ 8191 일 때 : 디지털 입력값이 –8000일 때의 아날로그 출력값

데이터 타입이 –192 ~ 16191 일 때 : 디지털 입력값이 0일 때의 아날로그 출력값

b) 게인값

데이터 타입이 –8192 ~ 8191 일 때 : 디지털 입력값이 0일 때의 아날로그 출력값

데이터 타입이 –192 ~ 16191 일 때 : 디지털 입력값이 8000일 때의 아날로그 출력값

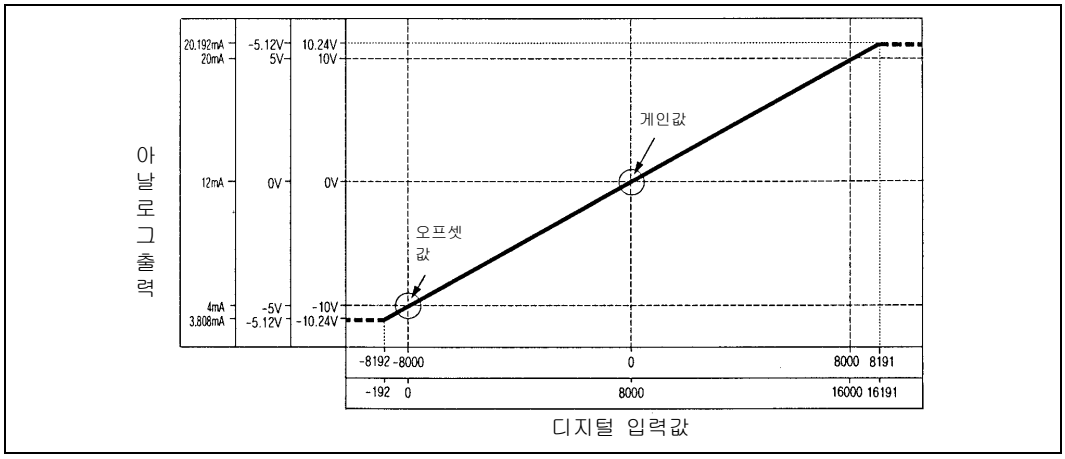
2) 공장 출하시 오프셋값과 게인값은 아래 같이 설정되어져 있습니다.

모 듈 명	아날로그 입력 범위	오프셋값	게인값
G3F-DA4I	DC 4～20 mA	DC 4 mA	DC 12 mA
G3F-DA4V	DC-10 ～10 V	DC-10 V	DC 0 V
G3F-DA1A	DC-10 ～10 V	DC-10 V	DC 0 V

3) 오프셋값과 게인값은 테스트 모드에서 각 채널마다 다르게 변경할 수 있습니다.

2.3 입출력 변환 특성 예

입출력 변환 특성 예를 그림 2.2에 나타냅니다.



[그림 2.2] 입출력 변환 특성 예

2.4 오프셋/게인값과 아날로그 출력의 관계

D/A 변환 모듈의 분해능은 오프셋값과 게인값 설정을 변경하여 임의로 조절할 수 있습니다. 오프셋값과 게인값의 설정을 변경할 경우, 아날로그값의 분해능 및 디지털 입력값에 의한 아날로그 출력값을 산출하는 식은 다음과 같습니다.

$$\text{분 해 능} = \frac{\text{게인값} - \text{오프셋값}}{8000}$$

$$\text{아날로그 출력} = \left[\frac{\text{게인값} - \text{오프셋값}}{8000} \times \text{디지털 입력값} \right] + \text{오프셋값}$$

아날로그 출력 = 분해능 X 디지털 입력값 X 오프셋값

예) 오프셋값 : DC–10V 게인값 : DC 0V 디지털 입력값 12000 일때

$$\text{분해능} = \frac{0-(-10)}{8000} = 0.00125$$

아날로그 출력 = 0.00125 X 12000 + (-10) = 5(V)

D/A 변환 모듈의 아날로그값의 최대 분해능은 표 2.1와 같으므로 디지털 입력값을 “1”씩 증가 또는 감소시킬 때 아날로그 출력값의 변화량은 위의 계산식과 다를 수 있습니다.

제 품 명	아날로그 출력 종류	최대 분해능
G3F-DA4I	DC 4～20 mA	1μ A
G3F-DA4V	DC -10 ～10 V	1.25 mV
	DC -5 ～5 V	0.625 mV
G4F-DA1A	DC 4～20 mA	1μ A
	DC -10 ～10 V	1.25 mV

[표 2.1] 최대 분해능

2.5 오프셋/게인값을 변경할 경우 입출력 변환 특성

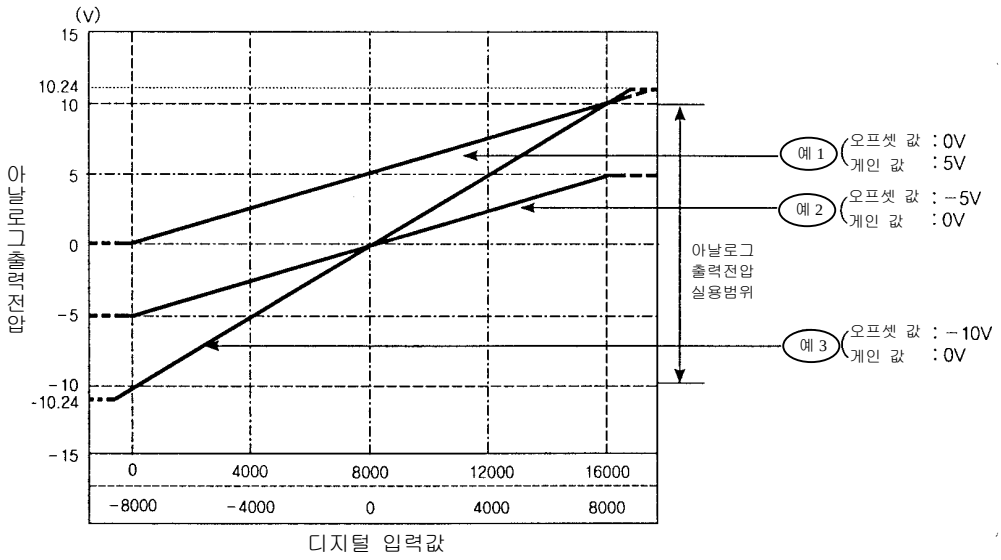
오프셋/게인값을 변경할 경우 입출력 변환 특성은 다음과 같습니다.

1) 전압 입력 특성

오프셋/게인값 설정의 변경에 따라 전압 출력 특성이 변하는 것을 그림 2.3, 2.4, 2.5에 표시합니다.

(1) DC –10 ~ 10V

DC –10 ~ 10V

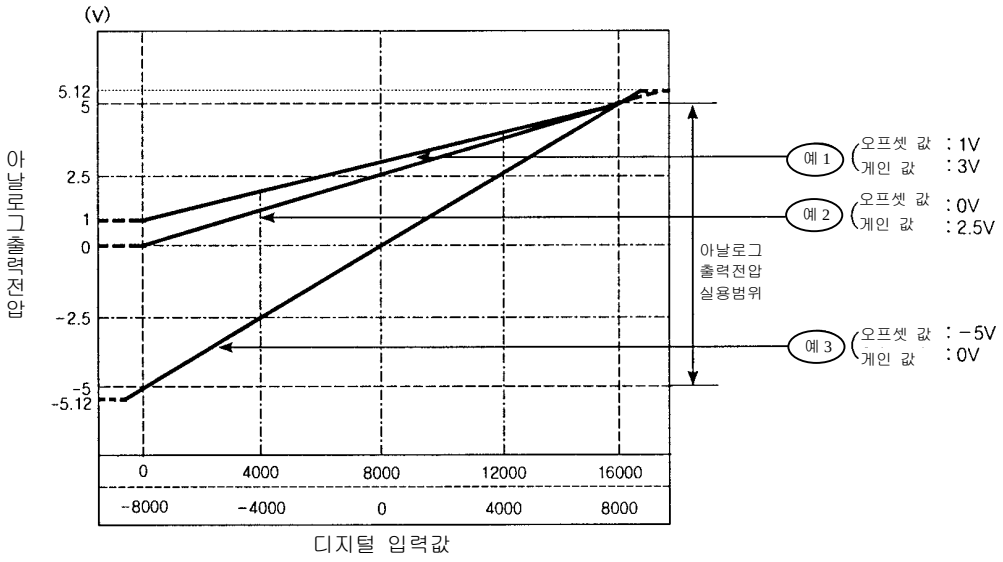


번호	디지털 입력범위	오프셋값	게인값	디지털 입력값						
				–8000	–4000	0	4000	8000	12000	16000
예 1	–192	0V	5V	–	–	0V	2.5V	5V	7.5V	10V
예 2	~	–5V	0V	–	–	–5V	–2.5V	0V	2.5V	5V
예 3	16191	–10V	0V	–	–	–10V	–5V	0V	5V	10V
예 1	–8192	0V	5V	0V	2.5V	5V	7.5V	10V	–	–
예 2	~	–5V	0V	–5V	–2.5V	0V	2.5V	5V	–	–
예 3	8191	–10V	0V	–10V	–5V	0V	5V	10V	–	–

[그림 2.3] 전압 출력 특성 (DC–10~10V)

(2) DC –5 ~ 5V

DC –5 ~ 5V



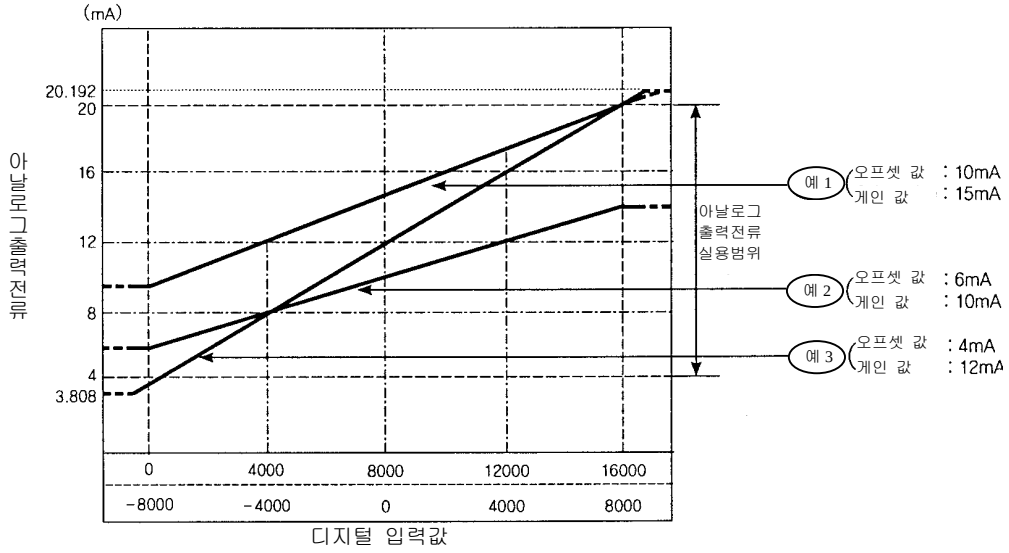
번호	디지털 입력범위	오프셋값	게인값	디지털 입력값						
				–8000	–4000	0	4000	8000	12000	16000
예 1	–192 ~ 16191	1V	3V	–	–	1V	2V	3V	4V	5V
예 2		0V	2.5V	–	–	0V	1.25V	2.5V	3.75V	5V
예 3		–5V	0V	–	–	–5V	–2.5V	0V	2.5V	5
예 1	–8192 ~ 8191	1V	3V	1V	2V	3V	4V	5V	–	–
예 2		0V	2.5V	0V	1.25V	2.5V	3.75V	5V	–	–
예 3		–5	0V	–5V	–2.5V	0V	2.5V	5V	–	–

[그림 2.4] 전압 출력 특성 (DC–5~5V)

2) 전류 입력 특성

오프셋/게인값 설정의 변경에 따라 전류 출력 특성이 변하는 것을 그림 1.5에 표시합니다.

DC –4 ~ 20mA



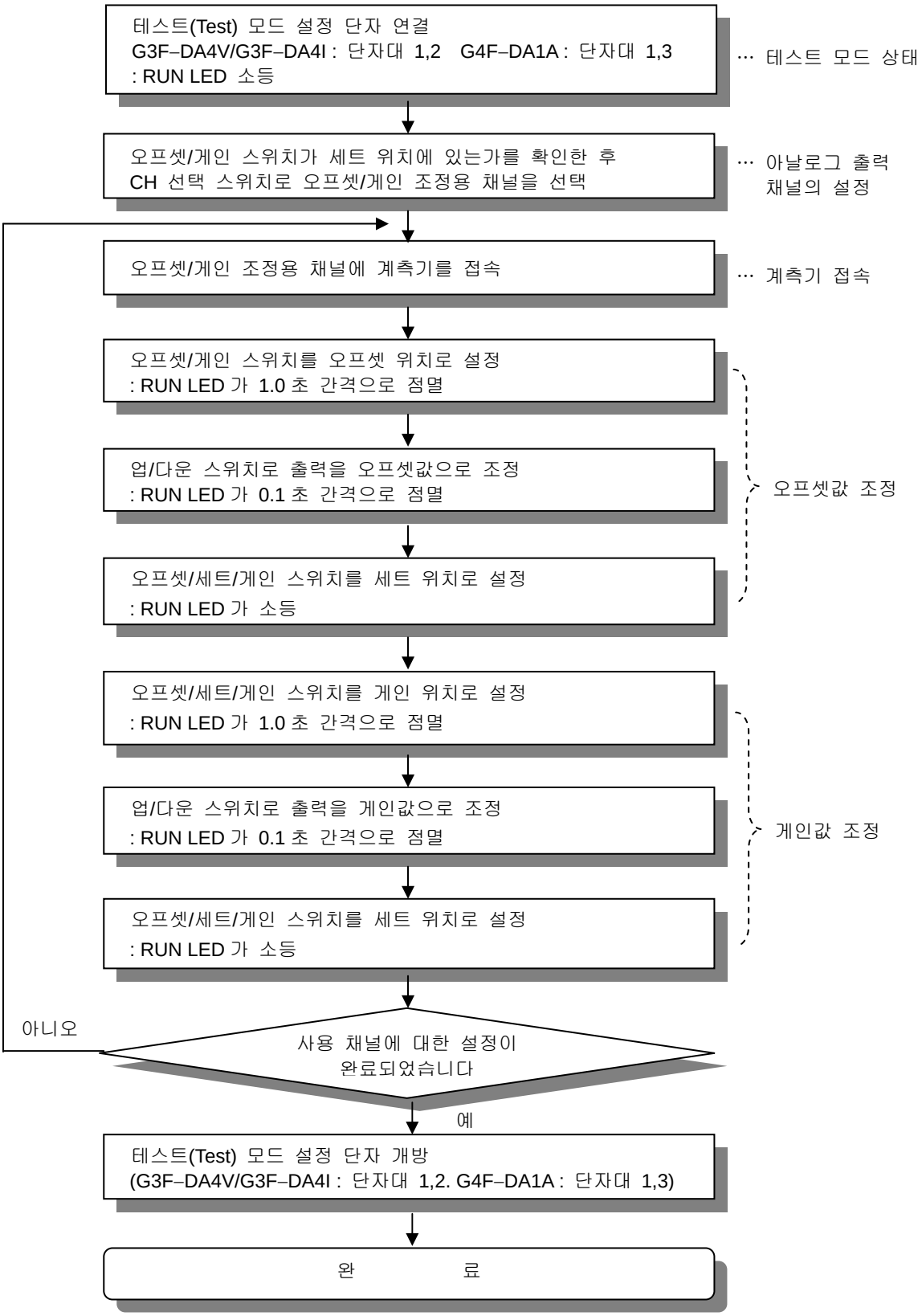
번호	디지털 입력범위	오프셋 값	게인값	디지털 입력값						
				–8000	–4000	0	4000	8000	12000	16000
예 1	–192 ~ 16191	10mA	15mA	–	–	10mA	12.5mA	15mA	17.5mA	20mA
예 2		6mA	10mA	–	–	6mA	6mA	10mA	12mA	14mA
예 3		4mA	12mA	–	–	4mA	4mA	12mA	16mA	20mA
예 1	–8192 ~ 8191	10mA	15mA	10mA	12.5mA	15mA	17.5mA	20mA	–	–
예 2		6mA	10mA	6mA	6mA	10mA	12mA	14mA	–	–
예 3		4mA	12mA	4mA	4mA	12mA	16mA	20mA	–	–

[그림 2.5] 전류 출력 특성 (DC 4~20mA)

2.6 오프셋/게인 설정 순서

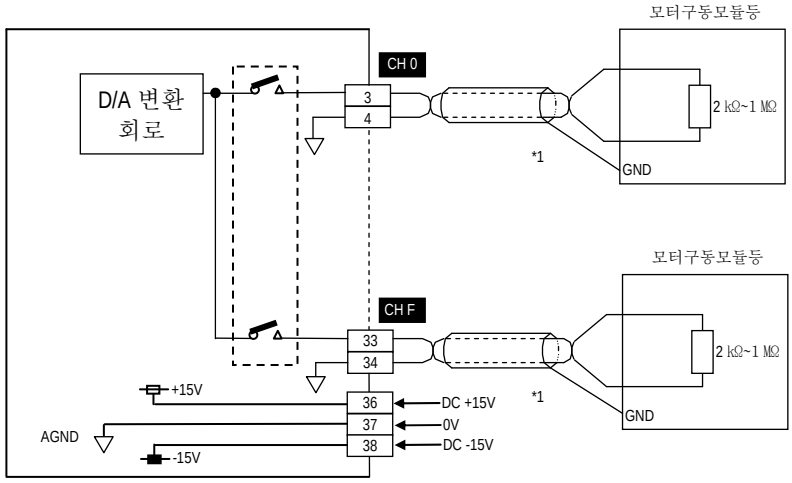
오프셋/게인 설정 순서를 나타냅니다.

오프셋/게인값은 각각의 채널에 대해서 조정합니다.

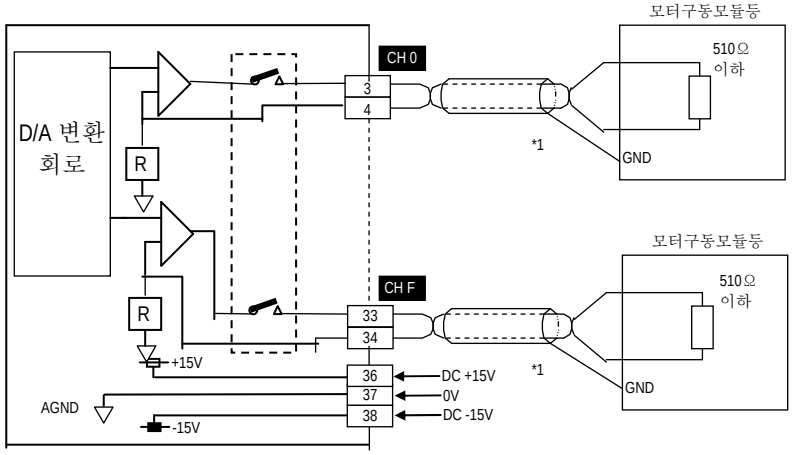


2.7 배선 예

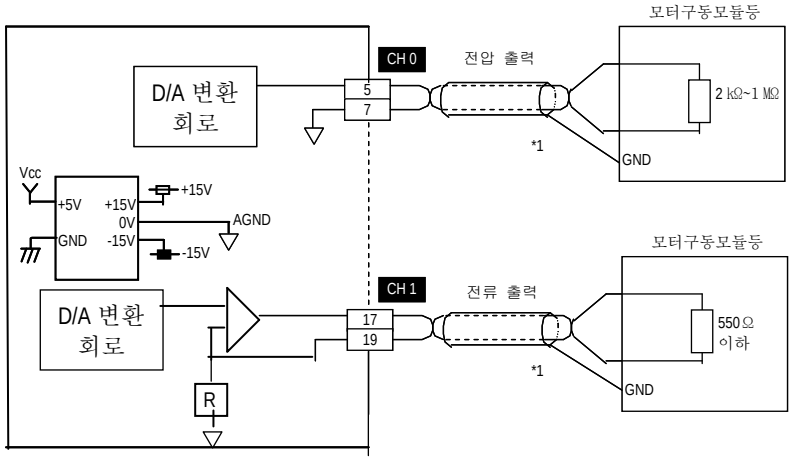
1) G3F-DA4V



2) G3F-DA4I

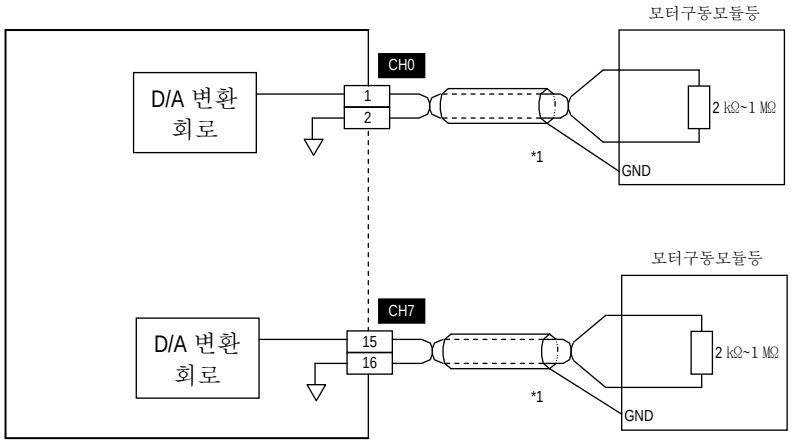


3) G4F-DA1A

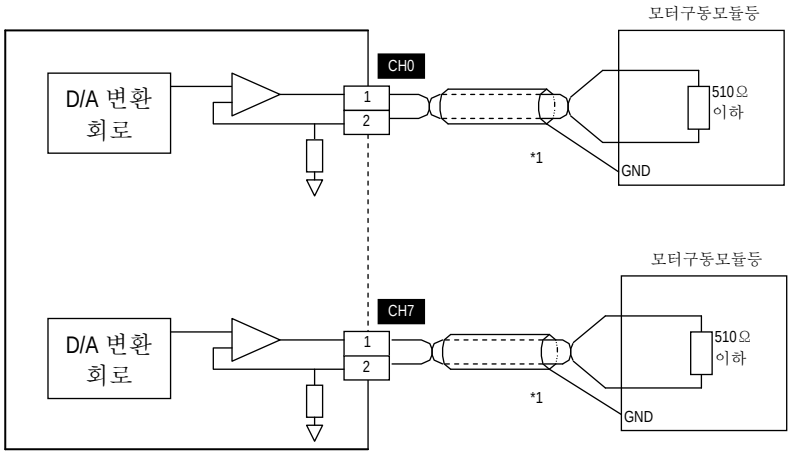


＊1 : 전선으로는 2심 트위스트 실드선을 사용하여 주십시오.

4) G3F-DA3V, G4F-DA3V, G4F-DA2V



5) G3F-DA3I, G4F-DA3I, G4F-DA2I

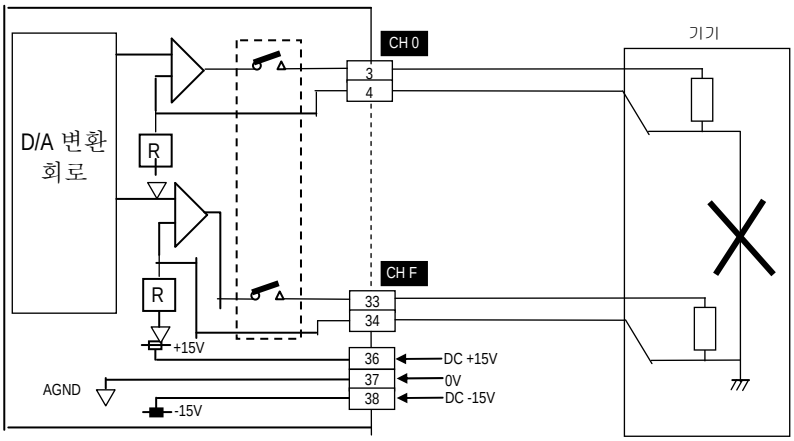


*1 : 전선은 2심 트위스트 실드선을 사용하여 주십시오.

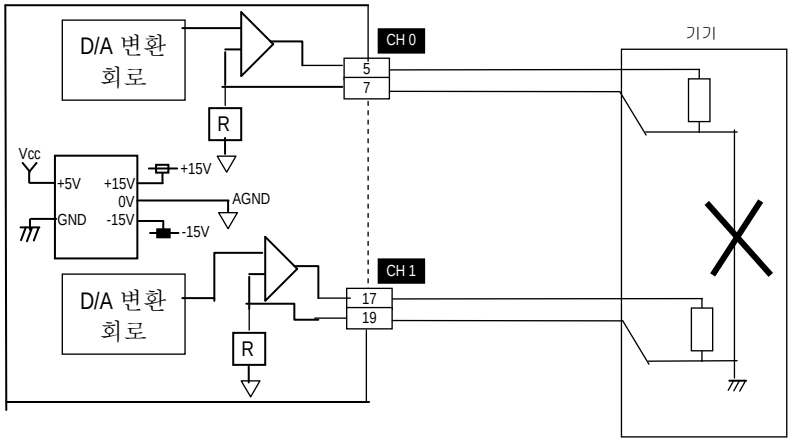
주 의

▶ 전류 출력 모듈에서 전류의 COMMON 선이 공통으로 되어 있는기기와는 접속할 수 없습니다. 전류 출력 모듈의 "-" 단자는 회로 내부적으로 각 채널별별도의 회로로 구성되어 있으므로 외부에서 공통으로 접속 시킬 경우 정상적인 출력이 되지 않습니다.

1) G3F-DA4I



2) G4F-DA1A (2 채널을 전류 출력형으로 사용할 때)



▶G3F-DA3I, G4F-DA3I, G4F-DA2I, G6F-DA2I 모듈에서도 전류 출력회로는 동일합니다.

▶G4F-DA1A 는 한 채널에서 전압과 전류를 동시에 사용할 수 없습니다.
한 채널에서 전압과 전류를 동시에 사용하면 내부 회로의 고장으로 인해 오출력, 오동작의 원인이 됩니다.

제3장 GLOFA-GM 프로그래밍

D/A 변환 모듈을 사용하기 위해서는 D/A변환 모듈의 내부 메모리와 PLC CPU간의 인터페이스를 위한 펑션블록을 사용해야 합니다.

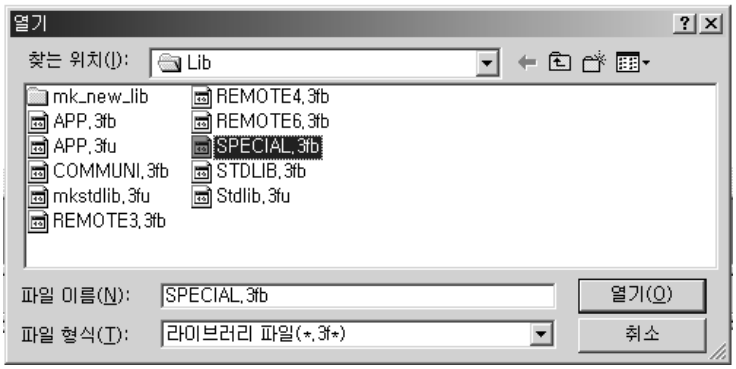
3.1 GMWIN에서 D/A 변환 모듈의 펑션블록 등록 절차

D/A 변환모듈용 펑션블록을 사용하기 위해서는 먼저 라이브러리의 등록이 필요합니다.

- Project창에서 Library창을 선택합니다.
- 마우스 오른쪽 버튼을 눌러 팝업 메뉴를 부릅니다.
- 팝업 메뉴에서 [프로젝트 항목 추가]-[라이브러리]선택.

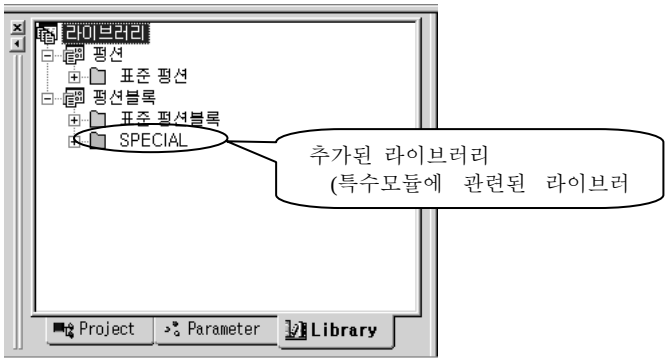
- 열기창에서 SPECIAL.□fb 또는 REMOTE*.□fb 파일을 선택하고 **열기**를 클릭합니다.

.(□ : GMR의 경우 "R" , GM1/2의 경우 "1", GM3의 경우 "3", GM4의 경우 "4", GM6의 경우 "6" 에 해당됩니다.)



(GM3의 경우 예)

- Library창에서 SPECIAL 라이브러리가 추가된 것을 확인합니다.



(SPECIAL 라이브러리를 추가한 경우 예)

3.2 펑션블록의 종류

GMWIN에서 사용되는 D/A변환 모듈용 펑션블록에 대해서 설명합니다.

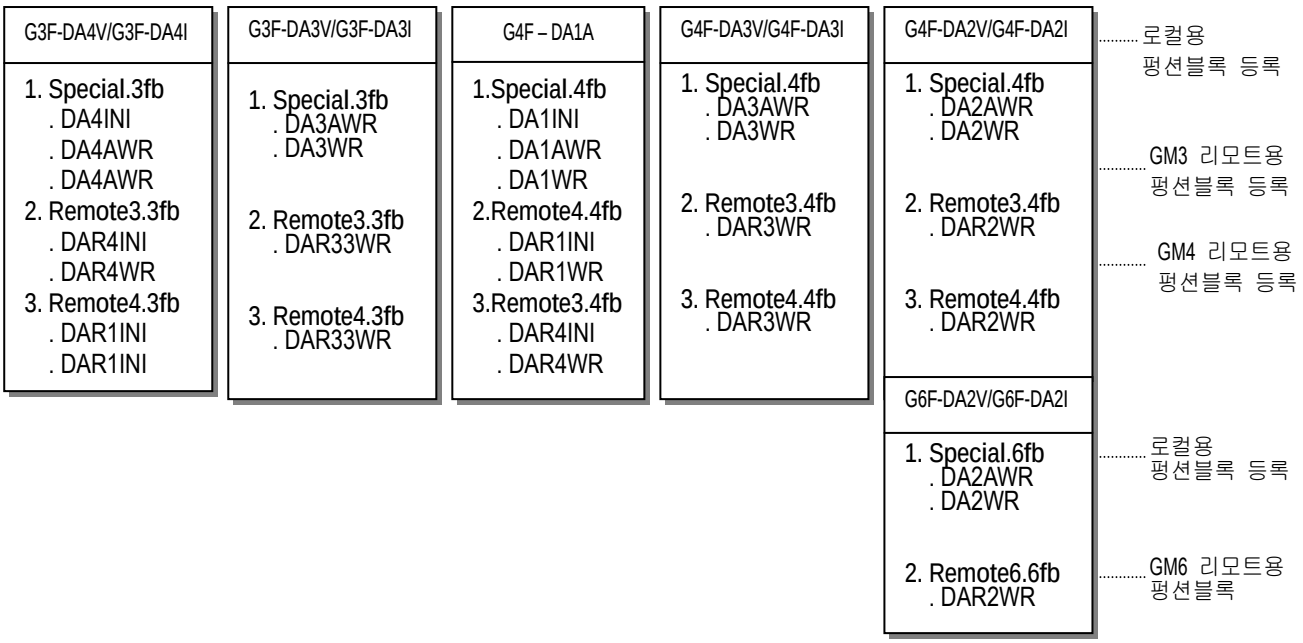
펑션블록의 종류는 다음과 같습니다.

No	G3F-DA4V / G3F-DA4I		G4F-DA1A		G6F-DA2V/ G6F-DA2I		기 능
	로컬	리모트	로컬	리모트	로컬	리모트	
1	DA4INI	DAR4INI	DA1INI	DAR1INI	-	-	모듈 초기화
2	DA4AWR	DAR4WR	DA1AWR	DAR1WR	DA2AWR	DAR2WR	D/A 변환 값 쓰기 (복수형)
3	DA4WR	-	DA1WR	-	DA2WR	-	D/A 변환 값 쓰기 (단독형)

No	G3F-DA3V/G3F-DA3I		G4F-DA3V / G4F-DA3I		G4F-DA2V / G4F-DA2I		기 능
	로컬	리모트	로컬	리모트	로컬	리모트	
1	DA3AWR	DAR33WR	DA3AWR	DAR3WR	DA2AWR	DAR2WR	D/A 변환 값 쓰기 (복수형)
2	DA3WR	-	DA3WR	-	DA2WR	-	D/A 변환 값 쓰기 (단일형)

알아두기

1.G3F-DA3V,G3F-DA3I,G4F-DA3V,G4F-DA3I의 펑션블록의 기능과 사용 방법은 동일합니다.
2.G4F-DA2V 와 G4F-DA2I의 펑션블록의 기능과 사용 방법은 동일합니다.
3.G3F-DA4V와 G3F-DA4I의 펑션블록의 기능과 사용 방법은 동일합니다.
4.G6F-DA2V 와 G6F-DA2I의 펑션블록의 기능과 사용 방법은 동일합니다
5.G3F-DA3V,G3F-DA3I,G4F-AD3V,G4F-DA3I,G4F-DA2V,G4F-DA2I, .G6F-DA2V / G6F-DA2I 는
초기화 펑션블록이 없습니다.



3.3 펑션블록의 공통사항

아래의 입출력 변수명의 기능과 사용법은 3.4항에서 설명되는 모든 펑션블록에 공통되는 내용으로 동일합니다.

구분	변수명	Datatype	내용
입력	REQ	BOOL	펑션블록 실행 요구 영역 ● 이 영역은 초기화 펑션블록의 실행을 요구하는 영역입니다. ● 프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 모듈 초기화 펑션블록이 실행됩니다.
	BASE	USINT	베이스 위치 번호 ● A/D 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. ● 설정 범위 : GM1시리즈(0 ~ 31), GM2시리즈(0 ~ 7), GM3/4시리즈(0 ~ 3) GM6시리즈(0,1)
	SLOT	USINT	슬롯의 위치 번호 ●위치 결정 모듈이 장착된 슬롯의 번호를 설정하는 영역입니다. ●설정 범위 : 0 ~ 7
	CH	BOOL [Array]*1	사용 채널 지정 영역 ● "0"이면 사용하지 않는 채널 ● "1"이면 사용하는 채널
	DATA TYPE	BOOL [Array] *주1	입력 데이터 타입 지정 영역 (G3F-DA4V/I, G4F-DA1A모듈만 해당됨) - 채널의 입력 디지털 데이터 타입의 종류를 설정하는 영역입니다. - "0"이면 데이터가 –192~16191 - "1"이면 데이터가 –8192~8191
	DATA	INT [Array] *주1	D/A 변환값 입력 영역 (G3F-DA3V/I, G4F-DA3V/I, G4F-DA2V/I, G6F-DA2V/I모듈만 해당됨) ▶채널의 입력 디지털 데이터를 설정하는 영역입니다. ▶설정 범위 : 0 ~ 4000
	SEL	USINT [Array] *주1	CPU 모듈이 Stop 상태이거나 해당 채널을 사용하지 않을 때 출력값을 선택하는 영역 (G3F-DA4V/I, G4F-DA1A모듈만 해당됨) - "0"이면 출력 범위의 중간값을 출력합니다 - "1"이면 이전값을 출력합니다 - "2"이면 출력 범위의 최대값을 출력합니다 - "3"이면 출력 범위의 최소값을 출력합니다
출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역 ● 초기화 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며, 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
	STAT	USINT	에러 상태 표시 영역 ●펑션블록 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다.
	ACT	BOOL [Array]*1	운전 채널 표시 영역 ● 초기화 펑션블록이 에러 없이 실행된 후 지정된 채널의 설정 조건이 정상이면 "1"이 출력되고, 비정상이면 "0"이 출력됩니다. ● 운전 지정이 안된 채널은 "0"이 출력됩니다.

알아두기

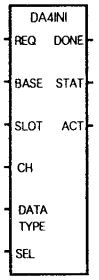
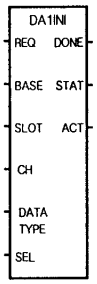
※1 : Array 수는 G3F-DA4V/I :16, G3F-DA3V/I : 8, G4F-DA3V/I :8, G4F-DA2V/I :4, G4F-DA1A : 2, G6F-DA2V/I :4 이며, 원소번호가 채널을 의미합니다.

3.4 로컬용 펑션블록

3.4.1 모듈 초기화 펑션블록

1) G3F-DA4V / G3F-DA4I : DA4INI, G4F-DA1A : DA1INI

모듈 초기화 펑션블록은 D/A 변환 모듈의 베이스 위치, 슬롯 장착위치, 사용 채널 지정, D/A 변환용 DATATYPE, CPU 모듈이 Stop 상태 일 때의 D/A 변환 모듈의 출력유지 정보를 설정하여 프로그램에 이용합니다.

펑션블록형태	구분	변수명	Datatype	내용
G3F - DA4V G3F - DA4I 	입력	REQ	BOOL	펑션블록 실행요구 영역 - 이 영역은 초기화 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행중 이 영역에 접속된 조건이 성립되어 "0→1"이되면 초기화 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정 범위 : GM1시리즈(0~31), GM2시리즈(0~7), GM3/4시리즈(0~3)
		SLOT	USINT	슬롯의 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 7
		CH	BOOL [Array] *주1	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다. - 사용하는 채널 "1", 사용하지 않는 채널은 "0"으로 지정합니다.
		DATA TYPE	BOOL [Array] *주1	입력 데이터 타입 지정 영역 - 채널의 입력 디지털 데이터 타입의 종류를 설정하는 영역입니다. - "0"이면 데이터가 –192~16191 - "1"이면 데이터가 –8192~8191
		SEL	USINT [Array] *주1	CPU 모듈이 Stop 상태이거나 해당 채널을 사용하지 않을 때 출력값을 선택하는 영역 - "0"이면 출력 범위의 중간값을 출력합니다 - "1"이면 이전값을 출력합니다 - "2"이면 출력 범위의 최대값을 출력합니다 - "3"이면 출력 범위의 최소값을 출력합니다
G4F - DA1A 	출력	DONE	BOOL	펑션블록 실행 완료 상태 - 초기화 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 초기화 펑션블록을 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러 내용은 4.4항을 참조하여 주십시오.
		ACT	USINT [Array] *주1	운전 채널 표시 영역 - 초기화 펑션블록이 에러 없이 실행된 후 지정된 채널이 정상이면 "1"이 출력되고, 운전 지정이 안된 채널은 "0"이 출력됩니다.

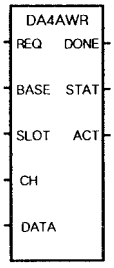
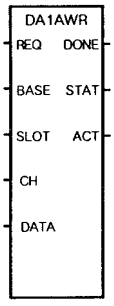
알 아 두 기

* 주1 – Array의 수는 G3F–DA4V와 G3F–DA4I : 16, G4F–DA1A : 2 입니다.

3.4.2 모듈 쓰기_Array형

1) G3F-DA4V / G3F-DA4I : DA4WR, G4F-DA1A : DA1AWR

Array형 모듈 쓰기 펑션블록은 D/A변환 모듈의 전 채널을 일괄로 처리하여 사용채널로 지정되면 D/A 변환하고자 하는 디지털값을 설정하여 프로그램에 이용합니다.

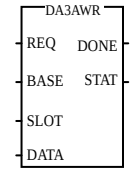
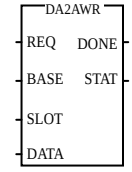
펑션블록형태	구분	변수명	Datatype	내용
G3F - DA4V G3F - DA4I 	입력	REQ	BOOL	펑션블록 실행요구 영역 - 이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행중 이 영역에 접속된 조건이 성립되어 "0→1"이되면 초기화 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정 범위 : GM1시리즈(0~31), GM2시리즈(0~7), GM3/4시리즈(0~3)
		SLOT	USINT	슬롯의 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 7
		CH	BOOL [Array] *주1	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다. - 사용하는 채널 "1", 사용하지 않는 채널은 "0"으로 지정합니다.
		DATA	INT [Array] *주1	D/A 변환값 입력 영역 - 채널의 입력 디지털 데이터를 설정하는 영역입니다.
G4F - DA1A 	출력	DONE	BOOL	펑션블록 실행 완료 상태 - 쓰기 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행때까지 "1"을 유지하며 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러 내용은 4.4항을 참조하여 주십시오.
		ACT	USINT [Array] *주1	운전 채널 표시 영역 - 쓰기 펑션블록이 에러없이 실행된 후 지정된 채널이 정상이면 "1"이 출력되고, 운전 지정이 안 된 채널은 "0"이 출력됩니다.

알 아 두 기

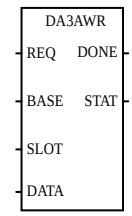
* 주1 – Array의 수는 G3F–DA4V와 G3F–DA4I : 16, G4F–DA1A : 2 입니다.

2) G3F-DA3V / G3F-DA3I, G4F-DA3V / G4F-DA3I:DA3AWR, G4F-DA2V / G4F-DA2I :DA2AWR

Array형 모듈 쓰기 펑션블록은 D/A변환 모듈의 전 채널을 사용채널로 지정 되어지고 D/A변환하고자 하는 디지털값을 설정하여 프로그램에 이용합니다.

펑션블록형태	구분	변수명	Datatype	내 용
G3F-DA3V/DA3I G4F-DA3V/DA3I 	입력	REQ	BOOL	펑션블록 실행 요구 영역 ▶이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. ▶프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 쓰기 펑션 블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 ▶D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 3
		SLOT	USINT	슬롯의 위치 번호 ▶D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 7
G4F-DA2V G4F-DA2I 		DATA	INT [Array]	D/A 변환값 입력 영역 ▶채널의 입력 : 디저털 데이터를 설정하는 영역입니다. ▶Array의 수는 G3F-DA3V,G3F-DA3I,G4F-DA3V,G4F-DA3I : 8, G4F-DA2V , G4F-DA2I : 4 입니다.
	출력	DONE	BOOL	펑션블록 실행 완료 상태 ▶쓰기 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때 까지 "1"을 유지하며 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 ▶쓰기 펑션블록 실행 중 에러가 발생되면 에러번호를 출력하는 영역입니다. ▶에러 내용은 4.4항을 참조하여 주십시오.

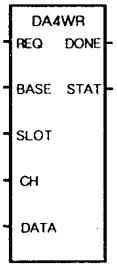
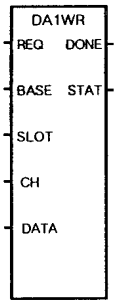
3) G6F-DA2V / G6F-DA2I : DA2AWR

펑션블록형태	구 분	변수명	Datatype	내 용
G6F-DA2V **G6F-DA2I** 	입력	REQ	BOOL	펑션블록 실행 요구 영역 - 이 영역은 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 쓰기 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 - A/D 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 1
		SLOT	USINT	슬롯의 위치 번호 - A/D 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 7
		DATA	INT [Array]	D/A 변환값 입력 영역 - 채널의 입력 디지털 데이터를 설정하는 영역입니다. - Array의 수는 4이며 채널수를 나타냅니다.
	출력	DONE	BOOL	펑션블록 실행 완료 상태 - 쓰기 펑션블록이 에러없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며, 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 에러 번호를 출력 영역입니다.

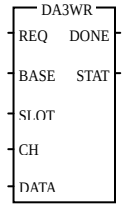
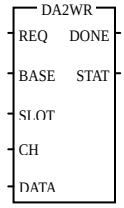
3.4.3 모듈 쓰기_단일형

1) G3F-DA4V / G3F-DA4I : DA4WR, G4F-DA1A : DA1WR

단일형 모듈 쓰기 펑션블록은 D/A 변환 모듈의 한 채널만 처리하며 D/A 변화하고자 하는 디지털값을 설정하여 프로그램에 이용합니다.

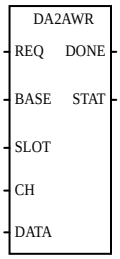
펑션블록형태	구분	변수명	Datatype	내용
G3F - DA4V G3F - DA4I 	입력	REQ	BOOL	펑션블록 실행요구 영역 - 이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행중 이 영역에 접속된 조건이 성립되어 "0→1"이되면 쓰기 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정 범위 : GM1시리즈(0~31), GM2시리즈(0~7), GM3/4시리즈(0~3)
		SLOT	USINT	슬롯의 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 7
		CH	USINT	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다. - 설정 범위 : 0 ~ 15 (G4F-DA1A : 0 ~ 1)
		DATA	INT	D/A 변환값 입력 영역 - 채널의 입력 디지털 데이터를 설정하는 영역입니다.
G4F - DA1A 	출력	DONE	BOOL	펑션블록 실행 완료 상태 - 쓰기 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행때까지 "1"을 유지하며 에러가 발생되면 "0"이 출력되면서 운전 정지상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러내용은 4.4항을 참조하여 주십시오.

2) G3F-DA3V / G3F-DA3I, G4F-DA3V / G4F-DA3I : DA3WR, G4F-DA2V / G4F-DA2I : DA2WR

펑션블록형태	구분	변수명	Datatype	내용
G3F-DA3V/DA3I G4F-DA3V/DA3I  G4F-DA2V G4F-DA2I  	입력	REQ	BOOL	펑션블록 실행 요구 영역 ▶이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. ▶프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 모듈 쓰기 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 ▶D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 3
		SLOT	USINT	슬롯의 위치 번호 ▶D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 7
		CH	USINT	사용 채널 지정 영역 ▶사용하는 채널을 지정하는 영역입니다. ▶설정범위 : G3F-DA3V / G3F-DA3I / G4F-DA3V / G4F-DA3I : 0~7 G4F-DA2V / G4F-DA2I : 0~3
		DATA	INT	D/A 변환값 입력 영역 ▶채널의 입력 디지털 데이터를 설정하는 영역입니다. ▶설정 범위 : 0 ~ 4000
	출력	DONE	BOOL	펑션블록 실행 완료 상태 ▶쓰기 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며, 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 ▶쓰기 펑션블록 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다. ▶에러내용은 4.4항을 참조하여 주십시오.

3) G6F-DA2V / G6F-DA2I : DA2WR

.

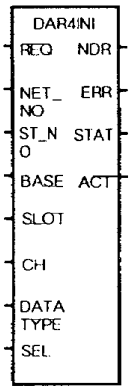
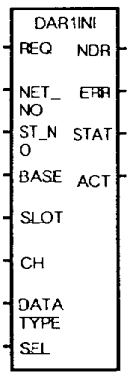
펑션블록형태	구분	변수명	Datatype	내 용
G6F-DA2V G6F-DA2I  	입력	REQ	BOOL	펑션블록 실행 요구 영역 - 이 영역은 읽기 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행 중 이 영역에 접속된 조건이 성립되어 "0→1"이 되면 쓰기 펑션블록이 실행됩니다.
		BASE	USINT	베이스 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 1
		SLOT	USINT	슬롯의 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정 범위 : 0 ~ 7
		CH	USINT	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다 - 설정 범위 : 0 ~ 3
		DATA	INT	D/A 변환값 입력 영역 - 채널의 입력 디지털 데이터를 설정하는 영역입니다.
	출력	DONE	BOOL	펑션블록 실행 완료 상태 표시 영역 - 읽기 펑션블록이 에러 없이 실행 완료되면 "1"이 출력되고, 다음 실행 때까지 "1"을 유지하며, 에러가 발생되면 "0"이 출력되면서 운전 정지 상태가 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러내용은 4.3항을 참조하여 주십시오.

3.5 리모용 펑션블록

3.5.1 모듈 초기화 펑션블록

1) G3F-DA4V / G3F-DA4I : DAR4INI, G4F-DA1A : DAR1INI

모듈 초기화 펑션블록은 자국의 통신 모듈 슬롯 위치번호, 리모트 I/O 국에 장착된 통신 모듈의 국번, D/A 변환 모듈이 장착된 베이스 위치번호, 슬롯 장착 위치 번호, 사용 채널 지정, D/A 변환용 디지털 DATATYPE, CPU 모듈이 Stop 상태일 때의 D/A 변환 모듈 출력 상태를 설정하며 이를 프로그램에 이용합니다.

펑션블록형태	구분	변수명	Datatype	내용
G3F - DA4V G3F - DA4I 	입력	REQ	BOOL	펑션블록 실행요구 영역 - 이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행중 이 영역에 접속된 조건이 성립되어 “0→1”이되면 초기화 펑션블록이 실행됩니다.
		NET_NO	USINT	펑션블록이 전송될 자국의 통신 모듈이 장착된 슬롯 위치번호 - 설정범위 : 0 ~ 7
		ST_NO	USINT	리모트 I/O 국에 장착된 통신 모듈의 국번 - 설정범위 : 0 ~ 63
		BASE	USINT	베이스 모듈 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정범위 : 0 ~ 3
		SLOT	USINT	슬롯 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정범위 : 0 ~ 7
		CH	BOOL [Array] *주1	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다. - 사용하는 채널 “ 1”, 사용하지 않는 채널은 “0”으로 지정합니다.
		DATA TYPE	BOOL [Array] *주1	입력 데이터 타입 지정 영역 - 채널의 입력 디지털 데이터 타입의 종류를 설정하는 영역입니다. - “0”이면 데이터가 –192 ~ 16191 - “1”이면 데이터가 –8192 ~ 8191
G4F - DA1A 		SEL	USINT [Array] *주1	CPU 모듈이 Stop 상태이거나 해당 채널을 사용하지 않을 때 출력값을 선택하는 영역 - “0”이면 출력 범위의 중간값을 출력합니다. - “1”이면 이전값을 출력합니다. - “2”이면 출력 범위의 최대값을 출력합니다. - “3”이면 출력 범위의 최소값을 출력합니다.
	출력	NDR	BOOL	펑션블록 실행이 에러 없이 종료된 경우 “1”이 되며, 실행조건이 성립된 스캔동안 “1”을 유지하고 다음 스캔에서 “0”이 됩니다.
		ERR	BOOL	에러 정보 표시 영역 - 초기화 펑션블록 실행중 에러가 발생되면 “1”이 출력되어 운전 정지 상태가 되고, 실행 조건이 성립된 스캔동안 “1”을 유지하며 다음 스캔에서 “0”이 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 초기화 펑션블록 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러내용은 4.4항을 참조하여 주십시오.
		ACT	USINT [Array] *주1	운전 채널 표시 영역 - 초기화 펑션블록이 에러 없이 실행된 후 지정된 채널이 정상이면 “1”이 출력되고, 운전 지정이 안된 채널은 “0”이 출력됩니다.

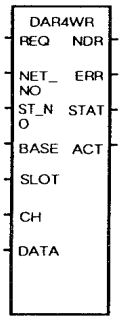
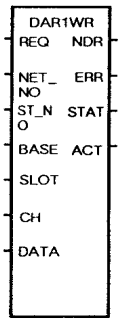
알 아 두 기

* 주1 – Array의 수는 G3F–DA4V와 G3F–DA4I : 16, G4F–DA1A : 2 입니다.

3.5.2 모듈 쓰기 펑션블록

1) G3F-DA4V / G3F-DA4I : DAR4WR, G4F-DA1A : DAR1WR

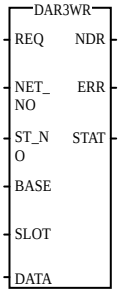
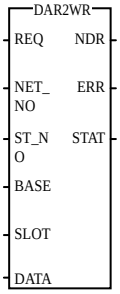
모듈 쓰기 펑션 블록은 D/A 변환 모듈이 전채널을 일괄로 처리하며 사용채널로 지정되면 D/A 변환 하고자 하는 디지털값을 설정하여 프로그램에 이용합니다.

펑션블록형태	구분	변수명	Datatype	내용
G3F - DA4V G3F - DA4I 	입력	REQ	BOOL	상승 Edge에서 펑션블록 실행요구 영역 - 이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. - 프로그램 수행중 이 영역에 접속된 조건이 성립되어 "0→1"로 변할 때 (⊥↑‾) 쓰기 펑션블록이 실행됩니다.
		NET_NO	USINT	펑션블록이 전송될 자국의 통신 모듈이 장착된 슬롯 위치번호 - 설정범위 : 0 ~ 7
		ST_NO	USINT	리모트 I/O 국에 장착된 통신 모듈의 국번 - 설정범위 : 0 ~ 63
		BASE	USINT	베이스 모듈 위치 번호 - D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. - 설정범위 : 0 ~ 3
		SLOT	USINT	슬롯 위치 번호 - D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. - 설정범위 : 0 ~ 7
		CH	BOOL [Array] *주1	사용 채널 지정 영역 - 사용하는 채널을 지정하는 영역입니다. - 사용하는 채널 " 1", 사용하지 않는 채널은 "0"으로 지정합니다.
		DATA	INT [Array] *주1	D/A 변환값 입력 영역 - 채널의 입력 디지털 데이터를 설정하는 영역입니다.
G4F - DA1A 	출력	NOR	BOOL	펑션블록 실행이 에러없이 종료된 경우 "1"이 출력되고, 실행조건이 성립된 스캔 동안 "1"을 유지하고 다음 스캔에서 "0" 이 됩니다.
		ERR	BOOL	에러 정보 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 "1"이 출력되어 운전 정지 상태가 되고, 실행 조건이 성립된 스캔동안 "1"을 유지하며 다음 스캔에서 "0"이 됩니다.
		STAT	USINT	에러 상태 표시 영역 - 쓰기 펑션블록 실행중 에러가 발생되면 에러 번호를 출력하는 영역입니다. - 에러내용은 4.4항을 참조하여 주십시오.
		ACT	BOOL [Array] *주1	운전 채널 표시 영역 - 쓰기 펑션블록이 에러없이 실행된 후 지정된 채널이 정상이면 "1"이 출력되고, 운전 지정이 안된 채널은 "0"이 출력됩니다.

알 아 두 기

* 주1 – Array의 수는 G3F–DA4V와 G3F–DA4I : 16, G4F–DA1A : 2 입니다.

2) G3F-DA3V / G3F-DA3I : DAR33WR,G4F-DA3V / G4F-DA3I : DAR3WR, G4F-DA2V / G4F-DA2I : DAR2WR

펑션블록형태	구분	변수명	Datatype	내용
G3F-DA3V/DA3I G4F-DA3V/DA3I  G4F-DA2V G4F-DA2I  	입력	REQ	BOOL	상승 Edge에서 펑션블록 실행 요구 영역 ▶이 영역은 쓰기 펑션블록의 실행을 요구하는 영역입니다. ▶프로그램 수행중 이 영역에 접속된 조건이 성립되어 "0→1"로 변할 때(상승 Edge) 쓰기 펑션블록이 실행됩니다.
		NET_NO	USINT	펑션블록이 전송될 자국의 통신 모듈이 장착된 슬롯 위치 번호 ▶설정 범위 : 0 ~ 7
		ST_NO	USINT	리모트 I/O국에 장착된 통신 모듈의 국번 ▶설정 범위 : 0 ~ 63
		BASE	USINT	베이스 위치 번호 ▶D/A 변환 모듈이 장착된 베이스의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 3
		SLOT	USINT	슬롯의 위치 번호 ▶D/A 변환 모듈이 장착된 슬롯의 번호를 쓰는 영역입니다. ▶설정 범위 : 0 ~ 7
		DATA	INT [Array]	D/A 변환값 입력 영역 ▶채널의 입력 디지털 데이터를 설정하는 영역입니다. ▶설정 범위 : 0 ~ 4000 ▶Array의 수는 G3F-DA3V/DA3I,G4F-DA3V/DA3I는 8, G4F-DA2V 와 G4F-DA2I는 4입니다.
	출력	NDR	BOOL	펑션블록 실행이 에러 없이 종료된 경우 "1"이 출력되고, 실행 조건이 성립된 스켄 동안 "1"을 유지하고 다음 스켄에서 "0"이 됩니다.
		ERR	BOOL	에러 정보 표시 영역 ▶쓰기 펑션블록 실행 중 에러가 발생되면 "1"이 출력되어 운전 정지 상태가 되고, 실행 조건이 성립된 스켄 동안 "1"을 유지하며, 다음 스켄에서 "0"이 됩니다.
		STAT	USINT	에러 상태 표시 영역 ▶쓰기 펑션블록 실행 중 에러가 발생되면 에러 번호를 출력하는 영역입니다. ▶에러내용은 4.4항을 참조하여 주십시오.

3.6 펑션블록상의 에러 종류

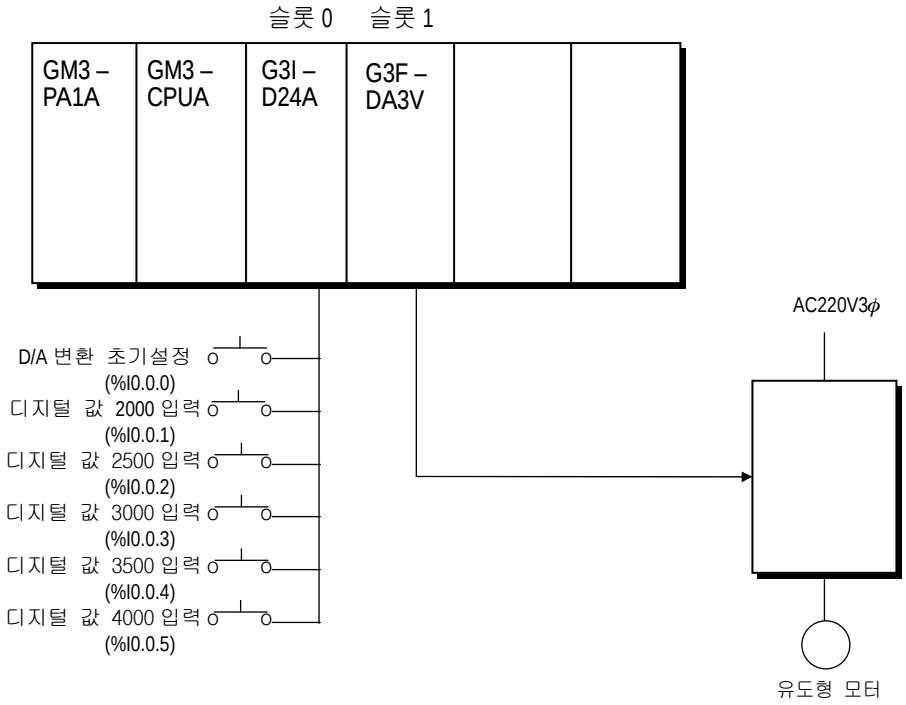
출력 변수 STAT에 나타나는 에러종류 및 조치 방법에 대해서 설명합니다.

STAT 번호	구분	내 용	펑션블록			조 치 방 법
			초기화	쓰 기		
				Array형	단일형	
0	로컬	정상 동작 중	○	○	○	-
1		베이스의 위치가 설정 범위 초과	○	○	○	설정 범위 내로 수정(4.2항 참조)
2		해당 베이스의 H/W 에러	○	○	○	A/S 의뢰
3		슬롯의 위치 번호가 설정 범위 초과	○	○	○	D/A 변환 모듈이 장착된 올바른 슬롯 번호 지정
4		지정한 슬롯에 D/A 변환 모듈이 비어 있음	○	○	○	지정된 슬롯에 D/A 변환 모듈을 장착
5		D/A 변환 모듈이 아닌 다른 모듈이 장착되어 있음	○	○	○	지정된 슬롯에 D/A 변환 모듈을 장착
6		채널 번호가 설정 범위 초과	-	-	○	사용 채널 지정을 바르게 설정
7		D/A 변환 모듈의 H/W 에러	○	○	○	A/S 의뢰
8		D/A 변환 모듈의 공용 메모리 에러	○	○	○	A/S 의뢰
9		초기화 펑션블록에서 사용채널 미지정	–	○	○	초기화 펑션블록에서 사용채널 바르게 지정
10		테스트 모드	○	○	○	테스트 모드에서 노멀 모드로 전환
128	리모트	리모트용 통신 모듈의 H/W 에러	–	○	-	리모트 통신 모듈 사용설명서 참조
129		베이스의 위치가 설정 범위 초과	○	○		설정 범위 내로 수정(4.2항 참조)
131		슬롯의 위치 번호가 설정 범위 초과	○	○		D/A 변환 모듈이 장착된 올바른 슬롯 번호 지정
133		D/A 변환 모듈이 아닌 다른 모듈이 장착되어 있음	○	○		지정된 슬롯에 D/A 변환 모듈을 장착
135		D/A 변환 모듈의 H/W 에러	○	○		A/S 의뢰
136		D/A 변환 모듈의 공용 메모리 에러	○	○		A/S 의뢰
137		초기화 펑션블록에서 사용채널 미지정	–	○		초기화 펑션블록에서 사용채널 바르게 지정
138		테스트 모드	○	○		테스트 모드에서 노멀 모드로 전환

제 4 장 프로그램 예제(GLOFA-GM 용)

4.1 5 단계의 아날로그 출력 전압으로 인버터의 속도를 제어하는 프로그램

1) 시스템 구성



2) 초기 설정 내용

(1) 사용 채널 : 채널 0

3) 프로그램 설명

(1) %I0.0.0 이 On 되면 D/A 변환 초기 설정을 한다.

(2) %I0.0.1 이 On 되면 채널 0 에 2000(5V)을 출력한다.

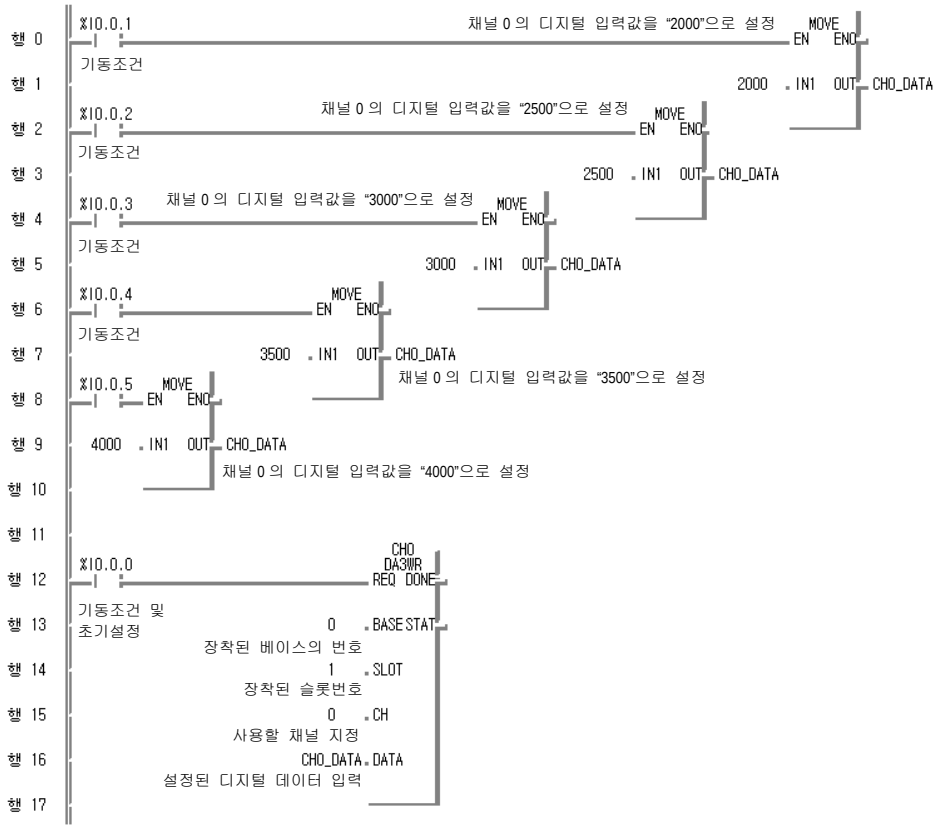
(3) %I0.0.2 가 On 되면 채널 0 에 2500(6.25V)을 출력한다.

(4) %I0.0.3 이 On 되면 채널 0 에 3000(7.5V)을 출력한다.

(5) %I0.0.4 이 On 되면 채널 0 에 3500(8.75V)을 출력한다.

(6) %I0.0.5 이 On 되면 채널 0 에 4000(10V)을 출력한다.

4) 프로그램

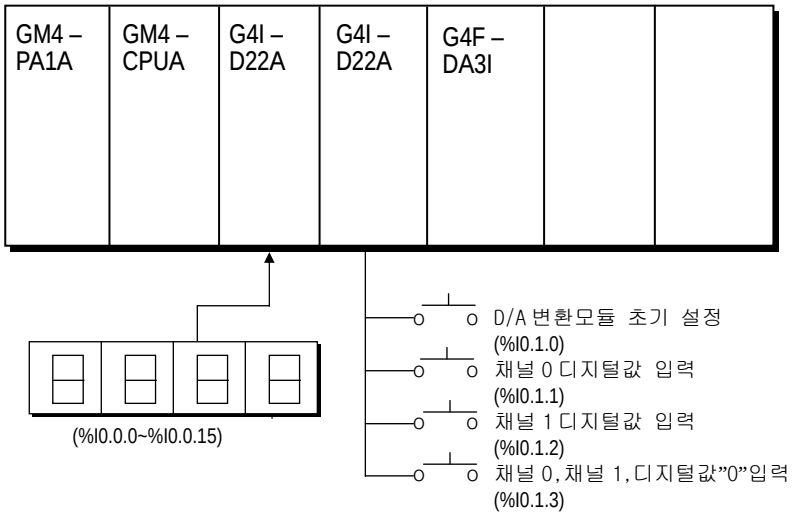


5) 프로그램에 사용된 입출력 변수

	변수명	변수종류	메모리할당	사용여부	데이터타입	초기값	설명문
1	CH0_DATA	VAR	<자동>	*	INT	0	
2	CH0	VAR	<자동>	*	FB Instance		

4.2 D/A 변환값을 디지털 스위치로 설정해서 출력하는 프로그램

1) **시스템 구성**



2) **초기 설정 내용**

(1)사용 채널 지정 : 채널 0, 1

3) **프로그램 설명**

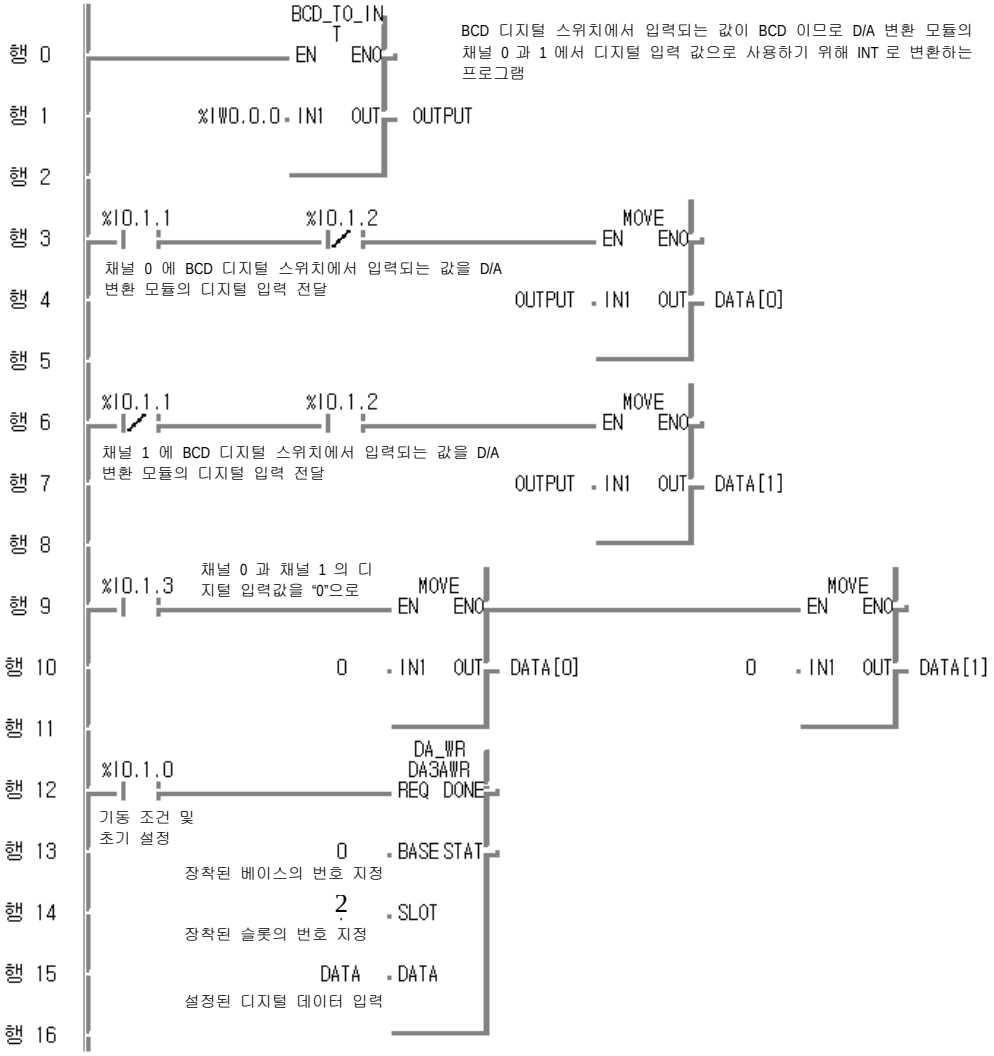
(1) %I0.1.0.이 "On"되면 D/A 변환 모듈을 초기화 합니다.

(2) %I0.1.1 이 "On"되면 BCD 디지털 스위치로 입력된 수치가 D/A 변환모듈의 채널 0 로 출력됩니다.

(3) %I0.1.2 이 "On"되면 BCD 디지털 스위치로 입력된 수치가 D/A 변환모듈의 채널 1 로 출력됩니다.

(4) %I0.1.3 이 "On" 되면 채널 0 과 채널 1 의 디저털 입력값을 "0" 으로 초기화 시킵니다.

4) 프로그램



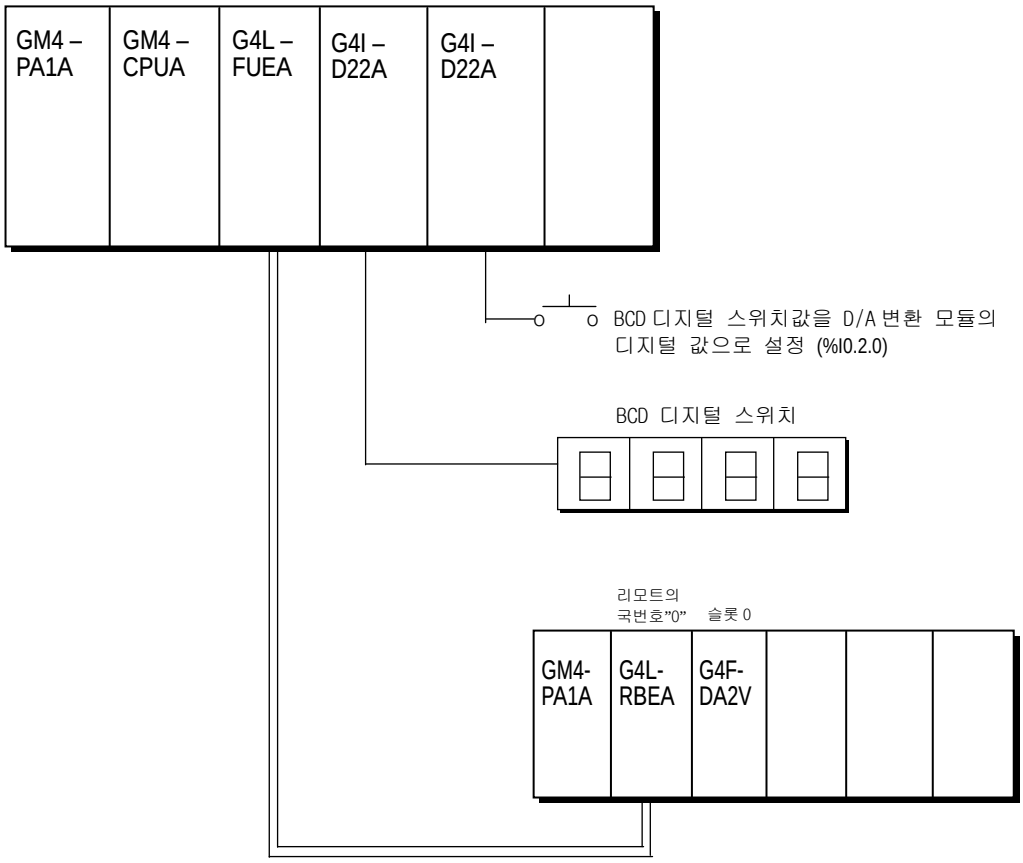
5) 프로그램에 사용된 입출력 변수

	변수명	변수종류	메모리귤당	사용여부	데이터타입	초기값	설명문
1	DA_WR	VAR	<자동>	*	FB Instance		
2	DATA	VAR	<자동>	*	ARRAY[8] OF INT		
3	OUTPUT	VAR	<자동>	*	INT		

4.3 리모트 I/O 국에 D/A 변환 모듈을 장착할 때의 프로그램

D/A 변환값을 디지털 스위치로 설정해서 출력하는 프로그램

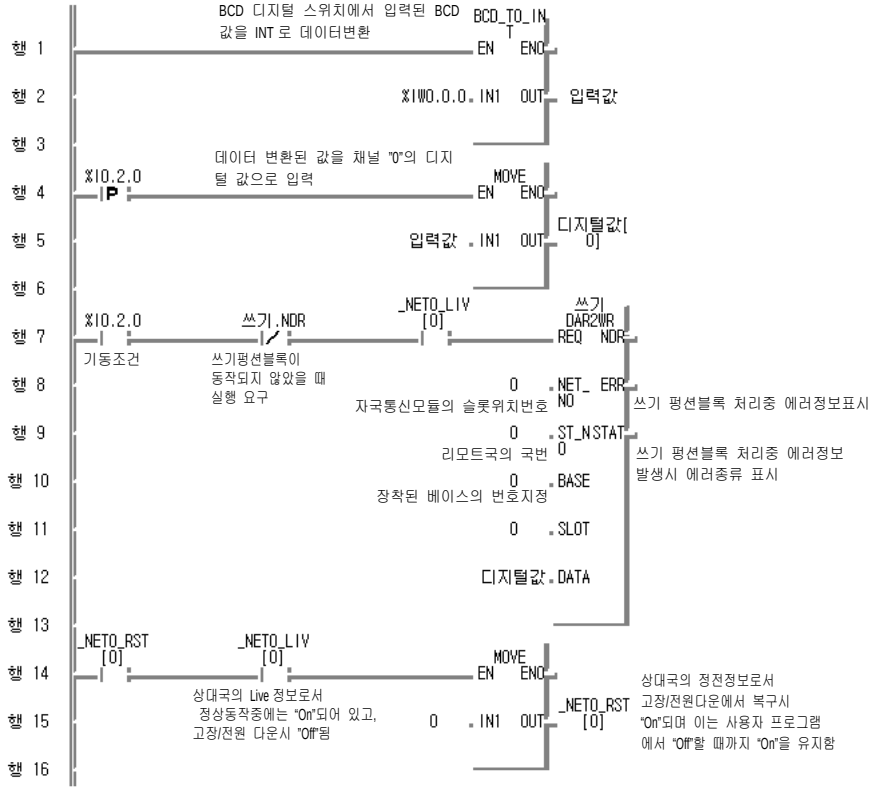
1) 시스템 구성



2) 프로그램 설명

(1) %I0.2.0 이 On 되면 디지털 스위치로 설정한 값을 D/A 변환 모듈의 채널 0 에 출력합니다.

3) 프로그램 설명

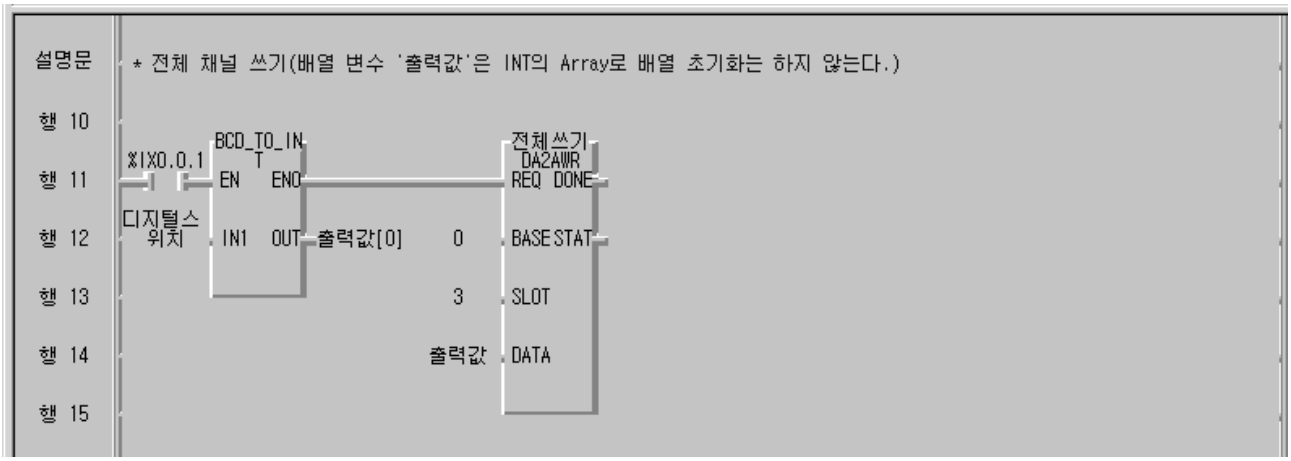


4) 프로그램에 사용된 입출력 변수 설명

	변수명	변수종류	메모리할당	사용여부	데이터타입	초기값	설명문
1	디지털값	VAR	<자동>	*	ARRAY[4] OF I		
2	쓰기	VAR	<자동>	*	FB Instance		
3	입력값	VAR	<자동>	*	INT		

4.4 예제 프로그램 실습

	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	디지털스위치	WORD	%IW0.0.1		VAR	*	
2	개별쓰기	FB Instance	<자동>		VAR	*	
3	전체쓰기	FB Instance	<자동>		VAR	*	
4	출력값	ARRAY[4] OF INT	<자동>		VAR	*	
5	CH0출력값	INT	<자동>		VAR	*	


설명문
* 전체 채널 쓰기(배열 변수 '출력값'은 INT의 Array로 배열 초기화는 하지 않는다.)
행 10
행 11
행 12
행 13
행 14
행 15
%IX0.0.1
디지털스
위치
BCD_TO_IN
T
EN
ENO
IN1
OUT
출력값[0]
0
3
출력값
전체쓰기
DA2AWR
REQ
DONE
BASESTAT
SLOT
DATA

부 록

부록A. 플래그 일람표

부록B. GLOFA-GM 명령어집

부록C. 태스크 프로그램

부록A. 플래그 일람표

부록 A. 플래그 일람표 (예약 변수)

1. 예약 변수

예약 변수는 시스템에서 미리 선언한 변수들입니다. 이 변수들은 특수한 용도로 사용하며, 사용자가 이 변수 이름으로 변수 선언을 할 수는 없습니다.

이 예약 변수를 사용할 때에는 변수 선언 없이 사용합니다.

1) 사용자 플래그

예약 변수	데이터 타입	내 용
_ERR	BOOL	연산 에러 접점
_LER	BOOL	연산 에러 래치 접점
_T20MS	BOOL	20 ms 클럭 접점
_T100MS	BOOL	100 ms 클럭 접점
_T200MS	BOOL	200 ms 클럭 접점
_T1S	BOOL	1 초 클럭 접점
_T2S	BOOL	2 초 클럭 접점
_T10S	BOOL	10 초 클럭 접점
_T20S	BOOL	20 초 클럭 접점
_T60S	BOOL	60 초 클럭 접점
_ON	BOOL	항시 On 접점
_OFF	BOOL	항시 Off 접점
_1ON	BOOL	1 스캔 On 접점
_1OFF	BOOL	1 스캔 Off 접점
_STOG	BOOL	스캔마다 반전
_INIT_DONE	BOOL	초기화 프로그램 완료
_RTC_DATE	DATE	RTC 의 현재 날짜
_RTC_TOD	TOD	RTC 의 현재 시간
_RTC_WEEK	UINT	RTC 의 현재 요일

2) 시스템 에러 대표 플래그

예 약 변 수	데 이 터 타 입	내 용
_CNF_ER	WORD	시스템의 에러(중고장)
_CPU_ER	BOOL	CPU 구성 에러
_IO_TYER	BOOL	모듈 타입 불일치 에러
_IO_DEER	BOOL	모듈 착탈 에러
_FUSE_ER	BOOL	Fuse 단선 에러
_IO_RWER	BOOL	입출력 모듈 읽기/쓰기 에러(고장)
_SP_IFER	BOOL	특수/통신 모듈 인터페이스 에러(고장)
_ANNUN_ER	BOOL	외부기기의 중고장 검출 에러
_WD_ER	BOOL	Scan Watch-Dog 에러
_CODE_ER	BOOL	프로그램 코드 에러
_STACK_ER	BOOL	Stack Overflow 에러
_P_BCK_ER	BOOL	프로그램 에러

3) 시스템 에러 해제 플래그

예 약 변 수	데 이 터 타 입	내 용
_CNF_ER_M	BYTE	시스템 에러(중고장) 해제
_IO_DEER_M	BOOL	모듈 착탈 에러 해제
_FUSE_ER_M	BOOL	퓨즈 단선 에러 해제
_IO_RWER_M	BOOL	입출력 모듈 읽기/쓰기 에러 해제
_SP_IFER_M	BOOL	특수/통신 모듈 인터페이스 에러 해제
_ANNUN_ER_M	BOOL	외부기기의 중고장 검출 에러 해제

4) 시스템 경고 대표 플래그

예 약 변 수	데 이 터 타 입	내 용
_CNF_WAR	WORD	시스템의 경고(경고장)
_RTC_ERR	BOOL	RTC 데이터 이상
_D_BCK_ER	BOOL	데이터 백업 에러
_H_BCK_ER	BOOL	핫 리스타트 수행 불가 에러
_AB_SD_ER	BOOL	비정상 전원 차단(Abnormal Shutdown)
_TASK_ERR	BOOL	태스크(Task) 충돌(정주기,외부 태스크)
_BAT_ERR	BOOL	배터리 이상
_ANNUN_WR	BOOL	외부기기의 경고장 검출

예 약 변 수	데 이 터 타 입	내 용
_HSPMT1_ER	BOOL	고속 링크 파라미터 1 이상
_HSPMT2_ER	BOOL	고속 링크 파라미터 2 이상
_HSPMT3_ER	BOOL	고속 링크 파라미터 3 이상
_HSPMT4_ER	BOOL	고속 링크 파라미터 4 이상

5) 시스템 에러 상세 플래그

예 약 변 수	데 이 터 타 입	내 용
_IO_TYER_N	UINT	모듈 타입 불일치 슬롯 넘버
_IO_TYERR	ARRAY OF BYTE	모듈 타입 불일치 위치
_IO_DEER_N	UINT	모듈 착탈 슬롯 넘버
_IO_DEERR	ARRAY OF BYTE	모듈 착탈 위치
_FUSE_ER_N	UINT	Fuse 단선 슬롯 넘버
_FUSE_ERR	ARRAY OF BYTE	Fuse 단선 슬롯 위치
_IO_RWER_N	UINT	입출력 모듈 읽기/쓰기 에러 슬롯 넘버
_IO_RWERR	ARRAY OF BYTE	입출력 모듈 읽기/쓰기 에러 슬롯 위치
_IP_IFER_N	UINT	특수/링크 모듈 인터페이스 에러 슬롯 넘버
_IP_IFERR	ARRAY OF BYTE	특수/링크 모듈 인터페이스 에러 슬롯 위치
_ANC_ERR	ARRAY OF UINT	외부기기의 중고장 검출
_ANC_WAR	ARRAY OF UINT	외부기기의 경고장 검출
_ANC_WB	ARRAY OF BIT	외부기기의 경고장 검출 비트 Map
_TC_BMAP	ARRAY OF BYTE	태스크 충돌 표시
_TC_CNT	UINT	태스크 충돌 카운터
_BAT_ER_TM	DT	배터리 전압 저하 시각
_AC_F_CNT	UINT	전원 차단 카운터
_AC_F_TM	ARRAY OF DT	순시정전 이력

6) 시스템 운전 상태 정보

예 약 변 수	데 이 터 타 입	내 용
CPU_TYPE	UINT	시스템의 형태
VER_NUM	UINT	PLC O/S 버전 번호
MEM_TYPE	UINT	메모리 모듈의 타입
SYS_STATE	WORD	PLC 모드 및 상태
GMWIN_CN	BYTE	PADT 연결 상태
RST_TY	BYTE	리스타트 모드 정보
INIT_RUN	BIT	초기화 수행 중
SCAN_MAX	UINT	최장 스캔 시간(ms)
SCAN_MIN	UINT	최단 스캔 시간(ms)
SCAN_CUR	UINT	현재 스캔 시간(ms)
STSK_NUM	UINT	실행시간 확인을 요하는 태스크 넘버
STSK_MAX	UINT	최장 태스크 실행 시간(ms)
STSK_MIN	UINT	최단 태스크 실행 시간(ms)
STSK_CUR	UINT	현재 태스크 실행 시간(ms)
RTC_TIME	ARRAY OF BYTE	현재 시각
SYS_ERR	UINT	이상 종류

7) 통신 모듈 정보 플래그 [n 은 통신 모듈이 장착되어 있는 슬롯 번호에 해당 (n = 0 ~ 7)]

예 약 변 수	데 이 터 타 입	내 용
_CnVERNO	UINT	통신 모듈의 버젼 No.
_CnSTNOH _CnSTNOL	UINT	통신 모듈의 국번
_CnTXECNT	UINT	통신 프레임 전송 에러
_CnRXECNT	UINT	통신 프레임 수신 에러
_CnSVCFCNT	UINT	통신 서비스 처리 에러
_CnSCANMX	UINT	통신 스캔 타임 최대(1ms 단위)
_CnSCANAV	UINT	통신 스캔 타임 평균(1ms 단위)
_CnSCANMN	UINT	통신 스캔 타임 최소(1ms 단위)
_CnLINF	UINT	통신 모듈 시스템 정보
_CnCRDER	BOOL	통신 모듈의 시스템 에러(에러 = 1)
_CnSVBSY	BOOL	공용 RAM 자원 부족(부족=1)
_CnIFERR	BOOL	인터페이스 에러(에러 = 1)
_CnINRING	BOOL	통신 참여(IN_RING = 1)

8) 리모트 I/O 제어 플래그[m 은 통신 모듈이 장착되어 있는 슬롯 번호에 해당(m = 0 ~ 7)]

예 약 변 수	데 이 터 타 입	내 용
_FSMm_reset	BOOL(Write 가능)	리모트 I/O 국 리셋 제어(리셋=1)
_FSMm_io_reset	BOOL(Write 가능)	리모트 I/O 국의 출력 접점 리셋 제어(리셋=1)
_FSMm_st_no	USINT(Write 가능)	해당 리모트 I/O 국의 국번호

9) 고속 링크 정보 상세 플래그 [m 은 고속 링크 파라미터의 번호(m = 1,2,3,4)에 해당]

예 약 변 수	데 이 터 타 입	내 용
_HSmRLINK	BIT	고속 링크의 RUN_LINK 정보
_HSmLTRBL	BIT	고속 링크의 비정상 정보(Link Trouble)
_HSmSTATE	ARRAY OF BIT	고속 링크의 파라미터에서 k 데이터 블록의 종합적 통신 상태 정보
_HSmMOD	ARRAY OF BIT	고속 링크의 파라미터에서 k 데이터 블록에 설정된 국의 모드 정보 (Run = 1, 이외 = 0)
_HSmTRX	ARRAY OF BIT	고속 링크의 파라미터에서 k 데이터 블록의 통신 상태 정보 (정상 = 1, 비 정상 = 0)
_HSmERR	ARRAY OF BIT	고속 링크의 파라미터에서 k 데이터 블록에 설정된 국의 상태 정보 (정상 = 0, 에러 = 1)

2. 예약어

예약어는 시스템에서 사용하기 위해 미리 정의한 단어들입니다. 따라서 식별자로 이 예약어를 사용할 수 없습니다.

예 약 어
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
데이터 타입 이름
DATE#, D# DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
펑션 블록의 이름들
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
연산자 (IL 언어) 연산자 (ST 언어)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION
TYPE ... END_TYPE
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR VAR_IN_OUT ... END_VAR VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

부록B. GLOFA-GM 명령어집

제 1 장. 펑션 및 펑션블록 일람표

펑션과 펑션 블록에 대한 목록 요약입니다. 각각의 펑션과 펑션 블록에 대해서는 8. 기본 펑션, 펑션 블록 라이브러리를 참고 하십시오.

1.1. 기본 펑션

1.1.1. 타입 변환 펑션

각각의 입력 데이터 타입을 출력 데이터 타입으로 변환합니다.

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
ARY_ASC_TO_***	ARY_ASC_TO_BYTE	WORD(ASCII)	BYTE	○	○	○
	ARY_ASC_TO_BCD	WORD(ASCII)	BYTE(BCD)	○	○	○
ARY_BYTE_TO_***	ARY_BYTE_TO_ASC	BYTE	WORD(ASCII)	○	○	○
ARY_BCD_TO_***	ARY_BCD_TO_ ASC	BYTE(BCD)	WORD(ASCII)	○	○	○
ASC_TO_***	ASC_TO_BCD	BYTE(BCD)	USINT	○	○	○
	ASC_TO_BYTE	WORD(BCD)	UINT	○	○	○
BCD_TO_***	BCD_TO_SINT	BYTE(BCD)	SINT	○	○	○
	BCD_TO_INT	WORD(BCD)	INT	○	○	○
	BCD_TO_DINT	DWORD(BCD)	DINT	○	○	○
	BCD_TO_LINT	LWORD(BCD)	LINT	○		
	BCD_TO_USINT	BYTE(BCD)	USINT	○	○	○
	BCD_TO_UINT	WORD(BCD)	UINT	○	○	○
	BCD_TO_UDINT	DWORD(BCD)	UDINT	○	○	○
	BCD_TO_ULINT	LWORD(BCD)	ULINT	○		
	BCD_TO_ASC	BYTE(BCD)	WORD	○	○	○
TRUNC	TRUNC	REAL	DINT	○		
		LREAL	LINT	○		
REAL_TO_***	REAL_TO_SINT	REAL	SINT	○		
	REAL_TO_INT	REAL	INT	○		
	REAL_TO_DINT	REAL	DINT	○		
	REAL_TO_LINT	REAL	LINT	○		
	REAL_TO_USINT	REAL	USINT	○		
	REAL_TO_UINT	REAL	UINT	○		
	REAL_TO_UDINT	REAL	UDINT	○		
	REAL_TO_ULINT	REAL	ULINT	○		
	REAL_TO_DWORD	REAL	DWORD	○		
	REAL_TO_LREAL	REAL	LREAL	○		
LREAL_TO_***	LREAL_TO_SINT	LREAL	SINT	○		
	LREAL_TO_INT	LREAL	INT	○		
	LREAL_TO_DINT	LREAL	DINT	○		
	LREAL_TO_LINT	LREAL	LINT	○		
	LREAL_TO_USINT	LREAL	USINT	○		

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
LREAL_TO_***	LREAL_TO_UINT	LREAL	UINT	○		
	LREAL_TO_UDINT	LREAL	UDINT	○		
	LREAL_TO_ULINT	LREAL	ULINT	○		
	LREAL_TO_LWORD	LREAL	LWORD	○		
	LREAL_TO_REAL	LREAL	REAL	○		
SINT_TO_***	SINT_TO_INT	SINT	INT	○	○	○
	SINT_TO_DINT	SINT	DINT	○	○	○
	SINT_TO_LINT	SINT	LINT	○		
	SINT_TO_USINT	SINT	USINT	○	○	○
	SINT_TO_UINT	SINT	UINT	○	○	○
	SINT_TO_UDINT	SINT	UDINT	○	○	○
	SINT_TO_ULINT	SINT	ULINT	○		
	SINT_TO_BOOL	SINT	BOOL	○	○	○
	SINT_TO_BYTE	SINT	BYTE	○	○	○
	SINT_TO_WORD	SINT	WORD	○	○	○
	SINT_TO_DWORD	SINT	DWORD	○	○	○
	SINT_TO_LWORD	SINT	LWORD	○		
	SINT_TO_BCD	SINT	BYTE(BCD)	○	○	○
	SINT_TO_REAL	SINT	REAL	○		
	SINT_TO_LREAL	SINT	LREAL	○		
INT_TO_***	INT_TO_SINT	INT	SINT	○	○	○
	INT_TO_DINT	INT	DINT	○	○	○
	INT_TO_LINT	INT	LINT	○		
	INT_TO_USINT	INT	USINT	○	○	○
	INT_TO_UINT	INT	UINT	○	○	○
	INT_TO_UDINT	INT	UDINT	○	○	○
	INT_TO_ULINT	INT	ULINT	○		
	INT_TO_BOOL	INT	BOOL	○	○	○
	INT_TO_BYTE	INT	BYTE	○	○	○
	INT_TO_WORD	INT	WORD	○	○	○
	INT_TO_DWORD	INT	DWORD	○	○	○
	INT_TO_LWORD	INT	LWORD	○		
	INT_TO_BCD	INT	WORD(BCD)	○	○	○
	INT_TO_REAL	INT	REAL	○		
	INT_TO_LREAL	INT	LREAL	○		

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
DINT_TO_***	DINT_TO_SINT	DINT	SINT	○	○	○
	DINT_TO_INT	DINT	INT	○	○	○
	DINT_TO_LINT	DINT	LINT	○		
	DINT_TO_USINT	DINT	USINT	○	○	○
	DINT_TO_UINT	DINT	UINT	○	○	○
	DINT_TO_UDINT	DINT	UDINT	○	○	○
	DINT_TO_ULINT	DINT	ULINT	○		
	DINT_TO_BOOL	DINT	BOOL	○	○	○
	DINT_TO_BYTE	DINT	BYTE	○	○	○
	DINT_TO_WORD	DINT	WORD	○	○	○
	DINT_TO_DWORD	DINT	DWORD	○	○	○
	DINT_TO_LWORD	DINT	LWORD	○		
	DINT_TO_BCD	DINT	DWORD(BCD)	○	○	○
	DINT_TO_REAL	DINT	REAL	○		
	DINT_TO_LREAL	DINT	LREAL	○		
LINT_TO_***	LINT_TO_SINT	LINT	SINT	○		
	LINT_TO_INT	LINT	INT	○		
	LINT_TO_DINT	LINT	DINT	○		
	LINT_TO_USINT	LINT	USINT	○		
	LINT_TO_UINT	LINT	UINT	○		
	LINT_TO_UDINT	LINT	UDINT	○		
	LINT_TO_ULINT	LINT	ULINT	○		
	LINT_TO_BOOL	LINT	BOOL	○		
	LINT_TO_BYTE	LINT	BYTE	○		
	LINT_TO_WORD	LINT	WORD	○		
	LINT_TO_DWORD	LINT	DWORD	○		
	LINT_TO_LWORD	LINT	LWORD	○		
	LINT_TO_BCD	LINT	LWORD(BCD)	○		
	LINT_TO_REAL	LINT	REAL	○		
	LINT_TO_LREAL	LINT	LREAL	○		
USINT_TO_***	USINT_TO_SINT	USINT	SINT	○	○	○
	USINT_TO_INT	USINT	INT	○	○	○
	USINT_TO_DINT	USINT	DINT	○	○	○
	USINT_TO_LINT	USINT	LINT	○		
	USINT_TO_UINT	USINT	UINT	○	○	○
	USINT_TO_UDINT	USINT	UDINT	○	○	○
	USINT_TO_ULINT	USINT	ULINT	○		
	USINT_TO_BOOL	USINT	BOOL	○	○	○
	USINT_TO_BYTE	USINT	BYTE	○	○	○
	USINT_TO_WORD	USINT	WORD	○	○	○
	USINT_TO_DWORD	USINT	DWORD	○	○	○
	USINT_TO_LWORD	USINT	LWORD	○		

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
USINT_TO_***	USINT_TO_BCD	USINT	BYTE(BCD)	○	○	○
	USINT_TO_REAL	USINT	REAL	○		
	USINT_TO_LREAL	USINT	LREAL	○		
UINT_TO_***	UINT_TO_SINT	UINT	SINT	○	○	○
	UINT_TO_INT	UINT	INT	○	○	○
	UINT_TO_DINT	UINT	DINT	○	○	○
	UINT_TO_LINT	UINT	LINT	○		
	UINT_TO_USINT	UINT	USINT	○	○	○
	UINT_TO_UDINT	UINT	UDINT	○	○	○
	UINT_TO_ULINT	UINT	ULINT	○		
	UINT_TO_BOOL	UINT	BOOL	○	○	○
	UINT_TO_BYTE	UINT	BYTE	○	○	○
	UINT_TO_WORD	UINT	WORD	○	○	○
	UINT_TO_DWORD	UINT	DWORD	○	○	○
	UINT_TO_LWORD	UINT	LWORD	○		
	UINT_TO_BCD	UINT	WORD(BCD)	○	○	○
	UINT_TO_REAL	UINT	REAL	○		
	UINT_TO_LREAL	UINT	LREAL	○		
	UINT_TO_DATE	UINT	DATE	○	○	○
UDINT_TO_***	UDINT_TO_SINT	UDINT	SINT	○	○	○
	UDINT_TO_INT	UDINT	INT	○	○	○
	UDINT_TO_DINT	UDINT	DINT	○	○	○
	UDINT_TO_LINT	UDINT	LINT	○		
	UDINT_TO_USINT	UDINT	USINT	○	○	○
	UDINT_TO_UINT	UDINT	UINT	○	○	○
	UDINT_TO_ULINT	UDINT	ULINT	○		
	UDINT_TO_BOOL	UDINT	BOOL	○	○	○
	UDINT_TO_BYTE	UDINT	BYTE	○	○	○
	UDINT_TO_WORD	UDINT	WORD	○	○	○
	UDINT_TO_DWORD	UDINT	DWORD	○	○	○
	UDINT_TO_LWORD	UDINT	LWORD	○		
	UDINT_TO_BCD	UDINT	DWORD(BCD)	○	○	○
	UDINT_TO_REAL	UDINT	REAL	○		
	UDINT_TO_LREAL	UDINT	LREAL	○		
	UDINT_TO_TOD	UDINT	TOD	○	○	○
	UDINT_TO_TIME	UDINT	TIME	○	○	○
ULINT_TO_***	ULINT_TO_SINT	ULINT	SINT	○		
	ULINT_TO_INT	ULINT	INT	○		
	ULINT_TO_DINT	ULINT	DINT	○		
	ULINT_TO_LINT	ULINT	LINT	○		
	ULINT_TO_USINT	ULINT	USINT	○		
	ULINT_TO_UINT	ULINT	UINT	○		

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
ULINT_TO_***	ULINT_TO_UDINT	ULINT	UDINT	○		
	ULINT_TO_BOOL	ULINT	BOOL	○		
	ULINT_TO_BYTE	ULINT	BYTE	○		
	ULINT_TO_WORD	ULINT	WORD	○		
	ULINT_TO_DWORD	ULINT	DWORD	○		
	ULINT_TO_LWORD	ULINT	LWORD	○		
	ULINT_TO_BCD	ULINT	LWORD(BCD)	○		
	ULINT_TO_REAL	ULINT	REAL	○		
	ULINT_TO_LREAL	ULINT	LREAL	○		
BOOL_TO_***	BOOL_TO_SINT	BOOL	SINT	○	○	○
	BOOL_TO_INT	BOOL	INT	○	○	○
	BOOL_TO_DINT	BOOL	DINT	○	○	○
	BOOL_TO_LINT	BOOL	LINT	○		
	BOOL_TO_USINT	BOOL	USINT	○	○	○
	BOOL_TO_UINT	BOOL	UINT	○	○	○
	BOOL_TO_UDINT	BOOL	UDINT	○	○	○
	BOOL_TO_ULINT	BOOL	ULINT	○		
	BOOL_TO_BYTE	BOOL	BYTE	○	○	○
	BOOL_TO_WORD	BOOL	WORD	○	○	○
	BOOL_TO_DWORD	BOOL	DWORD	○	○	○
	BOOL_TO_LWORD	BOOL	LWORD	○		
	BOOL_TO_STRING	BOOL	STRING	○	○	○
BYTE_TO_***	BYTE_TO_SINT	BYTE	SINT	○	○	○
	BYTE_TO_INT	BYTE	INT	○	○	○
	BYTE_TO_DINT	BYTE	DINT	○	○	○
	BYTE_TO_LINT	BYTE	LINT	○		
	BYTE_TO_USINT	BYTE	USINT	○	○	○
	BYTE_TO_UINT	BYTE	UINT	○	○	○
	BYTE_TO_UDINT	BYTE	UDINT	○	○	○
	BYTE_TO_ULINT	BYTE	ULINT	○		
	BYTE_TO_BOOL	BYTE	BOOL	○	○	○
	BYTE_TO_WORD	BYTE	WORD	○	○	○
	BYTE_TO_DWORD	BYTE	DWORD	○	○	○
	BYTE_TO_LWORD	BYTE	LWORD	○		
	BYTE_TO_STRING	BYTE	STRING	○	○	○
	BYTE_TO_ASC	BYTE	WORD(ASCII)			
WORD_TO_***	WORD_TO_SINT	WORD	SINT	○	○	○
	WORD_TO_INT	WORD	INT	○	○	○
	WORD_TO_DINT	WORD	DINT	○	○	○
	WORD_TO_LINT	WORD	LINT	○		
	WORD_TO_USINT	WORD	USINT	○	○	○
	WORD_TO_UINT	WORD	UINT	○	○	○

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
WORD_TO_***	WORD_TO_UDINT	WORD	UDINT	○	○	○
	WORD_TO_ULINT	WORD	ULINT	○		
	WORD_TO_BOOL	WORD	BOOL	○	○	○
	WORD_TO_BYTE	WORD	BYTE	○	○	○
	WORD_TO_DWORD	WORD	DWORD	○	○	○
	WORD_TO_LWORD	WORD	LWORD	○		
	WORD_TO_DATE	WORD	DATE	○	○	○
	WORD_TO_STRING	WORD	STRING	○	○	○
DWORD_TO_***	DWORD_TO_SINT	DWORD	SINT	○	○	○
	DWORD_TO_INT	DWORD	INT	○	○	○
	DWORD_TO_DINT	DWORD	DINT	○	○	○
	DWORD_TO_LINT	DWORD	LINT	○		
	DWORD_TO_USINT	DWORD	USINT	○	○	○
	DWORD_TO_UINT	DWORD	UINT	○	○	○
	DWORD_TO_UDINT	DWORD	UDINT	○	○	○
	DWORD_TO_ULINT	DWORD	ULINT	○		
	DWORD_TO_BOOL	DWORD	BOOL	○	○	○
	DWORD_TO_BYTE	DWORD	BYTE	○	○	○
	DWORD_TO_WORD	DWORD	WORD	○	○	○
	DWORD_TO_LWORD	DWORD	LWORD	○		
	DWORD_TO_REAL	DWORD	REAL	○		
	DWORD_TO_TIME	DWORD	TIME	○	○	○
	DWORD_TO_TOD	DWORD	TOD	○	○	○
	DWORD_TO_STRING	DWORD	STRING	○	○	○
LWORD_TO_***	LWORD_TO_SINT	LWORD	SINT	○		
	LWORD_TO_INT	LWORD	INT	○		
	LWORD_TO_DINT	LWORD	DINT	○		
	LWORD_TO_LINT	LWORD	LINT	○		
	LWORD_TO_USINT	LWORD	USINT	○		
	LWORD_TO_UINT	LWORD	UINT	○		
	LWORD_TO_UDINT	LWORD	UDINT	○		
	LWORD_TO_ULINT	LWORD	ULINT	○		
LWORD_TO_***	LWORD_TO_BOOL	LWORD	BOOL	○		
	LWORD_TO_BYTE	LWORD	BYTE	○		
	LWORD_TO_WORD	LWORD	WORD	○		
	LWORD_TO_DWORD	LWORD	DWORD	○		
	LWORD_TO_LREAL	LWORD	LREAL	○		
	LWORD_TO_DT	LWORD	DT	○		
	LWORD_TO_STRING	LWORD	STRING	○		
STRING_TO_***	STRING _TO_SINT	STRING	SINT	○	○	○
	STRING _TO_INT	STRING	INT	○	○	○
	STRING _TO_DINT	STRING	DINT	○	○	○

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	적용 기종		
				GMR~2	GM3	GM4~7
STRING_TO_***	STRING _TO_LINT	STRING	LINT	○		
	STRING _TO_USINT	STRING	USINT	○	○	○
	STRING _TO_UINT	STRING	UINT	○	○	○
	STRING _TO_UDINT	STRING	UDINT	○	○	○
	STRING _TO_ULINT	STRING	ULINT	○		
	STRING _TO_BOOL	STRING	BOOL	○	○	○
	STRING _TO_BYTE	STRING	BYTE	○	○	○
	STRING _TO_WORD	STRING	WORD	○	○	○
	STRING _TO_DWORD	STRING	DWORD	○	○	○
	STRING _TO_LWORD	STRING	LWORD	○		
	STRING _TO_REAL	STRING	REAL	○		
	STRING _TO_LREAL	STRING	LREAL	○		
	STRING _TO_DT	STRING	DT	○	○	○
	STRING _TO_DATE	STRING	DATE	○	○	○
	STRING _TO_TOD	STRING	TOD	○	○	○
	STRING _TO_TIME	STRING	TIME	○	○	○
NUM_TO_STRING	NUM_TO_STRING	ANY_NUM	STRING	○	○	○
TIME_TO_***	TIME_TO_UDINT	TIME	UDINT	○	○	○
	TIME_TO_DWORD	TIME	DWORD	○	○	○
	TIME_TO_STRING	TIME	STRING	○	○	○
DATE_TO_***	DATE_TO_UINT	DATE	UINT	○	○	○
	DATE_TO_WORD	DATE	WORD	○	○	○
	DATE_TO_STRING	DATE	STRING	○	○	○
TOD_TO_***	TOD_TO_UDINT	TOD	UDINT	○	○	○
	TOD_TO_DWORD	TOD	DWORD	○	○	○
	TOD_TO_STRING	TOD	STRING	○	○	○
DT_TO_***	DT_TO_LWORD	DT	LWORD	○		
	DT_TO_DATE	DT	DATE	○	○	○
	DT_TO_TOD	DT	TOD	○	○	○
	DT_TO_STRING	DT	STRING	○	○	○

1.1.2.수치 연산 펑션

1.1.2.1. 하나의 입력을 갖는 수치 연산 펑션

GMR, GM1, GM2 에서만 지원됩니다.(ABS 는 GM3, GM4, GM6, GM7 에서도 지원됩니다.)

No.	펑션 이름	기 능
일반 펑션		
1	ABS	절대값 연산(Absolute Value)
2	SQRT	제곱근 연산(Square Root)
로그 펑션		
3	LN	자연 대수 연산(Natural Logarithm)
4	LOG	상용 대수 연산(Logarithm Base To 1○)
5	EXP	자연 지수 연산(Natural Exponential)
삼각 펑션		
6	SIN	사인값 연산(Sine)
7	COS	코사인값 연산(Cosine)
8	TAN	탄젠트값 연산(Tangent)
9	ASIN	아크 사인값 연산(Arc Sine)
10	ACOS	아크 코사인값 연산(Arc Cosine)
11	ATAN	아크 탄젠트값 연산(Arc Tangent)
각도 펑션		
12	RAD_REAL	각도의 단위를 (°)에서 라디안(Radian)으로 변환
13	RAD_LREAL	
14	DEG_REAL	라디안(Radian)값을 각도(°)로 변환
15	DEG_LREAL	

1.1.2.2. 기본 수치 연산 펑션

EXPT 는 GMR,GM1, GM2 에서만 지원됩니다..(XCHG_***는 GM3,GM4,GM6,GM7 에서도 지원됩니다.)

No.	펑션 이름	기 능
입력 개수를 확장할 수 있는 연산 펑션(단,n은 8까지 가능)		
1	ADD	더하기 (OUT <= IN1 + IN2 + ... + INn)
2	MUL	곱하기 OUT <= IN1 * IN2 * ... * INn)
입력 개수가 일정한 연산 펑션		
3	SUB	빼기(OUT <= IN1 - IN2)
4	DIV	나누기(OUT <= IN1 / IN2)
5	MOD	나머지 구하기(OUT <= IN1 Modulo IN2)
6	EXPT	지수 연산(OUT <= $IN1^{IN2}$)
7	MOVE	데이터 복사(OUT <= IN)
입력 데이터 값 교환		
8	XCHG_***	입력 데이터 값을 서로 교환

1.1.3.비트열 펑션

1.1.3.1. 비트 시프트 펑션

No.	펑션 이름	기 능
1	SHL	입력을 N 비트 왼쪽으로 이동(오른쪽은 0 으로 채움)
2	SHR	입력을 N 비트 오른쪽으로 이동(왼쪽은 0 으로 채움)
3	SHIFT_C_***	입력을 N 비트만큼 지정된 방향으로 이동(Carry 발생)
4	ROL	입력을 N 비트 왼쪽으로 회전
5	ROR	입력을 N 비트 오른쪽으로 회전
6	ROTATE_C_***	입력을 N 비트만큼 지정된 방향으로 회전(Carry 발생)

1.1.3.2. 비트 연산 펑션

No.	펑션 이름	기 능(단, n은 8까지 가능)
1	AND	논리곱(OUT <= IN1 AND IN2 AND ... AND INn)
2	OR	논리합(OUT <= IN1 OR IN2 OR ... OR INn)
3	XOR	배타적 논리합(OUT <= IN1 XOR IN2 XOR ... XOR INn)
4	NOT	논리 반전(OUT <= NOT IN1)

1.1.4.선택 펑션

No.	펑션 이름	기 능(단, n은 8까지 가능)
1	SEL	입력 IN0 와 IN1 중에 선택하여 출력
2	MAX	입력 IN1,...INn 중에 최대값 출력
3	MIN	입력 IN1,...INn 중에 최소값 출력
4	LIMIT	상하한 제한값 출력
5	MUX	입력 IN0, ... INn 중 K 번째 입력을 출력

1.1.5.데이터 교환 펑션

No.	펑션 이름	기 능
1	SWAP_BYTE	BYTE 의 상 · 하위 Nibble 을 교환하여 출력
	SWAP_WORD	WORD 의 상 · 하위 BYTE 를 교환하여 출력
	SWAP_DWORD	DWORD 의 상 · 하위 WORD 를 교환하여 출력
	SWAP_LWORD	LWORD 의 상 · 하위 DWORD 를 교환하여 출력
2	ARY_SWAP_BYTE	Array 로 입력된 BYTE 의 상 · 하위 Nibble 을 교환하여 출력
	ARY_SWAP_WORD	Array 로 입력된 WORD 의 상 · 하위 BYTE 를 교환하여 출력
	ARY_SWAP_DWORD	Array 로 입력된 DWORD 의 상 · 하위 WORD 를 교환하여 출력
	ARY_SWAP_LWORD	Array 로 입력된 LWORD 의 상 · 하위 DWORD 를 교환하여 출력

1.1.6.비교 펑션

No.	펑션 이름	기 능(단, n은 8까지 가능)
1	GT	'크다' 비교 OUT <= (IN1>IN2) & (IN2>IN3) & ... & (INn-1 > INn)
2	GE	'크거나 작다' 비교 OUT <= (IN1>=IN2) & (IN2>=IN3) & ... & (INn-1 >= INn)
3	EQ	'같다' 비교 OUT <= (IN1=IN2) & (IN2=IN3) & ... & (INn-1 = INn)
4	LE	'작거나 같다' 비교 OUT <= (IN1<=IN2) & (IN2<=IN3) & ... & (INn-1 <= INn)
5	LT	'작다' 비교 OUT <= (IN1<IN2) & (IN2<IN3) & ... & (INn-1 < INn)
6	NE	'같지 않다' 비교 OUT <= (IN1<>IN2) & (IN2<>IN3) & ... & (INn-1 <> INn)

1.1.7.문자열 펑션

No.	펑션 이름	기 능
1	LEN	입력 문자열의 길이 구하기
2	LEFT	입력 문자열을 왼쪽으로부터 L 만큼 출력
3	RIGHT	입력 문자열을 오른쪽으로부터 L 만큼 출력
4	MID	입력 문자열의 P 번째부터 L 만큼 출력
5	CONCAT	입력 문자열을 붙여 출력
6	INSERT	첫번째 입력 문자열의 P 번째 문자 뒤에 두번째 입력 문자열을 삽입하여 출력
7	DELETE	입력 문자열의 P 번째 문자부터 L 개 문자를 삭제하여 출력
8	REPLACE	첫번째 입력 문자열의 P 번째 문자부터 L 개 문자를 두번째 입력 문자열으로 대치하여 출력
9	FIND	첫번째 입력 문자열중에 두번째 입력 문자열 패턴과 동일한 부분을 찾아 시작 문자 위치를 출력

1.1.8.날짜 시각 펑션

No.	펑션 이름	기 능
1	ADD_TIME	시간, 시각 및 날짜 시각에 시간 더하기
2	SUB_TIME	시간, 시각 및 날짜 시각에 시간 빼기
	SUB_DATE	날짜에서 날짜를 빼서 시간 산출하기
	SUB_TOD	시각에서 시각을 빼서 시간 산출하기
	SUB_DT	날짜 시각에서 날짜 시각을 빼서 시간 산출하기
3	MUL_TIME	시간에 숫자 곱하기
4	DIV_TIME	시간에 숫자 나누기
5	CONCAT_TIME	날짜와 시각을 붙여서 날짜 시각 만들기

1.1.9.시스템 제어 펑션

No.	펑션 이름	기 능
1	DI	태스크 프로그램 기동 불허
2	EI	태스크 프로그램 기동 허가
3	STOP	프로그램에 의한 운전정지
4	ESTOP	프로그램에 의한 비상 운전정지
5	DIREC_IN	입력 데이터 즉시 갱신(GM1- GM7에 적용)
6	DIREC_O	출력 모듈 데이터 즉시 갱신(GM1- GM7에 적용)
7	WDT_RST	Watch_Dog 타이머 갱신
8	MCS	Master Control
9	MCSCLR	Master Control Clear

1.1.10. 데이터 조작 명령 펑션

No.	펑션 이름	기 능
1	MEQ_***	입력 값을 Masking 한 후 이 값들을 비교
2	DIS_***	입력 값들을 지정된 Bit 개수 단위로 출력
3	UNI_***	Array로 입력된 값을 지정된 Bit수만큼 결합
4	BIT_BYTE	8개의 Bit들을 BYTE로 모음
5	BYTE_BIT	BYTE를 8개의 Bit로 나눔
6	BYTE_WORD	2개의 BYTE들을 WORD로 모음
7	WORD_BYTE	WORD를 2개의 BYTE로 나눔
8	WORD_DWORD	2개의 WORD들을 DWORD로 모음
9	DWORD_WORD	DWORD를 2개의 WORD로 나눔
10	DWORD_LWORD	2개의 DWORD들을 LWORD로 모음
11	LWORD_DWORD	LWORD를 2개의 DWORD로 나눔
12	GET_CHAR	지정된 문자열로부터 한문자(Character)를 추출
13	PUT_CHAR	지정된 문자를 지정된 문자열에 쓰기
14	STRING_TO_ARY	지정된 문자열을 BYTE Array로 변환
15	ARY_TO_STRING	BYTE Array를 지정된 문자열로 변환

1.1.11. 스택 연산 명령 펑션

No.	펑션 이름	기 능
1	FIFO_***	선입 선출 명령
2	LIFO_***	후입 선출 명령

1.2. MK(MASTER-K) 펑션

No.	펑션 이름	기 능(단, n은 8까지 가능)
1	ENCO_B,W,D,L	ON된 비트 위치를 숫자로 출력
2	DECO_B,W,D,L	지정된 비트 위치를 ON
3	BSUM_B,W,D,L	ON된 비트 개수를 숫자로 출력
4	SEG	BCD 또는 HEX값을 7세그먼트 디스플레이 코드로 변환
5	BMOV_B,W,D,L	비트 스트링의 일부분을 복사,이동
6	INC_B,W,D,L	IN 데이터를 하나 증가
7	DEC_B,W,D,L	IN 데이터를 하나 감소

1.3. Array 연산 명령 펑션

No.	펑션 이름	기 능
1	ARY_MOVE	Array Type의 데이터 복사(OUT <= IN)
2	ARY_CMP_***	2개의 Array로 입력된 값을 동일한 값이 있는지 비교
3	ARY_SCH_***	Array내에서 입력된 값과 동일한 값을 찾아 출력
4	ARY_FLL_***	입력 데이터 값으로 Array내부의 선택 영역을 채움.
5	ARY_AVE_***	Array내부의 지정된 영역에 대하여 평균값을 구함
6	ARY_SFT_C_***	Array의 Bit들을 정해진 개수만큼 지정된 방향으로 이동
7	ARY_ROT_C_***	Array의 Bit들을 정해진 개수만큼 지정된 방향으로 회전
8	SHIFT_A_***	Array 블록 중 지정된 범위의 값들을 지정된 방향으로 이동
9	ROTATE_A_***	Array 블록 중 지정된 범위의 값들을 지정된 방향으로 회전

1.4. 기본 펑션 블록

1.4.1.바이스테이블 펑션 블록

No.	펑션 블록 이름	기 능
1	SR	세트 우선 쌍안정 출력
2	RS	리세트 우선 쌍안정 출력
3	SEMA	시스템 자원 제어용 Semaphore

1.4.2.에지 검출 펑션 블록

No.	펑션 블록 이름	기 능
1	R_TRIG	상승 에지 검출(Rising Edge Detector)
2	F_TRIG	하강 에지 검출(Falling Edge Detector)

1.4.3.카운터

No.	펑션 블록 이름	기 능
1	CTU	가산 카운터(Up Counter)
2	CTD	감산 카운터(Down Counter)
3	CTUD	가감산 카운터(Up Down Counter)
4	CTR	링 카운터(Ring Counter)

1.4.4.타이머

No.	펑션 블록 이름	기 능
1	TP	펄스 타이머(Pulse Timer)
2	TON	On 딜레이 타이머(On-Delay Timer)
3	TOF	Off 딜레이 타이머(Off-Delay Timer)
4	TMR	적산 타이머(Integrating Timer)
5	TP_RST	펄스 타이머의 출력 Off가 가능한 노스테이블 타이머
6	TRTG	리트리거블 타이머(Retriggerable Timer)
7	TOF_RST	동작중 출력 Off가 가능한 Off 딜레이 타이머(Off-Delay Timer)
8	TON_UNIT	정수 설정 On 딜레이 타이머(On-Delay Timer)
9	TOF_UNIT	정수 설정 Off 딜레이 타이머(Off-Delay Timer)
10	TP_UNIT	정수 설정 펄스 타이머(Pulse Timer)
11	TMR_UNIT	정수 설정 적산 타이머(Integrating Timer)

1.4.5.기타 펑션 블록

No.	펑션 블록 이름	기 능
1	SCON	순차 스탭 및 스텝 점프
2	DUTY	지정된 Scan마다 On/Off 반복

제 2 장. 기본 펑션 및 펑션블록 라이브러리

2.1 기본 펑션 라이브러리

각각의 기본 펑션 라이브러리에 대한 설명입니다.

포인트 펑션 에러 발생시 다음을 참고하시어 프로그램을 작성 하십시오.

▷ 펑션 에러

펑션을 실행할 때 에러가 발생하면 ENO가 0이 되고,에러 플래그(_ERR, _LER)는 1이 됩니다.
에러가 발생되지않는 펑션의 ENO는 EN을 그대로 출력합니다.EN,ENO는 LD에서만 사용합니다.

▷ 에러 플래그

_ERR (Error)

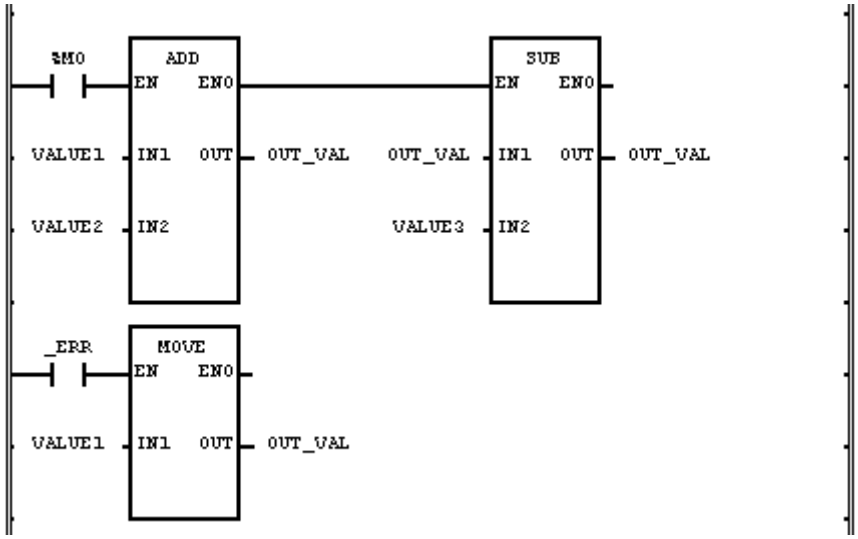
- 에러가 명시되어 있는 펑션의 연산 후 _ERR 값은 아래와 같이 변합니다.
  (에러가 명시되어 있지 않은 펑션은 연산 전의 _ERR 상태를 유지합니다.)
- 연산 에러인 경우, 1이 됩니다.
- 연산 에러가 아닌 경우, 0이 됩니다.

_LER (Latched Error)

- 연산 후 에러인 경우에 1이 되고, 그 값은 수행중인 프로그램의 끝까지 유지됩니다.
- 프로그램에서 0으로 써 넣는 것이 가능합니다.

■프로그램 예

ADD 펑션 에러시 SUB 펑션을 수행하지 않고, VALUE1의 값을 OUT_VAL에 저장하는 프로그램입니다.



(1)펑션(ADD)의 두입력이 아래와 같을 때, 펑션(ADD)는 에러가 발생합니다.
입력(IN1) : VALUE1(SINT) = 100(16#64)
(IN2) : VALUE2(SINT) = 50(16#32)
출력(OUT) : OUT_VAL(SINT) = -106(16#96)

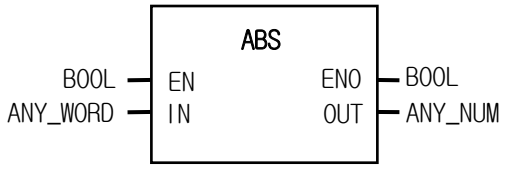
(2)출력값이 출력 데이타 타입의 범위를 벗어나, 비 정상적인 값이 OUT_VAL(SINT)에 저장됩니다.
이때, 펑션(ADD)의ENO는 0이 되어 펑션(SUB)는 실행되지 않고, 에러 플래그 _ERR과, _LER은 On 됩니다.

(3)_ERR이 On 되어, 펑션(MOVE)이 실행됩니다.
입력(IN1): VALUE1(SINT) = 100(16#64)
출력(OUT): OUT_VAL(SINT) = 100(16#64)

ABS

절대값 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 절대값 연산의 입력값 **출력** ENO 1을 출력 OUT : 절대값 IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

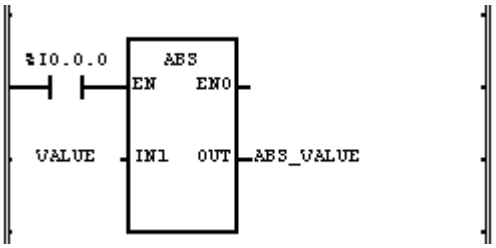
IN의 값을.
X의 절대값 |X| 는
X≥0 이면 |X| = X 이고,
X<0 이면 |X| = -X 입니다.
OUT = |IN|

■ 에러

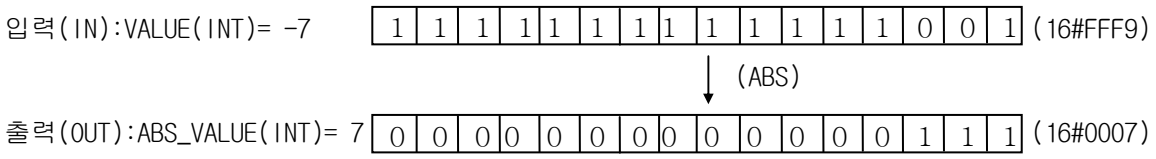
IN의 값이 (-)최소값일 때 _ERR, _LER 플래그가 셋(Set)됩니다.
예) 데이터 타입이 SINT일 때 IN의 값이 -128이면 에러

■ 프로그램 예

LD



(1)실행조건(% I0.0.0)이 On하면 펑션 ABS가 실행됩니다.
(2)VALUE = -7 이면, ABS_VALUE = |-7| = 7 이 됩니다.
VALUE = 200 이면, ABS_VALUE = |200| = 200 이 됩니다.

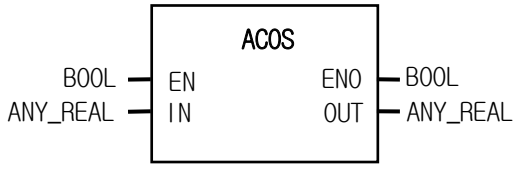


INT형의 음수 표시는 2의 Complement 표현 (3.2.4.데이터 타입별 구조 참조)

ACOS

Arc Cosine 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Cosine 연산의 입력값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산결과 각도출력 값(Radian) IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

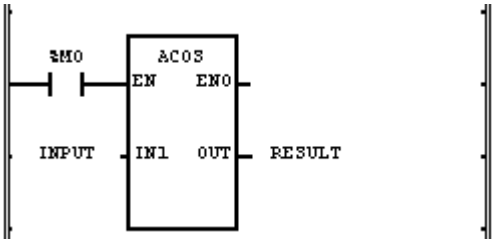
IN의 Arc Cosine값을 구해 OUT으로 출력시킵니다. 출력값은 0에서 π사이의 값이 됩니다.
OUT = ACOS (IN)

■ 에러

IN1의 범위가 -1.0부터 1.0 사이에 있지 않은 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 Arc Cosine연산 펑션 ACOS가 실행됩니다.
(2)INPUT 변수가 0.8660 .... ($\sqrt{3}$ / 2) 일 때 RESULT는 0.5235 ... (π/6 rad = 30°)입니다.

$ACOS(\sqrt{3} / 2) = \pi/6$

$(COS\ \pi/6 = \sqrt{3} /2)$

입력(IN1) : INPUT(REAL) = 0.866

↓ (ACOS)

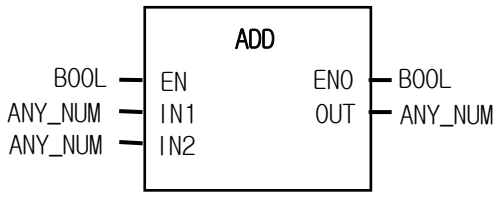
출력(OUT) : RESULT (REAL) = 5.23499966E-01

REAL형의 표시는 IEEE Standard 754-1984 기준 (3.2.4.데이터 타입별 구조 참조)

ADD

더하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 더해질 값 IN2 : 더할 값 입력은 8개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 더한 결과 값 IN1, IN2, ..., OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

■ 기능

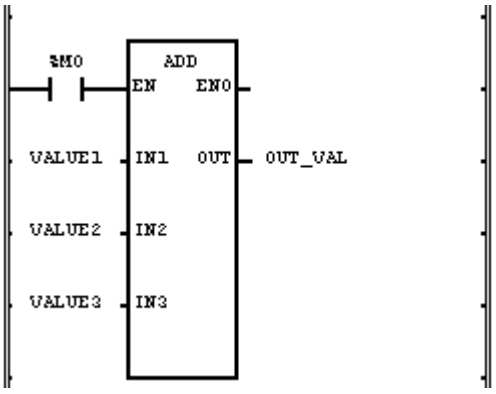
IN1, IN2,..., INn (n은 입력 개수)를 더해서 OUT으로 출력시킵니다.
OUT = IN1 + IN2 + ... + INn

■ 에러

출력값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 더하기 펑션 ADD가 실행됩니다.
(2)입력변수 값이VALUE1 = 300, VALUE2 = 200, VALUE3 = 100이면, 출력변수로 설정한 OUT_VAL = 300 + 200 + 100 = 600이 됩니다.

입력(IN1): VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

+ (ADD)

(IN2): VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

+ (ADD)

(IN2): VALUE3(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0

↓

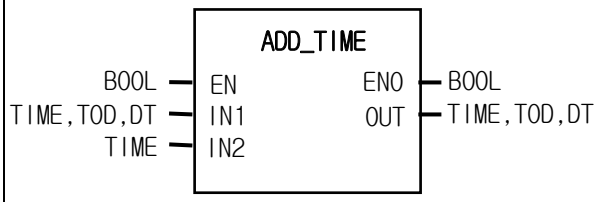
(OUT): OUT_VAL(INT) = 600(16#0258)

0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

ADD_TIME

시간 더하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준시각 또는 시간 IN2 : 더할 시간 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 더한 결과 시각 또는 시간 OUT의 타입은 입력 IN1의 타입을 따름. 즉 IN1의 타입이 TIME_OF_DAY이면, 출력 OUT의 타입도 TIME_OF_DAY임.

■ **기능**

▷ IN1이 TIME일 경우에는 시간과 시간을 더해서 합한 시간을 출력시킵니다.

▷ IN1이 TIME_OF_DAY일 경우에는 기준시각에 시간을 더해서 하루중의 시각을 출력시킵니다.

▷ IN1이 DATE_AND_TIME일 경우에는 기준이 되는 날짜와 시각에 시간을 더해서 날짜와 시각을 출력시킵니다.

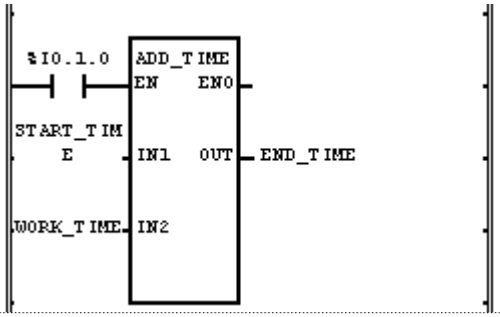
■ **에러**

▷ 출력값이 해당 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

▷ 시간과 시간을 더한 결과가 TIME 데이터 타입의 범위 T#49D17H2M47S295MS를 넘거나 시각(TOD)과시간을 더한 결과가 24시를 넘을 경우, 또는 날짜와 시각(DT)과 시간을 더한 결과가 2083년을 넘을 경우 에러가 됩니다.

■ **프로그램 예**

LD



(1)실행조건(%I0.1.0)이 On하면 시간 더하기 펑션 ADD_TIME 이 실행됩니다.

(2)작업을 시작한 START_TIME이 TOD#08:30:00이고 작업한 시간 WORK_TIME 이 T#2H10M20S500MS이면 작업이 종료된 시각 END_TIME에는 TOD#10:40:20.5가 출력됩니다.

입력(IN1) : START_TIME(TOD) = TOD#08:30:00

+ (ADD_TIME)

(IN2) : WORK_TIME(TIME) = T#2H10M20S500MS

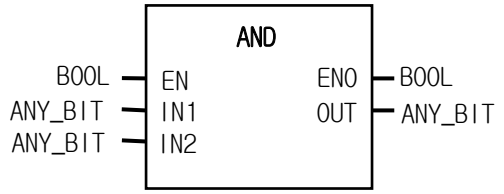
↓

출력(OUT) : END_TIME(TOD) = TOD#10:40:20.5

AND

논리곱

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : AND될 값 IN2 : AND될 값 입력이 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : AND된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

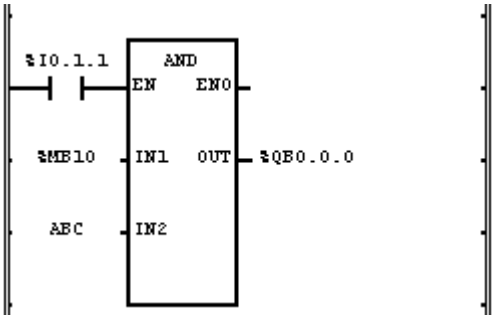
■ 기능

▷IN1을 IN2와 비트별로 AND해서 OUT으로 출력시킵니다.

IN1 1111 0000
&
IN2 1010 1010
OUT 1010 0000

■ **프로그램 예**

LD



(1)실행조건(% I0.1.1)이 On하면 펑션 AND가 실행됩니다.
(2)IN1= %MBRQ과 IN2 = ABC값을 AND시킨 결과가 OUT = %QB0.0.0에 출력됩니다.

입력(IN1) : %MB10 (BYTE) = 16#CC

1	1	0	0	1	1	0	0

& (AND)

(IN2) : ABC(BYTE) = 16#F0

1	1	1	1	0	0	0	0

↓

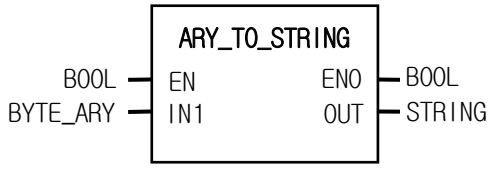
출력(OUT) : %QB0.0.0(BYTE) = 16#C0

1	1	1	1	0	0	0	0

ARY_TO_STRING

Byte Array를 문자열로 변환

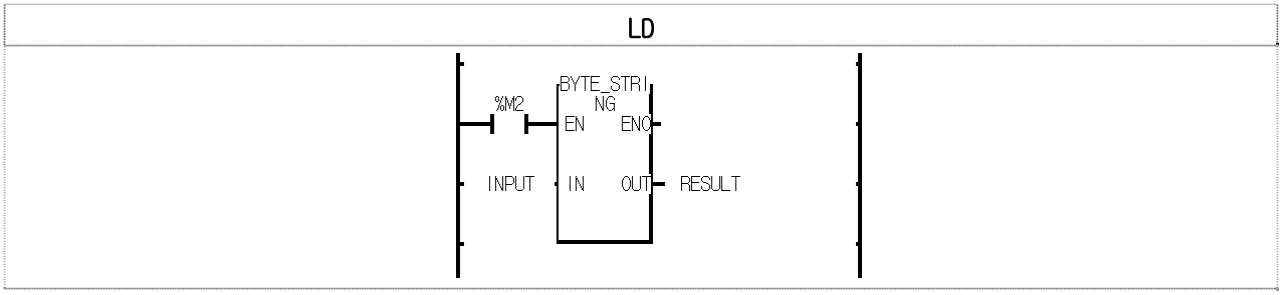
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : Byte Array입력 **출력** ENO : EN값을 그대로 출력 OUT : 변환된 String 출력

■ 기능

Byte Array를 하나의 String으로 변환합니다.

■ 프로그램 예



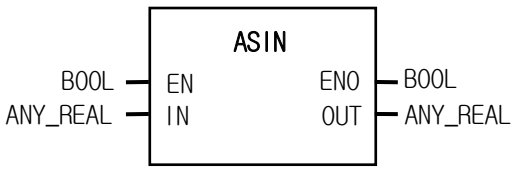
(1)실행조건(%M2)이 On하면 BYTE_STRING 펑션이 실행됩니다.

(2)입력변수인Input이 하위 바이트부터 순서대로
16#{22(“),47(G),4D(M),34(4),2D(-),43(C),50(P),55(U),41(A),22(“)}일 때 출력 변수인 RESULT로 “GM4-CPUA”가 출력됩니다.

ASIN

Arc Sine 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Sine 연산의 입력값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산결과 각도출력 값(Radian) IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

IN의 Arc Sine값을 구해 OUT으로 출력시킵니다. 출력 값은 $-\pi/2$에서 $\pi/2$ 사이의 값이 됩니다.

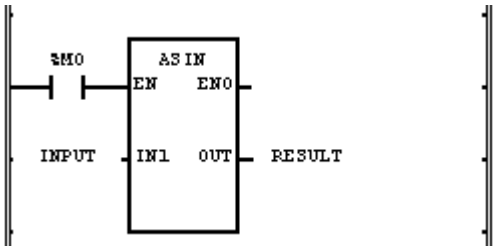
OUT = ASIN (IN)

■ 에러

입력 값의 범위가 -1.0과 1.0 사이에 있지 않을 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 Arc Sine 연산 펑션 ASIN가 실행됩니다.

(2)INPUT 변수가 0.8660 .... ($\sqrt{3}/2$) 일 때 펑션의 출력변수로 선언된RESULT는 1.0471 ....

($\pi/3$ rad = 60°)입니다.

ASIN ($\sqrt{3}$ / 2) = $\pi/3$

(SIN($\pi/3$) = $\sqrt{3}/2$)

입력(IN1) : INPUT(REAL) = 0.866

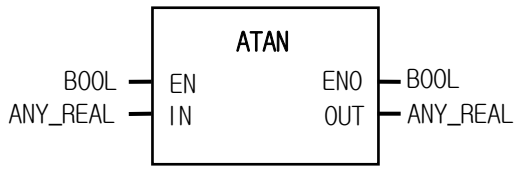
↓ (ASIN)

출력(OUT) : RESULT(REAL) =1.04714680E+00

ATAN

Arc Tangent 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Tangent 연산의 입력값 **출력** ENO : EN값이 그대로 출력 OUT : 연산결과 각도출력 값(Radian) IN, OUT은 같은 데이터 타입이어야 함.

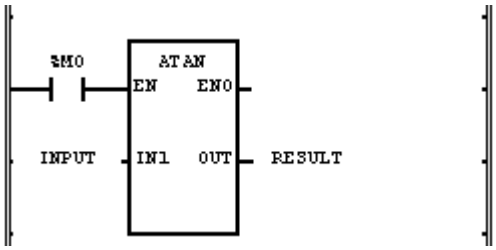
■ 기능

IN의 Arc Tangent 값을 구해 OUT으로 출력시킵니다. 출력값은 -π/2에서 π/2 사이의 값이 됩니다.

OUT = ATAN (IN)

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 Arc Trangent 연산 펑션 ATAN이 실행됩니다.

(2)펑션의 입력 변수로 선언된 INPUT = 1.0일 경우, 펑션의 출력변수로 선언된

RESULT = π/4 = 0.7853 .... 입니다.

ATAN (1) = π/4

(TAN(π/4) = 1)

입력(IN1) : INPUT(REAL) = 1.0

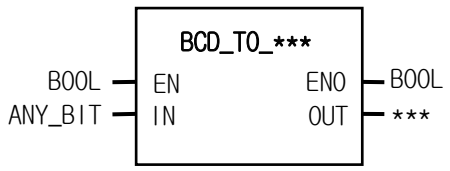
↓ (ATAN)

출력(OUT) : RESULT(REAL) = 7.85398185E

BCD_TO_***

BCD타입을 정수로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BCD형태의 데이터를 갖고 있는 ANY_BIT입력 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

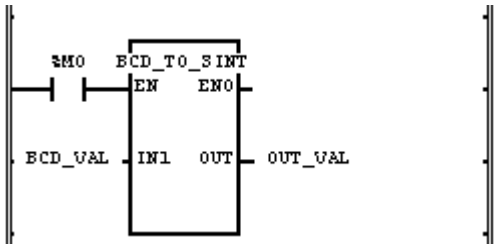
펑션	입력 타입	출력 타입	동작 설명
BCD_TO_SINT	BYTE	SINT	BCD를 출력 데이터 타입 타입으로 변환합니다. 입력이 BCD값일 경우에만 정상 변환됩니다. (입력 데이터 타입이 WORD일 경우 0～16#9999값만 정상 변환됩니다.)
BCD_TO_INT	WORD	INT	
BCD_TO_DINT	DWORD	DINT	
BCD_TO_LINT	LWORD	LINT	
BCD_TO_USINT	BYTE	USINT	
BCD_TO_UINT	WORD	UINT	
BCD_TO_UDINT	DWORD	UDINT	
BCD_TO_ULINT	LWORD	ULINT	

■ 에러

IN이 BCD 형태의 데이터가 아닌 경우, 출력은 0이 되고 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 펑션BCD_TO_*** 이 실행됩니다.

(2)BCD_VAL(BYTE 타입) = 16#22(2#0010_ 0010)이면, 펑션의 출력 변수로 선언된 OUT_VAL(SINT 타입) = 22(2#0001_ 0110)가 출력됩니다.

입력(IN1) :BCD_VAL(BYTE) = 16#22 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

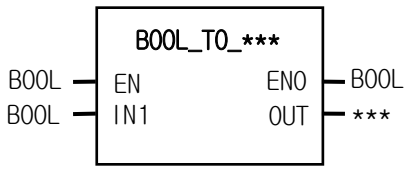
(BCD_TO_SINT)

출력(OUT): OUT_VAL(SINT) = 22 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

BOOL_TO_***

BOOL타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트(1 비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

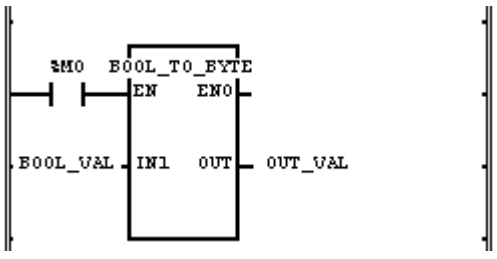
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
BOOL_TO_SINT	SINT	BOOL입력의 값이 2#0면 정수값 '0'을, 2#1이면 정수값 '1'을 출력 데이터 타입에 맞추어 출력합니다.
BOOL_TO_INT	INT	
BOOL_TO_DINT	DINT	
BOOL_TO_LINT	LINT	
BOOL_TO_USINT	USINT	
BOOL_TO_UINT	UINT	
BOOL_TO_UDINT	UDINT	
BOOL_TO_ULINT	ULINT	
BOOL_TO_BYTE	BYTE	BOOL을 상위 비트들을 0으로 채운 출력 데이터 타입 타입으로 변환합니다.
BOOL_TO_WORD	WORD	
BOOL_TO_DWORD	DWORD	
BOOL_TO_LWORD	LWORD	
BOOL_TO_STRING	STRING	BOOL을 STRING 타입으로 변환합니다. '0' 또는 '1'로 변환합니다. '0' 또는 '1' 로 변환됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 펑션 BOOL_TO_***이 실행됩니다.

(2)입력 변수로 선언된 BOOL_VAL(BOOL 타입) = 2#1이면,출력 변수로 선언된 OUT_VAL(BYTE 타입) = 2#0000_ 0001)이 출력 됩니다.

입력(IN1) : BOOL_VAL(BOOL) = 2#1 [1]

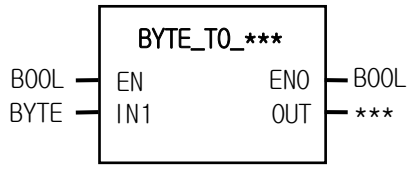
(BOOL_TO_SINT)

출력(OUT): OUT_VAL(BYTE) = 16#1 [0][0][0][0][0][0][0][1]

BYTE_TO_***

BYTE 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트열(8비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

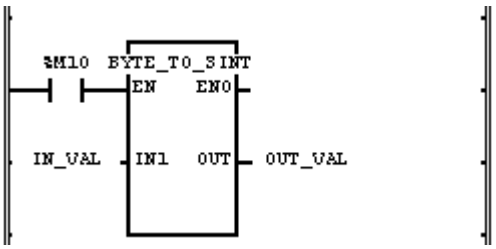
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
BYTE _TO_SINT	SINT	내부 비트 배열의 변환 없이 SINT 타입으로 변환합니다.
BYTE _TO_INT	INT	상위비트를 0으로 채워 INT 타입으로 변환합니다.
BYTE _TO_DINT	DINT	상위비트를 0으로 채워 DINT 타입으로 변환합니다.
BYTE _TO_LINT	LINT	상위비트를 0으로 채워 LINT 타입으로 변환합니다.
BYTE _TO_USINT	USINT	내부 비트 배열의 변환없이 SINT 타입으로 변환합니다.
BYTE _TO_UINT	UINT	상위비트를 0으로 채워 UINT 타입으로 변환합니다.
BYTE _TO_UDNT	UDINT	상위비트를 0으로 채워 UDINT 타입으로 변환합니다.
BYTE _TO_ULINT	ULINT	상위비트를 0으로 채워 ULINT 타입으로 변환합니다.
BYTE _TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
BYTE _TO_WORD	WORD	상위비트를 0으로 채워 WORD 타입으로 변환합니다.
BYTE _TO_DWORD	DWORD	상위비트를 0으로 채워 DWORD 타입으로 변환합니다.
BYTE _TO_LWORD	LWORD	상위비트를 0으로 채워 LWORD 타입으로 변환합니다.
BYTE _TO_STRING	STRING	입력값을 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)실행조건(%M10)이 On하면 펑션 BYTE_TO_SINT이 실행됩니다.
(2)IN_VAL(BYTE 타입) = 2#0001_1000이면, OUT_VAL(SINT 타입) = 24(2#0001_1000)가 됩니다.

입력(IN1) : IN_VAL(BYTE) = 16#18 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

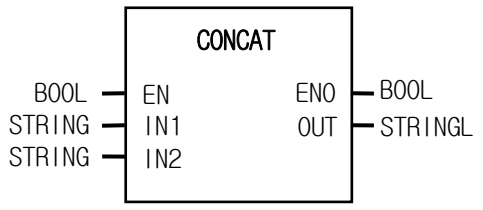
(BYTE_TO_SINT)

출력(OUT) : OUT_VAL(SINT) = 24 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

CONCAT

문자열 연결하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 입력 문자열 IN2 : 입력 문자열 입력 8개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

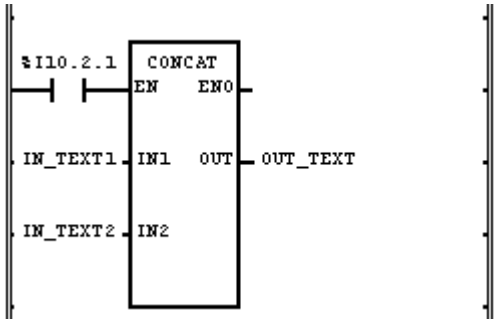
입력 문자열 IN1, IN2, IN3,...., INn(n은 입력 개수)을 순서대로 붙여서 출력 문자열 OUT에 출력시킵니다.

■ 에러

(각 입력 문자열의 문자수의 합) > 30인 경우, OUT 값은 각 입력 문자열을 30자까지 CONCAT한 값이 출력되고, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.2.1)이 On하면 펑션 CONCAT이 실행됩니다.
(2)펑션의 입력변수로 선언된 IN_TEXT1=`ABCD`, IN_TEXT2=`DEF`이면, 출력 변수로 선언된 OUT_TEXT=`ABCDDEF`가 됩니다.

입력(IN1) : IN_TEXT1(STRING) = `ABCD`

(CONCAT)

(IN2) : IN_TEXT2(STRING) = `DEF`

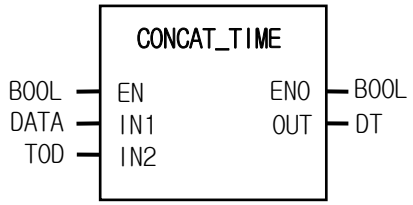
↓

출력(OUT) : OUT_TEXT(STRING) = 'ABCDDEF'

CONCAT_TIME

날짜와 시각 연결하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

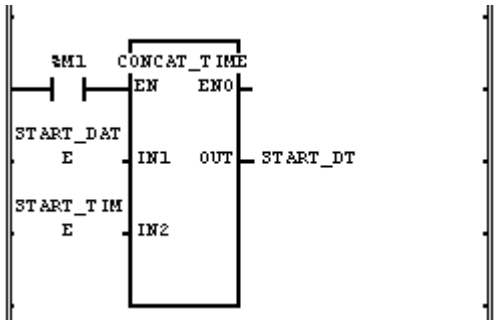
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 날짜 데이터 입력 IN2 : 시각 데이터 입력 **출력** ENO : EN값이 그대로 출력 OUT : 날짜와 시각 출력

■ 기능

IN1(날짜)과 IN2(시각)를 붙여서 날짜와 시각(DATE_AND_TIME) OUT으로 출력합니다.

■ 프로그램 예

LD



(1)실행조건(%M1)이 On하면 펑션CONCAT_TIME이 실행됩니다.

(2)운전시작 날짜 데이터 변수 START_DATE = D#1995-12-06이고 운전시작 시각 START_TIME = TOD#08:30:00 이면 날짜와 시각이 합쳐진 START_DT에는 DT#1995-12-06-08:30:00이 출력됩니다.

입력(IN1) : START_DATE1(DATE) = D#1995-12-06

(CONCAT_TIME)

(IN2) : START_TIME(TOD) = TOD#08:30:00

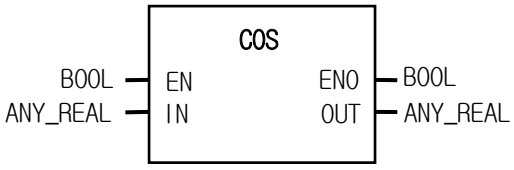
↓

출력(OUT) : START_DT(DT) = DT#1995-12-06-08:30:00

COS

Cosine 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Cosine연산의 각도 입력값(Radian) **출력** ENO: EN값이 그대로 출력 OUT: Cosine 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

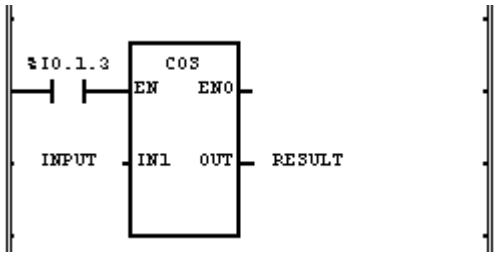
■ 기능

IN의 Cosine값을 구해 OUT으로 출력시킵니다.

OUT = COS (IN)

■ 프로그램 예

LD



(1)실행조건(%I0.1.3)이 On하면 펑션COS이 실행됩니다.

(2)INPUT으로 선언된 입력 변수의 값이 0.5235 (π/6 rad = 30°) 일 때 출력 변수로 선언된 RESULT은 0.8660 .... ($\sqrt{3}$ /2) 입니다.

COS (π/6) = $\sqrt{3}$/2 = 0.866

입력(IN1) : INPUT(REAL) = 0.5235

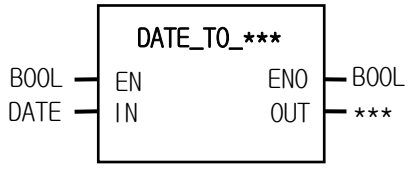
↓ (COS)

출력(OUT) : RESULT(REAL) = 8.66074800E-01

DATE_TO_***

DATE 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 날짜 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

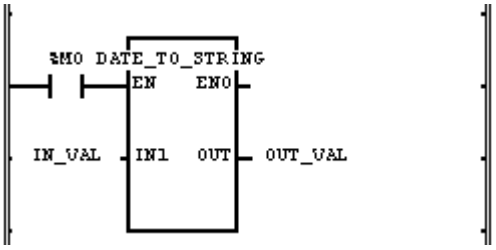
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
DATE_TO_UINT	UINT	DATE를 UINT타입으로 변환합니다.
DATE_TO_WORD	WORD	DATE를 WORD타입으로 변환합니다.
DATE_TO_STRING	STRING	DATE를 STRING타입으로 변환합니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 데이터 타입 변환 펑션DATE_TO_STRING이 실행됩니다.

(2)입력 변수로 선언된 IN_VAL(DATE 타입)의 값이 D#1995-12-01이면, 출력 변수로 선언된 OUT_VAL (STRING 타입)는 'D#1995-12-01'이 됩니다.

입력(IN1) : IN_VAL(DATE) = D#1995-12-01

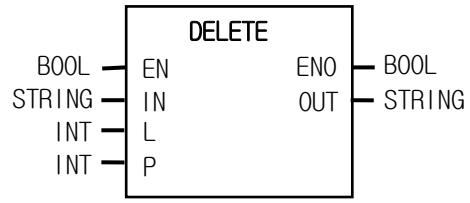
↓ (DATE_TO_STRING)

출력(OUT) : OUT_VAL(STRING) = 'D#1995-12-01'

DELETE

문자열을 삭제하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 삭제할 문자열 길이 P : 문자열의 삭제 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

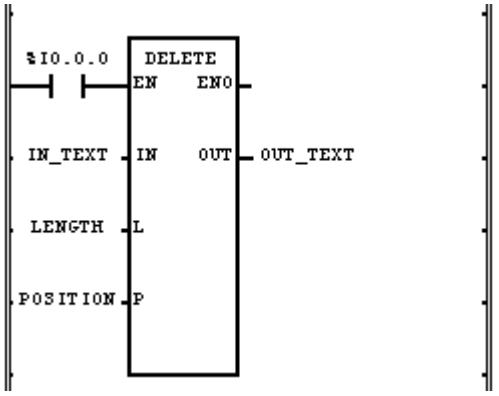
문자열 IN의 P번째 문자부터 길이 L숫자 만큼 삭제한 후, 문자열 OUT에 출력시킵니다.

■ 에러

P≤ 0 또는 L< 0인 경우 또는
P > (IN1의 입력 문자열의 문자수)인 경우, _ERR,_LER플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



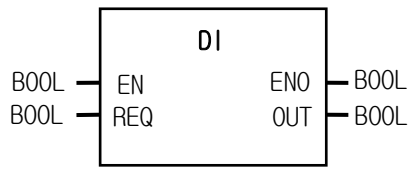
(1)실행조건(%I0.0.0)이 On하면 문자열 삭제 DELETE가 실행됩니다.
(2)입력 변수의 값이 IN_TEXT(입력한 문자)=`ABCDEF`,LENGTH(삭제할 문자열 길이)=3,POSITION(문자열의 삭제 위치)=3이면,출력 변수로 선언된 OUT_TEXT(STRING 타입)는 `ABF`가 됩니다.

입력(IN) : IN_TEXT(STRING) = `ABCDEF`
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 3
↓ (DELETE)
출력(OUT): OUT_VAL(STRING) = `ABF`

DI

태스크 프로그램 기동 불허

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 REQ : 태스크 프로그램 기동 불허 요구 **출력** ENO : EN값이 그대로 출력 OUT : DI동작이 실행되면 1출력

■ 기능

▷ EN이 1이고 REQ에 1의 값이 들어오면 사용자에 의해서 작성된 태스크 프로그램(싱글, 인터벌, 인터럽트)의 기동을 막습니다.

▷ 한번 'DI'명령이 수행되면 REQ입력이 0이 되어도 태스크 프로그램은 기동되지 않습니다.

▷ 태스크 프로그램이 정상적으로 기동하도록 할 때는 'EI' 펑션을 사용하여 주십시오.

▷ 프로그램의 수행 도중에 타 태스크 프로그램의 수행으로 연산 처리의 연속성을 잃을 경우 문제가 되는 부분에 대하여 부분적으로 태스크 프로그램의 수행을 막고자 할 때 사용할 수 있습니다.

▷ 태스크 프로그램의 기동불허 상태에서 발생한 태스크들은 태스크 종류에 따라 다음과 같이 수행됩니다.

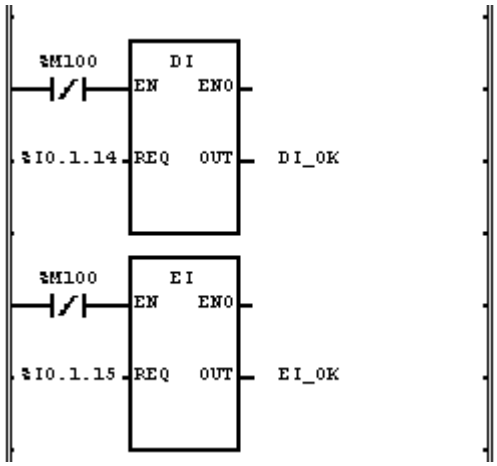
- 싱글 태스크 : 'EI'펑션 수행 후 또는 현재 수행 중인 태스크 프로그램의 종료 후에 수행됩니다. 이 때 싱글 변수의 상태 변화 횟수 만큼 태스크 프로그램을 반복하여 수행합니다.
- 인터벌 태스크, 인터럽트 : 태스크 프로그램의 기동 불허 상태에서 발생한 태스크는 'EI'펑션 수행 후 또는 현재 수행 중인 태스크 프로그램의 종료 후에 수행됩니다. 그러나, 태스크가 2번 이상 발생한 경우에는 태스크 충돌 경고(TASK_ERR)가 발생하고,충돌 횟수(TC_CNT)를 Count 합니다.

■ 프로그램 예

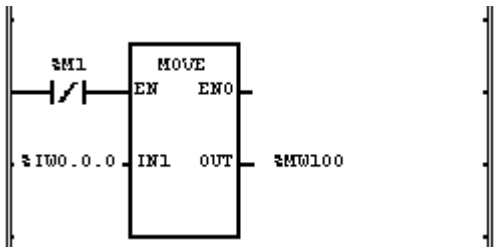
1초마다 값을 증가하는 태스크 프로그램을 태스크 프로그램 기동 불허 펑션 DI와 태스크 프로그램 기동 허가 펑션 EI를 이용하여 제어하는 프로그램

LD

(1)Scan 프로그램(TASK 프로그램 제어)



(2)1초마다 실행하여 증가하는 태스크 프로그램



(1)DI(태스크 프로그램 기동불허 펑션)의 기동불허 요구인 REQ (직접변수 %I0.1.14로지정)가 On하면 펑션 DI가 실행되어 출력 변수로 설정된 DI_OK의 값이 1이 됩니다.

(2)펑션 DI가 실행되면 1초마다 실행되던 태스크 프로그램의 실행이 정지됩니다.

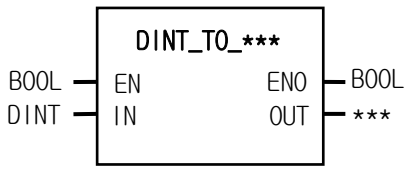
(3)EI (태스크 프로그램 기동허가 펑션)의 기동허가 요구인 REQ (직접변수 %I0.1.15로지정)가 On하면 펑션 EI가 실행되어 출력 변수로 설정한 EI_OK의 값이 1이 됩니다.

(4)펑션 EI가 실행하면 펑션 DI로 정지되었던 태스크 프로그램이 재실행 됩니다.

DINT_TO_***

DINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Double Integer값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
DINT_TO_SINT	SINT	입력이 -128 ~127일경우 정상 변화되나, 그외값은 에러가 발생합니다.
DINT_TO_INT	INT	입력이 -32768~32767일경우 정상변화되나,그외값은 에러가 발생합니다.
DINT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
DINT_TO_USINT	USINT	입력이 0~ 255일경우 정상 변화되나, 그외값은 에러가 발생합니다.
DINT_TO_UINT	UINT	입력이 0~ 65535일경우 정상 변화되나, 그외값은 에러가 발생합니다.
DINT_TO_UDINT	UDINT	입력이 0~2147483647일경우 정상 변화되나,그외값은 에러가 발생합니다.
DINT_TO_ULINT	ULINT	입력이 0~2147483647일경우 정상 변화되나,그외값은 에러가 발생합니다.
DINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
DINT_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
DINT_TO_WORD	WORD	하위 16 비트를 취해 WORD 타입으로 변환합니다.
DINT_TO_DWORD	DWORD	내부 비트 배열의 변화없이 DWORD 타입으로 변환합니다.
DINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD타입으로 변환합니다.
DINT_TO_BCD	DWORD	입력이 0~99,999,999일경우 정상변화되나, 그외값은 에러가 발생합니다.
DINT_TO_REAL	REAL	DINT를 REAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.
DINT_TO_LREAL	LREAL	DINT를 LREAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.

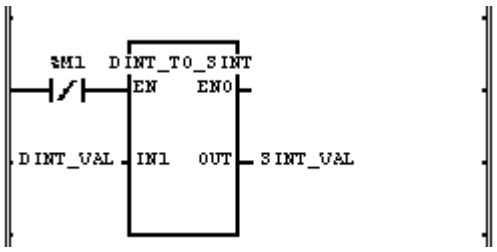
■ 에러

변환 에러 발생시 _ERR, _LER플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



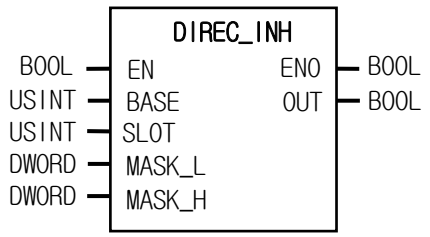
(1)실행조건(%M1)이 On하면 데이터 타입 변환 펑션 DINT_TO_SINT가 실행됩니다.
(2)INI = DINT_VAL(DINT 타입) = -77이면, SINT_VAL(SINT 타입) = -77이 됩니다.


입력(IN1) : DINT_VAL(DINT) = -77
상위
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
하위
1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1
(DINT_TO_SINT)
출력(OUT) : OUT_VAL(SINT) = -77
1 0 1 1 0 0 1 1

DIREC_IN

입력 데이터 즉시 갱신

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 BASE : 입력모듈이 장착된 베이스의 위치번호 SLOT : 입력모듈이 장착된 슬롯의 위치번호 MASK_L : 입력하위 32비트 데이터중 갱신하지 않을 비트 지정 MASK_H : 입력상위 32비트 데이터중 갱신하지 않을 비트 지정 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 입력데이터 갱신이 완료되면 1출력

■ 기능

▷ 펑션 DIREC_IN(입력 데이터 즉시 갱신)은 스캔 도중에 EN이 1이되면 BASE, SLOT에 지정된 위치의 입력 모듈의 64비트 데이터를 읽어서 입력 이미지 영역에 갱신하여 넣습니다.

▷ 이때 이미지 영역에는 해당 슬롯에 꽂혀있는 입력모듈의 점수 만큼만 갱신됩니다.

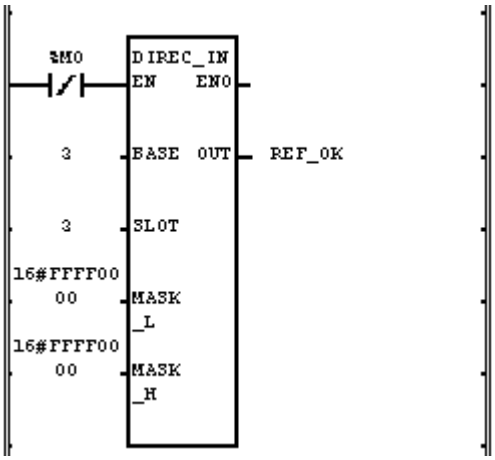
▷ 펑션 DIREC_IN은 스캔중에 입력(%I)의 On/Off 상태를 변화시키고 싶을때 사용이 가능합니다.

▷ 통상 스캔동기 일괄처리방식은 입력데이터 읽기와 출력 데이터의 출력을 스캔 프로그램의 종료 후에 일괄 처리하기 때문에, 1Scan 도중에 외부로 부터의 입력된 데이터의 갱신이 불가능합니다. 펑션 DIREC_IN을 사용하면 프로그램 실행도중에 해당하는 입력을 갱신할 수 있습니다.

■ 프로그램 예

1.3번째 증설 Base,3번째 Slot에 꽂힌 모듈이 16점 모듈이고,입력 데이터가 2# 1010_1010_1110_1011 로 즉시 갱신하는 프로그램

LD



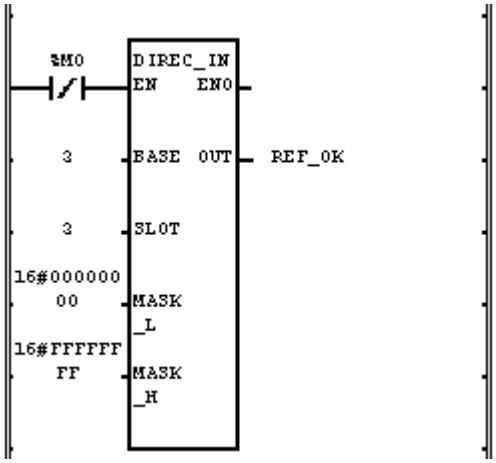
(1)입력조건(%M0)가 On하면 DIREC_IN(입력데이터 즉시 갱신)펑션이 실행합니다.

(2)장착된 모듈이 16점 모듈이므로 갱신대상 이미지 영역은 %IW3.3.0이 되고 갱신하지않을 비트의 지정 변수 MASK_L(입력하위 32비트)의 값에서 하위 16Bit가 갱신허용으로 설정되어 있으므로 %IW3.3.0은 스캔도중 #1010_1010_1110_1011로 갱신됩니다.

(3)비트의 지정 변수는 MASK_H(입력 상위 32비트)의 설정값은 현재 설정된 베이스, 슬롯에 16점 모듈이 꽂혀 있으므로 무시됩니다.

2. 3번째 증설 Base, 3번째 Slot에 꽂힌 모듈이 32점 모듈이고, 입력데이터가 2#0000_0000_1111_1111_1100_1100_0011_0011일때 하위 16Bit만 즉시 갱신하는 프로그램

LD

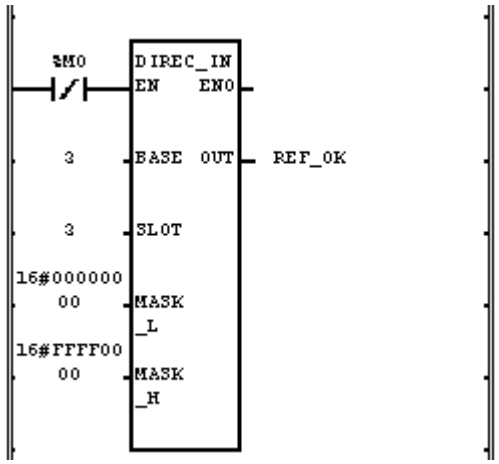


(1)입력조건(%M0)가 On하면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.

(2)장착된 모듈이 32점 모듈이므로 갱신대상 이미지 영역은 %ID3.3.0가 되나, 갱신하지 않을 때 비트 지정 MASK_L(입력하위 32비트)의 값에서 하위 16비트가 갱신허용으로 설정되어 있으므로 %IW3.3.0만 2#1100_1100_0011_0011로 갱신됩니다.

3. 3번째증설 Base, 3번째 Slot에 꽂힌 모듈이 64점 모듈이고 입력데이터가 16#0000_FFFF_AAAA_7777(2#0000_0000_0000_1111_1111_1111_1111_1010_1010_1010_1010_0111_0111_0111_0111)일때 64비트중 하위 48비트만 즉시 갱신할 프로그램.

LD



(1)입력조건(#M0)가 On하면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.
(2)장착된 모듈이 64점이므로 갱신 이미지 영역은 %IL3.3.0 즉 %ID3.3.0과 %ID3.3.1이 됩니다.
하위 32비트(MASK_L) 모두는 갱신 허용으로 되어있으므로 %ID3.3.0은 모두 갱신됩니다.
상위 32비트(MASK_H)는 이중 하위 32Bit만 갱신 허용으로 되어 있으므로 %ID3.3.1은 %IW.3.3.2는 갱신되고, %IW3.3.3은 갱신되지 않습니다.
따라서 이미지 영역의 데이터의 갱신은 아래와 같습니다.

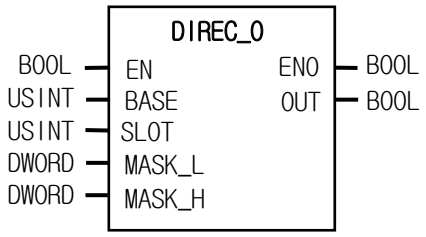
%IL3.3.0
- %ID3.3.0
  - %IW.3.3.0:2#0111_0111_0111_0111
  - %IW.3.3.1:2#1010_1010_1010_1010
- %ID3.3.1
  - %IW3.3.2:2#1111_1111_1111_1111
  - %IW3.3.3:이전값 유지

(3)입력 갱신이 완료되면 REF_OK(입력데이터 갱신 완료)에는 1이 출력됩니다.

DIREC_O

출력 모듈 데이터 즉시 갱신

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 BASE : 출력모듈이 장착된 베이스의 위치번호 SLOT : 출력모듈이 장착된 슬롯의 위치번호 MASK_L : 출력하위 32비트 데이터중 갱신하지 않을 비트 지정 MASK_H : 출력상위 32비트 데이터중 갱신하지 않을 비트 지정 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 출력데이터 갱신이 완료되면 1출력

■ 기능

▷펑션 DIREC_O(출력데이터 즉시 갱신)은 스캔도중에 EN(DIREC_O 실행조건)이 1이되면 BASE와 SLOT이 지정된 위치의 출력모듈의 64비트 데이터를 읽어서 MASK(1)되지 않은 비트만을 출렬모듈에 즉시 출합니다.

▷펑션 DIREC_O은 1스캔중에 출력(%Q)의 On/Off 상태를 변화시키고 싶을때 사용이 가능합니다.

▷통상 스캔중에 일괄처리방식은 입력데이터 읽기와 출력데이터의 출력을 스캔프로그램의 종료후에 일괄처리 하기 때문에 1스캔 도중에 외부로 신호를 출력하는 것이 불가능합니다.

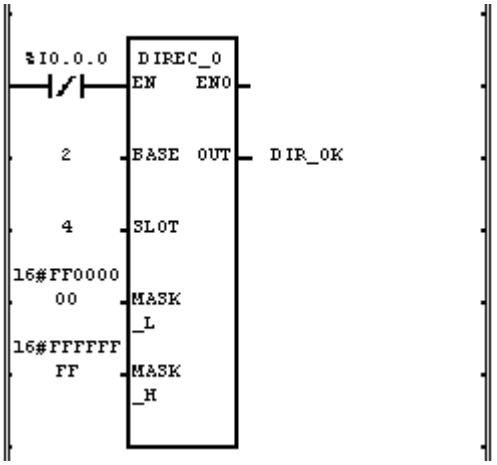
▷펑션 DIREC_O을 사용하면 프로그램 실행도중에 해당하는 비트의 데이터를 외부로 출력하는 것이 가능합니다.

▷해당위치에 다른 타입의 모듈이 꽂혀 있거나, 출력모듈에 데이터가 정상적으로 써지지 않으면 ENO와 OUT을 0으로 출력합니다.(정상동작시 1출력)

■ 프로그램 예

1.기본베이스 3번째 Slot에 장착된 16점 Relay 출력 모듈에 스캔 도중 출력 데이터 값이 2#0111_0111_0111_0111을 출력하는 프로그램

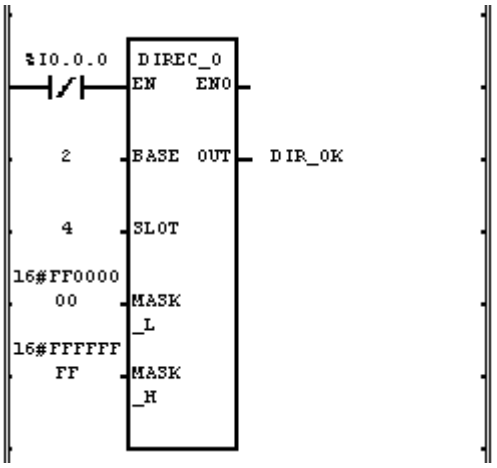
LD



(1)출력모듈이 장착된 Base의 위치번호 2와 SLOT번호 4를 입력합니다.
(2)스캔도중 출력하고져 하는 데이터가 16비트이므로 MASK_L의 값 중 하위 16비트만 출력 허용 값으로 설정합니다.(16#FFFF0000)
(3)실행조건(%I0.0.0)가 On하면 DIREC_0(출력모듈 데이터 즉시 갱신) 펑션이 실행되어 스캔도중에 출력모듈의 데이터가 2#0111_0111_0111_0111로 출력됩니다.

2.2번째 증설베이스 4번째 Slot에 장착된 32점 TR. 출력 모듈중 하위 24비트만 스캔도중 출력데이터 값이 2#1111_0000_1111_0000_1111_0000로 변경 출력하는 프로그램

LD



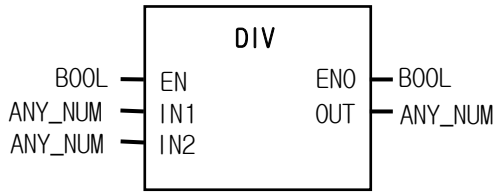
(1)출력모듈이 장착된 Base의 위치번호 2와 SLOT번호 4를 입력합니다.
(2)스캔도중 출력하고져 하는 데이터가 24비트이므로 MASK_L의 값중 하위 24비트만 출력허용값으로 설정합니다.(16#FF000000)
(3)실행조건(%I0.0.0)가 Off하면 DIREC_0(출력모듈 데이터 즉시 갱신) 펑션이 실행되어, 스캔도중에 출력모듈의 데이터가 2#□□□□_□□□□_1111_0000_1111_0000_1111_0000로 출력됩니다.

이전값 유지

DIV

나누기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나누어질 값(피제수) IN2 : 나눌 값(제수) **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 나눈 결과 값 (몫) IN1, IN2, OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

■ 기능

IN1을 IN2로 나눠서 그 몫중에서 소수점 이하를 버린 값을 OUT으로 출력시킵니다.

OUT = IN1/IN2

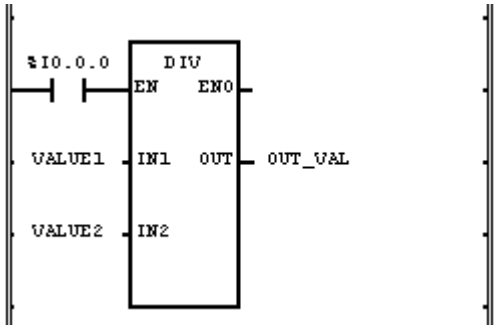
IN1	IN2	OUT	비 고
7	2	3	소수점 이하 버림
7	-2	-3	
-7	2	-3	
-7	-2	3	
7	0	×	에러

■ 에러

나눌 값(제수)이 '0'인 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 나누기 펑션DIV가 실행됩니다.

(2)입력 변수 VALUE1 = 300, VALUE2 = 100 이면, 출력 변수로 선언된 OUT_VAL = 300/100 = 3이 출력됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

/ (DIV)

(IN2): VALUE2(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

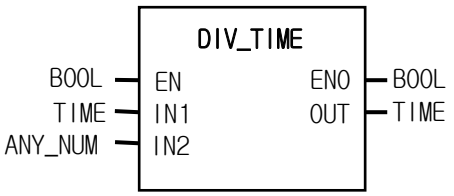
출력(OUT) : OUT_VAL(INT) = 3(16#3)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

DIV_TIME

시간 나누기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나눌 시간 IN2 : 나눌 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 나눈 결과 시간

■ 기능

IN1(시간)을 IN2(숫자)로 나누어서 나누어진 시간을 OUT으로 출력시킵니다.

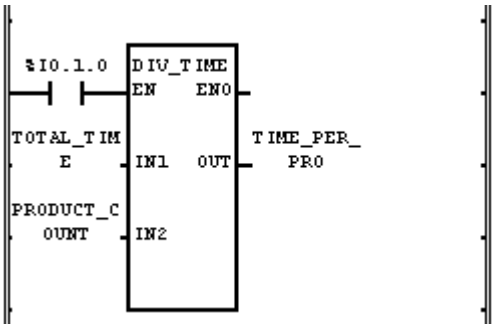
■ 에러

제수(IN2)가 0인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

어느 생산 Line의 하루 작업 시간이 12시간 24분24초이고, 하루 생산량이 12개일 때 제품 한 개를 생산하는데 걸리는 시간을 계산하는 프로그램입니다.

LD



(1)실행 조건(%I0.1.0)이 On하면 시간 나누기 펑션 DIV_TIME이 실행됩니다.

(2)하루 작업 시간 TOTAL_TIME(T#12H24M24S)을 하루 제품 생산량 PRODUCT_COUNT(12)로 나누면,제품 한 개당 걸리는 시간 TIME_PER_PRO(T#1H2M2S)이 출력됩니다. 즉, 1시간 2분 2초에 한 개씩 생산됩니다.

입력(IN1) : TOTAL_TIME(TIME) = T#12H24M24S

/ (DIV_TIME)

(IN2) : PRODUCT_COUNT(INT) = 12

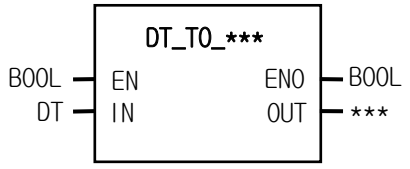
↓

출력(OUT) : TIME_PER_PRO(TIME) = T#1H2M2S

DT_TO_***

DT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 날짜와 시각 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

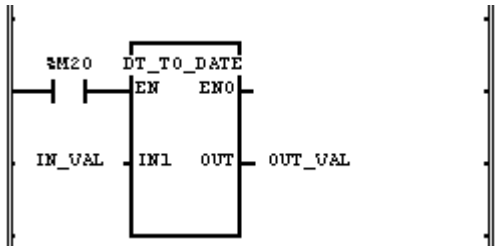
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
DT_TO_LWORD	LWORD	DT를 LWORD타입으로 변환합니다. (내부 데이터의 변환가 없어 역변환이 가능 합니다.)
DT_TO_DATE	DATE	DT를 DATE타입으로 변환합니다.
DT_TO_TOD	TOD	DT를 TOD 타입으로 변환합니다.
DT_TO_STRING	STRING	DT를 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)실행조건(%M20)이 On하면 DT 타입 변환 펑션 DT_TO_DATE가 실행됩니다.

(2)입력 변수로 선언된IN_VAL(DT 타입) = DT#1995-12-01-12:00:00이면, 출력 변수로 선언된 OUT_VAL (DATE 타입) = D#1995-12-01이 됩니다.

입력(IN1) : IN_VAL(DT) = DT#1995-12-01-12:00:00

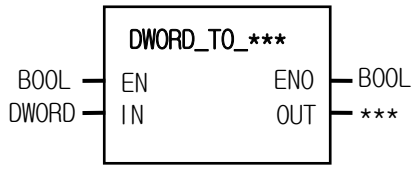
↓ (DT_TO_DATE)

출력(OUT) : OUT_VA(DATE) = D#1995-12-01

DWORD_TO_***

DWORD 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트열(32비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

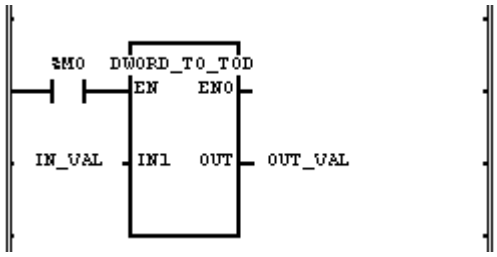
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
DWORD _TO_SINT	SINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
DWORD _TO_INT	INT	하위 16비트를 취해 INT 타입으로 변환합니다.
DWORD _TO_DINT	DINT	내부 비트 배열의 변환없이 DINT 타입으로 변환합니다.
DWORD _TO_LINT	LINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
DWORD _TO_USINT	USINT	하위 8비트를 취해 USINT 타입으로 변환합니다.
DWORD _TO_UINT	UINT	하위 16비트를 취해 UINT 타입으로 변환합니다.
DWORD _TO_UDINT	UDINT	내부 비트 배열의 변환없이 UDINT 타입으로 변환합니다.
DWORD _TO_ULINT	ULINT	상위 비트를 0으로 채운 ULINT 타입으로 변환합니다.
DWORD _TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
DWORD _TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
DWORD _TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환합니다.
DWORD _TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
DWORD _TO_REAL	REAL	내부 비트 배열의 변화없이 REAL 타입으로 변환합니다.
DWORD _TO_TIME	TIME	내부 비트 배열의 변화없이 TIME 타입으로 변환합니다.
DWORD _TO_TOD	TOD	내부 비트 배열의 변화없이 TOD 타입으로 변환합니다.
DWORD _TO_STRING	STRING	입력값을 Decimal로 바꾸어 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 타입 변환 펑션 DWIRD_TO_TOD가 실행됩니다.
(2)IN_VAL(DWORD 타입) = 16#3E8(1000)이면, OUT_VAL(TOD 타입) = TOD#1S가 됩니다.

입력(IN1):IN_VAL(DWORD)= 16#3E8(1000)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
상위	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
하위	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

데이터(내부 비트 배열 상태)의
변화없이 데이터 타입만 변환

출력(OUT) : OUT_VAL(TOD) = TOD#1S

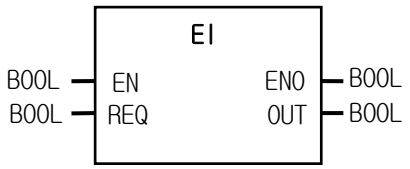
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
상위	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
하위	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

TIME, TOD는 10진 값을 MS 단위로 환산해 계산합니다. 즉 1000은 1000MS = 1S 로 계산합니다.
(3.2.4. 데이터 타입별 구조 참조)

EI

태스크 프로그램 기동 허가
(DI의 해제)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 REQ : 태스크 프로그램 기동 허가 요구 **출력** ENO : EN값이 그대로 출력 OUT : EI동작이 실행되면 1출력

■ 기능

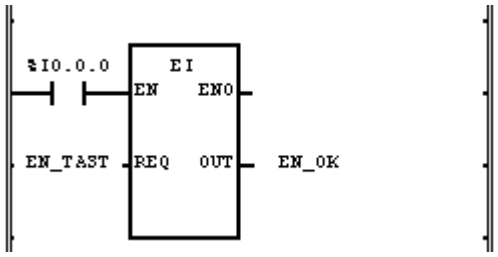
▷ EN이 1이고 REQ에 1의 값이 들어오면 'DI'펑션에 의해 막혀있던 태스크 프로그램이 정상적으로 기동합니다.

▷ 한번 'EI'명령이 수행되면 REQ입력이 0이 되어도 태스크 프로그램은 정상 기동합니다.

▷ 태스크 프로그램의 기동불허 상태에서 발생한 태스크들은 'EI'펑션 수행 후 또는 현재 수행중인 태스크 프로그램의 종료 후에 수행됩니다.

■ **프로그램 예**(DI 참조)

LD



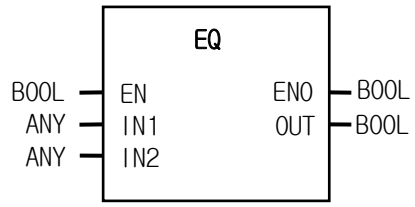
EN_TASK가 1이 되면 태스크 프로그램이 정상 기동 됩니다.
'EI'펑션에 의해서 태스크 실행 허가 상태가 되면 EN_OK에는 1이 출력됩니다.

.

EQ

'같다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

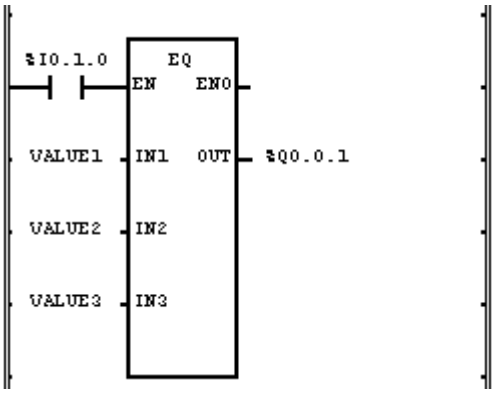
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

■ 기능

IN1=IN2=IN3...=INn(n은 입력개수)이면 OUT으로 1이 출력됩니다.
다른 경우에는 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.1.0)이 On하면 비교 펑션 'EQ'가 실행됩니다.
(2)VALUE1 = 300, VALUE2 = 300, VALUE3 = 300 이면, 비교결과 VALUE1 = VALUE2 = VALUE3이므로
출력값 %Q0.0.1 = 1이 됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

= (EQ)

(IN2) : VALUE2(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

= (EQ)

(IN3) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

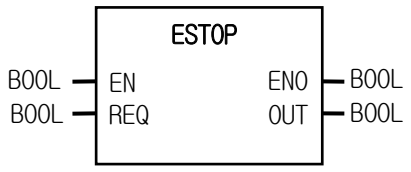
출력(OUT) : %Q0.0.1(BOOL) = 1(16#1)

1

ESTOP

프로그램에 의한 비상 운전정지

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 REQ: 프로그램에 의한 비상 운전정지 요구 **출력** ENO : EN값이 그대로 출력 OUT : ESTOP동작이 실행되면 1출력

■ 기능

▷ 펑션 실행 조건 EN이 1이고, 프로그램에 의한 비상 운전 정지 요구 신호 REQ가 1이되면 현재 수행중인 프로그램의 운전을 즉시 강제 정지하고 STOP모드로 갑니다.

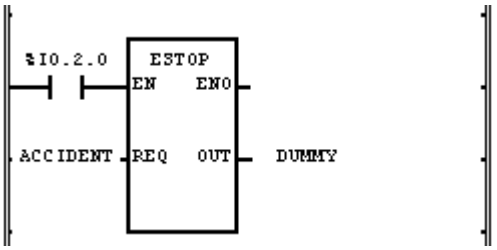
▷ 'ESTOP'펑션에 의해 정지된 경우는 전원을 재 투입하여도 기동되지 않습니다.

▷ 운전모드를 STOP으로 하였다가 RUN으로 하면 재 기동이 됩니다.

▷ 'ESTOP' 펑션이 수행되면 수행중인 프로그램을 중도에 중지하기 때문에 재 기동시 콜드 리스타트 모드가 아닐 경우 데이터의 연속성에 오류가 있을 수 있습니다.

■ 프로그램 예

LD



(1) 실행조건(%I0.2.0)이 On하면 프로그램에 의한 비상 운전 정지 펑션 'ESTOP'이 실행됩니다.

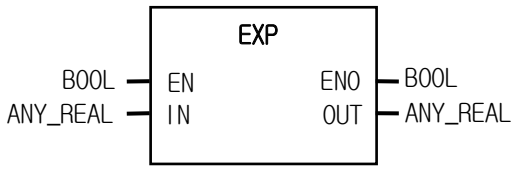
(2) 펑션 'ESTOP'의 ACCIDENT가 1이 되면 실행중인 프로그램을 즉시 중지하고 STOP모드로 됩니다.

※비상 사태 발생시 기계적 인터럽트와 함께 이중 안전장치로 사용할 수 있습니다.

EXP

자연지수 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 지수연산의 입력값 **출력** ENO : EN값이 그대로 출력 OUT : 지수연산 결과값 IN, OUT은 같은 데이터 타입이어야 함.

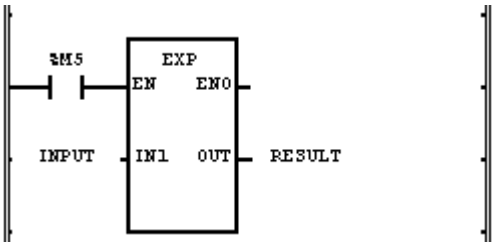
■ 기능

IN의 지수연산값을 구해 OUT으로 출력시킵니다.

$OUT = e^{IN}$

■ 프로그램 예

LD



(1)실행조건(%M5)이 On하면 자연 지수 펑션 'EXP'가 실행됩니다.

(2)입력 변수로 선언된 INPUT의 값이 2.0 일 경우, 출력 변수로 선언된 RESULT는 7.3890 .... 입니다.

$e^{2.0} = 7.3890.....$

입력(IN1) : INPUT(REAL) = 2.0 상위

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

하위

0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

(16#40000000)

↓ (EXP)

출력(OUT) : RESULT(REAL) = 7.38905621E+00 상위

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

하위

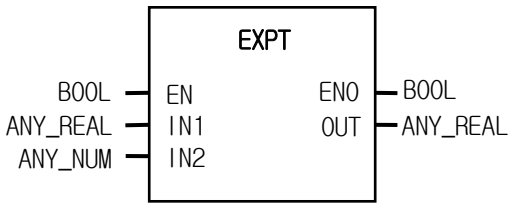
0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

(16#40EC7326)

EXPT

지수 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 진수 IN2 : 지수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 결과값 IN1과 OUT에 연결되는 변수는 같은 데이터 타입이어야 함.

■ 기능

IN1에 IN2를 지수승하여 OUT으로 출력시킵니다.

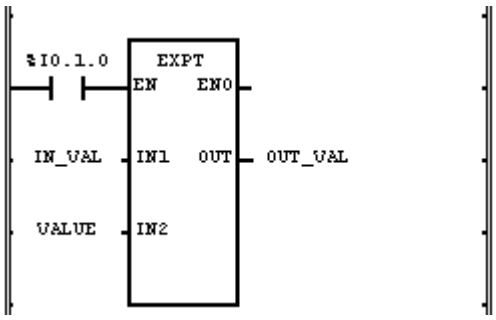
$OUT = IN1^{IN2}$

■ 에러

출력 값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.1.0)이 On하면 자연 지수 펑션 'EXPT'가 실행됩니다.

(2)입력된 변수로 선언된 IN_VAL = 1.5, VALUE = 3이면, 출력변수로 선언된 OUT_VAL = 1.5^3 = 1.5× 1.5× 1.5 = 3.375가 됩니다.

입력(IN1) : IN_VAL(REAL) = 1.5

(IN2) : VALUE(INT) = 3

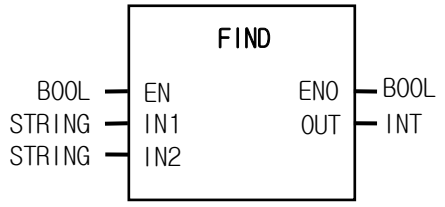
↓ (EXPT)

출력(OUT) : OUT_VAL(REAL) = 3.37500000E+00

FIND

문자열 찾기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

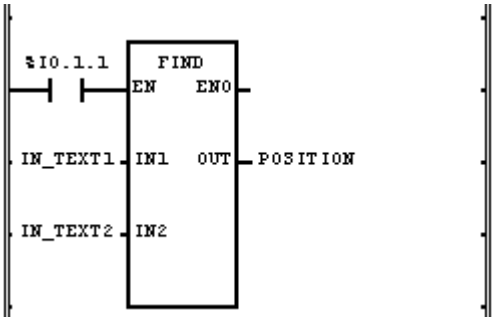
펑 션	설 명
	**입력** EN : 실행 허용 IN1 : 입력 문자열 IN2 : 찾을 문자열 **출력** ENO : EN값이 그대로 출력 OUT : 찾는 문자열의 위치

■ 기능

입력 문자열 IN1에서 문자열 IN2의 위치를 찾습니다. 찾으면 문자열 IN1에서 문자열 IN2가 있는 첫번째 문자위치를 OUT에 출력시키고, 없으면 0을 OUT에 출력시킵니다.

■ 프로그램 예

LD



(1)실행조건(%I0.1.1)이 On하면 FIND(문자열 찾기) 펑션이 실행됩니다.
(2)입력 변수로 선언된 입력문자열이 IN_TEXT1=`ABCEF, 찾을 문자열이 IN_TEXT2=`BC`이면, 출력 변수로 선언된 POSITION=2가 선언됩니다.
(3)입력문자열 IN_TEXT1=`ABCEF`에서 찾을 문자열 IN_TEXT2=`BC`의 위치는 2번째 입니다.

입력(IN1) : IN_TEXT1(STRING) = 'ABCEF'

(FIND)

(IN2) : IN_TEXT2(STRING) = 'BC'

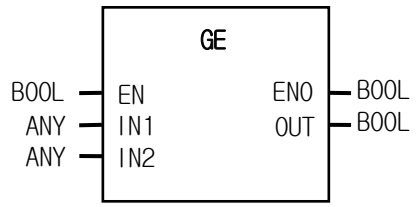
↓

출력(OUT) : POSITION(INT) = 2

GE

'크거나 같다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

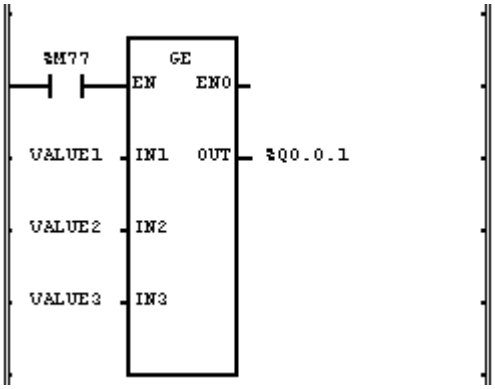
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2,...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

■ 기능

IN1≥IN2≥IN3...≥INn(n은 입력개수)이면 OUT으로1이 출력됩니다.
다른 경우에는 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M77)이 On하면 GE(비교:크거나 같다) 펑션이 실행됩니다.
(2)입력변수로 선언된 VALUE1=300, VALUE3=200 이면, 비교 결과 VALUE1≥VALUE2≥VALUE 3이므로,
출력 결과 값 %Q0.01=1이 됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

≥ (GE)

(IN2) : VALUE2(INT) = 200(16#00C8) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

≥ (GE)

(IN3) : VALUE3(INT) = 100(16#0064) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

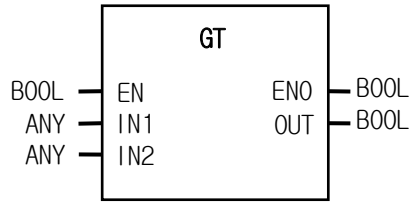
↓

출력(OUT) : %Q0.0.1(BOOL) = 1(16#1) | 1 |

GT

'크다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

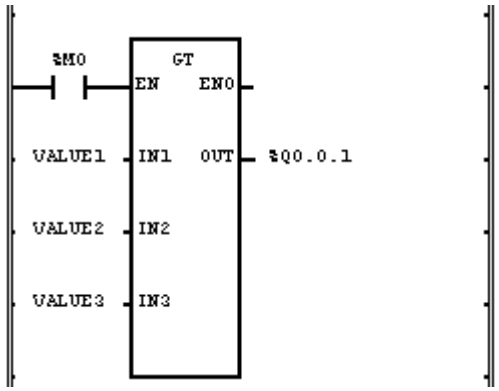
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과 값

■ 기능

IN1>IN2>IN3...>INn(n은 입력 개수)이면 OUT으로 1이 출력됩니다.
다른 경우에는 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 GT(비교:크다) 펑션이 실행됩니다.
(2)입력변수로 선언된 VALUE1 = 300, VALUE2 = 200, VALUE3 = 100이면, 비교결과 VALUE1 > VALUE2 > VALUE3이므로 출력결과값 %Q0.0.1=1이 됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C) | 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 |

> (GT)

(IN2) : VALUE2(INT) = 200(16#00C8) | 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 |

> (GT)

(IN3) : VALUE3(INT) = 100(16#0064) | 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 |

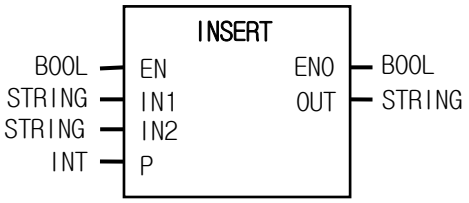
↓

출력(OUT) : %Q0.0.1(BOOL) = 1(16#1) → 1

INSERT

문자열 삽입하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 삽입될 문자열 IN2 : 삽입할 문자열 P : 문자열을 삽입할 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

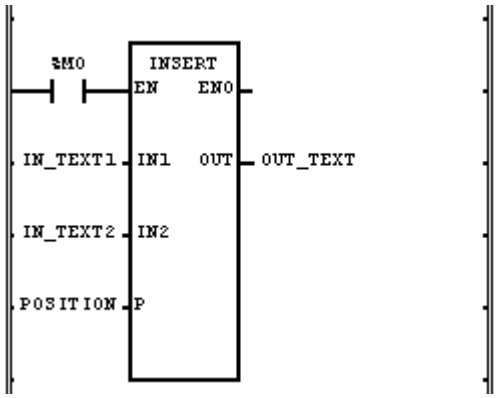
문자열IN1의 P번째 문자뒤에 문자열 IN2를 삽입한 후 문자열 OUT에 출력시킵니다.

■ 에러

P ≤0이거나 (변수 IN1의 문자수) < P인 경우, 또는 연산결과 문자수가 30자를 넘어서 OUT으로 30자까지 출력된 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



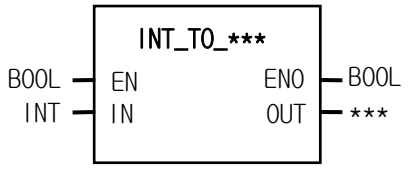
(1)실행조건(%M0)가 On하면, INSERT(문자열 삽입하기) 펑션이 실행됩니다.
(2)입력변수로 선언된 변수 IN_TEXT1= `ABCD`, IN_TEXT2=`XY`, POSITON=2이면, 출력변수 OUT_TEXT=`ABXYCD`가 됩니다.

입력(IN1) : IN_TEXT1(STRING) = 'ABCD'
(IN2) : IN_TEXT2(STRING) = 'XY'
(P) : POSITION(INT) = 2
↓(FIND)
출력(OUT): OUT_TEXT = `ABXYCD`

INT_TO_***

INT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Integer값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
INT_TO_SINT	SINT	입력이 -128 ~ 127일경우 정상 변화되나, 그외값은 에러가 발생합니다.
INT_TO_DINT	DINT	DINT 타입으로 정상 변환합니다.
INT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
INT_TO_USINT	USINT	입력이 0 ~ 255일경우 정상 변화되나, 그외값은 에러가 발생합니다.
INT_TO_UINT	UINT	입력이 0 ~ 32767일경우 정상 변화되나, 그외값은 에러가 발생합니다.
INT_TO_UDINT	UDINT	입력이 0 ~ 32767일경우 정상 변화되나, 그외값은 에러가 발생합니다.
INT_TO_ULINT	ULINT	입력이 0 ~ 32767일경우 정상 변화되나, 그외값은 에러가 발생합니다.
INT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
INT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
INT_TO_WORD	WORD	내부 비트 배열의 변화없이 WORD 타입으로 변환합니다.
INT_TO_DWORD	DWORD	상위 비트들을 0으로 채운 DWORD 타입으로 변환합니다.
INT_TO_LWORD	LWORD	상위 비트들을 0으로 채운 LWORD 타입으로 변환합니다.
INT_TO_BCD	WORD	입력이 0~9,999일 경우 정상변환되나, 그외값은 에러가 발생합니다.
INT_TO_REAL	REAL	INT를 REAL 타입으로 정상 변환합니다.
INT_TO_LREAL	LREAL	INT를 LREAL 타입으로 정상 변환합니다.

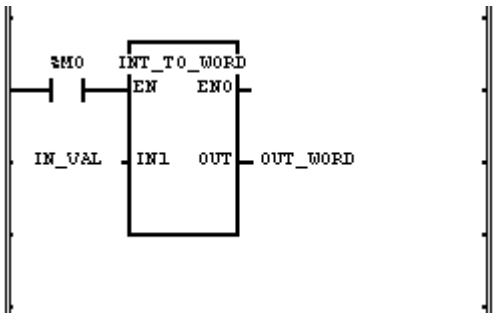
■ 에러

변환에러 발생시 _ERR _LER플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 INT_TO_WORD 펑션이 실행됩니다.
(2)입력 변수로 선언된 IN_VAL(INT 타입) = 512(16#200)이면, 출력 변수로 선언된 OUT_WORD(WORD 타입) =16#200이 됩니다.

입력(IN1) : IN_VAL(INT) = 512(16#200)

0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

(INT_TO_WORD)

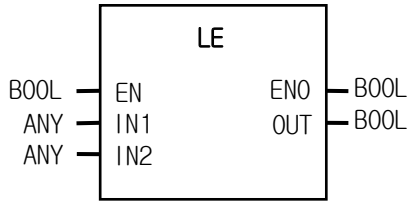
출력(OUT) : OUT_WORD(WORD) = 16#200

0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LE

'작거나 같다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

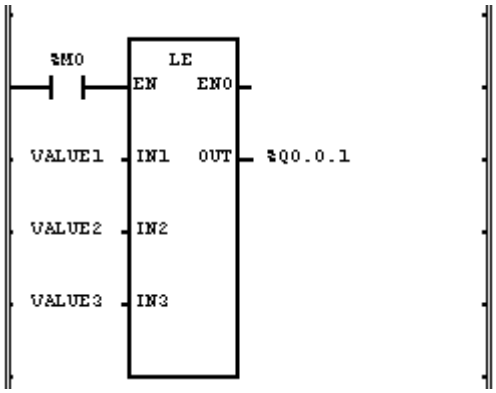
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

■ 기능

IN1 ≤ IN2 ≤ IN3... ≤ INn (n은 입력 개수)이면 OUT으로 1이 출력됩니다.
다른 경우에는 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)가 On하면 LE(비교:작거나 같다)펑션이 실행됩니다.
(2)입력 변수로 선언된 VALUE1=150, VALUE2=200, VALUE3 = 250이면, 비교 결과 VALUE1≤VALUE2≤VALUE 3이므로, 출력 결과값 %Q0.0.1=1이 됩니다.

입력(IN1) : VALUE1(INT) = 150(16#0096)

0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0

≤ (LE)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

≤ (LE)

(IN3) : VALUE1(INT) = 250(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

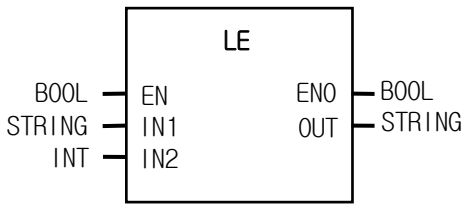
출력(OUT) : %Q0.0.1(BOOL) = 1(16#1)

1

LEFT

문자열의 왼쪽을 취하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

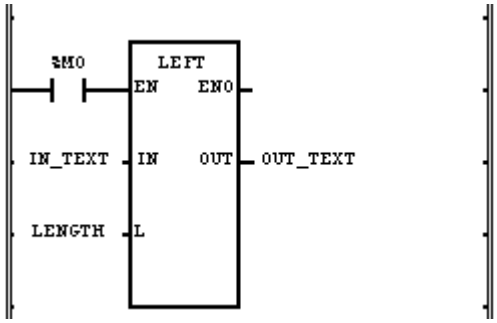
입력 문자열 IN에 대하여 왼쪽부터 문자열 길이 L만큼 출력 문자열 OUT에 출력시킵니다.

■ 에러

L < 0 인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)가 On하면 LEFT(문자열의 왼쪽 취하기) 펑션이 실행됩니다.

(2)입력 변수로 선언된 문자열이IN_TEXT=`ABCDEFG`이고, 출력할 문자열의 길이 LENGTH=3이면,출력 문자열 변수로 지정된 OUT_TEXT=`ABC`가 됩니다.

입력(IN1) : IN_TEXT(STRING) = 'ABCDEFG'
(IN2) : LENGTH(INT) = 3

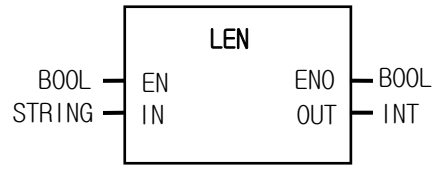
↓(LEFT)

출력(OUT) : OUT_TEXT(STRING) = `ABC`

LEN

문자열 길이 구하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

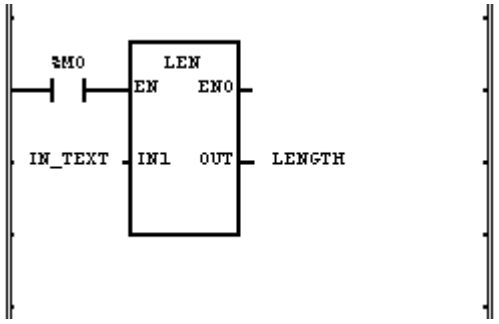
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 **출력** ENO : EN값이 그대로 출력 OUT : 문자열 길이

■ 기능

입력 문자열(IN)의 길이 (문자수)가 OUT 으로 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)가 On하면 LEN(문자열 길이 구하기) 펑션이 실행됩니다.
(2)입력 변수로 선언된 IN_TEXT=`ABCD`이면, 출력 변수로 선언된 LENGTH=4가 됩니다.

입력(IN1) : IN_TEXT(STRING) = 'ABCD'

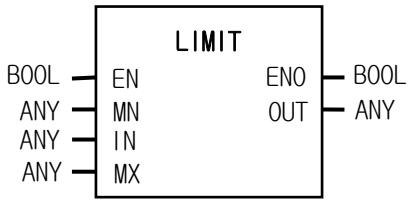
↓ (LEN)

출력(OUT) : LENGTH(INT) = 4

LIMIT

상하한 제한

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 MN : 최소값 IN : 제한될 값 MX : 최대값 **출력** ENO : EN값이 그대로 출력 OUT : 범위안에 든 값 MN, IN, MX, OUT은 데이터 타입이 모두 같아야 함.

■ 기능

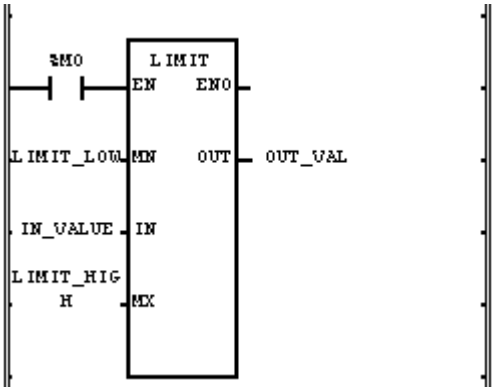
▷ 입력 IN값이 MN과 MX 사이에 있으면, OUT으로 IN이 출력됩니다.
즉, MN ≤ IN ≥MX이면 OUT = IN

▷ 입력 IN값이 MN보다 작으면, OUT으로 MN이 출력됩니다. 즉, IN < MN이면 OUT = MN

▷ 입력 IN값이 MX보다 크면, OUT으로 MX가 출력됩니다. 즉, IN > MX이면 OUT = MX

■ 프로그램 예

LD



(1)실행조건(%M0)가 On하면 LIMIT(상하한 제한)펑션이 실행합니다.

(2)하한값의 입력변수(LIMIT_LOW) , 상한값의 입력변수(LIMIT_HIGH), 제한된 값의 입력변수(IN_VALUE)에 대한 출력변수(OUT_VAL)의 값은 아래와 같습니다.

LIMIT_LOW	IN_VALUE	LIMIT_HIGH	OUT_VAL
1000	2000	3000	2000
1000	500	3000	1000
1000	4000	3000	3000

입력 (MN): LIMIT_LOW (INT) = 1000

(IN) : IN_VALUE (INT) = 4000

(MX) : IN_VALUE (INT) = 3000

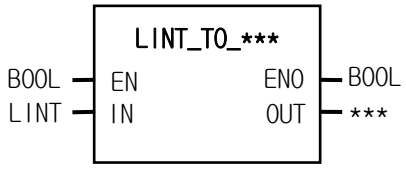
↓ (LIMIT)

출력 (OUT) : OUT_VAL (INT) = 3000

LINT_TO_***

LINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Long Integer값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
LINT_TO_SINT	SINT	입력이-128 ~ 127일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_INT	INT	입력이-32,768~32,767일경우 정상 변환되나,그외 값은 에러가 발생합니다.
LINT_TO_DINT	DINT	입력이-2^{31} ~ 2^{31}-1일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_USINT	USINT	입력이 0~ 255일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_UINT	UINT	입력이 0~ 65,535일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_UDINT	UDINT	입력이 0 ~ 2^{32}-1일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_ULINT	ULINT	입력이 0 ~ 2^{63}-1일경우 정상 변환되나, 그외 값은 에러가 발생합니다.
LINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
LINT_TO_BYTE	BYTE	하위 8비트를 취해 BOOL 타입으로 변환합니다.
LINT_TO_WORD	WORD	하위 16비트를 취해 BOOL 타입으로 변환합니다.
LINT_TO_DWORD	DWORD	하위 32비트를 취해 BOOL 타입으로 변환합니다.
LINT_TO_LWORD	LWORD	내부 비트 배열의 변화 없이 LWORD 타입으로 변환합니다.
LINT_TO_BCD	LWORD	입력이 0~9,999,999,999,999,999일경우 정상 변환되나 그외의 값은 에러가 발생합니다.
LINT_TO_REAL	REAL	LINT를 REAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.
LINT_TO_LREAL	LREAL	LINT를 LREAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.

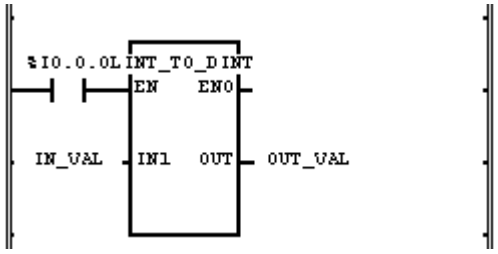
■ 에러

변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건 (%I0.0.0)이 On하면 LINT_TO_DINT펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(LINT 타입) = 123_456_789이면, OUT_VAL(DINT 타입) = 123_456_789가 됩니다.

입력(IN1) : IN_VAL(LINT) = 123,456,789
(16#75BCD15)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

(LINT_TO_DINT)

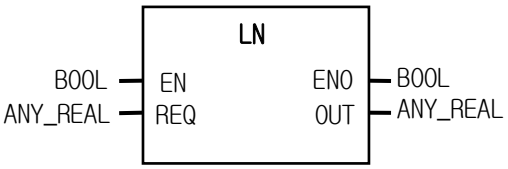
출력(OUT) : OUT_VAL(DINT) = 123,456,789
(16#75BCD15)

0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

LN

자연대수 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 자연대수 연산의 입력값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 자연대수값 IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

IN의 자연대수값을 구해 OUT으로 출력시킵니다.

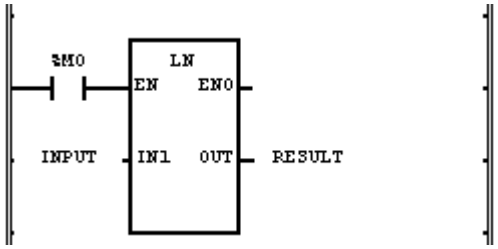
OUT = ln IN

■ 에러

입력값이 0또는 음수일 때 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On되면 LN(자연대수연수) 펑션이 실행됩니다.

(2)입력변수로 선언된 INPUT 값이 2.0일 때 출력변수로 선언된RESULT는 0.6931 .... 입니다.

ln(2.0) = 0.6931....

입력(IN1) : INPUT(REAL) = 2.0

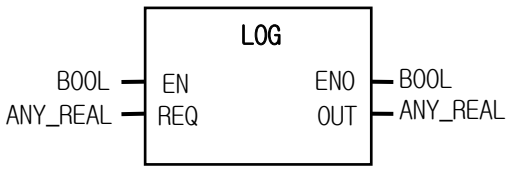
↓ (LN)

출력(OUT): RESULT(REAL) =6.93147182E-01

LOG

상용대수 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 상용대수 연산의 입력값 **출력** END : 에러 없이 실행되면 1을 출력 OUT: 상용대수 연산값 IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

IN의 상용대수값을 구해 OUT으로 출력시킵니다.

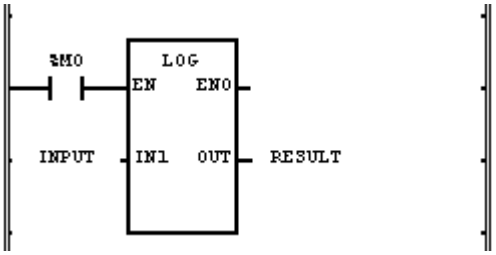
OUT = $\log_{10}$ IN = log IN

■ 에러

IN의 값이 0또는 음수일 때 _ ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On되면, LOG(상용대수연산) 펑션이 실행됩니다.

(2)입력변수로 선언된 INPUT 값이 2.0일 때 출력변수로 선언된RESULT는 0.3010 .... 입니다.

$\log_{10}(2.0) = 0.3010....$

입력(IN1) : INPUT(REAL) = 2.0

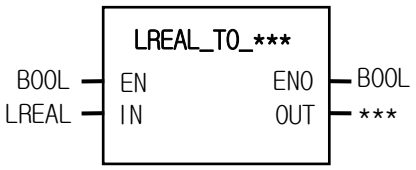
↓ (LOG)

출력(OUT) : RESULT(REAL) =3.01030010E-01

LREAL_TO_***

LREAL타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 LREAL값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

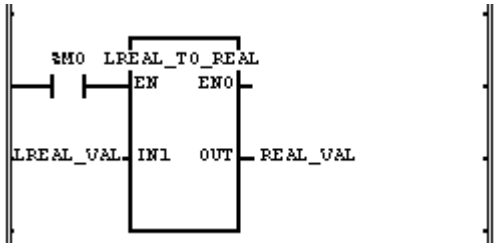
펑션	출력 타입	동작 설명
LREAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_INT	INT	입력의 정수 부분이 -32768 ~ 32767일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_LINT	LINT	입력의 정수 부분이 -2^{63} ~ $2^{63}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_USINT	USINT	입력의 정수 부분이 0 ~ 255일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_UINT	UINT	입력의 정수 부분이 0 ~ 65,535일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_UDINT	UDINT	입력의 정수 부분이 0 ~ $2^{32}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_ULINT	ULINT	입력의 정수 부분이 0 ~ $2^{64}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
LREAL_TO_LWORD	LWORD	내부 비트 배열의 변화없이 LWORD 타입으로 변환합니다.
LREAL_TO_REAL	REAL	LREAL을 REAL 타입으로 정상 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.

■ 에러

입력값이 출력타입에 저장할 수 있는 값보다 커서 오버 플로(Overflow) 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 0을 출력합니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면, LREAL_TO_REAL 펑션이 실행됩니다.
(2)입력변수로 선언된 LREAL_VAL(LREAL 타입) = -1.34E-12이면, 출력변수로 선언된REAL_VAL (REAL타입)=-1.34E-12가 됩니다.

입력(IN1) : LREAL_VAL (LREAL) = -1.34E-12

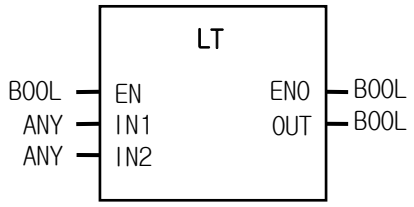
↓ (LREAL_TO_REAL)

출력(OUT) : REAL_VAL (REAL) = -1.34E-12

LT

'작다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

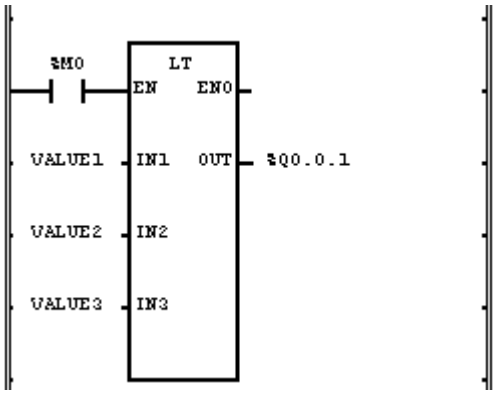
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

■ 기능

IN1<IN2<IN3...<INn(n은 입력개수)이면 OUT으로 1이 출력됩니다.
다른 경우에는 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 LT(비교:작다) 펑션이 실행됩니다.
(2)입력변수로 선언된 VALUE1 = 100, VALUE2 = 200, VALUE3 = 300 이면, 비교결과 VALUE1 < VALUE2 < VALUIE 3 이므로 출력 결과값 %Q0.0.1=1 이 됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064) 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0

< (LT)

(IN2) : VALUE2(INT) = 200(16#00C8) 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0

< (LT)

(IN3) : VALUE3(INT) = 300(16#012C) 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0

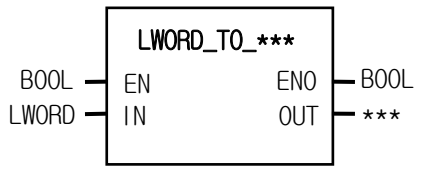
↓

출력(OUT) : %Q0.0.1(BOOL) = 1(16#1) 1

LWORD_TO_***

LWORD 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트열(64비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

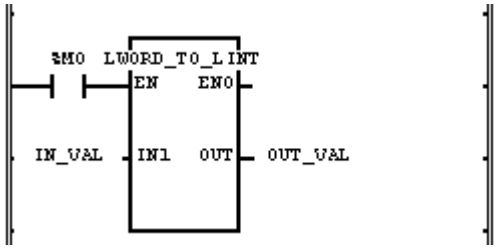
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
LWORD _TO_SINT	SINT	하위 8비트를 취하여 SINT 타입으로 변환합니다.
LWORD _TO_INT	INT	하위 16비트를 취하여 INT 타입으로 변환합니다.
LWORD _TO_DINT	DINT	하위 32비트를 취하여 DINT 타입으로 변환합니다.
LWORD _TO_LINT	LINT	내부 비트 배열의 변화없이 LINT 타입으로 변환합니다.
LWORD _TO_USINT	USINT	하위 8비트를 취하여 USINT 타입으로 변환합니다.
LWORD _TO_UINT	UINT	하위 16비트를 취하여 UINT 타입으로 변환합니다.
LWORD _TO_UDINT	UDINT	하위 32비트를 취하여 UDINT 타입으로 변환합니다.
LWORD _TO_ULINT	ULINT	내부 비트 배열의 변화없이 ULINT 타입으로 변환합니다.
LWORD _TO_BOOL	BOOL	하위 1비트를 취하여 BOOL 타입으로 변환합니다.
LWORD _TO_BYTE	BYTE	하위 8비트를 취하여 BYTE 타입으로 변환합니다.
LWORD _TO_WORD	WORD	하위 16비트를 취하여 WORD 타입으로 변환합니다.
LWORD _TO_DWORD	DWORD	하위 32비트를 취하여 DWORD 타입으로 변환합니다.
LWORD _TO_LREAL	LREAL	LWORD를 LREAL 타입으로 변환합니다.
LWORD _TO_DT	DT	내부 비트 배열의 변화없이 DT 타입으로 변환합니다.
LWORD _TO_STRING	STRING	입력값을 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 LWORD_TO_LINT 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(LWORD 타입) = 16#FFFFFFFFFFFFFFFF면, 출력변수로 선언된 OUT_VAL(LINT 타입) = -1(16#FFFFFFFFFFFFFFFF)이 됩니다.

입력(IN1) : IN_VAL(LWORD) = 16#FFFFFFFFFFFFFFFF

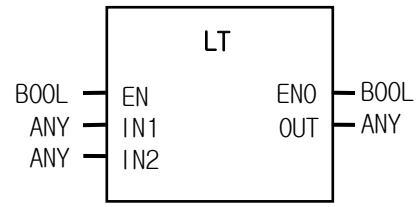
↓ (LWORD_TO_LINT)

출력(OUT) : OUT_VAL(LINT) = -1

MAX

최대값

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

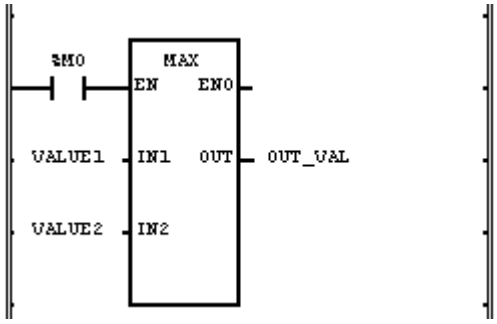
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : 입력값 중 최대값 IN1, IN2, ..., OUT은 모두 같은 타입이어야 함.

■ 기능

입력 IN1, IN2, ..., INn(n은 입력 개수)중에서 최대값을 OUT으로 출력시킵니다.

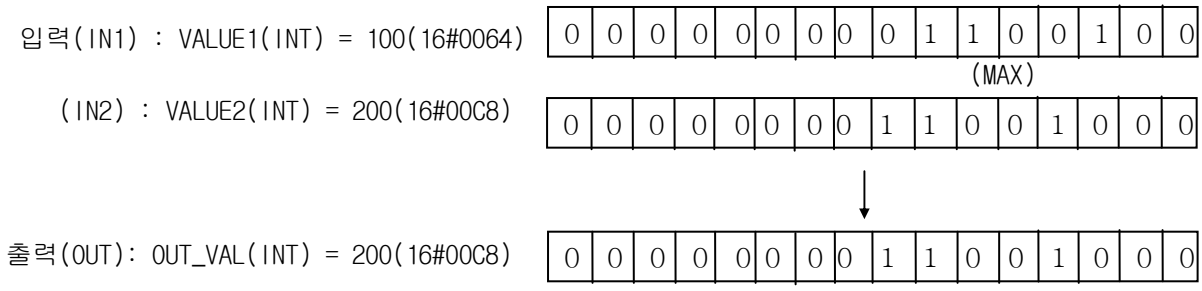
■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 MAX(최대값) 펑션이 실행됩니다.

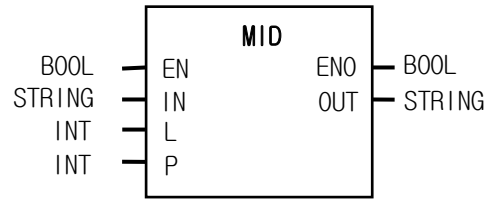
(2)입력변수로 선언된 VALUE1 = 100, VALUE2 = 200을 비교결과 최대값이 200이므로 출력변수로 선언된OUT_VALUE = 200으로 출력됩니다.



MID

문자열의 중간을 취하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 P : 출력할 문자열의 시작 위치 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

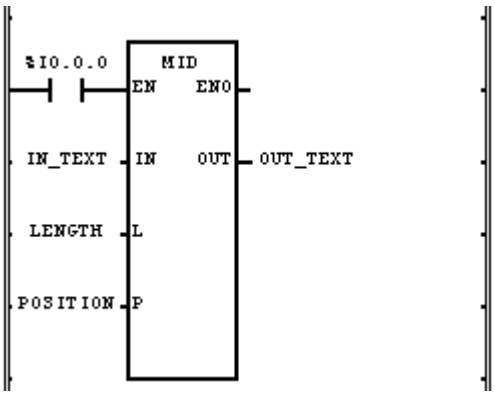
입력 문자열 IN에 대하여 입력 문자열의 P번째 문자부터 길이L 만큼을 출력 문자열OUT에 출력시킵니다.

■ 에러

(변수IN의 문자 수) < P인 경우, 또는 P <= 0 및 L < 0인 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면MID(문자열의 중간을 취하기) 펑션이 실행됩니다.

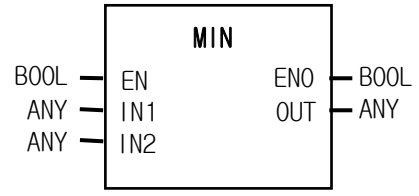
(2)입력된 문자열 IN_TEXT=`ABCDEFG`,이고, 출력할 문자열의 길이 LENGTH=3, 출력할 문자열의 시작위치 POSITION=2 이면, 출력문자열 변수로 선언된 OUT_TEXT=`BCD`가 됩니다.

```
입력(IN) : IN_TEXT1(STRING) = 'ABCDEFG'
    (L) :  LENGTH(INT)      =    3
    (P) :  POSITION(INT)    =    2
                                 ↓ (MID)
출력(OUT) : OUT_TEXT        =  `BCD`
```

MIN

최소값

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

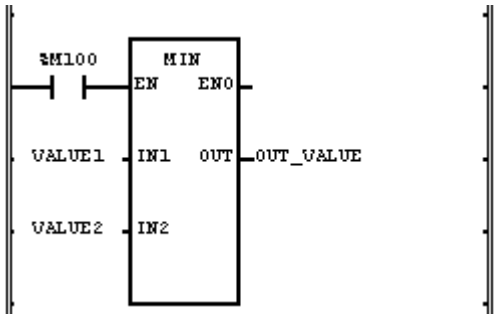
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT :입력값중 최소값 IN1, IN2, ..., OUT은 모두 같은 타입이어야 함.

■ 기능

입력 IN1, IN2,....., INn(n은 입력 개수)중에서 최소값을 OUT으로 출력시킵니다.

■ 프로그램 예

LD



(1)실행조건(%M100)이 On하면 MIN(최소값) 펑션이 실행됩니다.

(2)입력변수로 선언된 VALUE1 = 100, VALUE2 = 200을 비교결과 최소값이100이므로 출력변수로 선언된 OUT_VALUE = 100이 출력됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

(MIN)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

↓

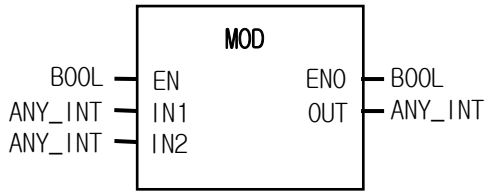
출력(OUT): OUT_VAL(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

MOD

나머지 구하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나누어 질 값(피제수) IN2 : 나눌 값(제수) **출력** ENO : EN값이 그대로 출력 OUT : 나눈 결과값(나머지) IN1, IN2, OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

■ 기능

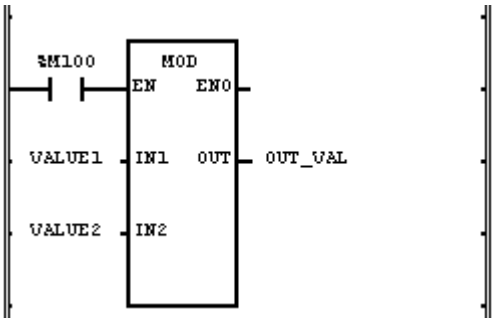
IN1을 IN2로 나눠서 그 나머지를 OUT으로 출력시킵니다.

OUT = IN1 - (IN1/IN2) × IN2 (단 IN2 = 0이면 OUT = 0)

IN1	IN2	OUT
7	2	1
7	-2	1
-7	2	-1
-7	-2	-1
7	0	0

■ 프로그램 예

LD



(1)실행조건(%M100)이 On하면 MON(나머지 구하기) 펑션이 실행합니다.

(2)입력변수중 나누어질 값 VALUE1=37이고, 나눌값 VALUE2=10이면, 출력변수로 선언된OUT_VAL의 값은 37을 10으로 나눈 나머지 7이 됩니다.

입력(IN1) : VALUE1(INT) = 37(16#0025)

0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1

(MOD)

(IN2) : VALUE2(INT) = 10(16#000A)

0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

↓

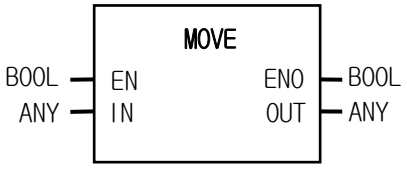
출력(OUT): OUT_VAL(INT) = 7(16#0007)

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

MOVE

데이터 복사

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : MOVE할 값 **출력** ENO : EN값이 그대로 출력 OUT : MOVE된 값 IN, OUT에 연결되는 변수는 같은 데이터 타입이어야 함.

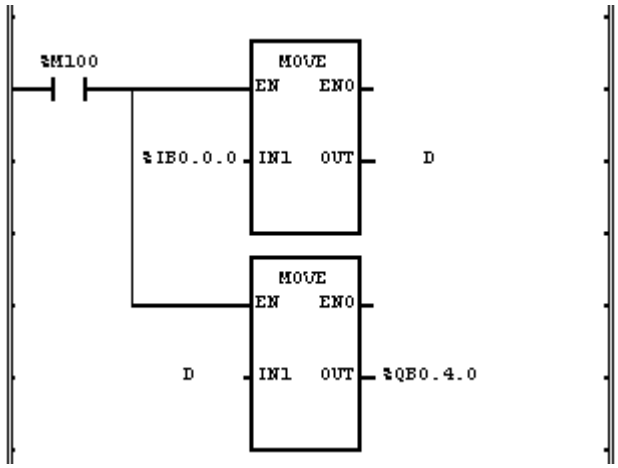
■ 기능

IN의 값을 OUT으로 옮긴다.

■ 프로그램 예

입력 %I0.0.0~%I0.0.7의 8점의 입력상태를 변수 D로 전송한 후, 전송된 데이터를 출력 %Q0.4.0 ~%Q0.4.7의 8점으로 출력시키는 프로그램

LD



(1)실행조건(%M100)가 On되면 MOVE(데이터 복사) 펑션이 실행됩니다.

(2)첫번째 MOVE 펑션에 의해 입력모듈의 8점 입력 데이터가 변수 D영역으로 옮겨지고 두번째 MOVE 펑션에 의해 변수 D에 저장된 입력모듈의 상태가 출력모듈로 출력됩니다.

입력(IN1) : %IB0.0.0(BYTE) = 16#18

0	0	0	1	1	0	0	0

(MOVE)

D(BYTE) = 16#18

0	0	0	1	1	0	0	0

(MOVE)

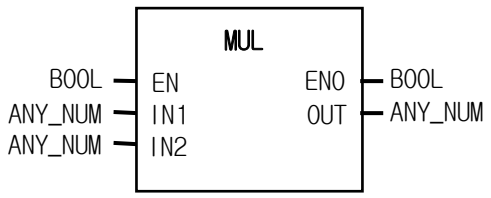
출력(OUT) : %QB0.4.0(BYTE) = 16#18

0	0	0	1	1	0	0	0

MUL

곱하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 곱해질 값 (피승수) IN2 : 곱할 값 (승수) 입력은 8개까지 확장 가능 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 곱한 결과 값 IN1, IN2, ..., OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함

■ 기능

IN1, IN2,..., INn (n은 입력 개수)를 곱해서 OUT으로 출력시킵니다.

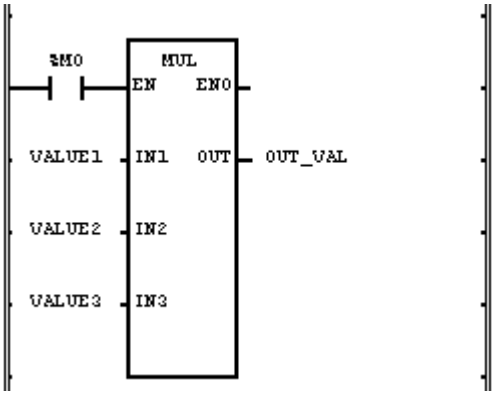
OUT = IN1 × IN2× ... × INn

■ 에러

출력값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

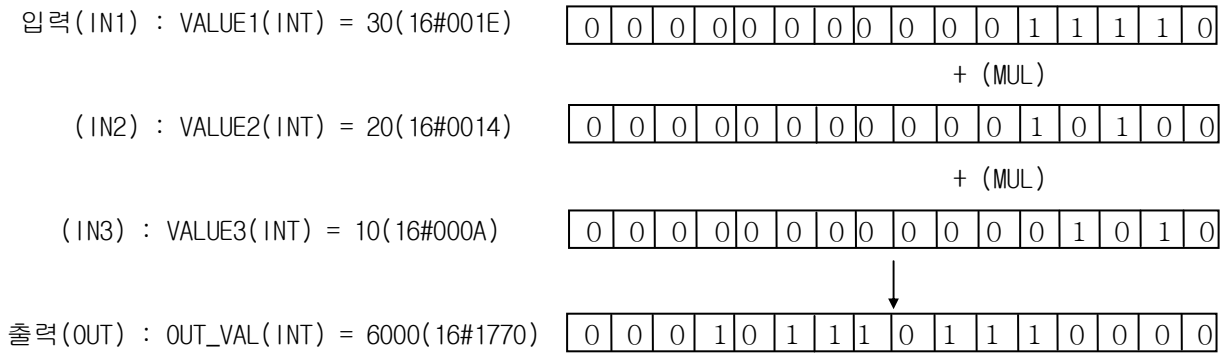
■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 MUL(곱하기) 펑션이 On합니다.

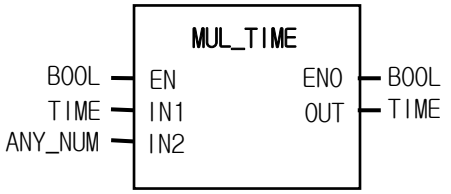
(2)MUL 펑션의 입력변수로 선언된 VALUE1 = 30, VALUE2 = 20, VALUE3 = 10이면, 출력변수로 선언된 OUT_VAL = 30 × 20× 10 = 6000이 됩니다.



MUL_TIME

시간 곱하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 일 때 펑션 실행 IN1 : 곱할 시간 IN2 : 곱할 값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 곱한 결과 시간

■ 기능

IN1(시간)을 IN2(숫자)로 곱해서 결과 시간을 OUT으로 출력시킵니다.

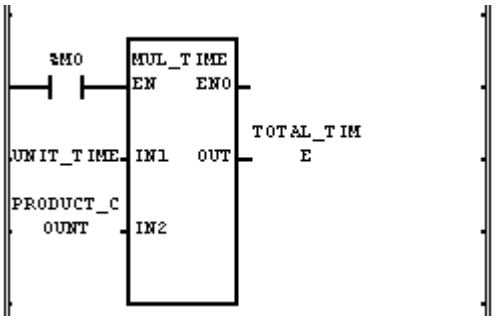
■ 에러

출력값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

어떤 제품생산 LINE에서 생산되는 제품의 단위 제품당 평균 계획시간이20분2초이고,하루 생산할 제품의 수가 20개일때 작업 소요시간을 설정하는 프로그램

LD



(1)입력변수(1N1:단위제품당 제작시간) UNIT_TIME:T#20M2S을 입력한다.
(2)입력변수(1N2:생산수량) PRODUCT_COUNT:20을 입력한다.
(3)출력변수 (OUT:총작업 소요시간)에 TOTAL_TIME을 입력한다.
(4)실행조건(% M0)이 On되면 출력변수로 설정한 TOTAL_TIME에 T#6H40M40S가 출력된다.

입력(IN1) : UNIT_TIME(TIME) = T#20MS2S
(MUL_TIME)
(IN2) : PRODUCT_COUNT(INT) = 16#18

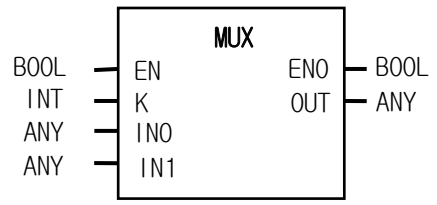
↓

출력(OUT): TOTAL_TIME(TIME) = T#6H40M40S

MUX

여러개중 선택

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 K : 선택 IN0 : 선택될 값 IN1 : 선택될 값 입력은 7개까지 확장 가능(IN0, IN1,..., IN6) **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 선택된 값 IN0, IN1, ..., OUT은 모두 같은 타입이어야 함.

■ 기능

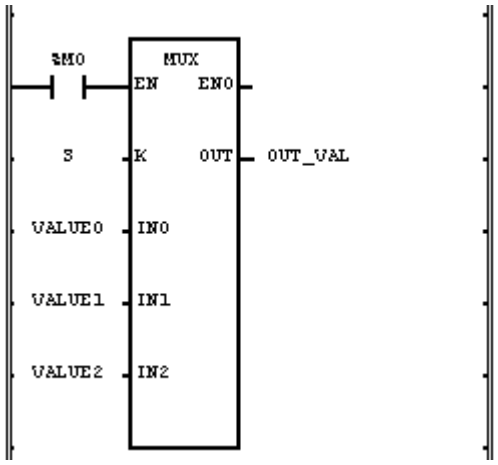
K값으로 여러 입력(IN0, IN1,..., INn)중 하나를 선택하여 출력시킵니다.
K = 0이면 IN0이, K = 1이면 IN1이, K = n이면 INn이 OUT으로 출력됩니다.

■ 에러

K의 값이 입력 변수 INn의 개수보다 크거나 같은 경우에 OUT으로는 IN0값이 출력되고, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이On하면 MUX(여러개를 선택) 펑션이 실행됩니다.
(2)입력변수로 설정된VALUE0, 1, 2 중 선택변수 S의 값에 따라 선택되어 출력변수로 선언된 OUT으로 출력 시킵니다.

입력 (K) : S(INT) = 2

(IN0) : VALUE0(WORD) = 16#11

(IN1) : VALUE1(WORD) = 16#22

(IN2) : VALUE2(WORD) = 16#33

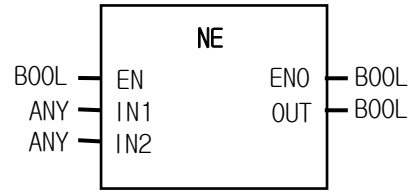
↓ (MUX)

출력 (OUT) : OUT_VAL(WORD) = 16#33

NE

'같지 않다' 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

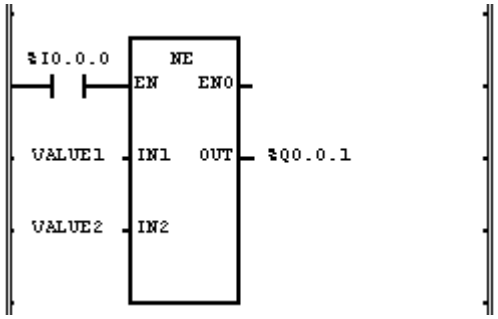
펑 션	설 명
	**입력** EN : 실행 허용 IN1 : 비교될 값 IN2 : 비교될 값 IN1, IN2는 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

■ 기능

IN1이 IN2와 같지 않으면 OUT으로 1이 출력됩니다.
같으면 OUT으로 0이 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 NE(비교:같지 않다) 펑션이 실행됩니다.
(2)입력변수로 선언된 VALUE1 = 300, VALUE2 = 200이면, 비교결과 VALUE1과 VALUE2가 다르므로
출력결과값 %Q0.0.1= 1이 됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

(NE)

(IN2) : VALUE2(INT) = 200(16#0C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

↓

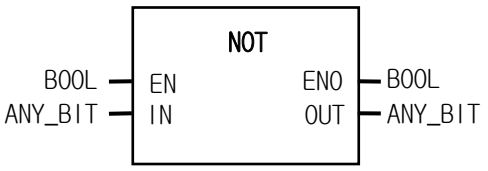
출력(OUT) : %Q0.0.1(BOOL) = 1(16#1)

1

NOT

논리 반전

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : NOT될 값 **출력** ENO : EN 값이 그대로 출력 OUT : NOT된 값 IN, OUT은 모두 같은 타입이어야 함.

■ 기능

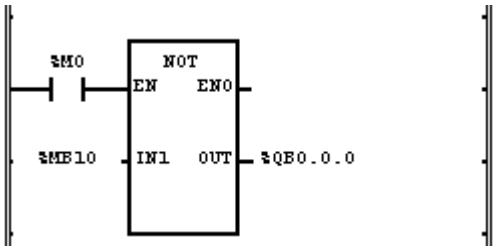
IN을 비트별로 NOT(반전)해서 OUT으로 출력시킵니다.

IN 1100 1010

OUT 0011 0101

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 NOT(논리반전) 펑션이 실행됩니다.

(2)NOT 펑션이 실행되면 입력변수로 선언된 %MB10의 데이터 값이 반전되어 출력변수로 선언된 %OB0.0.0.0에 출력됩니다.

입력(IN1) : %MB10(BYTE) = 16#CC | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

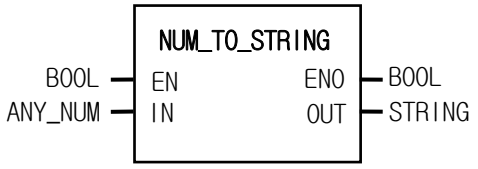
(NOT)

출력(OUT) : %QB0.0.0(BYTE) = 16#33 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

NUM_TO_STRING

숫자를 문자열로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

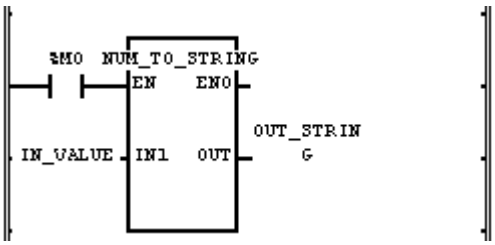
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : STRING으로 변환할 숫자 입력 **출력** ENO : EN값이 그대로 출력 OUT : 문자로 변환된 데이터

■ 기능

IN의 숫자 데이터를 문자 데이터로 변환해서 OUT으로 출력시킵니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 NUM_TO_STRING(숫자를 문자열로 변환) 펑션이 실행됩니다.
(2)NUM_TO_STRING 펑션의 입력 변수로 선언된 IN_VALUE(INT 타입) = 123이면 출력변수로 선언된 OUT_STRING = '123'이 되고, IN_VALUE(REAL 타입) = 123.0이면 OUT_STRING = '1.23E2'가 됩니다.

입력(IN1) : IN_VALUE(INT) = 123

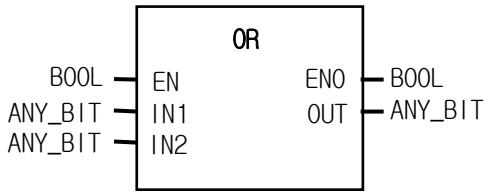
↓ (NUM_TO_STRING)

출력(OUT) : OUT_STRING(STRING) = '123'

OR

논리합

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : OR될 값 IN2 : OR될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : OR된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

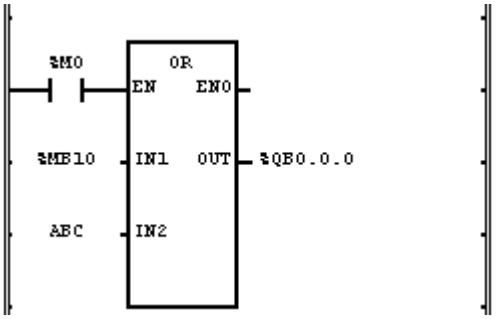
■ 기능

IN1을 IN2와 비트별로 OR해서 OUT으로 출력시킵니다.

```
  IN1  1111 ..... 0000
  OR
  IN2  1010 ..... 1010
  OUT 1111 ..... 1010
```

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 OR펑션이 실행됩니다.

(2)%MB10 = 11001100을 ABC = 11110000와 OR시킨 결과가 %QB0.0.0 = 11111100로 출력됩니다.

입력(IN1) : %MB10 (BYTE) = 16#CC

1	1	0	0	1	1	0	0

& (OR)

(IN2) : ABC(BYTE) = 16#F0

1	1	1	1	0	0	0	0

↓

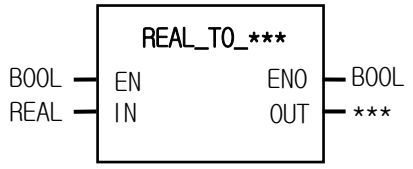
출력(OUT) : %QB0.0.0(BYTE) = 16#FC

1	1	1	1	1	1	0	0

REAL_TO_***

REAL타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 REAL값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

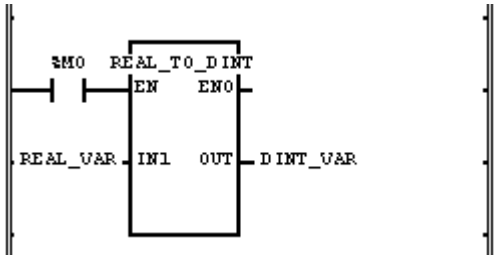
펑션	출력 타입	동작 설명
REAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127일 경우 정상 변환되나, 그외값은 에러가발생합니다. (소숫점 이하는 반올림)
REAL_TO_INT	INT	입력의 정수 부분이 -32768 ~ 32767일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_LINT	LINT	입력의 정수 부분이 -2^{63} ~ $2^{63}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_USINT	USINT	입력의 정수 부분이 0 ~ 255일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_UINT	UINT	입력의 정수 부분이 0 ~ 65,535일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_UDINT	UDINT	입력의 정수 부분이 0 ~ $2^{32}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_ULINT	ULINT	입력의 정수 부분이 0 ~ $2^{64}-1$일 경우 정상 변환되나, 그외값은 에러가 발생합니다. (소숫점 이하는 반올림)
REAL_TO_DWORD	DWORD	내부 비트 배열의 변화없이 DWORD 타입으로 변환합니다.
REAL_TO_LREAL	LREAL	REAL을 LREAL 타입으로 정상 변환합니다.

■ 에러

입력값이 출력타입에 저장할 수 있는 값보다 커서 오버 플로(Overflow)가 발생하면 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 0을 출력합니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 REAL_TO_DINT펑션이 실행됩니다.
(2)REAL_VAL(REAL 타입) = 1.234E4이면, DINT_VAL(DINT 타입) = 12340이 됩니다.

입력(IN1) : REAL_VAL(REAL) = 1.234E4

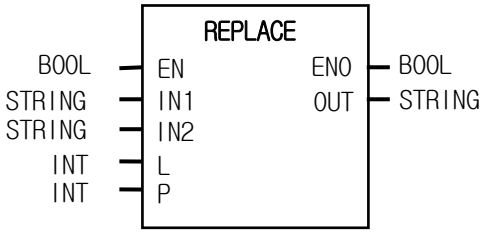
↓ (REAL_TO_DINT)

출력(OUT) : DINT_VAL(DINT) = 12340

REPLACE

문자열 대체하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 대체될 문자열 IN2 : 대체할 문자열 L : 대체될 문자열의 길이 P : 대체될 문자열의 위치 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

문자열 IN1의 P번째 문자부터 길이 L만큼의 문자를 문자열 IN2로 대체한 후 문자열 OUT에 출력시킵니다.

■ 에러

아래와 같은 경우 ERR, _LER 플래그가 셋(Set)됩니다.

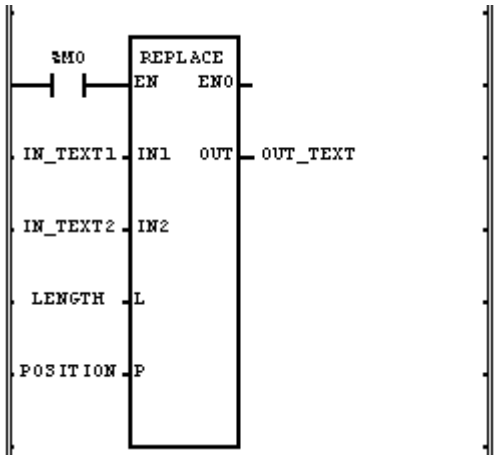
▷ P ≤0 또는 L < 0

▷ P > (IN1의 입력 문자열의 문자 수)

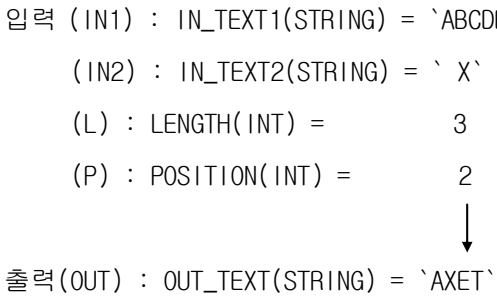
▷ 연산 결과 문자 수 > 30

■ 프로그램 예

LD



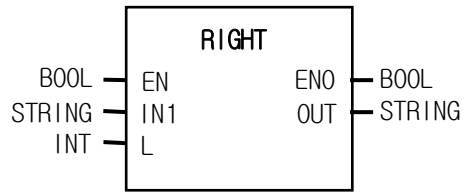
(1)실행조건(%M0)이 On하면 REPLACE(문자열 대체하기) 펑션이 실행됩니다.
(2)대체될 문자열 입력변수가 IN_TEXT1=`ABCDEF`이고, 대체할 문자열 입력변수 IN_TEXT2=`X` 이며, 대체될 문자열의 길이 입력변수 LENGTH=3, 대체될 문자열 위치 지정 입력변수 POSITION=2 이면 IN_TEXT의 `BCD`가 IN_TEXT2의 `X`로 대체되어 출력변수 OUT_TEXT에 `AXET`가 출력됩니다.



RIGHT

문자열의 오른쪽을 취하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

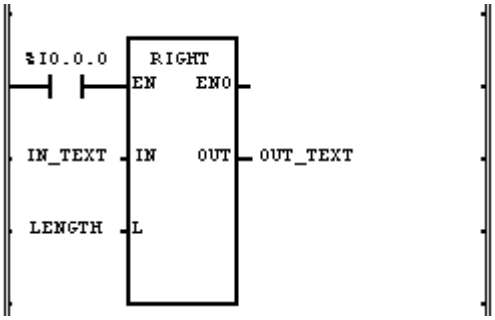
입력 문자열 IN에 대하여 오른쪽부터 문자열 길이 L만큼을 출력 문자열 OUT에 출력시킵니다.

■ 에러

L < 0 인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 RIGHT(문자열의 오른쪽 취하기) 펑션이 실행됩니다.

(2)입력변수로 선언된 문자열이 IN_TEXT=`ABCDEFG`이고, 출력할 문자열의 길이LENGTH=3이면,출력문자열 변수로 지정된 OUT_TEXT=`EFG`가 됩니다.

입력(IN1) : IN_TEXT(STRRING) = `ABCDEFG`

(L) : LENGTH(INT) = 3

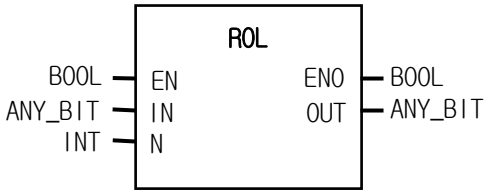
↓ (RIGHT)

출력(OUT) : OUT_TEXT(STRRING) = `EFG`

ROL

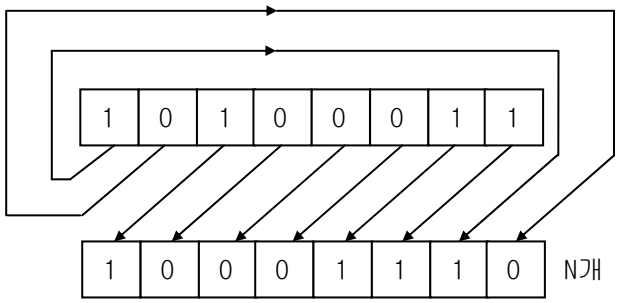
왼쪽으로 회전(Rotate Left)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트수 **출력** ENO : EN값이 그대로 출력 OUT : 회전된 값

■ 기능

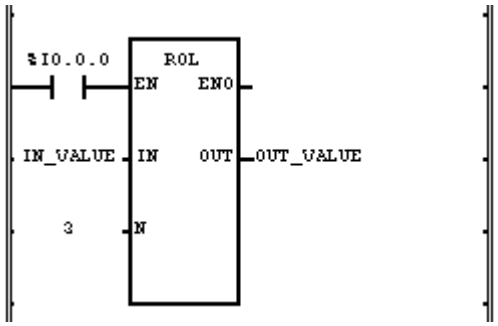
입력 IN을 N 비트수만큼 왼쪽으로 회전시킵니다.



■ 프로그램 예

입력 %I.0.0.0이 On하면 입력 데이터(1100_1100_1100_1100:16#CCCC)의 값을 좌로 3비트 만큼 회전시키는 프로그램

LD



(1)회전할 데이터 값을 입력한 변수 IN_VALUE로 설정한다.
(2)좌회전할 비트수 3을 : 회전할 비트수 지정 입력(N)에 살다
(3)회전된 데이터 값을 출력할 출력변수를 OUT_VALUE로 설정한다.
(4)실행조건 %I0.0.0이 On하면 ROL(왼쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트를 좌로 3비트 회전하여 출력변수로 선언된 OUT_VALVE값에 출력된다.

입력(IN1) : IN_VALUE(WORD) = 16#CCCC

1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

(N) : 3 (ROL)

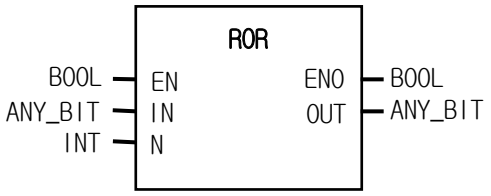
출력(OUT) :OUT_VALUE(WORD)-16#6666

0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

ROR

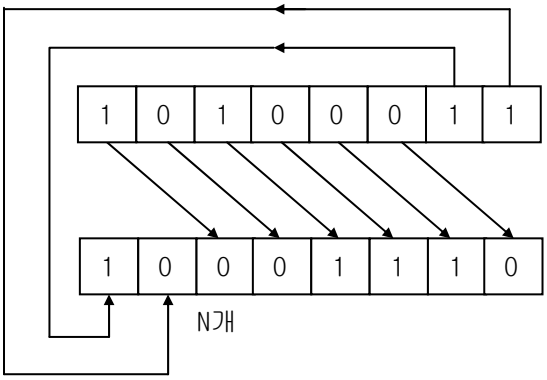
오른쪽으로 회전(Rotate Right)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트수 **출력** ENO : EN값이 그대로 출력 OUT : 회전된 값

■ 기능

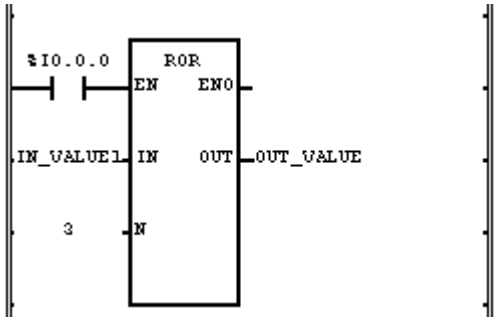
입력 IN을 N 비트수만큼 오른쪽으로 회전시킵니다.



■ 프로그램 예

입력 %I.0.0.0이 On하면 입력 데이터값(1110001100110001:16#E331)을 우로 3비트 만큼 회전시키는 프로그램

LD



(1)회전할 데이터 값을 입력한 변수를 IN_VALUE1로 설정한다.
(2)우회전할 비트수 3을 : 회전할 비트수 지정 입력(N)에 살다
(3)실행조건 %I0.0.0이 On하면 ROR(오른쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 우로 3비트만큼 회전되어 출력변수로 선언된 OUT_VALUE값에 출력된다.

입력(IN1) :IN_VALUE1(WORD)=16#E331

1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	1

(N) : 3 (ROR)

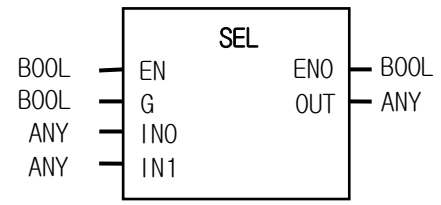
출력(OUT) :OUT_VALUE(WORD)=16#3C

0	0	1	1	1	1	0	0	0	1	1	0	0	1	1	0

SEL

둘중 선택

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

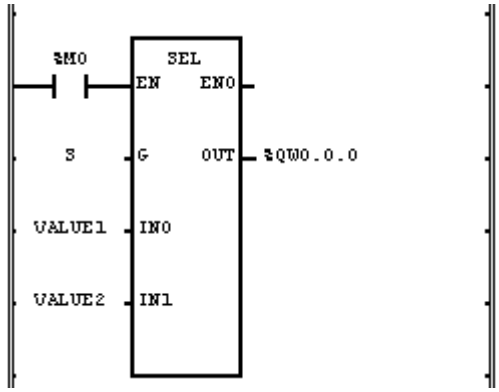
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 G : 선택 IN0 : 선택될 값 IN1 : 선택될 값 **출력** ENO : EN 값이 그대로 출력 OUT : 선택된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

■ 기능

G가 0이면 IN0이 OUT으로, G가 1이면 IN1이 OUT으로 출력됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 SEL(둘중 선택) 펑션이 실행됩니다.

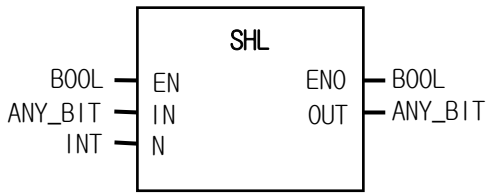
(2)SEL 펑션이 실행되면 S = 1일 때, VALUE1 = 16#1110, VALUE2 = 16#FF00이면 %QW0.0.0 = 16#FF00이 됩니다.

입력 (G) : S = 1
(IN0) : VALUE1(WORD) = 16#1110
(IN1) : VALUE2(WORD) = 16#FF00
↓ (SEL)
출력(OUT) : %QW0.0.0(WORD) = 16#FF00

SHL

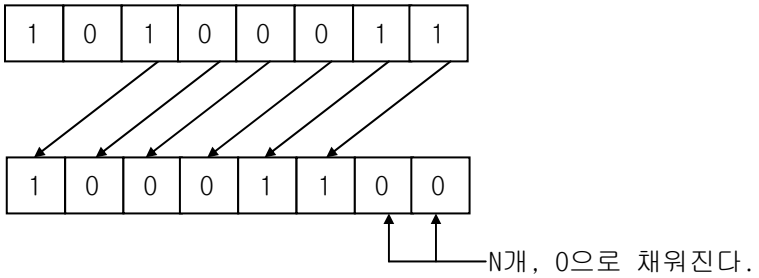
왼쪽으로 이동(Shift Left)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트열 N : 이동할 비트수 **출력** ENO : EN값이 그대로 출력 OUT : 이동된 값

■ 기능

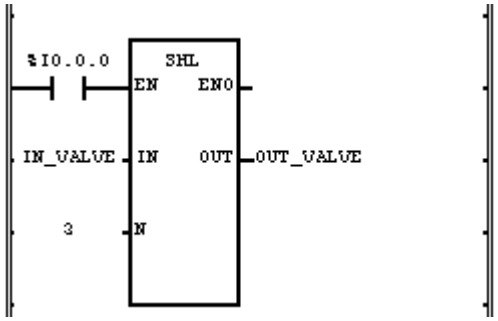
입력 IN을 N 비트 수만큼 왼쪽으로 이동합니다.
입력 IN의 맨 오른쪽에 있는 N개 비트는 0으로 채워집니다.



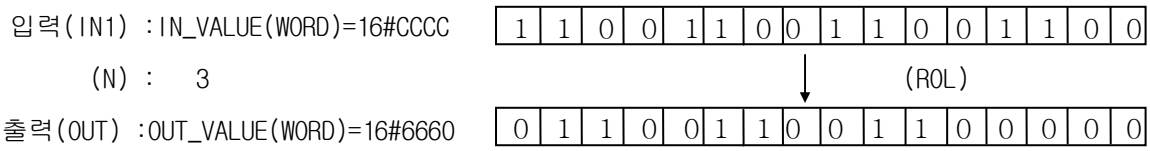
■ 프로그램 예

입력 %I.0.0.00이 On하면 입력 데이터값(1100_1100_1100_1100:16#CCCC)을 좌로 3비트 만큼 이동시키는 프로그램

LD



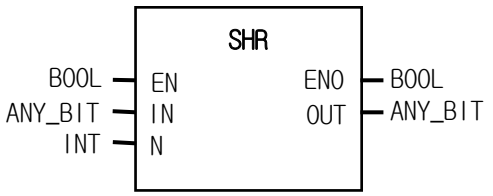
(1)이동할 데이터 값을 입력할 변수를 IN_VALUE(11001110:16#CE)로 설정한다.
(2)좌로 이동한 비트수 3을 지정입력(N)에 살다. (변수 지정후 쓰기로 가능)
(3)실행조건(%Z0.0.0)이 On하면 SHL(왼쪽으로 이동) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 좌로 3비트 이동하여, 출력변수로 선언된 OUT_VALUE에 출력됩니다.



SHR

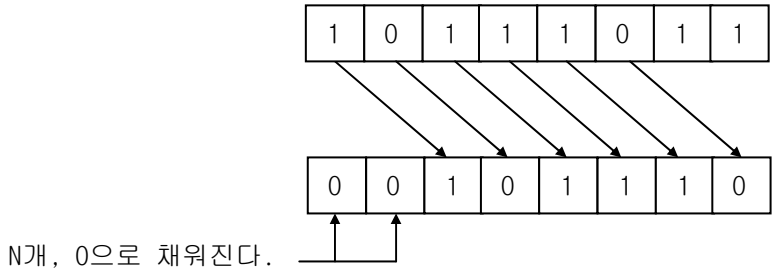
오른쪽으로 이동(Shift Right)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트열 N : 이동할 비트수 **출력** ENO : EN값이 그대로 출력 OUT: 이동된 값

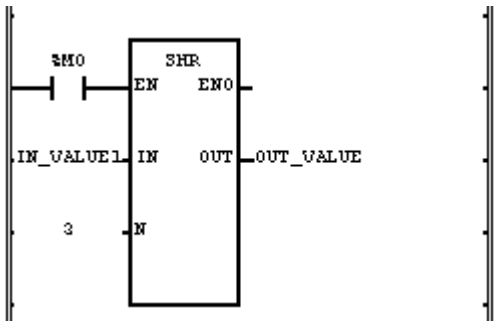
■ 기능

입력 IN을N 비트수만큼 오른쪽으로 이동합니다.
입력 IN의 맨 왼쪽에 있는 N개 비트는 0으로 채워집니다.



■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 SHL(왼쪽으로 이동) 펑션이 실행됩니다.
(2)입력변수로 설정된 데이터 비트가 우로 3비트 이동하여, 출력변수로 선언된 OUT_VALUE에 출력됩니다.

입력(IN1): IN_VALUE(WORD) = 16#E331

1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	1

(N) : 3 (ROR)

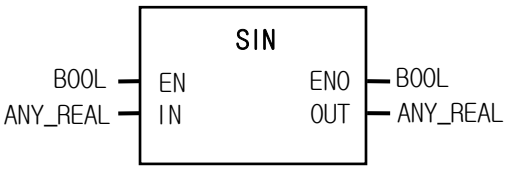
출력(OUT) : OU2T_VALUE(WORD)-16#1C66

0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	0

SIN

Sine 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Sine 연산의 각도 입력값(Radian) **출력** ENO : EN값이 그대로 출력 OUT : Sine 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

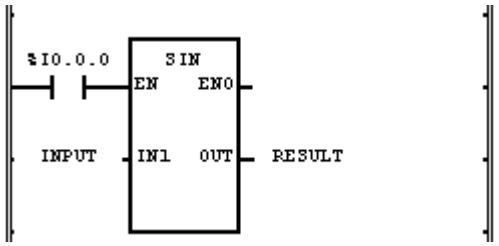
■ 기능

IN의 Sine값을 구해 OUT으로 출력시킵니다.

OUT = SIN (IN)

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 SIN(Sin연산) 펑션이 실행됩니다.

(2)INPUT으로 선언된 입력변수의 값이 1.0471 .... ($\pi/3$ rad = 60°) 일 때 출력변수로 선언된 RESULT는 0.8660 .... ($\sqrt{3}/2$)이 출력됩니다..

$SIN(\pi/3) = \sqrt{3}/2 = 0.8660$

입력(IN1) : INPUT(REAL) = 1.0471

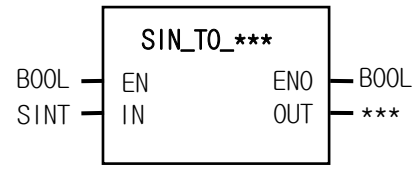
↓ (SIN)

출력(OUT) : RESULT(REAL) = 8.65976572E-01

SINT_TO_***

SINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Short Integer값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
SINT_TO_INT	INT	INT 타입으로 정상 변환합니다.
SINT_TO_DINT	DINT	DINT 타입으로 정상 변환합니다.
SINT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
SINT_TO_USINT	USINT	입력이 0 ~ 127 일경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_UINT	UINT	입력이 0 ~ 127 일경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_UDINT	UDINT	입력이 0 ~ 127 일경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_ULINT	ULINT	입력이 0 ~ 127 일경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
SINT_TO_BYTE	BYTE	내부 비트 배열의 변화없이 BYTE 타입으로 변환합니다.
SINT_TO_WORD	WORD	상위 비트들을 0으로 채운 WORD 타입으로 변환합니다.
SINT_TO_DWORD	DWORD	상위 비트들을 0으로 채운 DWORD 타입으로 변환합니다.
SINT_TO_LWORD	LWORD	상위 비트들을 0으로 채운 LWORD 타입으로 변환합니다.
SINT_TO_BCD	BYTE	입력이 0~99일 경우 정상 변환 되나, 그 외 값은 에러가 발생합니다.
SINT_TO_REAL	REAL	SINT를 REAL 타입으로 정상 변환합니다.
SINT_TO_LREAL	LREAL	SINT를 LREAL 타입으로 정상 변환합니다.

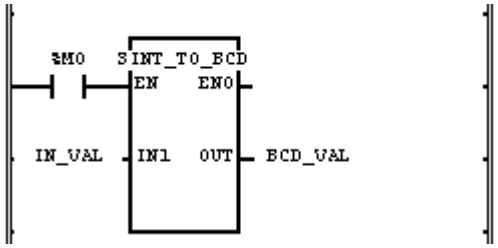
■ 에러

변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(% M0)이 On하면 SINT_TO_BCD 펑션이 실행됩니다.
(2)입력변수로 선언된IN_VAL(SINT타입)=64(2#0100_0000)이면,OUT_VAL(BCD타입)=16#64(2#0110_0100)가 됩니다.

입력(IN1) : IN_VAL(SINT) = 64(16#40)

0	1	0	0	0	0	0	0

↓ (SINT_TO_BCD)

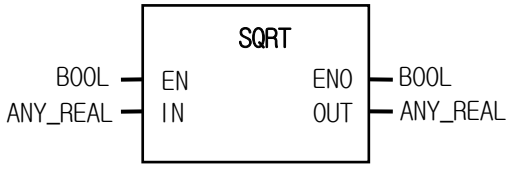
출력(OUT) : OUT_VAL(BCD) = 16#64(16#64)

0	1	1	0	0	1	0	0

SQRT

제곱근 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 제곱근 연산의 입력값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 제곱근값 IN, OUT은 같은 데이터 타입이어야 함.

■ 기능

IN의 제곱근값을 구해OUT으로 출력시킵니다.

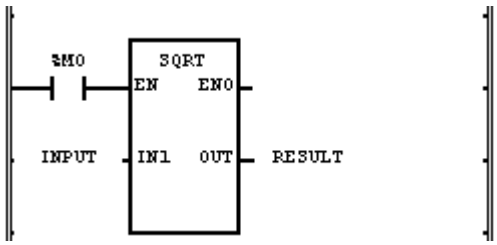
$OUT = \sqrt{IN}$

■ 에러

IN의 값이 음수일 때 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 SQRT(제곱근 연산) 펑션이 실행됩니다.

(2)INPUT으로 선언된 입력변수의 값이 9.0 일 때 출력변수로 선언된 RESULT 는 3.0이 출력됩니다.

$\sqrt{9.0} = 3.0$

입력(IN1) : INPUT(REAL) = 9.0

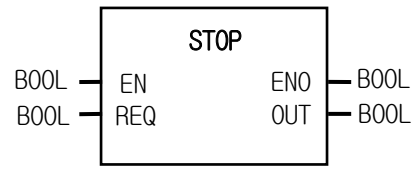
↓ (SQRT)

출력(OUT) : RESULT(REAL) = 3.0

STOP

프로그램에 의한 운전정지

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

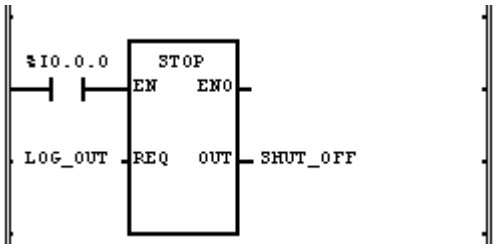
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 RE : 프로그램에 의한 운전정지 요구 **출력** ENO : EN값이 그대로 출력 OUT : STOP동작이 실행되면 1출력

■ 기능

▷ EN이 1이고 REQ에 1의 값이 들어오면 운전을 정지하고 STOP모드로 됩니다.
▷ 'STOP'펑션이 수행되면 수행중인 스캔 프로그램의 수행을 완료한 후 정지합니다.
▷ 전원을 재 투입하거나 운전모드를 STOP으로 하였다가 RUN으로 하면 재 기동이 됩니다.

■ 프로그램 예

LD

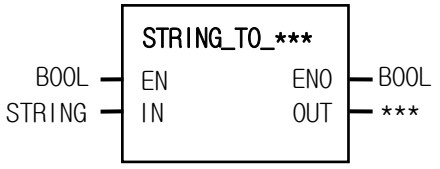


(1)실행조건(%I0.0.0)이 On하고 LOG_OUT가 1이 되면 실행중인 스캔 프로그램을 완료한 후 STOP 모드로 됩니다.
(2)입력변수로 선언한 'STOP'펑션 수행 후 안정된 상태에서 PLC의 전원을 끄는 것이 좋습니다.

STRING_TO_***

STRING 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 문자열 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

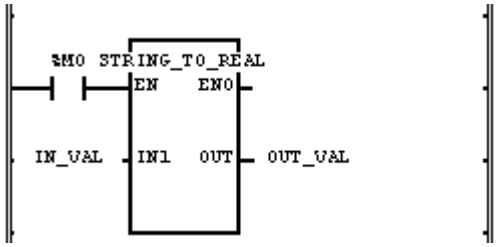
펑 션	출력 타입	동작 설명
STRING _TO_SINT	SINT	STRING을 SINT 타입으로 변환합니다.
STRING _TO_INT	INT	STRING을 INT 타입으로 변환합니다.
STRING _TO_DINT	DINT	STRING을 DINT 타입으로 변환합니다.
STRING _TO_LINT	LINT	STRING을 LINT 타입으로 변환합니다.
STRING _TO_USINT	USINT	STRING을 USINT 타입으로 변환합니다.
STRING _TO_UINT	UINT	STRING을 UINT 타입으로 변환합니다.
STRING _TO_UDINT	UDINT	STRING을 UDINT 타입으로 변환합니다.
STRING _TO_ULINT	ULINT	STRING을 ULINT 타입으로 변환합니다.
STRING _TO_BOOL	BOOL	STRING을 BOOL 타입으로 변환합니다.
STRING _TO_BYTE	BYTE	STRING을 BYTE 타입으로 변환합니다.
STRING _TO_WORD	WORD	STRING을 WORD 타입으로 변환합니다.
STRING _TO_DWORD	DWORD	STRING을 DWORD 타입으로 변환합니다.
STRING _TO_LWORD	LWORD	STRING을 LWORD 타입으로 변환합니다.
STRING _TO_REAL	REAL	STRING을 REAL 타입으로 변환합니다.
STRING _TO_LREAL	LREAL	STRING을 LREAL 타입으로 변환합니다.
STRING _TO_DT	DT	STRING을 DT 타입으로 변환합니다.
STRING _TO_DATE	DATE	STRING을 DATE 타입으로 변환합니다.
STRING _TO_TOD	TOD	STRING을 TOD 타입으로 변환합니다.
STRING _TO_TIME	TIME	STRING을 TIME 타입으로 변환합니다.

■ 에러

입력 문자 패턴이 출력 데이터 타입과 맞지 않을 때 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 STRING_TO_REAL 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(STRING 타입) = '-1.34E12'면,출력변수로 선언된 OUT_VAL(REAL= -1.34E12 가 됩니다.

입력(IN1) : IN_VAL(STRING) = '-1.34E12'

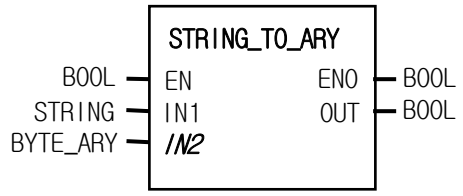
↓ (STRING_TO_REAL)

출력(OUT) : OUT_VAL(REAL) = -1.34E12

STRING_TO_ARY

문자열을 Byte Array로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

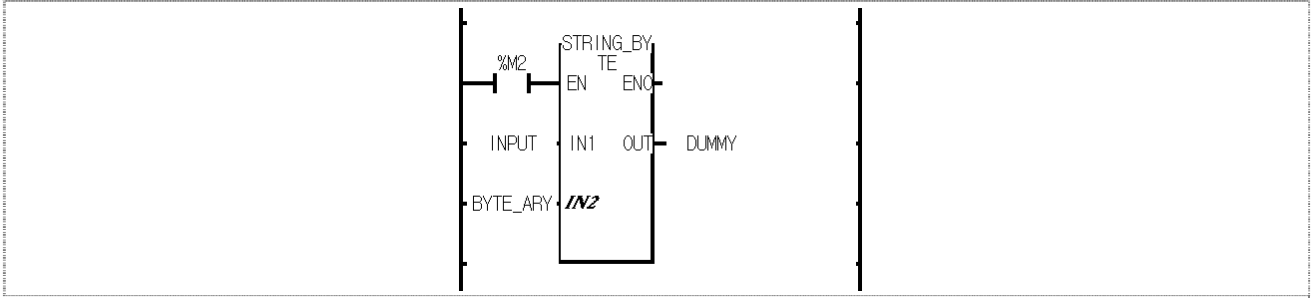
펑 션	설 명
	**입력** EN : 1 일때 펑션 실행 IN1 : String입력 **출력** ENO : EN값을 그대로 출력 OUT : dummy 출력 **입출력** IN2 : 변환된 Byte Array 출력

■ 기능

하나의 String을 30개의 Byte Array로 변환합니다.

■ 프로그램 예

LD



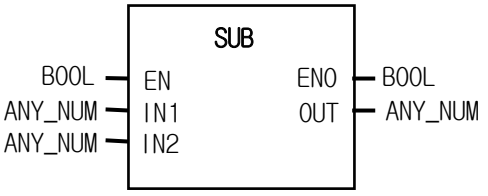
(1)실행조건(%M2)이 On하면 STRING_BYTE 펑션이 실행됩니다.

(2)입력변수인 INPUT이 "GM4-CPUA"일 경우 입출력 변수인 BYTE_ARY에 하위 바이트부터 순서대로 16#{22("),47(G),4D(M),34(4),2D(-),43(C),50(P),55(U),41(A),22(")}가 출력됩니다.

SUB

빼기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 빼어질 값 IN2 : 뺄 값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 뺀 결과 값 IN1,IN2,OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함

■ 기능

IN1에서 IN2를 빼서 OUT으로 출력시킵니다.

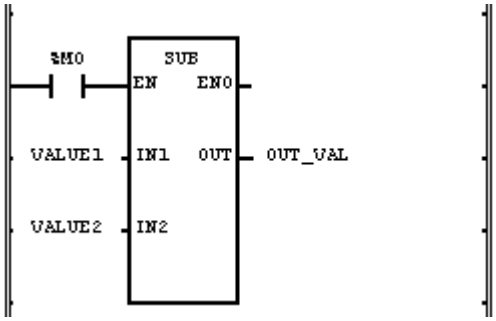
OUT = IN1 − IN2

■ 에러

출력값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 SUB(빼기) 펑션이 실행됩니다.

(2)입력변수 값이 VALUE1 = 300,VALUE2 = 200이면, 출력변수로 설정한 OUT_VAL는 연산을 실시한 결과 (300-200=100)가 출력됩니다.

입력(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

- (SUB)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

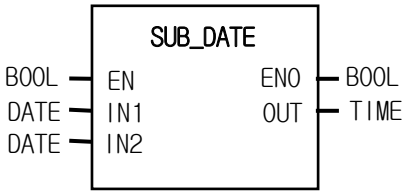
출력(OUT) : OUT_VAL(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0

SUB_DATE

날짜 빼기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 날짜 IN2 : 뺄 날짜 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 두 날짜간의 차이를 시간으로 출력

■ 기능

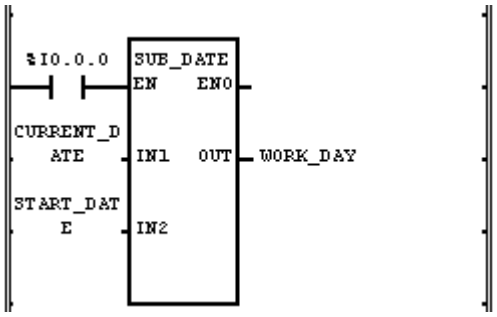
IN1(기준날짜)에서 IN2(특정날짜)를 빼서 날짜 차이를 OUT으로 출력시킵니다.

■ 에러

출력값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다.
날짜 차이가 TIME 데이터 타입의 범위 T#49D17H2M47S295MS를 넘거나 날짜 연산의 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 SUB_DATE(날짜빼기) 펑션이 실행됩니다.
(2)입력변수로 선언한 현재의 날짜 CURRENT_DATE가 D#1995-12-15이고 시스템이 가동을 시작한 날짜 START_DATE가 D#1995-11-1이면, 출력변수로 선언한 조업한 날짜 WORK_DAY에는 T#44D 가 출력됩니다.

입력(IN1) : CURRENT_DATE(DATE) = D#1995-12-15

(SUB_DATE)

(IN2) : START_DATE(DATE) = D#1995-11-1

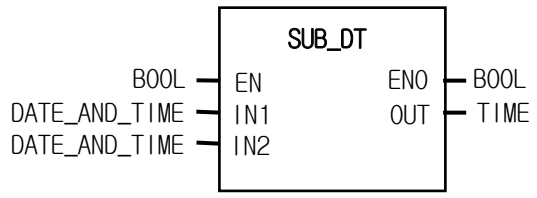
↓

출력(OUT) : WORK_DAY(TIME) = T#44D

SUB_DT

시간과 날짜 빼기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 기준 날짜와 시각 IN2 : 뺄 날짜와 시각 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 뺀 결과 시간

■ 기능

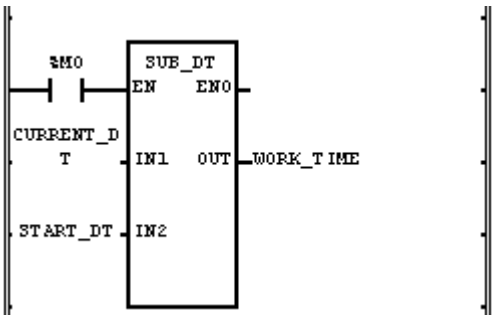
IN1(기준 날짜와 시각)에서 IN2(특정 날짜와 시각)를 빼서 시간 차이를 OUT으로 출력시킵니다.

■ 에러

출력값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다.
날짜와 시각 빼기 연산의 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 SUB_DT(시간과 날짜빼기) 펑션이 실행됩니다.
(2)입력변수로 선언한 현재의 날짜와 시각 CURRENT_DT가 DT#1995-12-15-14:30:00이고 조업이 재개된 날짜와 시각 START_DT가 DT#1995-12-13-12:00:00이면, 출력변수로 선언된 연속 조업한 시간 WORK_TIME에는T#2D2H30M가 출력됩니다.

입력(IN1) : CURRENT_DT(DT) = DT#1995-12-15-14:30:00

(SUB_DATE)

(IN2) : START_DT(DT) = DT#1995-12-13-12:00:00

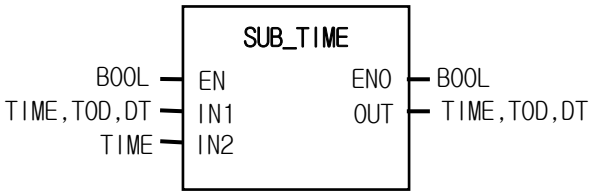
↓

출력(OUT) : WORK_TIME(TIME) = T#2D2H30M

SUB_TIME

시간 빼기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준시각 또는 시간 IN2 : 뺄 시간 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 뺀 결과 시각 또는 시간 OUT의 타입은 입력 IN1의 타입을 따름. 즉 IN1의 타입이 TIME이면, 출력 OUT의 타입도 TIME임.

■ 기능

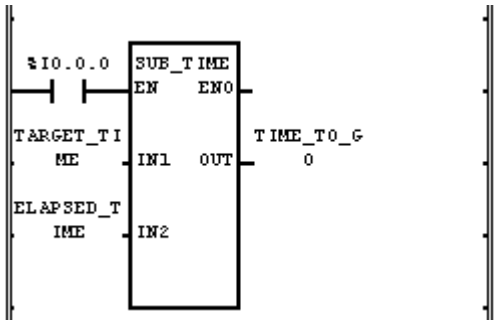
▷ IN1이 TIME일 경우에는 시간에서 시간을 빼서 차이 시간을 출력시킵니다.
▷ IN1이 TIME_OF_DAY일 경우에는 기준시각에서 시간을 빼서 하루중의 시각을 출력시킵니다.
▷ IN1이 DATE_AND_TIME일 경우에는 기준이 되는 날짜와 시각에서 시간을 빼서 날짜와 시각을 출력시킵니다.

■ 에러

출력값이 해당 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다.
시간에서 시간을 뺀 결과가 음수가 되거나 시각(TOD)에서 시간을 뺀 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 SUB_TIME(시간빼기) 펑션이 실행됩니다.
(2)입력변수로 선언된 전체 작업시간 TARGET_TIME이 T#2H30M이고, 경과된 시간 ELAPSED_TIME이 T#1H10M30S300MS면 출력변수로 선언된 남아 있는 작업시간 TIME_TO_GO에는 T#1H19M29S700MS가 출력됩니다.

입력(IN1) : TARGET_TIME(TIME) = T#2H30M

(SUB_TIME)

(IN2) : ELAPSED_TIME(TIME) = T#1H10M30S300MS

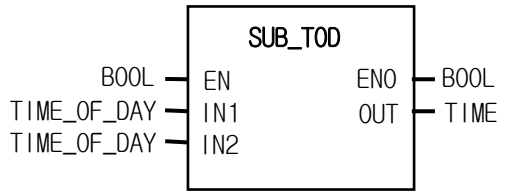
↓

출력(OUT) : TIME_TO_GO(TIME) = T#1H19M29S700MS

SUB_TOD

시각에서 시각 빼기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 시각 IN2 : 뺄 시각 **출력** ENO : 에러없이 실행되면 1 출력 OUT : 뺀 결과 시간

■ 기능

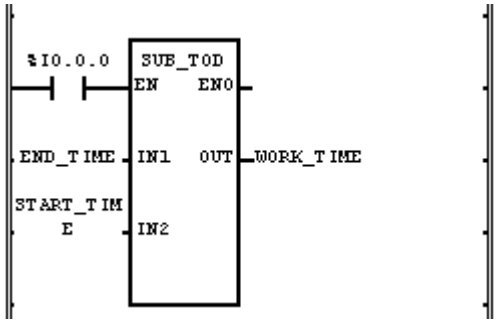
IN1(기준시각)에서 IN2(특정시각)를 빼서 시간 차이를 OUT으로 출력시킵니다.

■ 에러

시각에서 시각을 뺀 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

LD



(1)실행조건(%I0.0.0)이 On하면 SUB_TOD(시각에서 시각빼기) 펑션이 실행됩니다.

(2)입력변수로 선언된 END_TIME이 TOD#14:20:30.5이고 작업을 시작한 START_TIME이 TOD#12:00:00이면, 출력변수로 선언된 작업에 소요된 시간 WORK_TIME에는 T#2H20M30S500MS 가 출력됩니다.

입력(IN1) : END_TIME(TOD) = TOD#14:20:30.5

(SUB_TOD)

(IN2) : START_TIME(TOD) = TOD#12:00:00

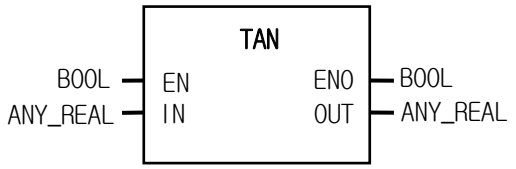
↓

출력(OUT) : WORK_TIME(TIME) = T#2H20M30S500MS

TAN

Tangent 연산

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Tangent 각도입력 값(Radian) **출력** ENO : EN값이 그대로 출력 OUT : Tangent연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

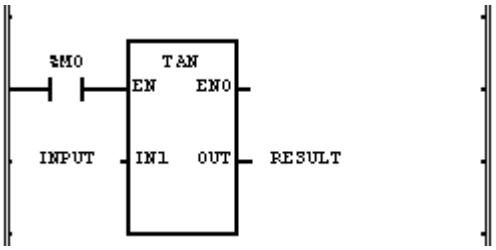
■ 기능

IN의Tangent값을 구해 OUT으로 출력시킵니다.

OUT = TAN(IN)

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 TAN(Tangent연산) 펑션이 실행합니다.

(2)INPUT으로 선언된 입력변수의 값이 0.7853 .... (π/4 rad = 45°)일 때 출력 변수로 선언된 RESULT는 1.0000 입니다.

$$TAN(\pi/4) = 1$$

입력(IN1) : INPUT(REAL) = 0.7853

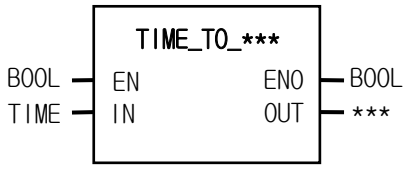
↓ (TAN)

출력(IN2) : RESULT(REAL) = 9.99803722E-01

TIME_TO_***

TIME 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 시간 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

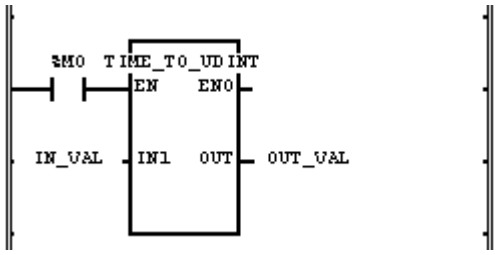
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
TIME_TO_UDINT	UDINT	TIME를 UDINT 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화없이 데이터 타입만 변환합니다.
TIME_TO_DWORD	DWORD	TIME를 DWORD 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화없이 데이터 타입만 변환합니다.
TIME_TO_STRING	STRING	TIME를 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 TIME_TO_UDINT 펑션이 실행됩니다.

(2)입력변수로 선언된 IN_VAL(TIME 타입) = T#120MS이면,출력변수로 선언된 OUT_VAL(UDINT 타입) = 120 이 됩니다.

입력(IN1) : IN_VAL(TIME) = T#120MS(16#78)

0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

(TIME_TO_UDINT)

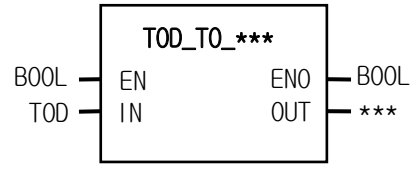
출력(OUT) : OUT_VAL(UDINT) = 120(16#78)

0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

TOD_TO_***

TOD 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 하루중 시각 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

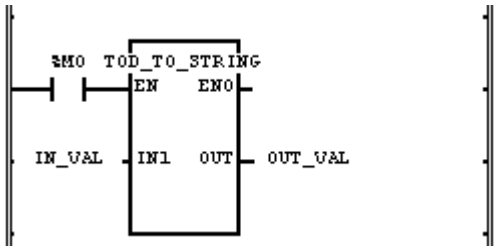
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
TOD_TO_UDINT	UDINT	TOD를 UDINT 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화없이 데이터 타입만 변환합니다.
TOD_TO_DWORD	DWORD	TOD를 DWORD 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화없이 데이터 타입만 변환합니다.
TOD_TO_STRING	STRING	TOD를 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 TOD_TO_STRING 펑션이 실행됩니다.

(2)입력변수로 선언된 IN_VAL(TOD 타입) = TOD#12:00:00이면,출력변수로 선언된 OUT_VAL(STRING타입) = 'TOD#12:00:00'이 됩니다.

입력(IN1) : IN_VAL(TOD) = TOD#12:00:00

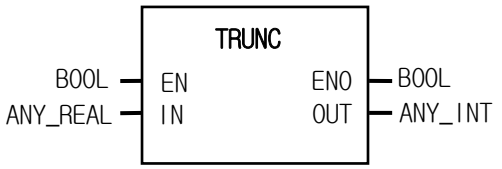
↓ (TOD_TO_STRING)

출력(IN2) : OUT_VAL(STRING) = 'TOD#12:00:00'

TRUNC

소수점 이하 값을 버리고 정수로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 변환될 Real값 **출력** ENO : 에러없이 실행되면 1을 출력 OUT: 정수로 변환된 값

■ 기능

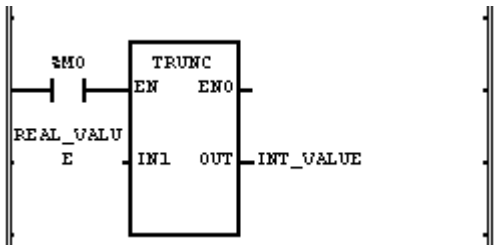
펑션	입력 타입	출력 타입	동 작 설 명
TRUNC	REAL LREAL	DINT LINT	입력 IN으로 들어온 부동소수의 소수점 이하 값은 버리고 OUT으로 정수값을 출력합니다.

■ 에러

변환된 값이 OUT에 연결된 데이터 타입의 최대값보다 큰 경우 또는 OUT에 연결된 변수가 Unsigned Integer이고 변환된 출력값이 음수일 때 OUT으로 0을 출력한 경우에는 _ERR, _LER 플래그가 셋(Set) 됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 TRUNC (소수점 이하값을 버리고 정수변환) 펑션이 실행합니다.
(2)입력변수로 선언된 REAL_VALUE(REAL 타입)= 1.6이면 INT_VALUE(INT타입) = 1이 됩니다.
REAL_VALUE(REAL 타입)= -1.6이면 INT_VALUE(INT타입) = -1이 됩니다.

입력(IN1) : REAL_VALUE(REAL) = 1.6

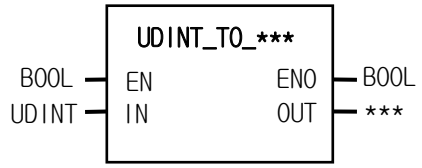
↓ (TRUNC)

출력(OUT) : INT_VALUE(INT) = 1

UDINT_TO_***

UDINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Double Integer값 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
UDINT_TO_SINT	SINT	입력이 0~127일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UDINT_TO_INT	INT	입력이 0~32767일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UDINT_TO_DINT	DINT	입력이 0~2,147,483,64일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UDINT_TO_LINT	LINT	UDINT를 LINT 타입으로 정상 변환합니다.
UDINT_TO_USINT	USINT	입력이 0~255일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UDINT_TO_UINT	UINT	입력이 0~65535일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UDINT_TO_ULINT	ULINT	UDINT를 ULINT 타입으로 정상 변환합니다.
UDINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
UDINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
UDINT_TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환됩니다.
UDINT_TO_DWORD	DWORD	내부 비트 배열의 변환없이 DWORD 타입으로 변환합니다.
UDINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
UDINT_TO_BCD	DWORD	0~99,999,999의 값은 정상변환되나, 그외의 값은 에러가 발생합니다.
UDINT_TO_REAL	REAL	UDINT를 REAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.
UDINT_TO_LREAL	LREAL	UDINT를 LREAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.
UDINT_TO_TOD	TOD	내부 비트 배열의 변환없이 TOD 타입으로 변환합니다.
UDINT_TO_TIME	TIME	내부 비트 배열의 변환없이 TIME 타입으로 변환합니다.

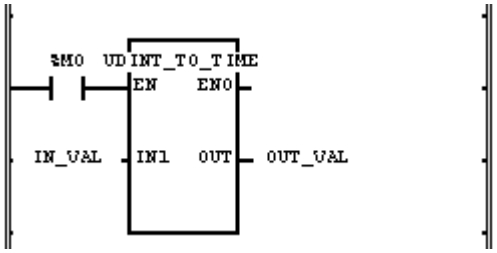
■ 에러

변환에러 발생시 _ERR , _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 UDINT_TO_TIME 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(UDINT 타입) = 123이면,출력변수로 선언된 OUT_VAL(TIME 타입) = T#123MS 가 됩니다.

입력(IN1) : IN_VAL(UDINT) = 123

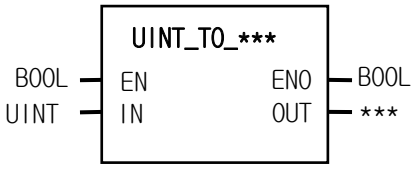
↓

출력(OUT) : OUT_VAL(TIME) = T#123MS

UINT_TO_***

UINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Integer값 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
UINT_TO_SINT	SINT	입력이 0~127일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UINT_TO_INT	INT	입력이 0~32,767일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UINT_TO_DINT	DINT	UINT를 UDINT 타입으로 정상 변환합니다.
UINT_TO_LINT	LINT	UINT를 ULINT 타입으로 정상 변환합니다.
UINT_TO_USINT	USINT	입력이 0~255일 경우 정상변환되나 그외 값은 에러가 발생합니다.
UINT_TO_UDINT	UDINT	UINT를 UDINT 타입으로 정상 변환합니다.
UINT_TO_ULINT	ULINT	UINT를 ULINT 타입으로 정상 변환합니다.
UINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
UINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
UINT_TO_WORD	WORD	내부 비트 배열의 변환없이 WORD 타입으로 변환합니다.
UINT_TO_DWORD	DWORD	상위 비트를 0으로 채운 DWORD 타입으로 변환합니다.
UINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
UINT_TO_BCD	BCD	0~99,999,999의 값은 정상변환되나, 그외의 값은 에러가 발생합니다.
UINT_TO_REAL	REAL	UINT를 REAL타입으로 변환합니다.
UINT_TO_LREAL	LREAL	UINT를 LREAL타입으로 변환합니다.
UNIT_TO_DATE	DATE	내부 비트 배열의 변환없이 DATE 타입으로 변환합니다.

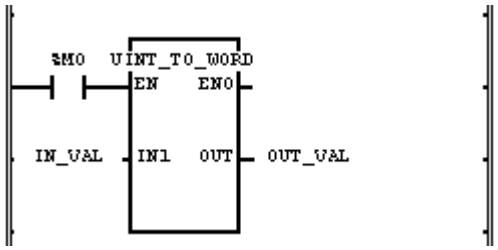
■ 에러

변환에러 발생시 _ERR , _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 UINT_TO_WORD 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(UINT 타입) = 255(2#0000_0000_1111_1111)이면, 출력변수로 선언된 OUT_VAL(WORD 타입) = 2#0000_0000_1111_1111이 됩니다.

입력(IN1) : IN_VAL(UINT) = 255

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

(UINT_TO_WORD)

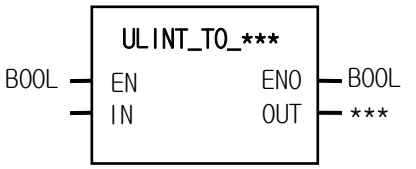
출력(OUT) : OUT_VAL(WORD) = 16#FF

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

ULINT_TO_***

ULINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Long Integer값 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
ULINT_TO_SINT	SINT	입력이 0~127일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_INT	INT	입력이 0~32,767일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_DINT	DINT	입력이 0~2^{31}-1일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_LINT	LINT	입력이 0~2^{63}-1일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_USINT	USINT	입력이 0~255일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_UINT	UINT	입력이 0~65,535일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_UDINT	UDINT	입력이 0~2^{32}-1일 경우 정상변환되나 그외 값은 에러가 발생합니다.
ULINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
ULINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
ULINT_TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환합니다.
ULINT_TO_DWORD	DWORD	하위 32비트를 취해 DWORD 타입으로 변환합니다.
ULINT_TO_LWORD	LWORD	내부 비트 배열의 변환없이 LWORD 타입으로 변환합니다.
ULINT_TO_BCD	BCD	0~9,999,999,999,999,999의 값은 정상 변환되나, 그외 값은 에러가 발생합니다.
ULINT_TO_REAL	REAL	ULINT를 REAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.
ULINT_TO_LREAL	LREAL	ULINT를 LREAL 타입으로 변환합니다. 변환중 정밀도에 따른 오차가 발생할 수 있습니다.

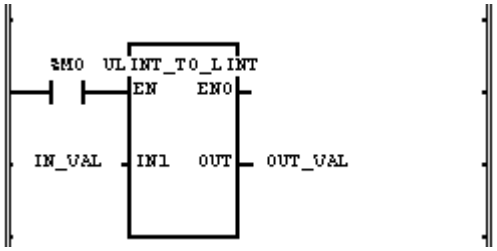
■ 에러

변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 ULINT_TO_LINT 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(ULINT 타입) = 123,567,899이면,출력변수로 선언된 OUT_VAL(LINT 타입) = 123,567,899가 됩니다.

입력(IN1) : IN_VAL(ULINT) = 123,567,899

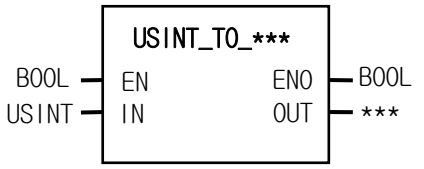
↓ (ULINT_TO_LINT)

출력(OUT) : OUT_VAL(LINT) = 123,567,899

USINT_TO_***

USINT 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Short Integer값 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
USINT_TO_SINT	SINT	입력이 0~127일 경우 정상변환되나 그외 값은 에러가 발생합니다.
USINT_TO_INT	INT	입력을 INT 타입으로 정상 변환합니다.
USINT_TO_DINT	DINT	입력을 DINT 타입으로 정상 변환합니다.
USINT_TO_LINT	LINT	입력을 LINT 타입으로 정상 변환합니다.
USINT_TO_UINT	UINT	입력을 UINT 타입으로 정상 변환합니다.
USINT_TO_UDINT	UDINT	입력을 UDINT 타입으로 정상 변환합니다.
USINT_TO_ULINT	ULINT	입력을 ULINT 타입으로 정상 변환합니다.
USINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
USINT_TO_BYTE	BYTE	내부 비트 배열의 변환없이 BYTE 타입으로 변환합니다.
USINT_TO_WORD	WORD	상위 비트를 0으로 채운 WORD 타입으로 변환합니다.
USINT_TO_DWORD	DWORD	상위 비트를 0으로 채운 DWORD 타입으로 변환합니다.
USINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
USINT_TO_BCD	BCD	0~99의 값은 정상변환되나, 그외의 값은 에러가 발생합니다.
USINT_TO_REAL	REAL	USINT를 REAL 타입으로 변환합니다.
USINT_TO_LREAL	LREAL	USINT를 LREAL 타입으로 변환합니다.

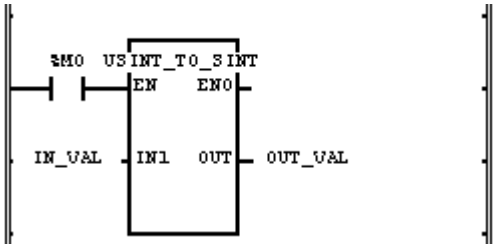
■ 에러

변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다.

에러 발생시 출력 타입의 비트수 만큼 하위 비트를 취해 내부 비트 배열의 변환없이 출력 시킵니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 ULINT_TO_SINT 펑션이 실행됩니다.
(2)입력변수로 선언된 IN_VAL(USINT 타입) = 123이면, 출력변수로 선언된 OUT_VAL(SINT 타입) = 123이 됩니다.

입력(IN1) : IN_VAL(USINT) = 123(16#7B) | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

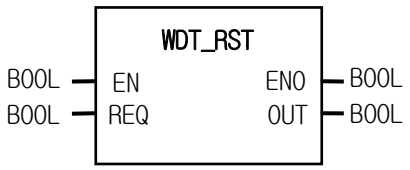
(ULINT_TO_SINT)

출력(OUT) : OUT_VAL(SINT) = 123(16#7B) | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

WDT_RST

Watch_Dog타이머 초기화

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 REQ : Watch_Dog타이머 초기화 요구 **출력** ENO : EN값이 그대로 출력 OUT : Watch_Dog 타이머를 초기화 후 1출력

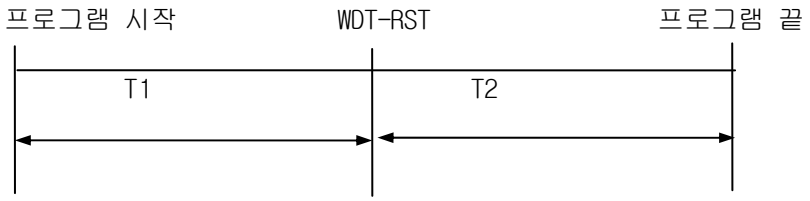
■ 기능

▷ 프로그램 중에서 Watch-Dog Timer를 Reset 시킵니다.

▷ 프로그램에서 스캔타임이 조건에 의해 설정된 Watch-Dog Time 시간을 넘는 경우 사용합니다.

▷ 스캔 타임이 매번 설정된 스캔 Watch Dog Time을 넘는 경우는 GMWIN의 기본파라미터에서 스캔 워치독 타임의 설정값으로 변경해 주십시오.

▷ 프로그램의 0 Line부터 WDT-RST 펑션까지의 시간 T1과 WDT-RST 펑션부터 프로그램 끝까지의 시간T2 어느쪽도 스캔워치독 타임 설정값을 넘지 않도록 해 주십시오.

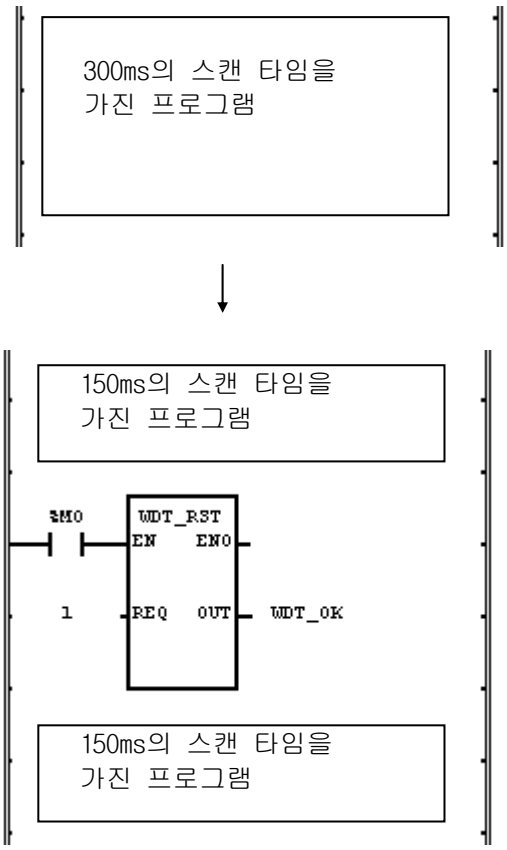


- WDT-RST 펑션은 1스캔중에 여러번 사용 가능합니다.

■ 프로그램 예

스캔워치독 타임이200ms로 설정된 프로그램에서 실행조건에 따라 프로그램의 두행 시간이 300ms가 될 때의 프로그램

LD



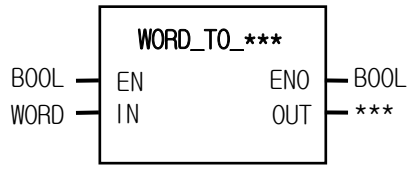
(1)실행조건(%M0)가 On하면 WDT-RST(Watch Dog 타이머 초기화) 펑션이 실행합니다.

(2)WDT-RST 펑션이 실행되면, 프로그램의 실행조건에 따라 스캔타임이 300ms까지 늘어나는 프로그램을 스캔워치독 타임 이내(200ms)으로 맞출 수 있습니다.

WORD_TO_***

WORD 타입 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트열(16비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

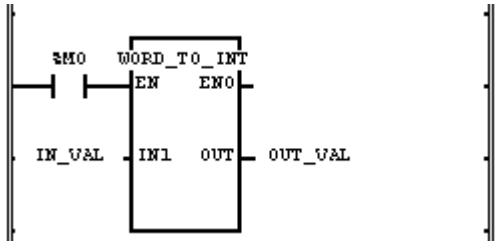
■ 기능

IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
WORD _TO_SINT	SINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_INT	INT	내부 비트 배열의 변환없이 INT 타입으로 변환합니다.
WORD _TO_DINT	DINT	상위 비트를 0으로 채운 DINT 타입으로 변환합니다.
WORD _TO_LINT	LINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
WORD _TO_USINT	USINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_UINT	UINT	내부 비트 배열의 변환없이 INT 타입으로 변환합니다.
WORD _TO_UDINT	UDINT	상위 비트를 0으로 채운 DINT 타입으로 변환합니다.
WORD _TO_ULINT	ULINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
WORD _TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
WORD _TO_BYTE	BYTE	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_DWORD	DWORD	상위 비트를 0으로 채운 DWORD 타입으로 변환합니다.
WORD _TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
WORD _TO_DATE	DATE	내부 비트 배열의 변환없이 DATE 타입으로 변환합니다.
WORD _TO_STRING	STRING	WORD를 STRING 타입으로 변환합니다.

■ 프로그램 예

LD



(1)입력조건(%M0)이 On하면 WORD-TO-INT 펑션이 실행됩니다.

(2)입력변수로 선언된 IN_VAL(WORD 타입) = 2#0001_0001_0001_0001이면, 출력변수로 선언된 OUT_VAL(INT 타입) = 4096 + 256 + 16 + 1 = 4,369가 됩니다.

입력(IN1) : IN_VAL(WORD) = 16#1111

0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

↓ (WORD-TO-INT)

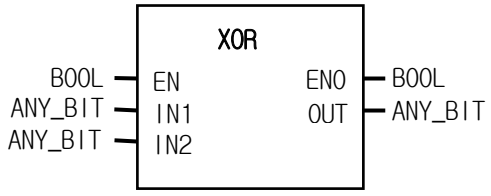
출력(OUT) : OUT_VAL(INT) = 4,369(16#1111)

0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

XOR

배타적 논리합

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : XOR될 값 IN2 : XOR될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : XOR 된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

■ 기능

IN1을 IN2와 비트별로 XOR해서 OUT으로 출력시킵니다.

IN1 1111 0000

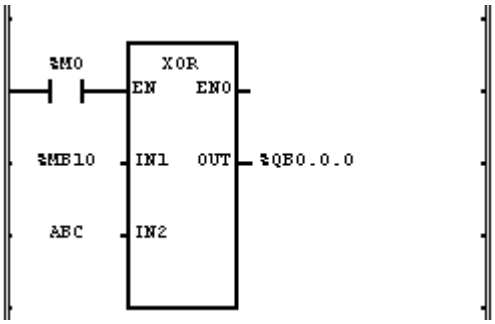
XOR

IN2 1010 1010

OUT 0101 1010

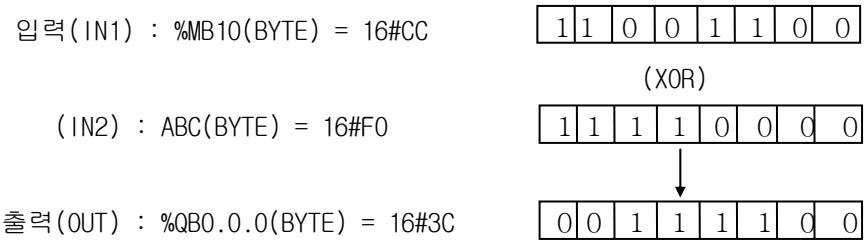
■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 XOR(배타적 논리합) 펑션이 실행됩니다.

(2)입력변수 %MB10=11001100, ABC=11110000이면,두 값을 XOR시킨 결과가 출력변수 %QB0.0.0 = 00111100 로 출력됩니다.

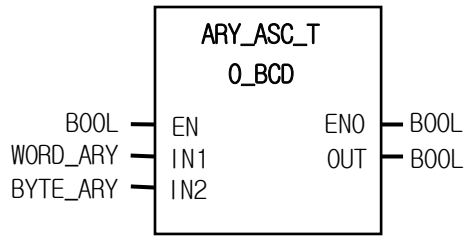

입력(IN1) : %MB10(BYTE) = 16#CC
1 1 0 0 1 1 0 0
(XOR)
(IN2) : ABC(BYTE) = 16#F0
1 1 1 1 0 0 0 0
출력(OUT) : %QB0.0.0(BYTE) = 16#3C
0 0 1 1 1 1 0 0

2.2 응용 펑션 라이브러리

ARY_ASC_TO_BCD

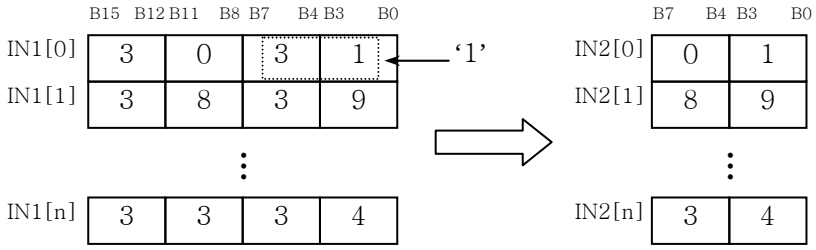
입력 ASCII Array 출력 BCD Array

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : ASCII Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : BCD Array 출력

■ 기능

ASCII 데이터를 지니고 있는 WORD Array를 입력 받아서, BCD(Binary Coded Decimal)값인 BYTE Array로 변환합니다



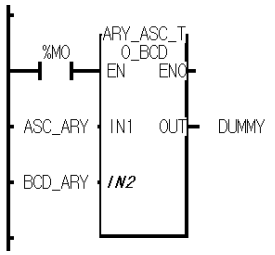
■ 에러

▷ 입력단의 2 개 Array 의 개수가 서로 다를 경우, IN2 값에는 변화가 없으며 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ IN1 Array 원소의 값이 16진 값으로 '0'~'9'이외의 값이 입력되면, IN1에 해당하는 IN2 Array 원소의 값은 16#00이 되고 나머지 원소들의 값은 정상적으로 변환이 이루어 지지만 _ERR/_LER 플래그가 셋(Set) 됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면, ARY_ASC_TO_BCD 펑션이 실행됩니다.

(2)펑션의 입력 변수로 선언된 ASC_ARY값이 다음과 같이 선언되어 있다면

ASC_ARY[0]	3031H
ASC_ARY[1]	3839H
ASC_ARY[2]	3334H

펑션이 실행된 후에 입출력 변수로 선언된 BCD_ARY의 값은

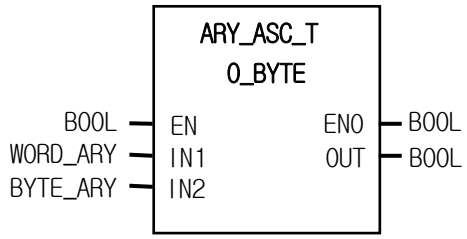
BYTE_ARY[0]	01H
BYTE_ARY[1]	89H
BYTE_ARY[2]	34H

과 같이 출력됩니다.

ARY_ASC_TO_BYTE

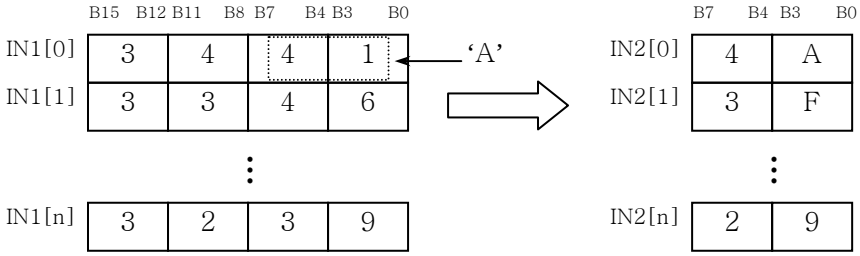
입력 ASCII Array 출력 BYTE Array

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : ASCII Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : BYTE Array 출력

■ 기능

ASCII 데이터를 지니고 있는 WORD Array를 입력 받아서, 16진(HEX) 값인 BYTE Array로 변환합니다.

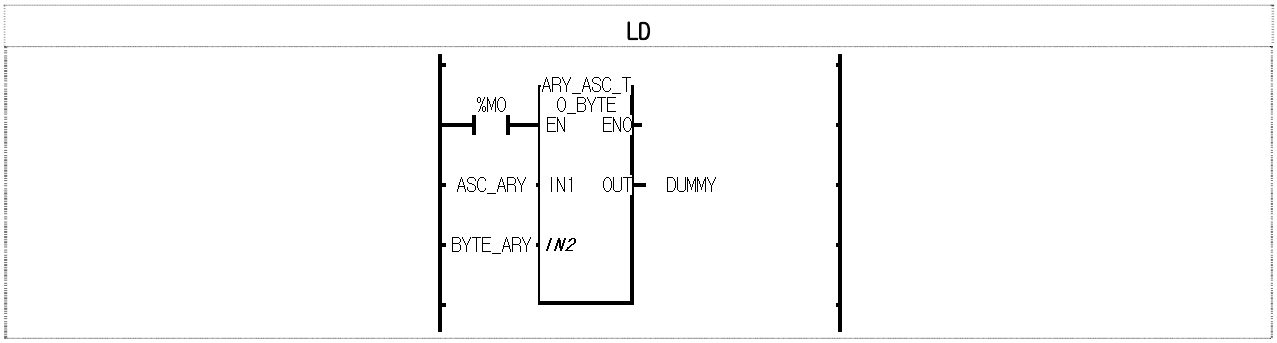


■ 에러

▷ 입력 단의 2 개 Array 의 개수가 서로 다를 경우, IN2 값에는 변화가 없으며 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ IN1 Array 원소의 값이 16진 값으로 '0'~'F' 이외의 값이 입력되면, IN1에 해당하는 IN2 Array 원소의 값은 00이 되고 나머지 원소들의 값은 정상적으로 변환이 이루어 지지만 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예



(1)실행조건(%M0)이 On하면, ARY_ASC_TO_BYTE 펑션이 실행됩니다.

(2)펑션의 입력 변수로 선언된 ASC_ARY값이 다음과 같이 선언되어 있다면

ASC_ARY[0]	3441H
ASC_ARY[1]	3346H
ASC_ARY[2]	3239H

펑션이 실행된 후에 입출력 변수로 선언된 BYTE_ARY의 값은

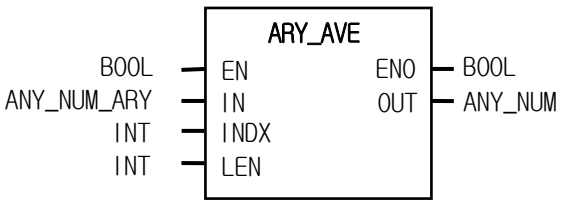
BYTE_ARY[0]	4AH
BYTE_ARY[1]	3FH
BYTE_ARY[2]	29H

과 같이 출력됩니다.

ARY_AVE_***

Array의 평균값 구하기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 평균을 위한 데이터 Array INDX : Array 내에서 연산을 시작할 위치 LEN : 평균값을 구할 Array 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산후의 평균값을 출력

■ 기능

▷ ARY_AVE_*** 펑션은 Array 내부의 지정된 영역에 대하여 평균값을 구합니다.

▷ 출력 타입은 입력 Array 타입과 동일하게 설정되어 있습니다.

▷ LEN이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 – |LEN|) 사이의 원소들에 대한 평균값을 구합니다.

▷ 평균값은 반올림하여 출력합니다.

펑션	출력 타입	동작 설명
ARY_AVE_SINT	SINT	SINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_INT	INT	SINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_DINT	DINT	DINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_LINT	LINT	LINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_USINT	USINT	USINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_UINT	UINT	UINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_UDINT	UDINT	UDINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_ULINT	ULINT	ULINT값들의 평균값을 구합니다.(소수점 이하는 반올림)
ARY_AVE_REAL	REAL	REAL값들의 평균값을 구합니다.
ARY_AVE_LREAL	LREAL	LREAL값들의 평균값을 구합니다.

■ 에러

▷ Array의 범위를 초과하여 지정한 경우 _ERR/ _LER 플래그가 셋(Set)됩니다.

▷ 에러발생시 OUT에는 0이 출력됩니다.

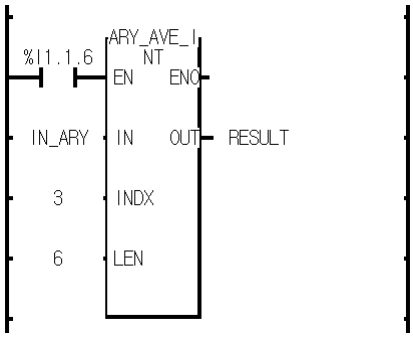
※에러가 발생하는 범위 지정

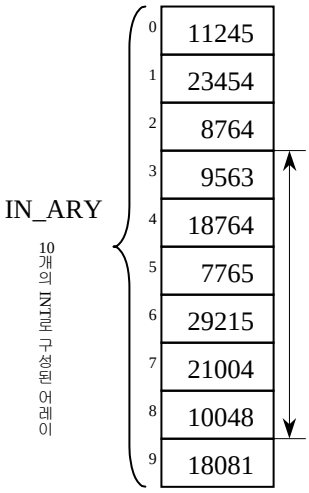
INDX < 0 또는 INDX > IN의 최대 원소번호

INDX + LEN > IN의 최대 원소번호

■ 프로그램 예

LD





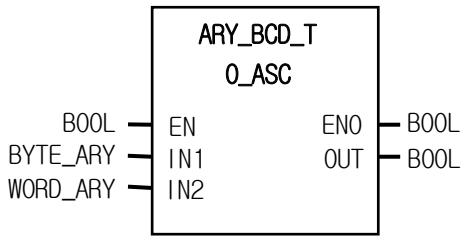
$$\frac{9563+18764+7765+29215+21004+10048}{6}=16044.83=16045$$

(1)입력조건(%I1.1.6)이 On하면, ARY_AVE_INT 펑션이 실행됩니다.
(2)Array 내의 값이 위의 그림과 같을 경우 Array 인덱스 3번째로부터 6개의 원소에 대한 평균값을 구합니다.
(3)평균값이16044.8이지만 출력 타입이 INT이므로 반올림을 수행하여 16045를 출력합니다.

ARY_BCD_TO_ASC

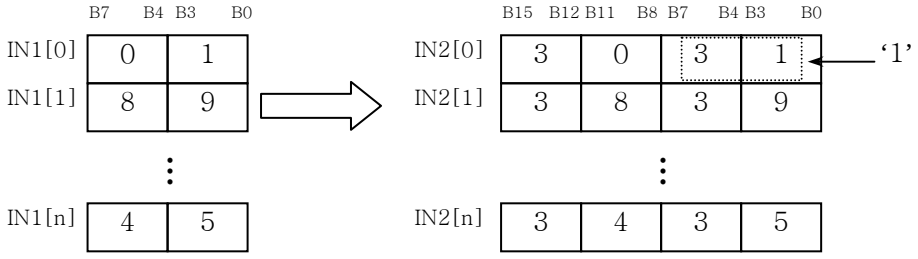
입력 BCD Array
출력 ASCII Array

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : BCD Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : ASCII Array 출력

■ 기능

BCD값을 지닌 BYTE Array를 입력 받아서, ASCII 데이터를 지니고 있는 WORD Array로 변환합니다.



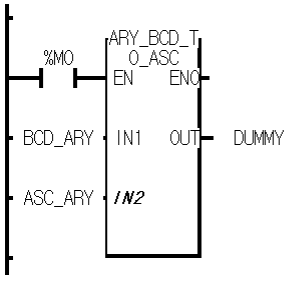
■ 에러

▷ 입력 단의 2 개 Array 의 개수가 서로 다를 경우, IN2 값에는 변화가 없으며 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ IN1Array 원소의 값이 16 진 값으로 0~9 이외의 값이 입력되면, IN1 에 해당하는 IN2 Array 원소의 값은 16#3030("00")이 되고 나머지 원소들의 값은 정상적으로 변환이 되지만 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면, ARY_BCD_TO_ASC 펑션이 실행 됩니다.

(2)펑션의 입력 변수로 선언된 BCD_ARY의 값이 다음과 같이 선언되어 있다면

BYTE_ARY[0]	01H
BYTE_ARY[1]	89H
BYTE_ARY[2]	45H

펑션이 실행된 후에 입출력 변수로 선언된 ASC_ARY의 값은

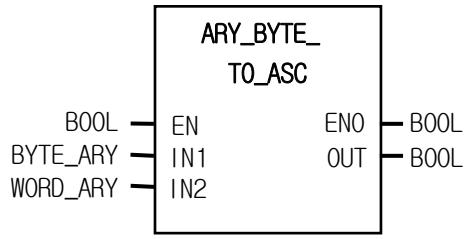
ASC_ARY[0]	3031H
ASC_ARY[1]	3839H
ASC_ARY[2]	3435H

과 같이 출력됩니다.

ARY_BYTE_TO_ASC

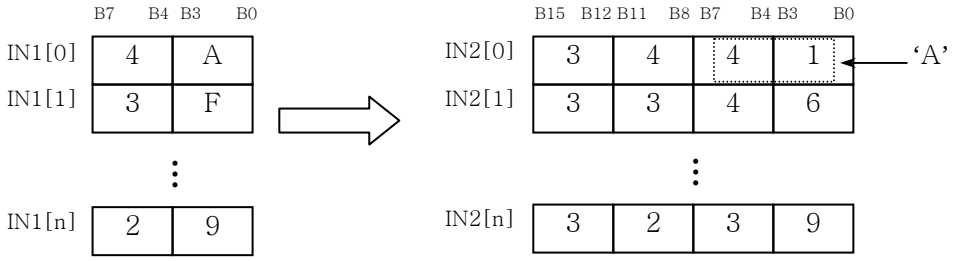
입력 BYTE Array
출력 ASCII Array

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : BYTE Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : ASCII Array 출력

■ 기능

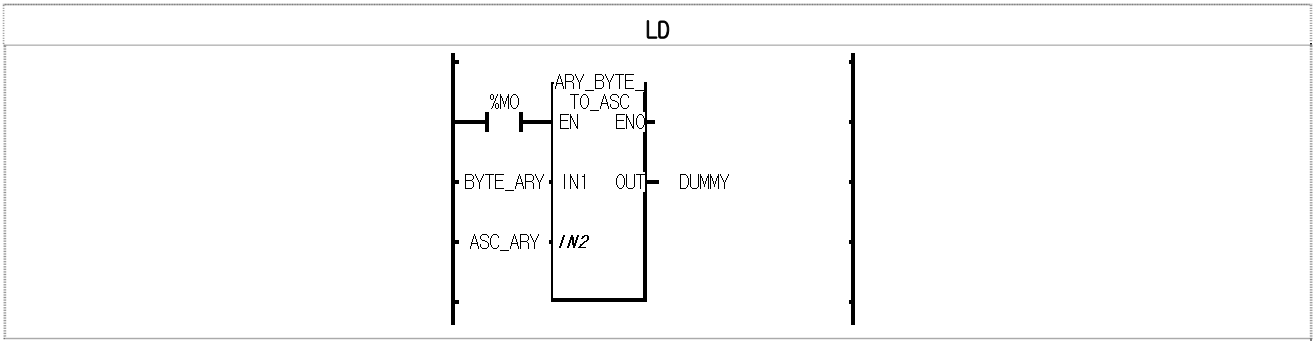
16진(HEX)값을 가지고 있는 BYTE Array를 입력 받아서, ASCII 데이터를 지니고 있는 WORD Array로 변환합니다.



■ 에러

입력단의 2 개 Array 의 개수가 서로 다를 경우, IN2 값에는 변화가 없으며 _ERR/_LER 플래그가 (Set)됩니다.

■ 프로그램 예



(1)실행조건(%M0)이 On하면, ARY_BYTE_TO_ASC 펑션이 실행됩니다.
(2)펑션의 입력 변수로 선언된 BYTE_ARY의 값이 다음과 같이 선언되어 있다면

BYTE_ARY[0]	4AH
BYTE_ARY[1]	3FH
BYTE_ARY[2]	29H

펑션이 실행된 후에 입출력 변수로 선언된 ASC_ARY의 값은

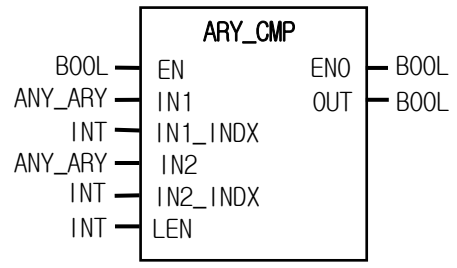
ASC_ARY[0]	3441H
ASC_ARY[1]	3346H
ASC_ARY[2]	3239H

과 같이 출력됩니다.

ARY_CMP_***

Array 비교

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : 비교할 첫번째 Array IN1_INDX : 첫번째 Array에서 비교할 시작 위치 IN2 : 비교할 두번째 Array IN2_INDX : 두번째 Array에서 비교할 시작 위치 LEN : 비교할 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Array 비교값이 동일하면 1을 출력

■기능

▷ ARY_CMP_*** 펑션은 2개의 Array를 입력 받아 서로 동일한 값을 가지고 있는지를 비교합니다.

▷ LEN이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 – |LEN|) 사이의 원소들을 비교합니다.

펑션	입력 Array 타입	동작 설명
ARY_CMP_BOOL	BOOL	2개의 BOOL Array를 서로 비교합니다.
ARY_CMP_BYTE	BYTE	2개의 BYTE Array를 서로 비교합니다.
ARY_CMP_WORD	WORD	2개의 WORD Array를 서로 비교합니다.
ARY_CMP_DWORD	DWORD	2개의 DWORD Array를 서로 비교합니다.
ARY_CMP_LWORD	LWORD	2개의 LWORD Array를 서로 비교합니다.
ARY_CMP_SINT	SINT	2개의 SINT Array를 서로 비교합니다.
ARY_CMP_INT	INT	2개의 INT Array를 서로 비교합니다.
ARY_CMP_DINT	DINT	2개의 DINT Array를 서로 비교합니다.
ARY_CMP_LINT	LINT	2개의 LINT Array를 서로 비교합니다.
ARY_CMP_USINT	USINT	2개의 USINT Array를 서로 비교합니다.
ARY_CMP_UINT	UINT	2개의 UINT Array를 서로 비교합니다.
ARY_CMP_UDINT	UDINT	2개의 UDINT Array를 서로 비교합니다.
ARY_CMP_ULINT	ULINT	2개의 ULINT Array를 서로 비교합니다.
ARY_CMP_REAL	REAL	2개의 REAL Array를 서로 비교합니다.
ARY_CMP_LREAL	LREAL	2개의 LREAL Array를 서로 비교합니다.
ARY_CMP_TIME	TIME	2개의 TIME Array를 서로 비교합니다.
ARY_CMP_DATE	DATE	2개의 DATE Array를 서로 비교합니다.
ARY_CMP_TOD	TOD	2개의 TOD Array를 서로 비교합니다.
ARY_CMP_DT	DT	2개의 DT Array를 서로 비교합니다.

■ 에러

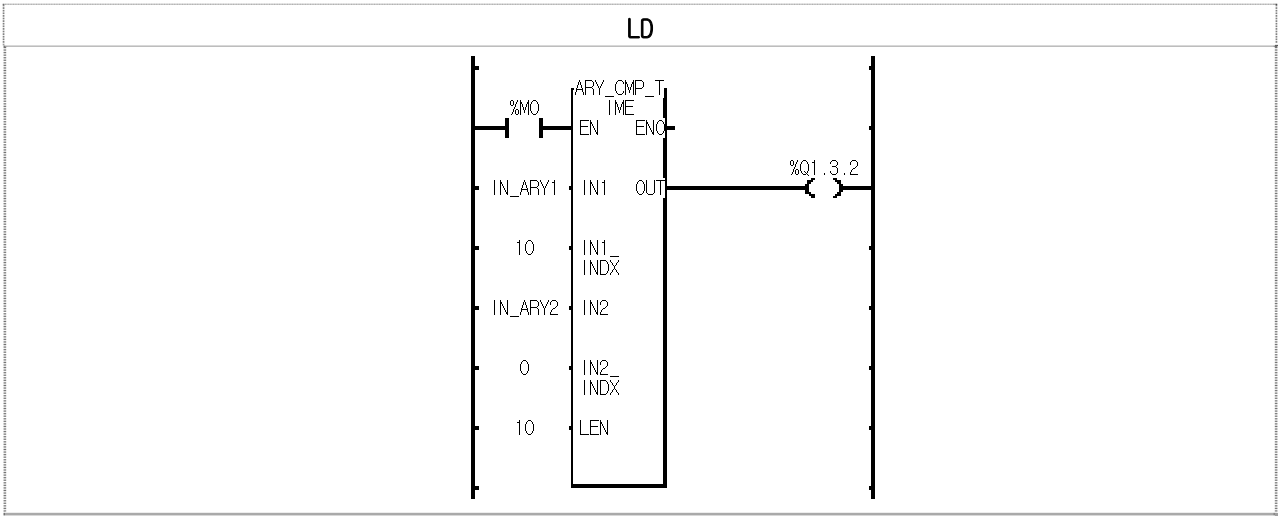
▷ Array의 범위를 초과하여 지정한 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

※ 에러가 발생하는 범위 지정

IN1_INDX < 0 또는 IN1_INDX > IN1 최대 원소 번호
IN2_INDX < 0 또는 IN2_INDX > IN2 최대 원소 번호
IN1_INDX + LEN ≥ IN1 최대 원소 번호
IN2_INDX + LEN ≥ IN2 최대 원소 번호

■ 프로그램 예

LD

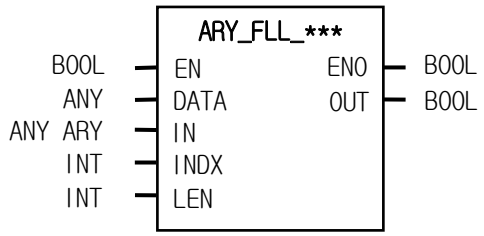


(1)입력조건(%M0)이 On하면 ARY_CMP_TIME 펑션이 실행됩니다.
(2)IN_ARY1이 100개의 원소를 지닌 TIME Array이고, IN_ARY2가 10개의 원소를 지닌TIME Array일 경우 IN_ARY1의 11번째 원소로부터 20번째 까지 10개의 원소를 IN_ARY 2의 첫번째 원소로부터 10번째까지 10개의 원소와 각각 비교하여 모두 동일하면 접점 출력 %Q1.3.2가 On됩니다.

ARY_FLL_***

Array 내부 영역 채우기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 DATA : Array 내를 채울 값 INDX : 값을 채울 Array 내의 처음 위치 LEN : 값을 채울 Array 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 동작이 성공적으로 끝나면 1을 출력 **입출력** IN : 값이 채워질 Array

■ 기능

▷ ARY_FLL_*** 펑션은 입력 DATA 값으로 Array 내부의 선택 영역을 채웁니다.

▷ LEN이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 – |LEN|) 사이의 원소들을 비교합니다.

펑션	입출력 Array 타입	동작 설명
ARY_FLL_BOOL	BOOL	BOOL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_BYTE	BYTE	BYTE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_WORD	WORD	WORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_DWORD	DWORD	DWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_LWORD	LWORD	LWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_SINT	SINT	SINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_INT	INT	INT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_DINT	DINT	DINT Array 내부를 지정된 값으로 채웁니다.
, ARY_FLL_LINT	LINT	LINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_USINT	USINT	USINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_UINT	UINT	UINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_UDINT	UDINT	UDINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_ULINT	ULINT	ULINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_REAL	REAL	REAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_LREAL	LREAL	LREAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_TIME	TIME	TIME Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_DATE	DATE	DATE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_TOD	TOD	TOD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL_DT	DT	DT Array 내부를 지정된 값으로 채웁니다.

■ 에러

▷ Array의 범위를 초과하여 지정한 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

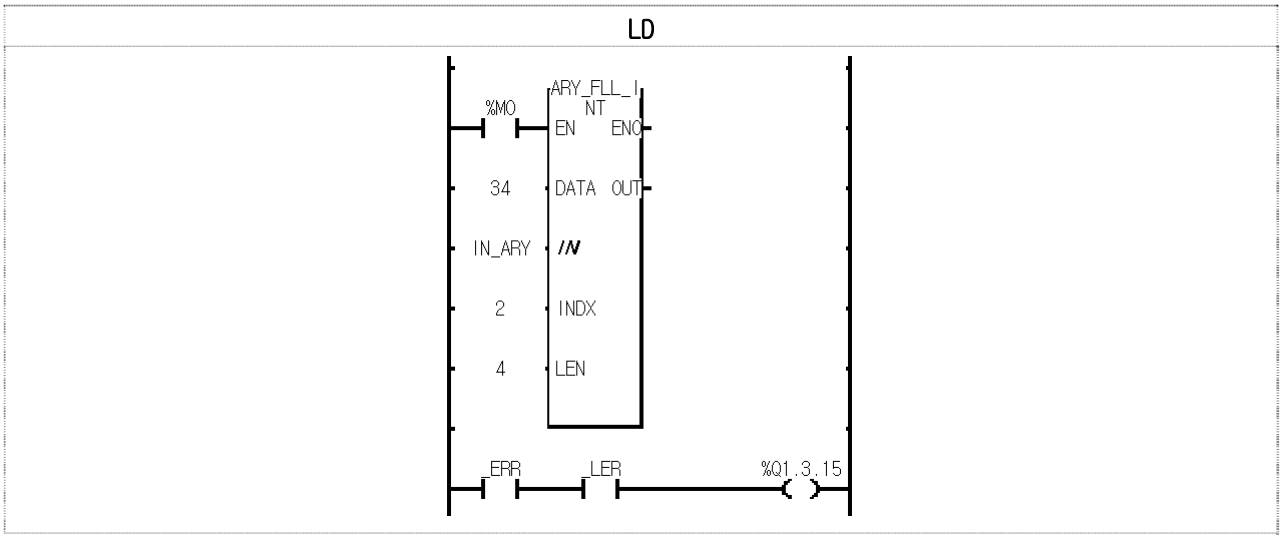
▷ 에러 발생시 OUT은 OFF되고, IN의 Array 값은 변하지 않습니다.

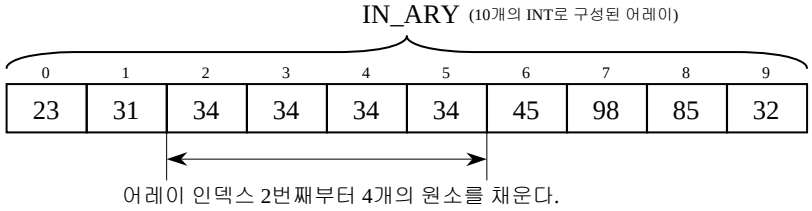
※ 에러가 발생하는 범위 지정

INDX < 0 또는 INDX > IN의 최대 원소번호

INDX + LEN ≥ IN 의 최대 원소번호

■ 프로그램 예





(1)입력조건(%M0)이 On하면, ARY_FLL_INT 펑션이 실행됩니다.

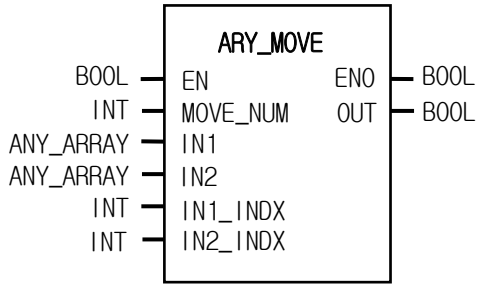
(2)Array 인덱스 2번째부터 4개의 원소를 지정된 값 34로 채웁니다.

(3)만약 LEN을 9로 대체하면 Array의 전체 개수를 초과하므로 에러가 발생하여 _ERR과 _LER 플래그가 On되므로 출력 접점 %Q1.13.15가 On됩니다.

ARY_MOVE

Array 복사

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 MOVE_NUM : Move할 어레이 개수 IN1 : Move할 어레이 변수(STRING Type은 사용 불가) IN2 : Move될 어레이 변수(STRING Type은 사용 불가) IN1_INDX : 어레이변수의 Move할 시작 Pointer 위치 IN2_INDX : 어레이변수의 Move 될 시작 Pointer 위치 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 에러없이 실행되면 1을 출력

■ 기능

▷ENO이 1이면, IN1 어레이 변수의 데이터를 IN2 어레이 변수에 복사합니다.

▷IN1의 IN1_INDX번째 값부터 MOVE_NUM 개수 만큼 데이터를 복사하여, IN2의 IN2_INDX번째 값부터 붙여넣기를 실행합니다.

▷MOVE가 가능하기위해서는 IN1과 IN2의 데이터 타입과 Size가 동일해야 합니다. 단, IN1과 IN2의 어레이 개수는 다를 수 있습니다.

▷데이터 타입의 Size는 아래표와 같습니다.

데이터 Size	변수 타입
1 Bit	BOOL
8 Bit	BYTE, SINT, USINT
16 Bit	WORD / INT / UINT / DATE
32 Bit	DWORD / DINT / UDINT / TIME / TOD
64 Bit	DT

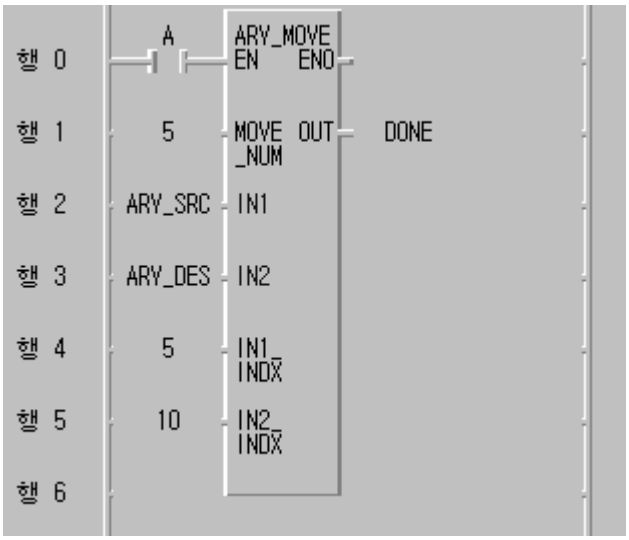
■ 에러

▷IN1과 IN2데이터 타입의 Size가 서로 다른 경우 에러가 발생합니다.

▷IN1의 Array 개수가 (IN1_INDX + MOVE_NUM) 보다 작은 경우나, IN2의 Array 개수가 (IN2_INDX + MOVE_NUM) 보다 작은 경우 에러가 발생합니다. 이때, 데이터 복사는 수행하지 않고 OUT은 0이 됩니다. 또한, ENO가 OFF되고 _ERR,_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



변수명	변수 타입	어레이 개수
ARY_SRC	INT	10
ARY_DES	WORD	15

(1)실행조건(A)이 On하면 ARY_MOVE 펑션이 실행됩니다.

(2)아래표와 같이 ARY_SRC[5]부터 5개의 데이터를 ARY_DES[10]에 복수합니다.

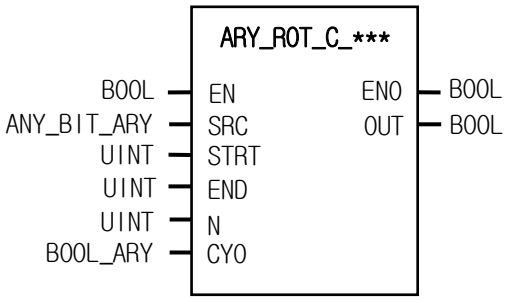
이 때, ARY_DES의 데이터 타입이 WORD이므로 16진수로 저장됩니다.

실행전				실행후			
ARY_SRC[0]	0	ARY_DES[0]	16#0	ARY_SRC[0]	0	ARY_DES[0]	16#0
ARY_SRC[1]	11	ARY_DES[1]	16#1	ARY_SRC[1]	11	ARY_DES[1]	16#1
ARY_SRC[2]	22	ARY_DES[2]	16#2	ARY_SRC[2]	22	ARY_DES[2]	16#2
ARY_SRC[3]	33	ARY_DES[3]	16#3	ARY_SRC[3]	33	ARY_DES[3]	16#3
ARY_SRC[4]	44	ARY_DES[4]	16#4	ARY_SRC[4]	44	ARY_DES[4]	16#4
ARY_SRC[5]	55	ARY_DES[5]	16#5	ARY_SRC[5]	55	ARY_DES[5]	16#5
ARY_SRC[6]	66	ARY_DES[6]	16#6	ARY_SRC[6]	66	ARY_DES[6]	16#6
ARY_SRC[7]	77	ARY_DES[7]	16#7	ARY_SRC[7]	77	ARY_DES[7]	16#7
ARY_SRC[8]	88	ARY_DES[8]	16#8	ARY_SRC[8]	88	ARY_DES[8]	16#8
ARY_SRC[9]	99	ARY_DES[9]	16#9	ARY_SRC[9]	99	ARY_DES[9]	16#9
		ARY_DES[10]	16#A			ARY_DES[10]	16#37
		ARY_DES[11]	16#B			ARY_DES[11]	16#42
		ARY_DES[12]	16#C			ARY_DES[12]	16#4D
		ARY_DES[13]	16#D			ARY_DES[13]	16#58
		ARY_DES[14]	16#E			ARY_DES[14]	16#63

ARY_ROT_C_***

Array의 Bit Rotate with Carry

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

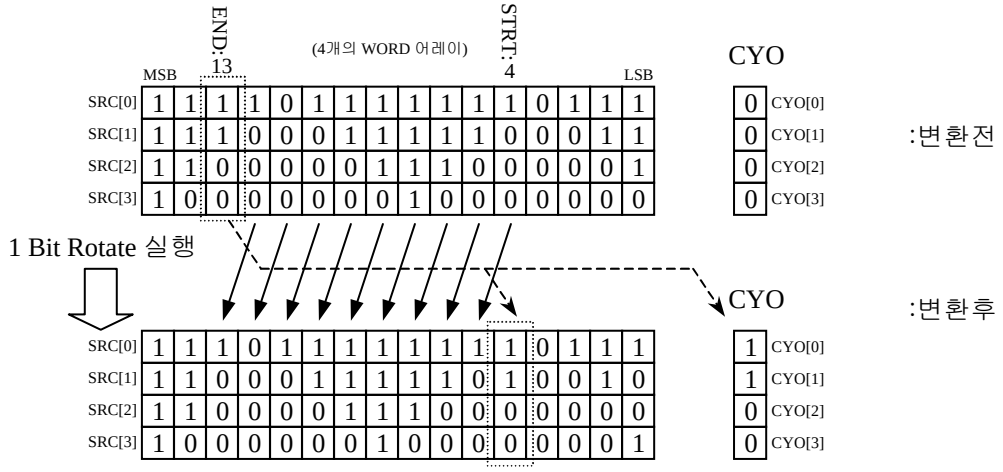
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 STRT : Rotate할 bit의 시작위치 END : Rotate할 bit의 종료위치 N : Rotate 시킬 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산 에러가 없으면 1을 출력 **입출력** SRC : Rotate 시킬 Source Array CYO : Rotate후에 출력되는 Carry bit Array

■ 기능

▷ ARY_ROT_C_*** 펑션은 지정된 범위의 Array 원소들의 bit들을 정해진 개수만큼 Rotate시킵니다.

▷ 동작의 지정

- 범위 지정 : STRT와 END로 이동할 비트의 범위를 지정합니다.
- 이동 방향 및 횟수 : STRT에서 END방향으로 지정된 횟수(N)만큼 Rotate 합니다
- 출력 : 데이터 조작의 결과는 SRC에 지정된 ANY_BIT_ARY에 저장되며, 회전동작으로 END 위치에서 STRT로 돌아 들어가는 비트열 데이터는 CYO 비트 Array로 출력됩니다.



펑션	입출력 Array 타입	동작 설명
ARY_ROT_C_BYTE	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트수 만큼 Rotate시킵니다.
ARY_ROT_C_WORD	WORD	
ARY_ROT_C_DWORD	DWORD	
ARY_ROT_C_LWORD	LWORD	

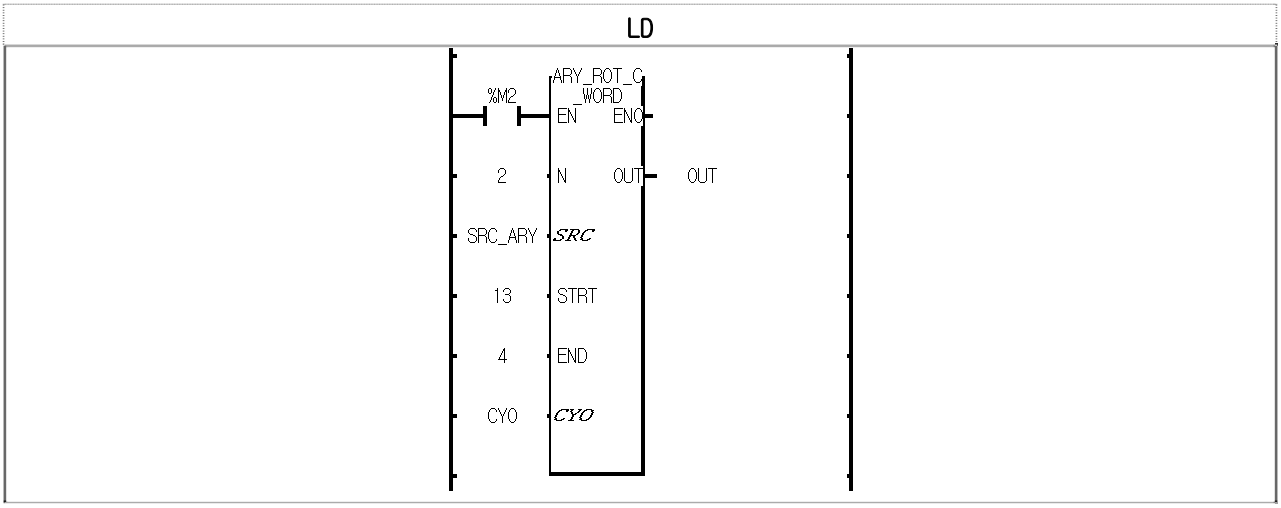
■ 에러

▷ SRC과 CYO Array의 개수가 서로 동일하지 않을 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ STRT와 END가 SRC 원소의 비트 범위를 벗어나면 에러가 발생합니다.

▷ 에러발생시 SRC와 CYO는 변하지 않습니다.

■ 프로그램 예

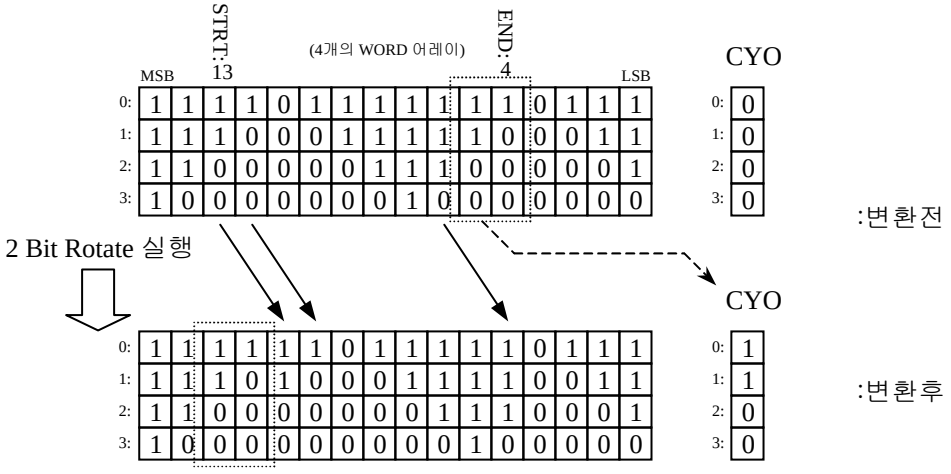


(1)입력조건(%M2)이 On되면, ARY_ROT_C_WORD 펑션이 실행됩니다.

(2)STRT인 13번째 bit열과 END인 4번째 비트열로 지정된 범위에서 STRT로부터 END방향으로 2번 회전합니다.

(3)회전된 결과는 SRC_ARY에 저장되며 이때 출력되는 캐리 비트열은 CYO BOOL Array로 출력됩니다.

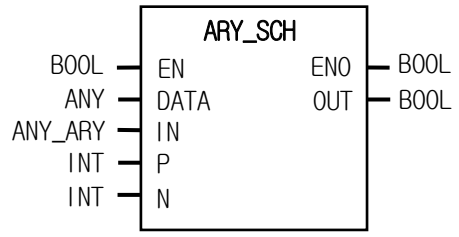
(연산전)
SRC_ARY : 16#F7F7
16#E3E3
16#C1C1
16#8080
(N) : 2
(연산후)
SRC_ARY : 16#FDF7
16#E8F3
16#C071
16#8020
CYO : 2#1100



ARY_SCH_***

Array 탐색(search)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 DATA : 탐색할 DATA IN : 탐색할 Array **출력** ENO : EN값을 그대로 출력 OUT : 원하는 값을 찾으면 1을 출력 **입출력** P : 목표값에 해당하는 Array의 첫번째 위치 N : 목표값과 동일한 Array 원소들의 개수

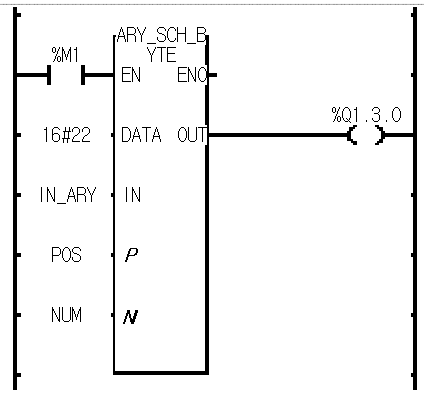
■ 기능

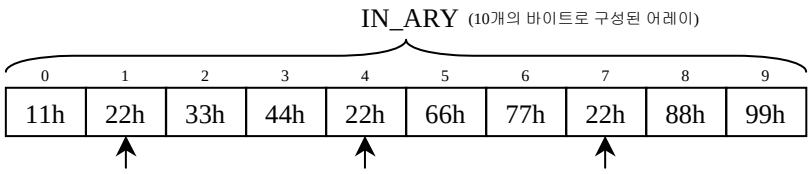
ARY_SCH_*** 펑션은 Array 내에서 입력된 값과 동일한 값을 찾아 Array 내에서의 처음 위치와 전체 개수를 출력합니다. Array 내에서 목표 값과 동일한 값이 하나 이상일 경우 OUT에 1을 출력합니다.

펑션	입력 Array 타입	동작 설명
ARY_SCH_BOOL	BOOL	BOOL Array 내에서 탐색합니다.
ARY_SCH_BYTE	BYTE	BYTE Array 내에서 탐색합니다.
ARY_SCH_WORD	WORD	WORD Array 내에서 탐색합니다.
ARY_SCH_DWORD	DWORD	DWORD Array 내에서 탐색합니다.
ARY_SCH_LWORD	LWORD	LWORD Array 내에서 탐색합니다.
ARY_SCH_SINT	SINT	SINT Array 내에서 탐색합니다.
ARY_SCH_INT	INT	INT Array 내에서 탐색합니다.
ARY_SCH_DINT	DINT	DINT Array 내에서 탐색합니다.
ARY_SCH_LINT	LINT	LINT Array 내에서 탐색합니다.
ARY_SCH_USINT	USINT	USINT Array 내에서 탐색합니다.
ARY_SCH_UINT	UINT	UINT Array 내에서 탐색합니다.
ARY_SCH_UDINT	UDINT	UDINT Array 내에서 탐색합니다.
ARY_SCH_ULINT	ULINT	ULINT Array 내에서 탐색합니다.
ARY_SCH_REAL	REAL	REAL Array 내에서 탐색합니다.
ARY_SCH_LREAL	LREAL	LREAL Array 내에서 탐색합니다.
ARY_SCH_TIME	TIME	TIME Array 내에서 탐색합니다.
ARY_SCH_DATE	DATE	DATE Array 내에서 탐색합니다.
ARY_SCH_TOD	TOD	TOD Array 내에서 탐색합니다.
ARY_SCH_DT	DT	DT Array 내에서 탐색합니다.

■ 프로그램 예

LD



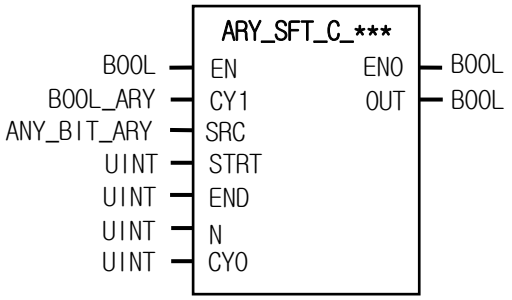


(1)입력조건(%M1)이 On하면 ARY_SCH_BYTE 펑션이 실행됩니다.
(2)위의 그림에서처럼 IN_ARY가 10개의 바이트 Array로 구성되어 있고 이 Array 내부에서 22h이라는 값을 탐색하면 화살표로 표시된 바와 같이 3개가 탐색 됩니다.
(3)POS에는 Array 내부의 첫번째 위치인 1이 출력되고, NUM에는 전체 개수인 3이 출력됩니다. 펑션 출력으로는 하나 이상의 값이 탐색 되었으므로 1이 출력되어 출력접점 %Q1.3.0은 On이 됩니다.

ARY_SFT_C_***

Array의 Bit Shift Left with Carry

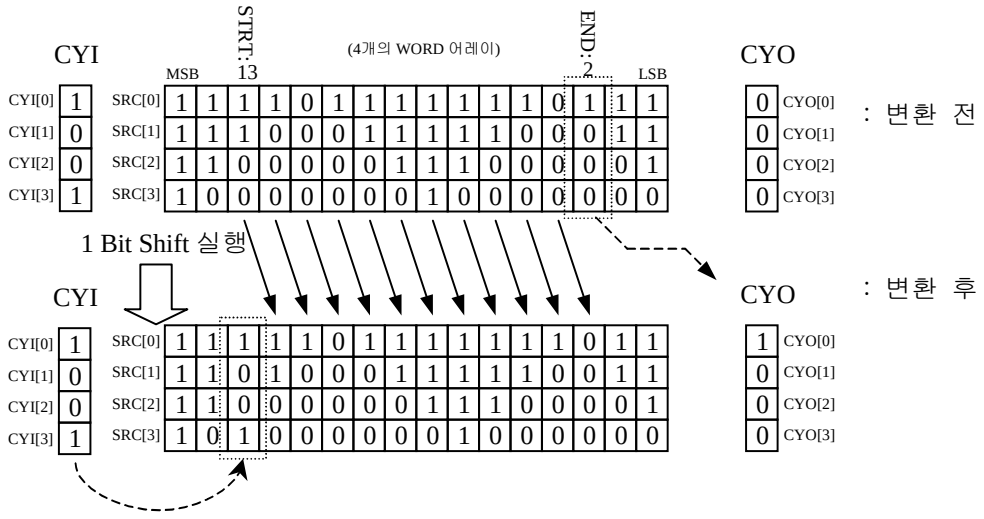
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 CYI : Array에 입력되는 Carry bit Array STRT : Shift할 bit의 시작 END : Shift할 bit의 종료위치 N : Shift 시킬 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산 에러가 없으면 1을 출력 **입출력** SRC : Shift 시킬 Source Array CYO : Shift후에 출력되는 Carry bit Array

■ 기능

▷ ARY_SFT_C_*** 펑션은 Array 원소들의 bit들을 정해진 개수만큼 지정된 방향으로 이동시킵니다.

▷ 동작의 지정
- 범위 지정 : STRT와 END로 이동할 비트의 범위를 지정됩니다.
- 이동 방향 및 횟수 : STRT에서 END방향으로 지정된 횟수(N)만큼 이동합니다
- 입력 데이터 지정 : Shift한 후에 비워지는 위치를 입력 데이터(CYI)로 채웁니다.
- 출력 : 데이터 조작의 결과는 SRC에 지정된 ANY_BIT_ARY에 저장되며, END 위치에서 Shift 동작으로 밀려 나온 비트열 데이터는 CYO로 출력됩니다.



펑션	입출력 Array 타입	동작 설명
ARY_SFT_C_BYTE	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트수 만큼 Shift시킵니다.
ARY_SFT_C_WORD	WORD	
ARY_SFT_C_DWORD	DWORD	
ARY_SFT_C_LWORD	LWORD	

■ 에러

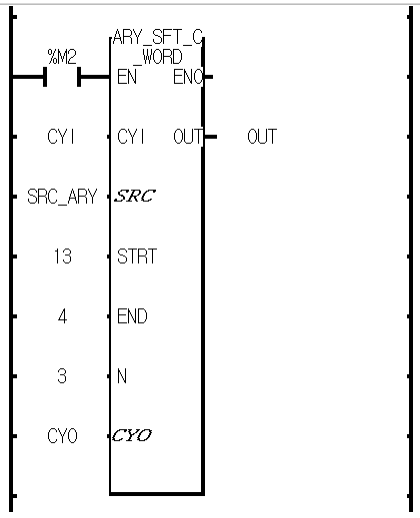
▷ CYI, SRC, CYO Array의 개수가 모두 동일하지 않을 경우 _ERR/_LER 플래그가 셋(Set) 됩니다.

▷ STRT와 END가 SRC 원소의 비트 범위를 벗어나면 에러가 발생합니다.

▷ 에러발생시 SRC와 CYO는 변하지 않습니다.

■ 프로그램 예

LD



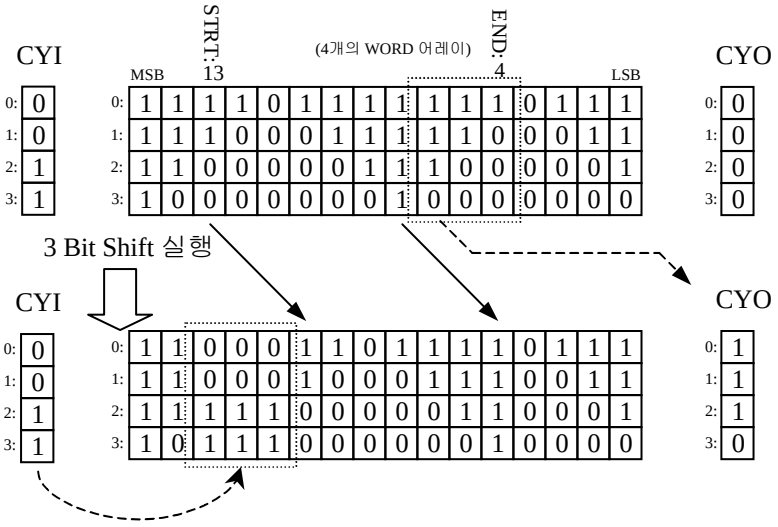
(1)입력조건(%M2)이 On되면, ARY_SFT_C_WORD 펑션이 실행됩니다.

(2)STRT인 13번째 bit열과 END인 4번째 비트열로 지정된 범위에서 STRT로부터 END방향으로 3번 shift 합니다.

(3)Shift후에 비워지는 비트열은 CYI(2#0011)로 채워집니다.

(4)Shift후의 결과는 SRC_ARY로 다시 출력되고 캐리 비트열은 CYO로 출력됩니다.

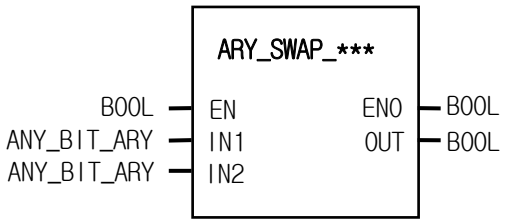
(연산전)
CYI : 2#0011
SRC_ARY : 16#F7F7
16#E3E3
16#C1C1
16#8080
(N) : 3
(연산후)
SRC_ARY : 16#C6F7
16#C473
16#F831
16#B810
CYO : 2#1110



ARY_SWAP_***

Array 원소 데이터의 상위 하위 바꾸기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : Swap된 Array 출력

■ 기능

입력된 Array 원소를 2개의 크기로 구분하여 상위와 하위를 서로 교환 합니다.

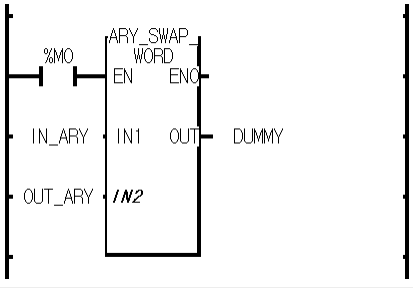
펑션	입력 타입	동작 설명
ARY_SWAP_BYTE	BYTE	BYTE 원소의 상 · 하위 니블(Nibble)을 서로 교환하여 출력합니다.
ARY_SWAP_WORD	WORD	WORD 원소의 상 · 하위 BYTE 를 서로 교환하여 출력합니다.
ARY_SWAP_DWORD	DWORD	DWORD 원소의 상 · 하위 WORD 를 서로 교환하여 출력합니다.
ARY_SWAP_LWORD	LWORD	LWORD 원소의 상 · 하위 DWORD 를 서로 교환하여 출력합니다.

■ 에러

입력단의 2 개 Array 의 개수가 서로 다를 경우, IN2 Array 의 원소 값에는 변화가 없으며 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면, ARY_SWAP_WORD 펑션이 실행됩니다.
(2)펑션의 입력 변수로 선언된 IN_ARY의 값이 다음과 같을 경우

IN_ARY[0]	12ABH
IN_ARY[1]	23BCH
IN_ARY[2]	34CDH

펑션이 실행된 후에 입출력 변수로 선언된 OUT_ARY의 값은

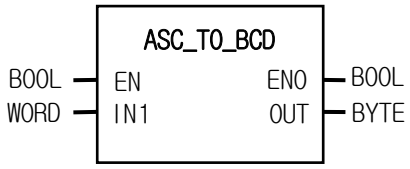
OUT_ARY[0]	AB12H
OUT_ARY[1]	BC23H
OUT_ARY[2]	CD34H

과 같이 출력됩니다.

ASC_TO_BCD

ASCII를 BCD로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : ASCII입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BCD출력

■ 기능

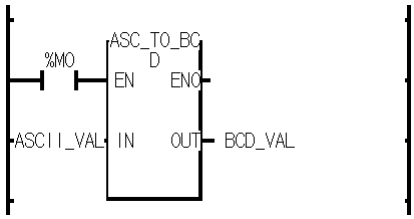
2개의 ASCII값을 받아서 2자리의 BCD(Binary Coded Decimal)로 출력 합니다.

■ 에러

IN 값을 통해 16 진 값으로 '0'~'9'이외의 값이 입력되면, 출력은 16#00 이 되고 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



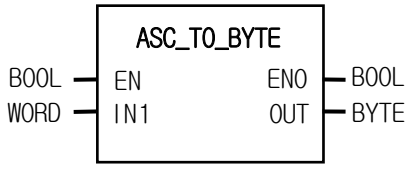
(1)실행조건(%M0)이 On되면, ASC_TO_BCD 펑션이 실행됩니다.

(2)펑션의 입력변수로 선언된 ASCII_VAL(WORD 타입)=16#3732="72"일 경우, 펑션의 출력 변수로 선언된 BCD_VAL(BYTE 타입)=16#72 가 출력됩니다.

ASC_TO_BYTE

ASCII를 BYTE로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : ASCII입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BYTE 출력

■ 기능

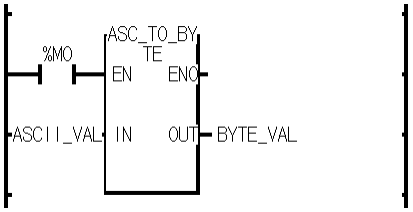
2개의 ASCII 데이터를 입력을 받아서 2자리의 16진(HEX) 값으로 출력합니다.

■ 에러

IN값을 통해 16진 값으로 '0'~'F'이외의 값이 입력되면, 출력은 00이 되고 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



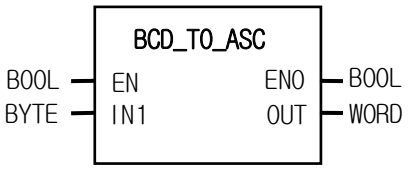
(1)실행조건(%M0)이 On하면, ASC_TO_BYTE 펑션이 실행됩니다.

(2)펑션의 입력변수로 선언된 ASCII_VAL(WORD 타입)=16#4339일 경우, 펑션의 출력 변수로 선언된 BYTE_VAL(BYTE 타입) = 16#C9가 출력됩니다.

BCD_TO_ASC

BCD를 ASCII로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : BCD 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : ASCII 출력

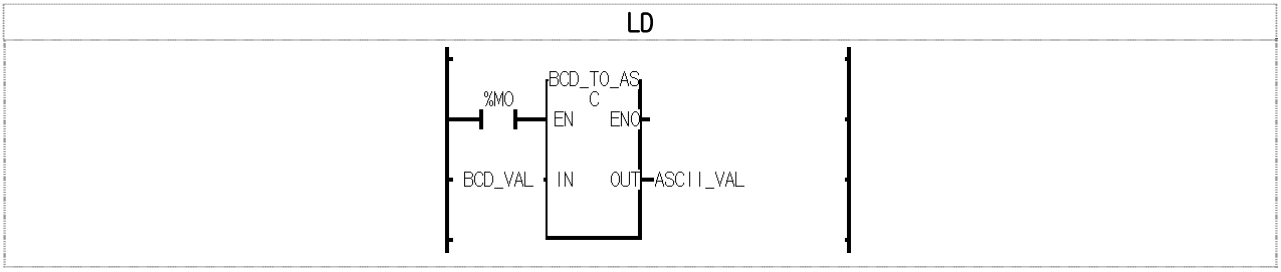
■ 기능

2자리의 BCD(Binary Coded Decimal) 값을 받아서 2개의 아스키 값으로 출력 합니다.

■ 에러

IN 값을 통해 16 진값으로 0~9 이외의 값이 입력되면, 출력은 16#3030("00")이 되고 _ERR/_LER 플래그가 셋(Set) 됩니다.

■ 프로그램 예



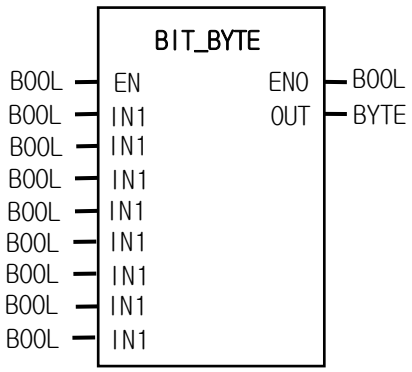
(1)실행조건(%M0)이 On 하면, BCD_TO_ASC 펑션이 실행됩니다.

(2)펑션의 입력변수로 선언된 BCD_VAL(BYTE 타입)=16#85 일 경우, 펑션의 출력 변수로 선언된 ASCII_VAL(WORD 타입)=16#3835= "85"가 출력됩니다.

BIT_BYTE

8개 BIT들을 BYTE로 묶음

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 ~ IN8 : Bit입력 **출력** ENO : EN값을 그대로 출력 OUT : Byte 출력

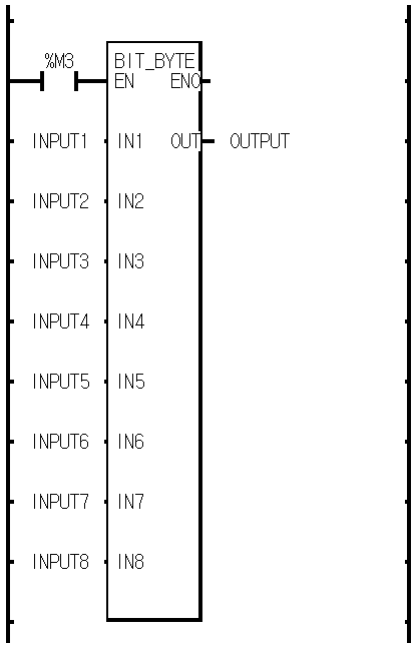
■ 기능

8개의 비트를 하나의 바이트로 조합합니다.

IN8: MSB(최상위 비트), IN1: LSB(최하위 비트)

■ 프로그램 예

LD



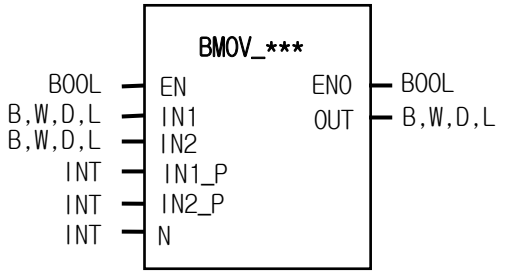
(1)실행조건(%M3)이 On 되면, BIT_BYTE 펑션이 실행됩니다.

(2)8 개의 입력이 각각 INPUT1 부터 8 까지 순서대로 {0,1,1,0,1,1,0,0}이면, 출력변수로 선언된 OUTPUT(BYTE 타입)= 2#00110110 입니다.

BMOV_***

비트 스트링의 일부분을 복사,이동

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1: 조합할 비트 데이터를 가진 스트링 데이터 IN2: 조합할 비트 데이터를 가진 스트링 데이터 IN1_P : IN1지정 데이터상의 시작 비트 위치 IN2_P : IN2지정 데이터상의 시작 비트 위치 N: 조합할 비트의 수 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 조합된 비트 스트링 데이터 출력

■ 기능

▷EN이 1이 되면 IN1의 비트 스트링중 IN1_P로 지정된 비트 위치부터 큰 방향으로 N개의 비트를 취하여, IN2의 비트 스트링에서 IN2_P로 지정된 비트 위치부터 큰 방향으로 대치한 후 OUT으로 출력합니다.

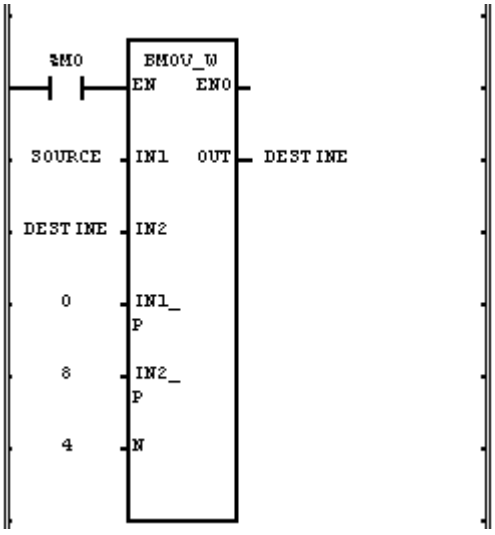
▷IN1 = 1111 0000 1111 0000, IN2 = 0000 1010 1010 1111이고 IN1_P = 4, IN2_P = 8, N = 4면, 출력되는 데이터는 OUT = 0000 1111 1010 1111이 됩니다.입력에는 B(BYTE), W(WORD), D(DWORD), L(LWORD) 타입의 데이터가 접속가능하며, L(LWORD)는GM1,2에만 적용됩니다.입출력데이터에 맞추어 'ENCO_B', 'ENCO_W', 'ENCO_D', 'ENCO_L' 중 하나의 펑션을 골라서 사용할 수 있습니다.

■ 에러

IN1_P, IN2_P의 값이 데이터 범위를 벗어나거나, N의 값이 음수 또는 IN1_P, IN2_P부터 N개를 취한 것이 데이터의 범위를 벗어나면 데이터를 출력하지 않고, _ERR,_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면 BMOV_W 펑션이 실행됩니다.
(2)입력변수로 선언된 SOURCE = 2#0101 1111 0000 1010, DESTINE = 2#0000 0000 0000 0000이고, IN1_ P = 0, IN2_P = 8, N = 4 이므로 연산 결과는 2#0000 1010 0000 0000이 되고, 출력을 DESTINE으로 지정하였으므로 DESTINE = 2#0000 1010 0000 0000으로 바뀌게 됩니다.

입력(IN1) : SOURCE(WORD) = 16#5F0A

0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0

(IN2) : DESTINE(WORD) = 16#0000

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(IN1_ P) = 0
(IN2_P) = 8
(N) = 4

(BMOV_W)

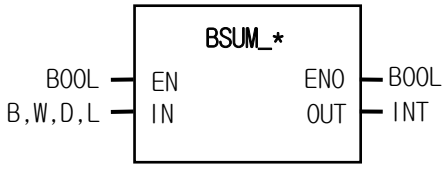
출력(OUT) : DESTINE(WORD) = 16#0A00

0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0

BSUM_***

ON된 비트 개수를 숫자로 출력

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : ON비트를 검색할 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : ON된 비트 개수를 합한 결과 데이터

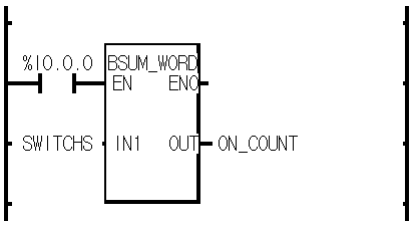
■ 기능

EN이 1이면, IN의 비트 스트링 데이터중, 1로 되어있는 비트의 숫자를 세어서 OUT으로 출력합니다.
입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속가능하며, LWORD는 GM1,2에만 적용됩니다.

FUNCTION	IN변수 타입	동작 설명
BSUM_BYTE	BYTE	입력 테이터에 맞추어 4가지 펑션중 하나를 골라서 사용합니다.
BSUM _WORD	WORD	
BSUM _DWORD	DWORD	
BSUM _LWORD	LWORD	

■ 프로그램 예

LD

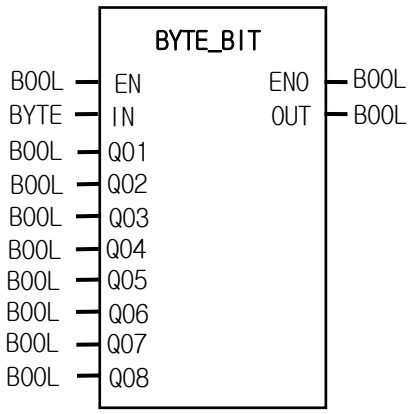


(1)실행조건(%M0) 이 On하면 BSUM_WORD 펑션이 실행됩니다.
(2)입력변수로 선언된 SWITCHS(WORD타입) = 2#0000 0100 0010 1000 이라면, ON되어있는 비트의 개수, 즉 '3'을 출력하여 ON_COUNT(INT타입)에 정수값 '3'이 저장됩니다.

BYTE_BIT

BYTE를 8개 BIT들로 나눔

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

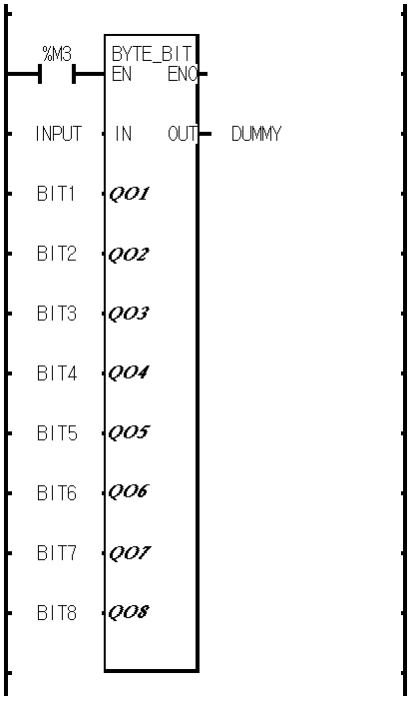
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : Byte 입력 **출력** ENO : EN값을 그대로 출력 OUT : Dummy 출력 **입출력** Q01~8 : Bit 출력

■ 기능

1개의 바이트를 입출력 변수로 선언된 8개의 비트(Q01~Q02)로 분산합니다.
Q08: MSB(최상위 비트), Q01: LSB(최하위 비트)

■ 프로그램 예

LD

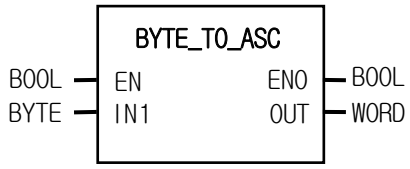


(1)실행조건(%M0)이 On 되면, BYTE_BIT 펑션이 실행됩니다.
(2)입력변수로 선언된 INPUT=16#AC=2#10101100 이면, 입출력 변수로 선언된 Q01~8 에 Q01 부터 순서대로 2#{0, 0, 1, 1, 0, 1, 0, 1}이 저장됩니다.

BYTE_TO_ASC

BYTE를 ASCII로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : BYTE입력 **출력** ENO : EN값을 그대로 출력 OUT : ASCII 출력

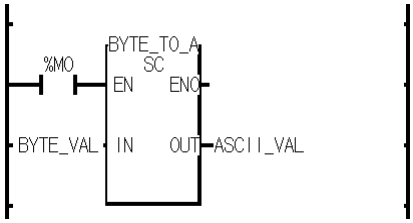
■ 기능

▷ 2자리의 16진(HEX) 데이터를 입력 받아서 2개의 아스키 값으로 출력합니다.
예) 16#12 -> 3132

▷ 16#A~F 는 대문자의 아스키 값으로 출력합니다.

■ 프로그램 예

LD



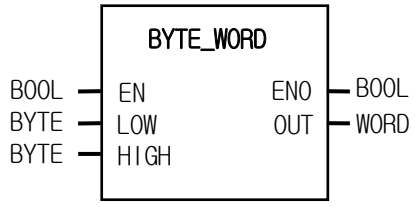
(1)실행조건(%M0)이 On 하면 BYTE_TO_ASC 펑션이 실행됩니다.

(2)펑션의 입력 변수로 선언된 BYTE_VAL(BYTE 타입) =16#3A 일 경우, 펑션의 출력변수로 선언된 ASCII_VAL(WORD 타입) = 16#3341 = '3','A'가 출력됩니다.

BYTE_WORD

2개의 BYTE를 WORD로 모음

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

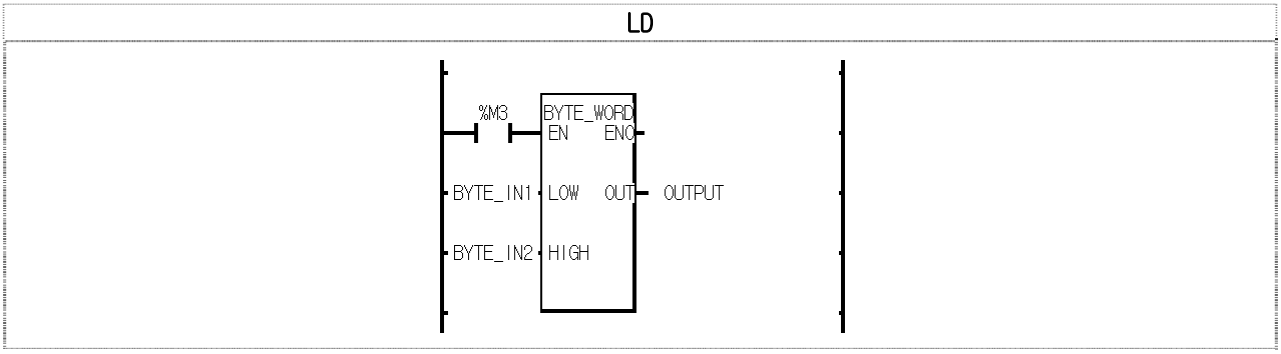
펑 션	설 명
	**입력** EN : 1일때 펑션 블록 실행 LOW : 하위 BYTE 입력 HIGH : 상위 BYTE 입력 **출력** ENO : EN값을 그대로 출력 OUT : WORD출력

■ 기능

2개의 바이트를 하나의 워드로 조합합니다.

LOW: 하위 바이트 입력, HIGH: 상위 바이트 입력

■ 프로그램 예



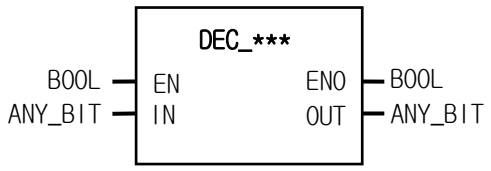
(1)실행조건(%M3)이 On 되면, BYTE_WORD 펑션이 실행됩니다.

(2)입력변수로 선언된 BYTE_IN1=16#56 이고 BYTE_IN2=16#AD 이면, 출력변수로 선언된 OUTPUT=16#AD56 입니다.

DEC_***

IN 데이터를 하나 감소

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 감소시킬 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 감소시킨 결과 데이터

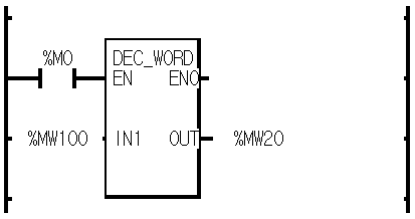
■ 기능

ENO이 1이면, IN의 비트스트링 데이터를 1만큼 감소시켜서 OUT으로 출력합니다.
언더플로우가 발생해도 에러는 발생하지 않으며, 결과는 16#0000인 경우에 16#FFFF가 됩니다.
입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속가능하며, LWORD는 GM1,2에만 적용됩니다.

FUNCTION	IN/OUT변수 타입	동작 설명
DEC_BYTE	BYTE	입출력 테이터에 맞추어 4가지 펑션중 하나를 골라서 사용합니다.
DEC_WORD	WORD	
DEC_DWORD	DWORD	
DEC_LWORD	LWORD	

■ 프로그램 예

LD

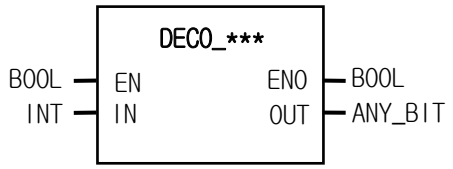


(1)실행조건(%M0)이 On하면 DEC_WORD 펑션이 실행됩니다.
(2)입력 변수로 선언된 %MW100 = 16#0007(2#0000 0000 0000 0111) 이라면, 연산이 실행된 후에는
%MW20 = 16#0006(2#0000 0000 0000 0110)이 됩니다.

DECO_***

지정된 비트 위치를 ON

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : Decoding할 입력 데이터 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : Decoding한 결과 데이터

■ 기능

EN이 1이면, IN의 값 즉 비트 위치지정 데이터에 따라서 출력의 비트 스트링 데이터중 지정된 위치의 비트만 1로 하여 출력합니다.
출력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속가능하며, LWORD는 GM1,2에만 적용됩니다.

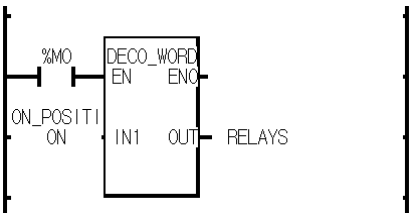
FUNCTION	OUT변수 타입	동작 설명
DECO_BYTE	BYTE	출력 테이터에 맞추어 4가지 펑션중 하나를 골라서 사용합니다.
DECO_WORD	WORD	
DECO_DWORD	DWORD	
DECO_LWORD	LWORD	

■ 에러

입력데이터가 음수이거나, 비트 위치지정 데이터가 출력타입의 비트한계를 넘으면(DECO_WORD의 경우 16 이상), OUT은 0이 되고 _ERR,_LER 플래그가 셋됩니다.

■ 프로그램 예

LD

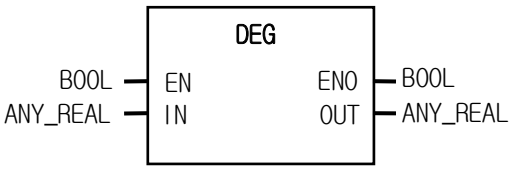


(1)실행조건(%M0)이 On하면 DECO_WORD펑션이 실행됩니다.
(2)입력변수로 선언된 ON_POSITON(INT타입) = 5 라면, 출력의 5번 비트만 ON되므로,
RELAYS(WORD타입) = 2#0000 0000 0010 0000 이 됩니다.

DEG_***

Radian값을 각도로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑션	설 명
	**입력** EN : 1 일때 펑션 실행 IN : 라디안(Radian)값 입력 **출력** ENO : EN값을 그대로 출력 OUT : 각도 출력

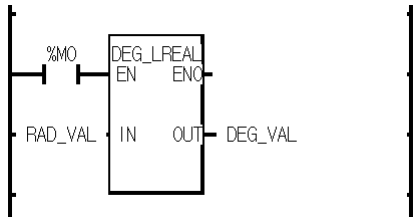
■ 기능

라디안(Radian) 값을 입력 받아서 각도(Degree)로 출력 합니다.

펑션	입력 타입	출력 타입	동작 설명
DEG_REAL	REAL	REAL	라디안(Radian)값을 출력 데이터 타입에 해당하는 각도 값으로 변환합니다.
DEG_LREAL	LREAL	LREAL	

■ 프로그램 예

LD



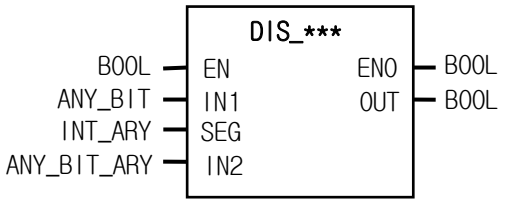
(1)실행조건(%M0)이 On 되면, DEG_LREAL 펑션이 실행됩니다.

(2)펑션의 입력변수로 선언된 RAD_VAL=1.0 일 경우, 펑션의 출력변수 DEG_VAL=5.7295779513078550e+001 가 됩니다.

DIS_***

데이터 분산(Distribution)

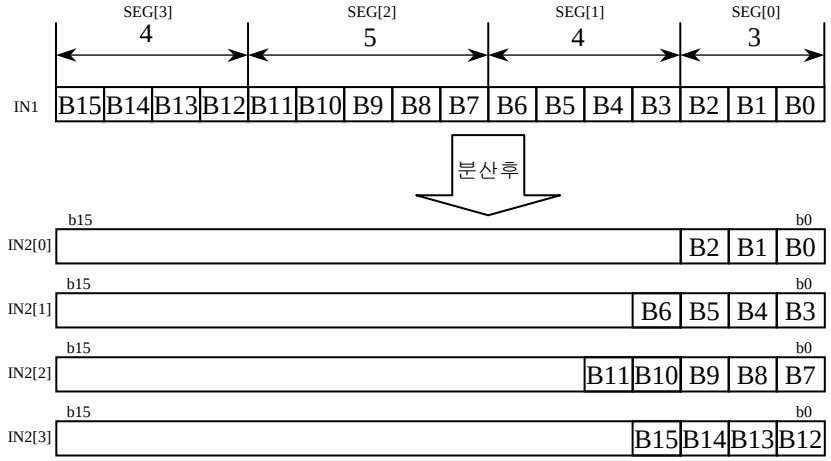
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : 입력 데이터 SEG : 데이터 분산 비트수 지정 어레이 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : Dummy 출력 **입출력** IN2 : 분산된 WORD 어레이 출력

■ 기능

입력 데이터를 SEG에서 지정된 비트 개수 단위로 구분하여 IN2로 출력합니다.

펑션	입력 타입	동작 설명
DIS_BYTE	BYTE	각 타입에 해당하는 IN1 입력을 SEG 입력 어레이 값으로 비트열을 구분하여 IN1과 동일한 데이터 타입의 IN2 어레이로 출력합니다.
DIS_WORD	WORD	
DIS_DWORD	DWORD	
DIS_LWORD	LWORD	

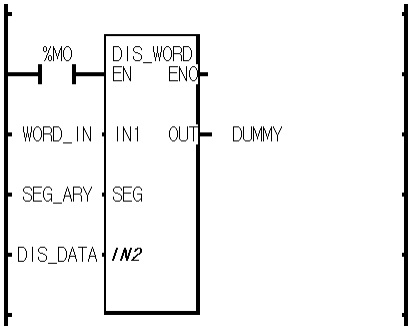


■ 에러

SEG로 지정된 값들의 합이 입력 변수 타입의 비트수를 초과할 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면, DIS_WORD 펑션이 실행됩니다.

(2)입력변수로 선언된 WORD_IN의 값이 16#3456이고,SEG_ARY={3,4,5,4}이면,펑션이 실행된 후에 DIS_DATA로 출력되는 값은

DIS_DATA[0]=16#0003

DIS_DATA[1]=16#000A

DIS_DATA[2]=16#0008

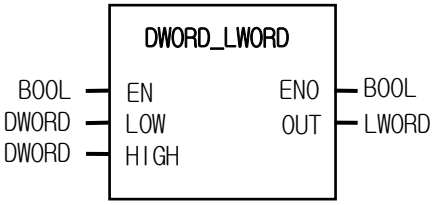
DIS_DATA[3]=16#0003

가 출력됩니다.

DWORD_LWORD

2개의 DWORD를 LWORD로 모음

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

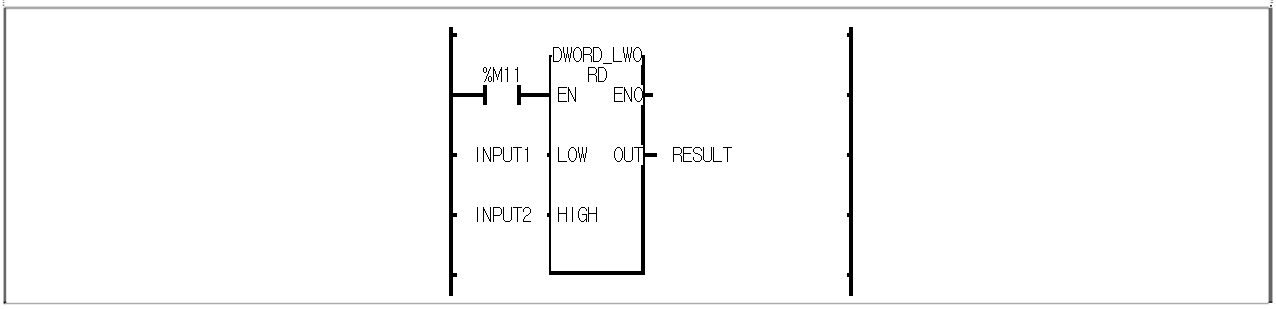
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 LOW : 하위 DWORD 입력 HIGH : 상위 DWORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : LWORD 출력

■ 기능

2개의 DWORD를 하나의 LWORD로 조합합니다.
LOW: 하위 더블워드 입력, HIGH: 상위 더블워드 입력

■ 프로그램 예

LD

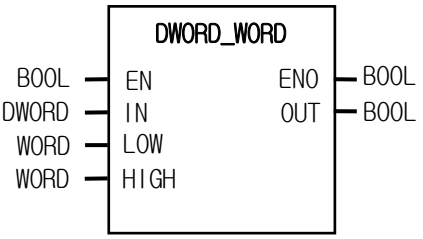


(1)실행조건(%M11)이 On 되면, DWORD_LWORD 펑션이 실행됩니다.
(2)입력변수로 선언된 INPUT1=16#1A2A3A4A5A6A7A8A 이고 INPUT2=16#8C7C7C6C5C4C3C2C1C 일 때 출력 변수로 선언된 RESULT=16#8C7C6C5C4C3C2C1C1A2A3A4A5A6A7A8A 가 출력됩니다.

DWORD_WORD

DWORD를 2개의 WORD로 나눔

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : DWORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : Dummy 출력 **입출력** LOW : 하위 WORD 출력 HIGH : 상위 WORD 출력

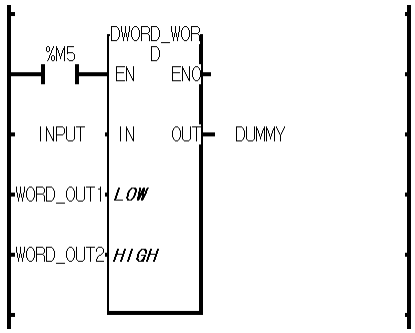
■ 기능

하나의 DWORD를 2개의 WORD로 분산 합니다.

LOW: 하위 워드 출력, HIGH: 상위 워드 출력

■ 프로그램 예

LD



(1)실행조건(%M5)이 On 되면, DWORD_WORD 펑션이 실행됩니다.

(2)입력변수로 선언된 INPUT=16#11223344AABBCCDD 일 때, 입출력 변수로 선언된

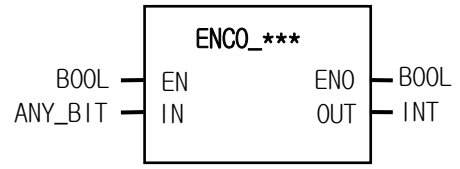
WORD_OUT1=16#AABBCCDD

WORD_OUT2=16#11223344 이 저장됩니다.

ENCO_***

ON된 비트 위치를 숫자로 출력

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : Encoding할 입력 데이터 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : Encoding한 결과 데이터

■ 기능

EN이 1이면, IN의 비트 스트링 데이터중, 1로 되어있는 비트중 최상위 비트의 위치를 OUT으로 출력합니다.

입력에는 B(BYTE), W(WORD), D(DWORD), L(LWORD) 타입의 데이터가 접속가능하며, L(LWORD)는GM1,2에만 적용됩니다.

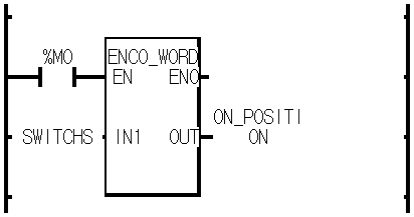
FUNCTION	IN변수 타입	동작 설명
ENCO_BYTE	BYTE	각 입력 변수 타입에 따라 원하는 ENCO 펑션을 골라서 사용합니다.
ENCO_WORD	WORD	
ENCO_DWORD	DWORD	
ENCO_LWORD	LWORD	

■ 에러

입력데이터중 하나의 비트도 1이 되어있지 않은 경우는 OUT은 -1이 되고, _ERR,_LER 플래그가 셋(Set) 됩니다.

■ 프로그램 예

LD



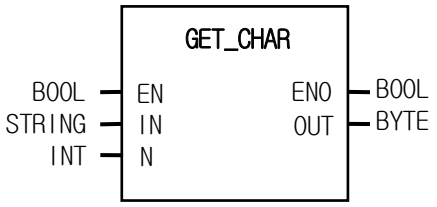
(1)실행조건(%M0)이 On하면 ENCO_WORD 펑션이 실행됩니다.

(2)SWITCHS(WORD타입) = 2#0000 1000 0000 0010 이라면, ON되어있는 2비트의 위치, 즉 '11'과'1'중 상위 위치인 '11'을 출력하여 ON_POSITON(INT타입)에 정수값 '11'이 저장됩니다.

GET_CHAR

문자열로부터 한 문자(CHAR)추출

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : STRING 입력 N : STRING내의 위치 지정 **출력** ENO : EN값을 그대로 출력 OUT : Byte 출력

■ 기능

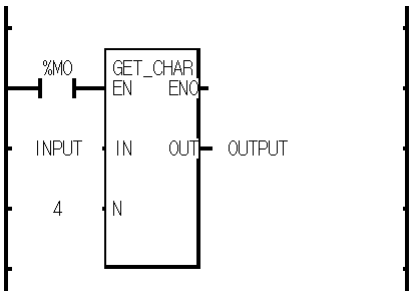
STRING의 지정된 위치로부터 하나의 바이트를 추출합니다.

■ 에러

▷ N값이 스트링의 바이트 개수를 넘는 경우 _ERR/_LER 플래그가 셋(set)됩니다.
▷ 에러가 발생했을 경우 16#00이 출력됩니다.

■ 프로그램 예

LD

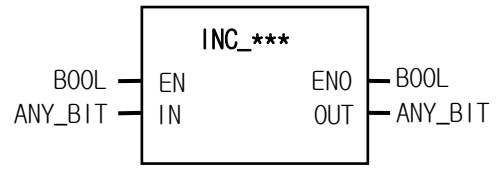


(1)실행조건(%M0)이 On되면, GET_CHAT 펑션이 실행됩니다.
(2)입력 변수로 선언된 INPUT(STRING 타입)="LG GLOFA PLC"일 때 이 String의 4번째 문자를 추출하게 되면 출력 변수로 선언된 OUTPUT으로 16#47("G")가 출력됩니다.

INC_***

IN 데이터를 하나 증가

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 증가시킬 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 증가시킨 결과 데이터

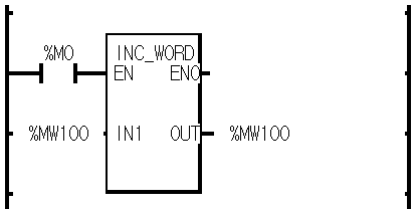
■ 기능

EN이 1이면, IN의 비트스트링 데이터를 1만큼 증가시켜서 OUT으로 출력합니다.
오버플로우가 발생해도 에러는 발생하지 않으며, 결과는 16#FFFF인 경우에 16#0000가 됩니다.
입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속가능하며, LWORD는 GM1,2에만 적용됩니다.

FUNCTION	IN/OUT변수 타입	동작 설명
INC_BYTE	BYTE	입출력 데이터에 맞추어 4가지 펑션중 하나를 골라서 사용합니다.
INC_WORD	WORD	
INC_DWORD	DWORD	
INC_LWORD	LWORD	

■ 프로그램 예

LD

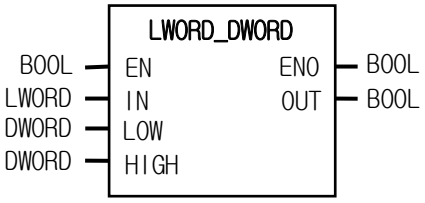


(1)실행조건(%M0)이 On하면 INC_WORD 펑션이 실행됩니다.
(2)입력변수 %MW100 = 16#0007(2#0000 0000 0000 0111) 이라면, 연산이 실행된 후에는
%MW100 = 16#0008(2#0000 0000 0000 1000)이 됩니다.

LWORD_DWORD

LWORD를 2개의 DWORD로 나눔

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

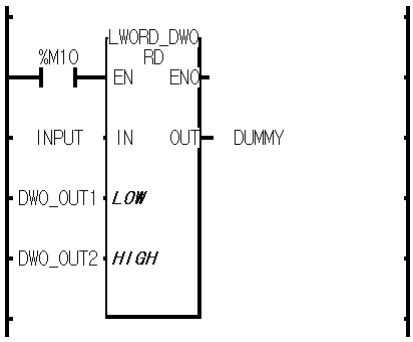
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : LWORD입력 **출력** ENO : EN값을 그대로 출력 OUT : Dummy 출력 **입출력** LOW : 하위 DWORD 출력 HIGH : 상위 DWORD 출력

■ 기능

▷ 하나의 LWORD를 2개의 DWORD로 분산 합니다.
LOW: 하위 더블워드 출력, HIGH: 상위 더블워드 출력

■ 프로그램 예

LD

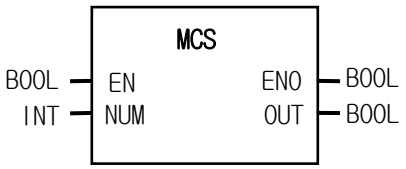


(1)실행조건(%M10)이 On되면, LWORD_DWORD 펑션이 실행됩니다.
(2)입력변수로 선언된 INPUT=16#AAAABBBBCCCCDDDDABCDABCDABCDABCD일 때, 입출력 변수로 선언된
DWO_OUT1=16#ABCDABCDABCDABCD
DWO_OUT2=16#AAAABBBBCCCCDDDD
가 출력됩니다.

MCS

Master Control

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 NUM : Nesting (0~15) **출력** ENO : MCS 명령이 실행되면 1을 출력 OUT : Dummy(항상 0 을 출력합니다.)

■ 기능

▷EN이 On이면, Master Control이 수행됩니다. 이경우, MCS 펑션에서 MCSCLR 펑션사이의 프로그램은 정상적으로 수행됩니다.

▷EN이 Off 인경우, MCS 펑션에서 MCSCLR 펑션사이의 프로그램은 아래와 같이 수행됩니다.

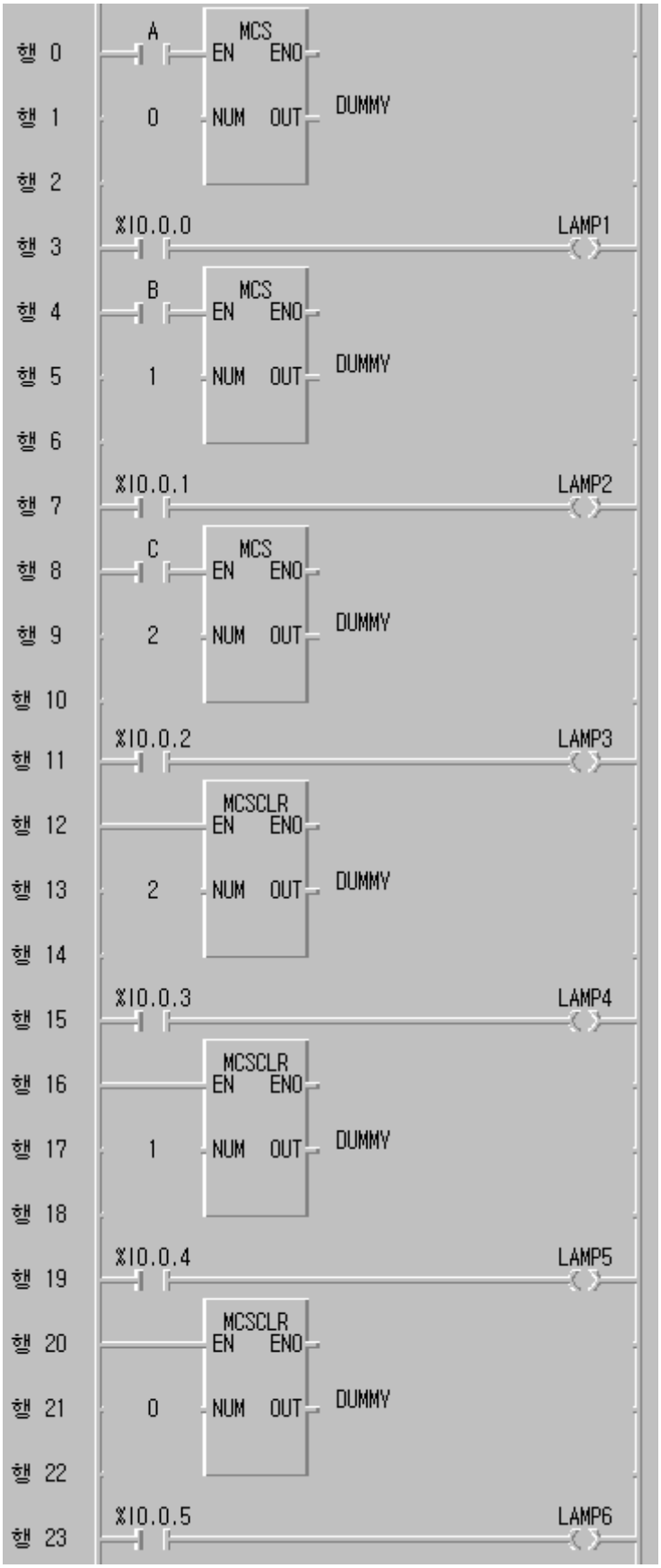
명령어	명령어 상태
Timer	현재값은 0이되고, 출력(Q)은 Off됩니다.
Counter	출력(Q)은 Off되고, 현재값은 현재 상태를 유지합니다.
코일	모두 Off됩니다.
역코일	모두 Off됩니다.
셋코일, 리셋코일	현재 값을 유지합니다.
펑션, 펑션 블록	현재 값을 유지합니다.

▷EN이 Off 인경우에도 MCS 펑션에서 MCSCLR 펑션사이의 명령들이 위와 같이 수행되기 때문에 스캔 타임이 감소되지 않습니다.

▷Master Control 명령은 Nesting해서 사용될 수 있습니다. 즉, Master Control 영역이 Nesting(NUM)에 의해 구분될 수 있습니다. Nesting(NUM)은 0에서 15까지 설정이 가능하고, 만약 16이상으로 설정한 경우 Master Control이 정상적으로 동작하지 않습니다.

* MCSCLR없이 MCS 명령을 사용한 경우,MCS 펑션에서 프로그램의 마지막 행까지 Master Control이 수행되니 주의 바랍니다.

■ 프로그램 예



A가 On 일때 실행합니다.

A와 B가 On 일때 실행합니다.

A와 B와 C가 On 일때 실행합니다.

A와 B가 On 일때 실행합니다.

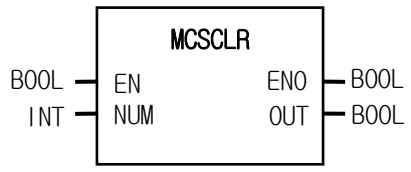
A가 On 일때 실행합니다.

A,B,C와 상관없이 실행합니다.

MCSCLR

Master Control 해제 명령

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 NUM : Nesting (0~15) **출력** ENO : MCSCLR 명령이 실행되면 1을 출력 OUT : MCSCLR 명령이 실행되면 1을 출력

■ 기능

▷Master Control 명령을 해제합니다. 그리고, Master Control 영역의 마지막을 가리킵니다.

▷MCSCLR 펑션 동작시 Nesting(NUM)의 값보다 같거나 작은 모든 MCS 명령을 해제합니다.

* MCSCLR 펑션 앞에는 접점을 사용하지 않습니다.

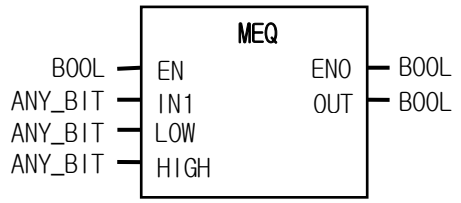
■ 프로그램 예

MCS 펑션의 프로그램예를 참조 바랍니다.

MEQ_***

Masked Equal

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN1 : 입력1 IN2 : 입력2 MASK : masking할 입력값 **출력** ENO : EN값을 그대로 출력 OUT : 동일하면 1을 출력

■ 기능

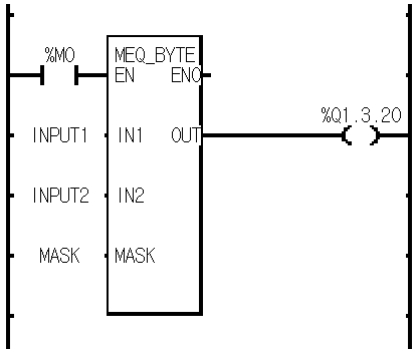
▷ 입력된 변수값들을 Masking한 후 두개의 변수값이 서로 동일한지를 비교합니다. 만약 8비트 변수값을 2#11111100으로 Masking하면 하위 2비트는 입력값 비교에서 제외됩니다.

▷ 하나의 변수값에서 특정 비트들이 ON되어 있는지 검사하는 용도로도 사용이 가능합니다. 즉, 8비트 변수 비교시 IN1에 비교하고자 하는 변수를 입력하고 IN2에는 16#FF로 설정한 후 MASK에 검색하고자 하는 비트조합을 입력하면(i.e. 2#00101100) 입력되는 변수와 비교하여 동일한 경우 ON이 출력됩니다.

펑션	입력변수 타입	동작 설명
MEQ_BYTE	BYTE	입력값들을 Masking 한 후 이 값들이 서로 동일한 지를 비교합니다.
MEQ_WORD	WORD	
MEQ_DWORD	DWORD	
MEQ_LWORD	LWORD	

■ 프로그램 예

LD



(1)실행조건(%M0)이 On 되면, MEQ_BYTE 펑션이 실행됩니다.

(2)입력 변수 INPUT1(BYTE 타입)= 2#01011100

INPUT2(BYTE 타입)= 2#01110101

MASK(BYTE 타입) = 2#11010110 일 경우 Making 된 후의 입력 변수들의 비교할 비트는

INPUT1'(BYTE 타입) = 2#01010100

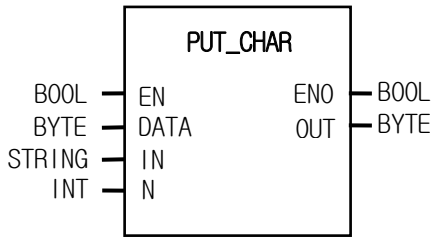
INPUT2'(BYTE 타입)= 2#01010100

과 같이 되어 서로 동일한 값을 가지므로 출력접점 %Q1.3.20 이 On 됩니다

PUT_CHAR

문자열에 한 문자 써넣기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 DATA : String에 삽입할 Byte입력 IN : String 입력 N : String내의 위치 지정 **출력** ENO : EN값을 그대로 출력 OUT : String 출력

■ 기능

하나의 바이트 입력값을 STRING상의 지정된 위치(N 숫자)에 덮어쓰기(Overwrite)합니다.

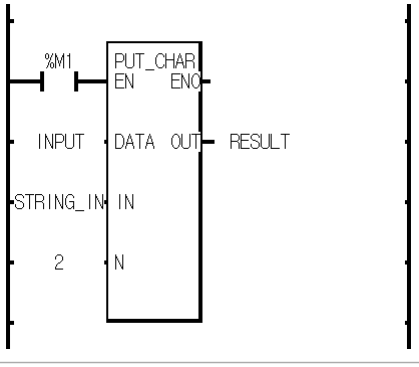
■ 에러

▷ N값이 스트링의 바이트 개수를 넘는 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ 에러가 발생했을 경우 16#00이 출력됩니다.

■ 프로그램 예

LD



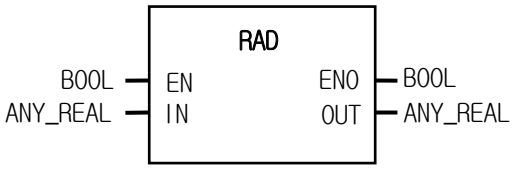
(1)실행조건(%M1)이 On하면 PUT_CHAR 펑션이 실행됩니다.

(2)입력변수인 INPUT=16#41("A")이고 STRING_IN="TOKEN" 일 때 입력 변수 STRING_IN의 2번째 위치에 입력변수 INPUT이 지닌 값을 덮어쓰기 하게 되면 출력 변수인 RESULT는 "TAKEN"이 됩니다.

RAD_***

각도(DEG)를 Radian 값으로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●				

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 각도 입력 **출력** ENO : EN값을 그대로 출력 OUT : 라디안 값 출력

■ 기능

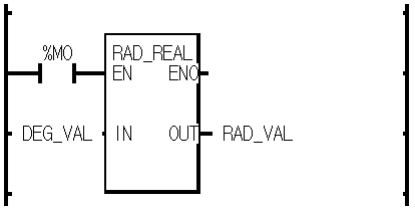
▷ 각도의 단위를 도(°)에서 라디안(Radian)값으로 출력 합니다.

▷ 각도가 360°를 넘어서더라도 정상적으로 변환시켜 줍니다.(예를 들어, 입력이 370°이면 출력은 360°를 뺀 10°에 해당하는 라디안 값을 출력합니다.)

펑션	입력 타입	출력 타입	동작 설명
RAD_REAL	REAL	REAL	각도의 단위를 도(°)에서 출력 데이터 타입에 해당하는 라디안 값으로 변환합니다.
RAD_LREAL	LREAL	LREAL	

■ 프로그램 예

LD



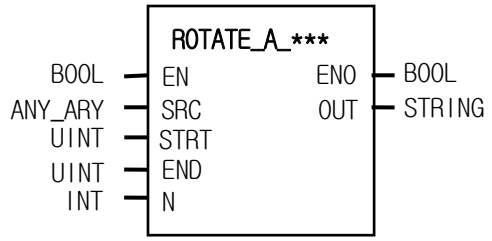
(1)실행조건(%M0)이 On 하면, RAD_REAL 펑션이 실행됩니다.

(2)펑션의 입력 변수로 선언된 DEG_VAL=127(°)일 경우, 펑션의 출력변수 값은 RAD_VAL=2.21656823 이 됩니다.

ROTATE_A_***

지정된 어레이 원소의 ROTATE

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

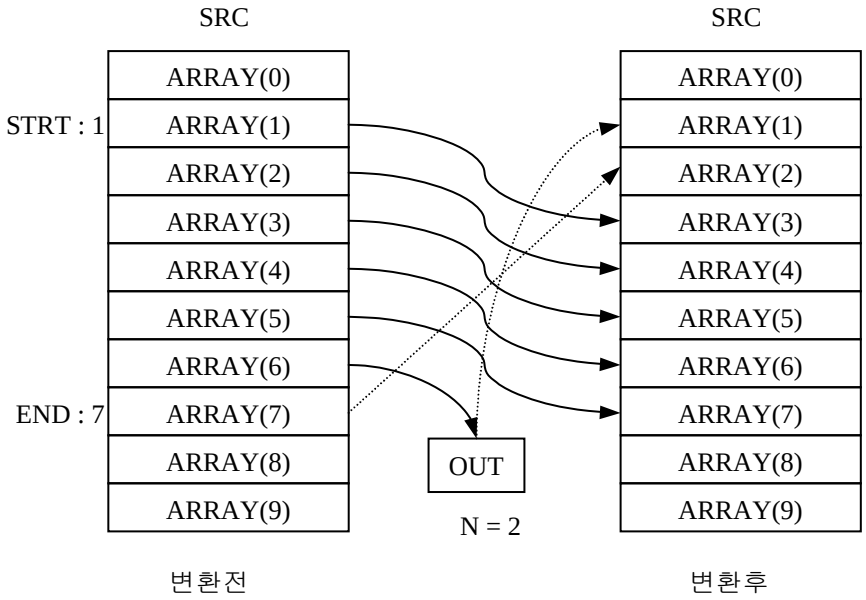
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 N : Rotate 시킬 개수 STRT : 어레이 블록중 Rotate할 원소의 시작위치 END : 어레이 블록중 Rotate 할 원소의 종료위치 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : Rotate하여 밀려나온 데이터를 출력 **입출력** SRC : Rotate조작이 가해질 어레이 블록

■ 기능

▷ ROTATE_A_*** 펑션은 어레이 블록중 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.

▷ 동작의 지정:

- 범위지정 : STRT와 END로 이동할 데이터 원소의 범위를 지정합니다.
- 이동방향 및 횟수 : STRT에서 END방향으로 지정된 횟수(N)만큼 Rotate됩니다.
- 입력 데이터 지정 : Rotate하여 비워지는 위치를 입력 데이터(IN)로 채웁니다.
- 출력 : 데이터 조작의 결과는 *SRC*에 지정된 ANY_ARY에 저장되며, END 위치에서 Rotate동작으로 STRT로 돌아들어가는 데이터는 OUT으로 출력됩니다.



펑션	입출력 어레이 타입	동작 설명
ROTATE_A_BOOL	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 ROTATE 시킵니다.
ROTATE_A_BYTE	BYTE	
ROTATE_A_WORD	WORD	
ROTATE_A_DWORD	DWORD	
ROTATE_A_LWORD	LWORD	
ROTATE_A_SINT	SINT	
ROTATE_A_INT	INT	
ROTATE_A_DINT	DINT	
ROTATE_A_LINT	LINT	
ROTATE_A_USINT	USINT	
ROTATE_A_UINT	UINT	
ROTATE_A_UDINT	UDINT	
ROTATE_A_ULINT	ULINT	
ROTATE_A_REAL	REAL	
ROTATE_A_LREAL	LREAL	
ROTATE_A_TIME	TIME	
ROTATE_A_DATE	DATE	
ROTATE_A_TOD	TOD	
ROTATE_A_DT	DT	

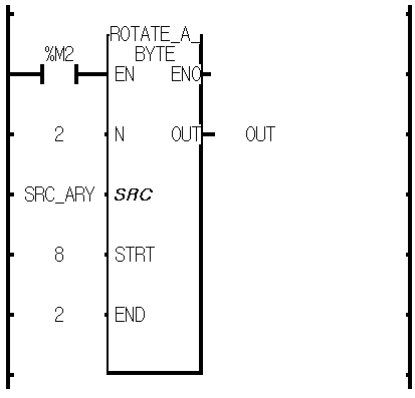
■ 에러

▷ STRT 또는 END값이 SRC 어레이의 원소 개수 범위를 벗어나면 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ 에러발생시 SRC는 변하지 않고 출력은 각 변수타입의 초기화값(i.e. INT=0, TIME=T#0S)이 출력됩니다.

■ 프로그램 예

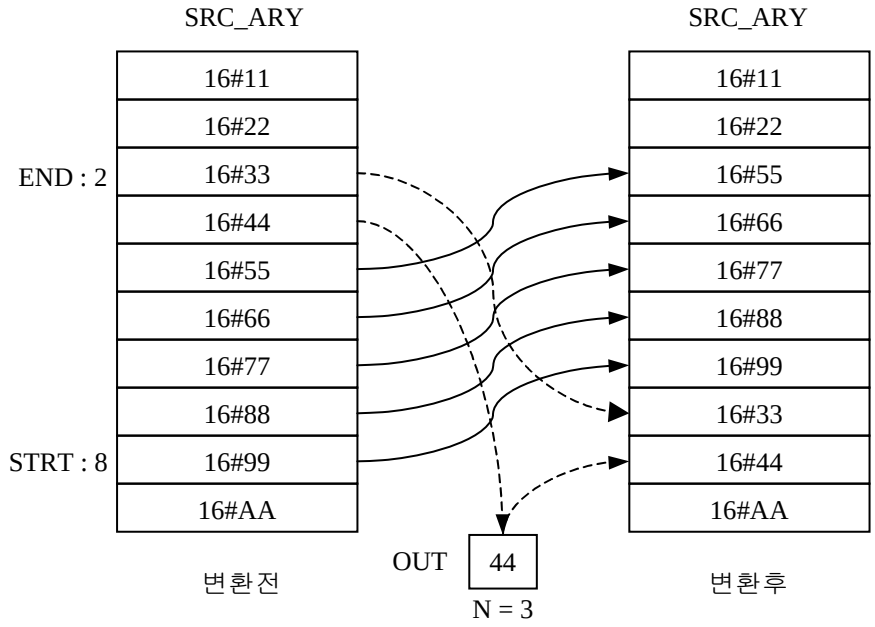
LD



(1)입력 조건(%M2)이 On되면, ROTATE_A_BYTE 펑션이 실행됩니다.

(2)입출력 변수로 선언된 SRC_ARY의 인덱스 8의 원소부터 인덱스 2의 원소의 방향으로 ROTATE동작이 2번 일어납니다.

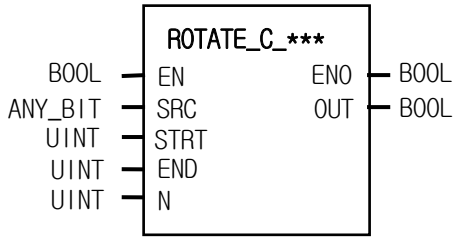
(3)출력 값에는 캐리 출력에 해당하는 원소의 값 16#44가 출력됩니다.


SRC_ARY
SRC_ARY
16#11
16#22
16#33
16#44
16#55
16#66
16#77
16#88
16#99
16#AA
END : 2
STRT : 8
16#11
16#22
16#55
16#66
16#77
16#88
16#99
16#33
16#44
16#AA
OUT
44
N = 3
변환전
변환후

ROTATE_C_***

지정된 어레이 원소의 ROTATE

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

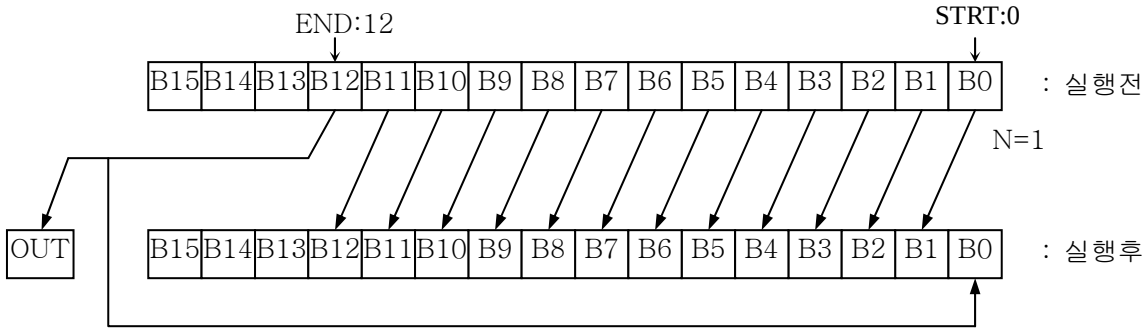
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 STRT : SRC의 비트열 중 회전할 범위의 시작 bit 위치 END : SRC의 비트열 중 회전할 범위의 끝 bit 위치 N : Shift할 비트수 **출력** ENO : 에러없이 수행되면 1을 출력 OUT : Carry 출력 **입출력** *SRC* : 회전할 변수

■ 기능

▷ SRC의 비트열중 지정된 범위의 원소들을 지정된 방향으로 회전합니다.

▷ 동작의 지정:

- 범위지정: STRT와 END로 이동할 비트의 범위를 지정합니다.
- 이동방향 및 횟수: START에서 END방향으로 지정된 횟수(N)만큼 회전됩니다.
- 출력: 데이터 조작의 결과는 *SRC* 에 지정된 ANY_BIT에 바뀌어 저장되며, 회전동작으로 END위치에서 STRT로 돌아 들어가는 비트 데이터는 OUT으로도 출력됩니다.



펑션	SRC변수 타입	동작 설명
ROTATE_C_BYTE	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Rotate시킵니다.
ROTATE_C_WORD	WORD	
ROTATE_C_DWORD	DWORD	
ROTATE_C_LWORD	LWORD	

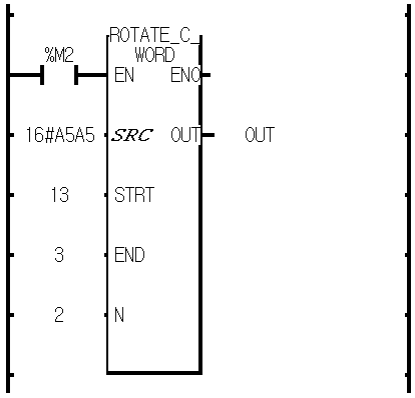
■ 에러

▷ START 와 END 값이 SRC 변수 타입의 비트 개수를 넘는 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

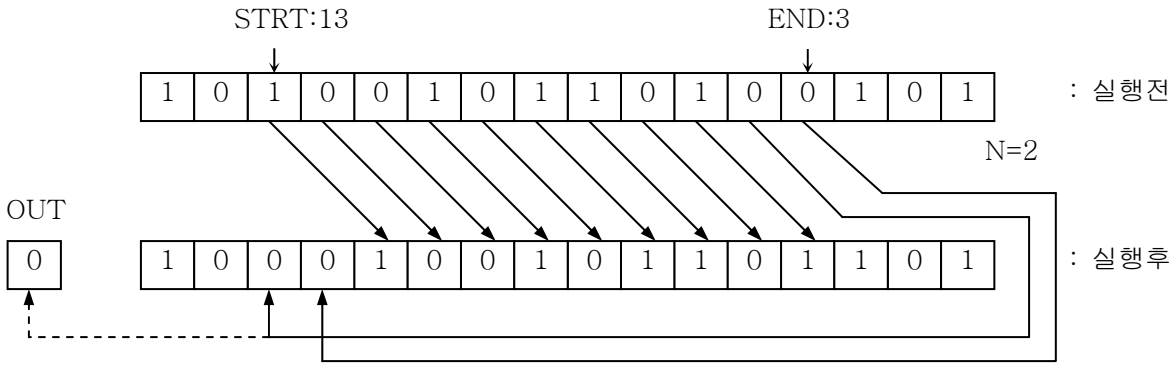
▷ SRC의 값은 변화가 없습니다.

■ 프로그램 예

LD



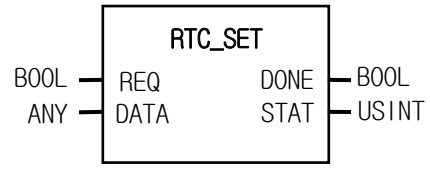
(1)실행조건(%M2)이 On되면, ROTATE_C_WORD 펑션이 실행됩니다.
(2)16#A5A5값이 STRT(13)와 END(3)로 지정된 범위에서 STRT부터 END방향으로 2번을 회전합니다.
(3)Rotate된 후의 값이 SRC(16#896D)로 다시 출력되고 회전되면서 END위치에서 STRT로 되돌아가는 비트인 0이 OUT으로 출력됩니다.



RTC_SET

시간 데이터 쓰기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** REQ : Rising Edge 시 펑션 블록 실행 DATA : 입력할 시간 데이터 **출력** DONE : RTC_SET을 정상적으로 수행되면1을 출력 STAT : 에러 발생시 에러 코드 출력

■ 기능

▷REQ 변수의 Rising Edge 에서 아래와 같이 Setting한 RTC시간(DATA)을 PLC의 Clock Device에 저장합니다.

변수	내용	예	변수	내용	예
DATA[0]	년도	16#01	DATA[4]	분	16#30
DATA[1]	월	16#03	DATA[5]	초	16#45
DATA[2]	일	16#15	DATA[6]	요일	16#03
DATA[3]	시	16#18	DATA[7]	년대	16#20

*위 예는 2001년 3월 15일 (목) 18시 30분 45초인 경우입니다.

*요일 표시는 월 : 0, 화 : 1, 수 : 2, 목 : 3, 금 : 4, 토 : 5, 일 : 6 으로 합니다.

▷ 위 DATA 변수는 반드시 어레이 BYTE 변수로 선언하고, 각각의 데이터는 BCD값으로 Setting 합니다.

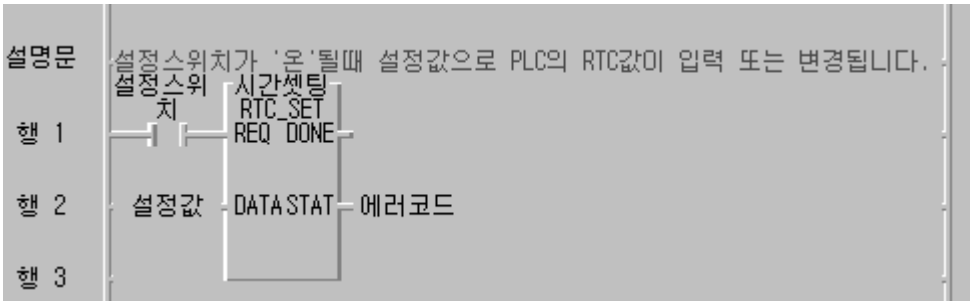
■ 에러

CPU가 RTC를 지원하지 않는 경우나, RTC 데이터가 범위를 벗어난 경우 출력(DONE)은 '0'이 STAT에 아래 표와 같이 에러 코드를 출력합니다.

에러 코드	내용
00	No error
01	RTC 모듈이 없습니다. * GM6인 경우 GM6-CPUB, GM6-CPUC 만 RTC를 지원합니다. * GM7 인 경우G7E-RTCA를 장착하십시오.
02	부적절한 RTC 데이터 입니다. 예) 14(월) 32(일) 25(시간) * RTC 데이터를 옳바르게 설정하여 주십시오.

■ 프로그램 예

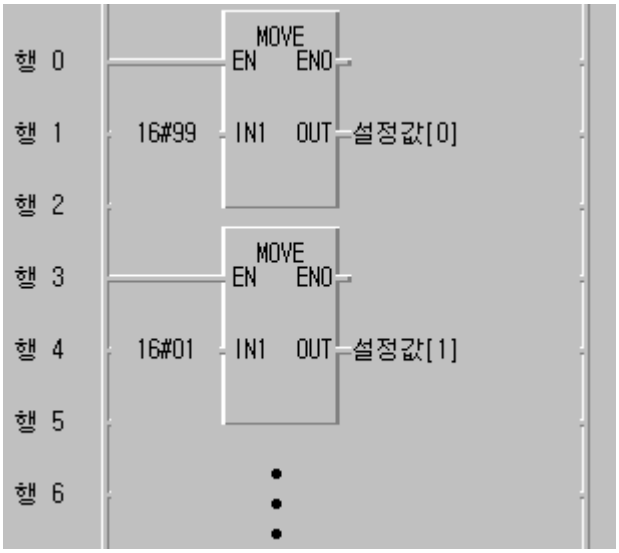
RTC 데이터가 1999. 1. 17. 11:53:24, 일요일인 경우의 예입니다.



(1) 설정스위치가 '온'될때 설정값으로 PLC의 RTC값이 입력 또는 변경됩니다.

(2) 아래는 변수(설정값) 설정 방법입니다.

(3) 위와 같이 초기값으로 주는 방법외에 설정값은 펑션 MOVE를 사용하여 각각의 설정치를 변수(설정값)로 저장할 수 있습니다.



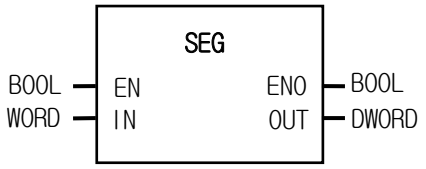
(4) RTC 값을 읽을 경우 아래의 플래그를 사용합니다.
예 : 1998. 12. 22. 19:37:46, 화

키워드	Type	내용	설명	Data
_RTC_TOD	TOD	현재 시각	현재 시각 데이터	TOD#19:37:46
_RTC_WEEK	UINT	현재 요일	요일 데이터 *(0: 월, 1: 화, 2: 수, 3: 목, 4: 금, 5: 토, 6: 일)	1
_INT_DATE	DATE	현재 날짜	현재 날짜 데이터(1984년1월1일~2083년12월31일)	D#1998-12-22
_RTC_ERR	BOOL	RTC 에러	RTC 데이터 에러시 '1'출력	0
_RTC_TIME[n] * n : 0 to 7	BCD	현재 시각	RTC 현재시각의BCD 데이터 _RTC _TIME[0]:년, _RTC _TIME[1]:월, _RTC _TIME[2]:일, _RTC _TIME[3]:시, _RTC _TIME[4]:분, _RTC _TIME[5]:초, _RTC _TIME[6]:요일,_RTC _TIME[7]:세기 요일 (0: 월, 1: 화, 2: 수, 3: 목, 4: 금, 5: 토, 6: 일)	_RTC _TIME[0]: 16#98 _RTC _TIME[1]: 16#12 _RTC _TIME[2]: 16#22 _RTC _TIME[3]: 16#19 _RTC _TIME[4]: 16#37 _RTC _TIME[5]: 16#46 _RTC _TIME[6]: 16#1 _RTC _TIME[7]: 16#19

SEG

BCD 또는 HEX값을 7세그먼트 디스플레이 코드로 변환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 7세그먼트 코드로 변환할 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 7세그먼트 코드로 변환된 결과 데이터

■ 기능

EN이 1이면, IN의 BCD 또는 HEX(16진) 숫자를 아래표와 같이 7세그먼트 디스플레이를 위한 코드로 변환하여OUT으로 출력합니다. BCD입력의 경우 0000 ~ 9999까지의 값이 4개의 7세그먼트에 표시 가능하며, HEX입력의 경우 0000 ~ FFFF의 값이 4개의 7세그먼트에 표시 가능합니다.

표시 예

1) 4자리 BCD -> 4자리 7세그먼트 코드 : 'SEG'펑션 사용

2) 4자리 HEX -> 4자리 7세그먼트 코드 : 'SEG'펑션 사용

3) 정수 -> 4자리 BCD형의 7세그먼트 코드 : 'INT_TO_BCD'펑션을 거쳐서 'SEG'펑션 사용

4) 정수 -> 4자리 HEX형의 7세그먼트 코드 : 'INT_TO_WORD'펑션을 거쳐서 'SEG'펑션 사용

5) 7세그먼트의 자릿수가 4자리를 넘을 때

가) BCD, HEX는 4자리씩 나누어 'SEG'펑션을 사용

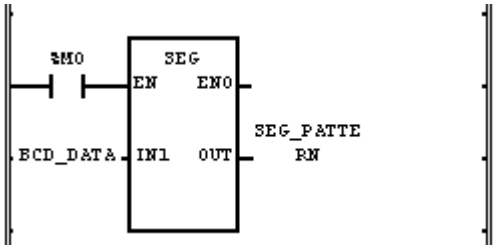
나) 정수 -> 8자리 BCD형 7세그먼트 코드:

정수를 10,000으로 나누어 몫과 나머지를 각각 'INT_TO_BCD'펑션을 거쳐서 'SEG'로 변환하여

상위 4자리와 하위 4자리의 7세그먼트 코드를 생성

■ 프로그램 예

LD



(1) 실행조건(%M0)이 On하면 SEG펑션이 실행됩니다.
(2) 입력변수로 선언된BCD_DATA(WORD타입) = 16#1234라면, 7세그먼트 디스플레이에 '1234'가 표시
되는 코드 '2#00000110_01011011_01001111_01100110'이 출력되어 SEG_PATTERN(DWORD타입)에 저장됩니다.

입력(IN1) : BCD_DATA(WORD) = 16#1234

출력(OUT) : SEG_PATTERN(DWORD) = 상위
16#065B4F66 하위

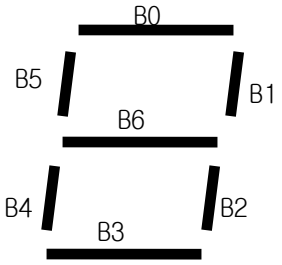
0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

(SEG)

0	0	0	0	0	1	1	0	0	1	0	1	1	0	1	1

0	1	0	0	1	1	1	1	0	1	1	0	0	1	1	0

7 세그먼트의 구성



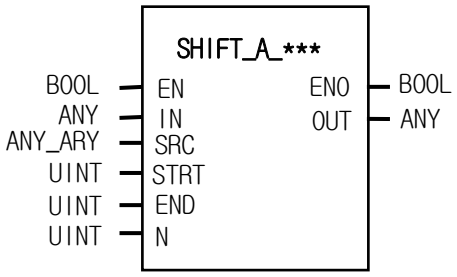
7세그먼트 코드 변환표

입력 (BCD)	입력 (16진수)	정수값	출력 B7	B6	B5	B4	B3	B2	B1	B0	표시 데이터
0	0	0	0	0	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	1	1	0	1
2	2	2	0	1	0	1	1	0	1	1	2
3	3	3	0	1	0	0	1	1	1	1	3
4	4	4	0	1	1	0	0	1	1	0	4
5	5	5	0	1	1	0	1	1	0	1	5
6	6	6	0	1	1	1	1	1	0	1	6
7	7	7	0	0	1	0	0	1	1	1	7
8	8	8	0	1	1	1	1	1	1	1	8
9	9	9	0	1	1	0	1	1	1	1	9
	A	10	0	1	1	1	0	1	1	1	A
	B	11	0	1	1	1	1	1	0	0	B
	C	12	0	0	1	1	1	0	0	1	C
	D	13	0	1	0	1	1	1	1	0	D
	E	14	0	1	1	1	1	0	0	1	E
	F	15	0	1	1	1	0	0	0	1	F

SHIFT_A_***

지정된 어레이 원소의 SHIFT

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

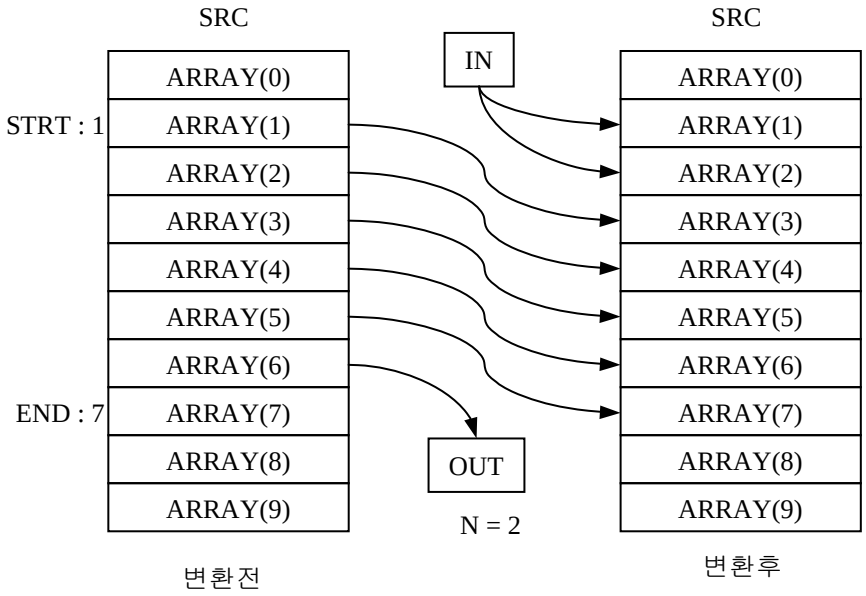
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : Shift된 후 비워진 원소들의 자리에 입력될 값 N : Shift 시킬 개수 STRT: 어레이 블록중 Shift할 원소의 시작위치 END : 어레이 블록중 Shift할 원소의 종료위치 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : Shift하여 밀려나온 데이터를 출력 **입출력** SRC : Shift조작이 가해질 어레이 블록

■ 기능

▷ SHIFT_A_*** 펑션은 어레이 블록중 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.

▷ 동작의 지정:

- 범위지정 : STRT와 END로 이동할 데이터 원소의 범위를 지정합니다.
- 이동방향 및 횟수 : STRT에서 END방향으로 지정된 횟수(N)만큼 Shift됩니다.
- 입력 데이터 지정 : Shift하여 비워지는 위치를 입력 데이터(IN)로 채웁니다.
- 출력 : 데이터 조작의 결과는 *SRC*에 지정된 ANY_ARY에 저장되며, END 위치에서 Shift동작으로 밀려나온 데이터는 OUT으로 출력됩니다.



펑션	입출력 어레이 타입	동작 설명
SHIFT_A_BOOL	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.
SHIFT_A_BYTE	BYTE	
SHIFT_A_WORD	WORD	
SHIFT_A_DWORD	DWORD	
SHIFT_A_LWORD	LWORD	
SHIFT_A_SINT	SINT	
SHIFT_A_INT	INT	
SHIFT_A_DINT	DINT	
SHIFT_A_LINT	LINT	
SHIFT_A_USINT	USINT	
SHIFT_A_UINT	UINT	
SHIFT_A_UDINT	UDINT	
SHIFT_A_ULINT	ULINT	
SHIFT_A_REAL	REAL	
SHIFT_A_LREAL	LREAL	
SHIFT_A_TIME	TIME	
SHIFT_A_DATE	DATE	
SHIFT_A_TOD	TOD	
SHIFT_A_DT	DT	

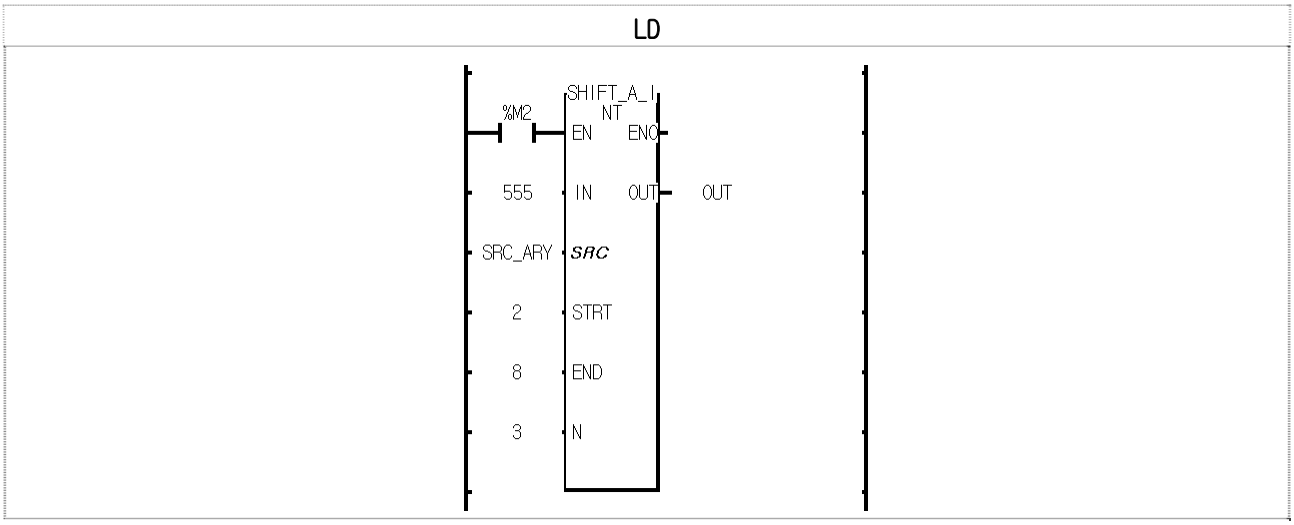
■ 에러

▷ STRT 또는END값이 SRC 어레이의 원소 개수 범위를 벗어나면 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ 에러발생시 SRC는 변하지 않고 출력은 각 변수타입의 초기화값(i.e. INT=0, TIME=T#0S)이 출력됩니다.

■ 프로그램 예

LD



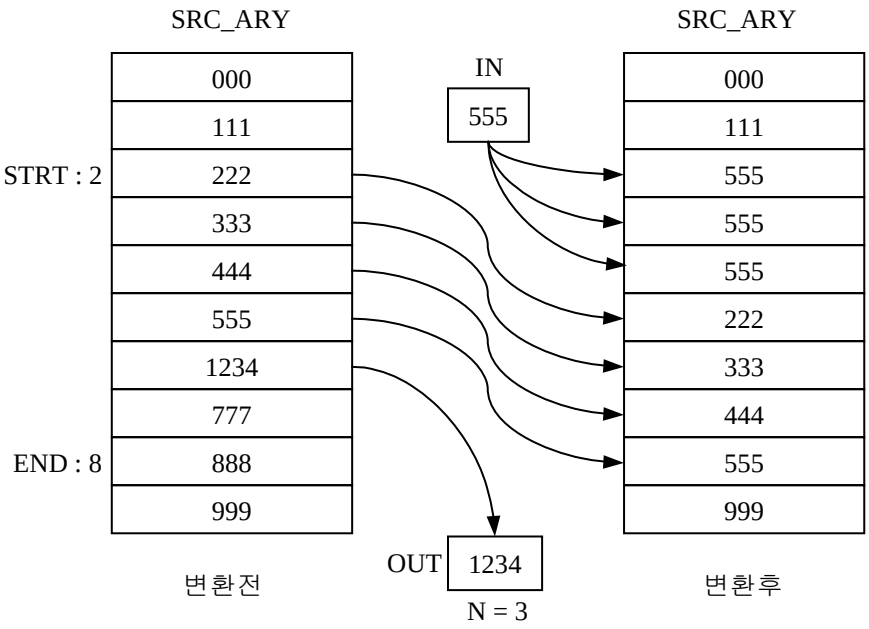
(1)입력 조건(%M2)이 On되면, SHIFT_A_INT 펑션이 실행됩니다.

(2)입출력 변수로 선언된 SRC_ARY의 인덱스 2의 원소부터 인덱스 8의 원소까지 SHIFT동작이 일어납니다.

(3)지정된 영역의 원소들이 3번 SHIFT됩니다.

(4)SHIFT된 후에 비워지는 원소들 즉, 어레이 인덱스 2의 원소부터 3개의 원소가 입력값 555로 채워집니다.

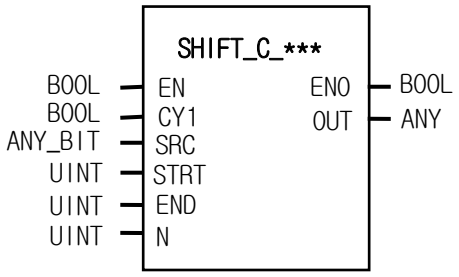
(5)출력 값에는 캐리(Carry) 출력에 해당하는 원소의 값 1234가 출력됩니다.


SRC_ARY
SRC_ARY
IN
555
000
111
222
333
444
555
1234
777
888
999
STRT : 2
END : 8
000
111
555
555
555
222
333
444
555
999
OUT
1234
N = 3
변환전
변환후

SHIFT_C_***

Shift with Carry

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

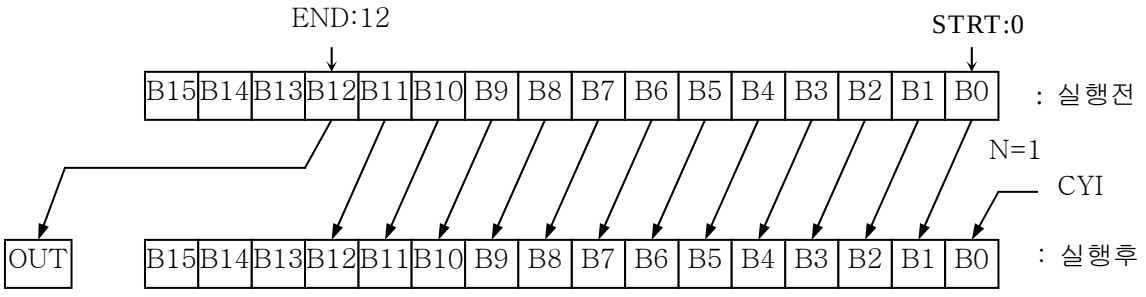
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 *CYI : Carry 입력* STRT : SRC의 비트열 중 Shift할 시작 bit 위치 END : SRC의 비트열 중 Shift할 끝 bit 위치 N : Shift할 비트수 **출력** ENO : 에러없이 수행되면 1을 출력 OUT : Carry 출력 **입출력** *SRC* : Shift할 변수

■ 기능

▷ SRC의 비트열중 지정된 범위의 원소들을 지정된 방향으로 이동합니다.

▷ 동작의 지정:

- 범위지정: STRT와 END로 이동할 비트의 범위를 지정합니다.
- 이동방향 및 횟수: STRT에서 END방향으로 지정된 횟수(N)만큼 Shift됩니다.
- 입력 데이터 지정: Shift하여 비워지는 위치를 입력 데이터(CYI)로 채웁니다.
- 출력: 데이터 조작의 결과는 *SRC* 에 지정된 ANY_BIT에 바뀌어 저장되며, END위치에서 Shift동작으로 밀려 나온 비트 데이터는 OUT으로 출력됩니다.



펑션	SRC 변수 타입	동작 설명
SHIFT_C_BYTE	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Shift시킵니다.
SHIFT_C_WORD	WORD	
SHIFT_C_DWORD	DWORD	
SHIFT_C_LWORD	LWORD	

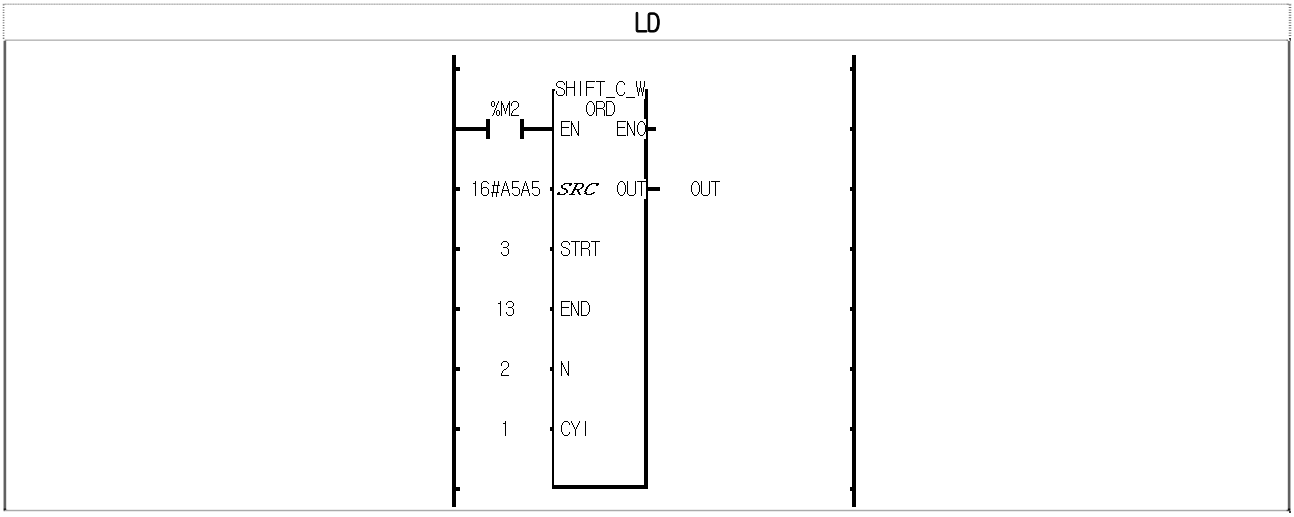
■ 에러

▷ STRT 와 END 값이 SRC 변수 타입의 비트 개수를 넘는 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ SRC의 값은 변화가 없습니다.

■ 프로그램 예

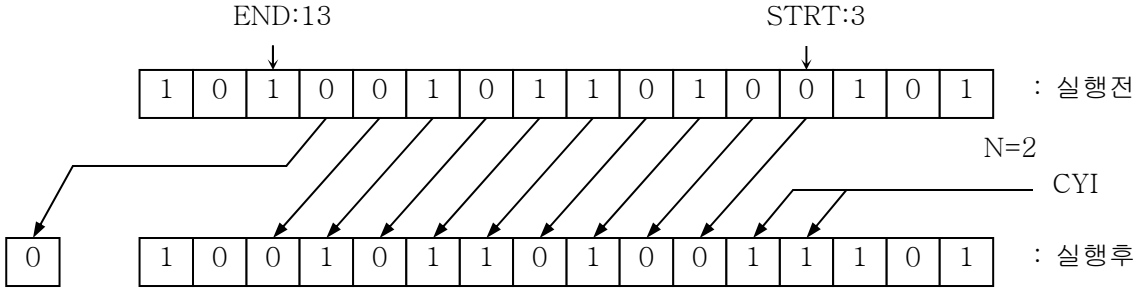
LD



(1)실행 조건(%M2)이 On되면, SHIFT_C_WORD 펑션이 실행됩니다.

(2)16#A5A5값이 STRT에서 END 방향으로 2비트 shift하고 shift 후에 비워지는 비트는 CYI값인 1로 채워집니다.

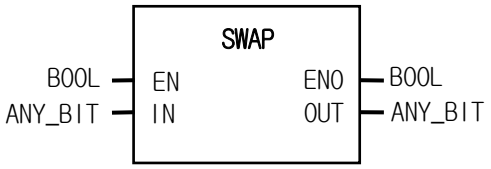
(3)Shift된 후의 값은 다시 SRC(16#969D)로 출력되고 2비트 shift되면서 밀려나온 값 00이 OUT으로 출력됩니다.



SWAP_***

데이터의 상위 하위 바꾸기

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 입력 **출력** ENO : EN값을 그대로 출력 OUT : Swap된 값

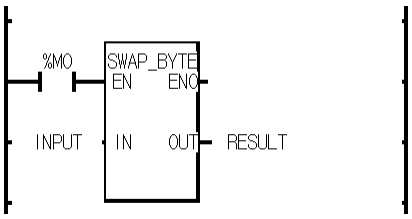
■ 기능

입력된 변수를 2개의 크기로 구분하여 상위와 하위를 서로 교환합니다.

펑 션	입력 타입	동작 설명
SWAP_BYTE	BYTE	BYTE 의 상하위 니블(Nibble)을 서로 교환하여 출력합니다.
SWAP_WORD	WORD	WORD 의 상하위 BYTE 를 서로 교환하여 출력합니다.
SWAP_DWORD	DWORD	DWORD 의 상하위 WORD 를 서로 교환하여 출력합니다.
SWAP_LWORD	LWORD	LWORD 의 상하위 DWORD 를 서로 교환하여 출력합니다.

■ 프로그램 예

LD



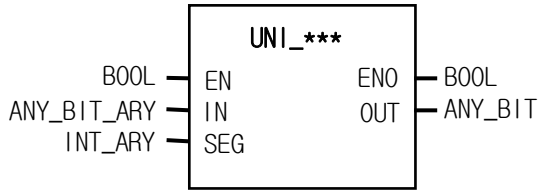
(1)실행조건(%M0)이 On 되면, SWAP_BYTE 펑션이 실행됩니다.

(2)펑션의 입력 변수 INPUT(BYTE 타입)=16#5F 일 경우, 펑션의 출력 변수 RESULT(BYTE 타입)=16#F5 가 됩니다.

UNI_***

데이터 결합(Union)

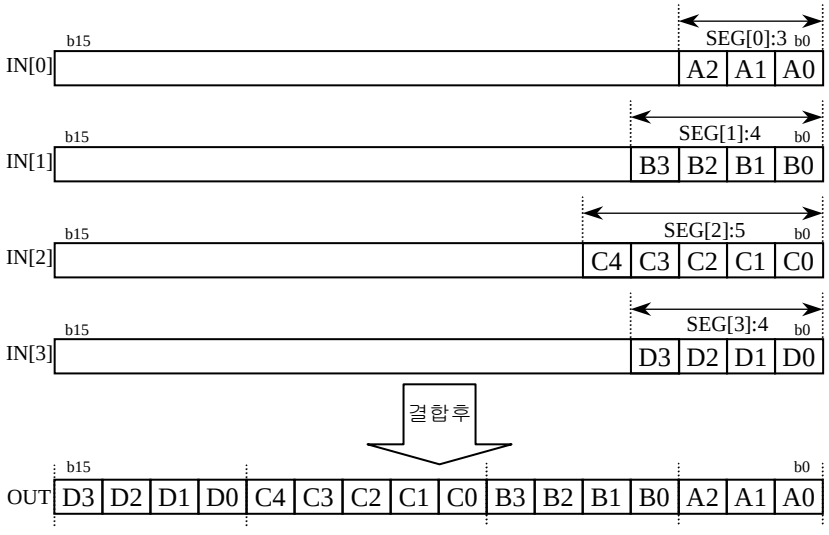
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : 입력 데이터 어레이 SEG : 데이터 결합 비트수 지정 어레이 **출력** ENO : 에러없이 실행되면 1을 출력 OUT : 결합된 데이터 출력

■ 기능

입력 데이터 어레이를 SEG 어레이에서 지정한 비트수 별로 하위 비트부터 지정된 비트수 만큼 결합하여 출력합니다.

펑션	입력 타입	출력 타입	동작 설명
UNI_BYTE	BYTE	BYTE	입력 어레이를 SEG가 지정하는 비트의 개수만큼씩 자른 후, 입력 어레이 타입과 동일한 하나의 값으로 묶어 출력합니다.
UNI_WORD	WORD	WORD	
UNI_DWORD	DWORD	DWORD	
UNI_LWORD	LWORD	LWORD	

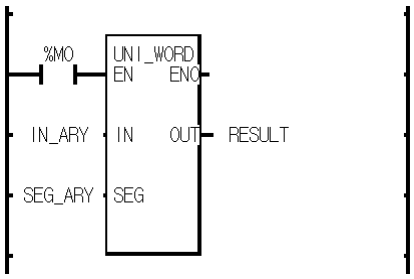


▷ SEG로 지정된 값들의 합이 입력 변수 타입의 비트수를 초과할 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

▷ IN과 SEG 어레이의 개수가 서로 다를 경우, 0이 출력되고 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On하면, UNI_WORD 펑션이 실행됩니다.
(2)입력 변수로 선언된 IN_ARY와 SEG_ARY의 값이 다음과 같을 경우

IN_ARY[0]	A3B5
IN_ARY[1]	B4C6
IN_ARY[2]	C5D7
IN_ARY[3]	D6E8

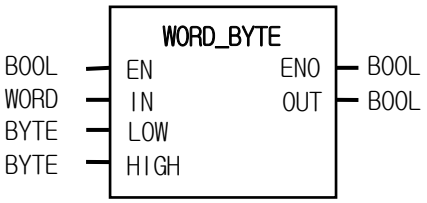
SEG_ARY[0]	3
SEG_ARY[1]	4
SEG_ARY[2]	7
SEG_ARY[3]	2

펑션이 실행된 후에 출력변수의 RESULT의 값은 2#00 1010111 0110 101 = 16#2BB5로 출력됩니다.

WORD_BYTE

WORD를 2개의 BYTE로 나눔

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

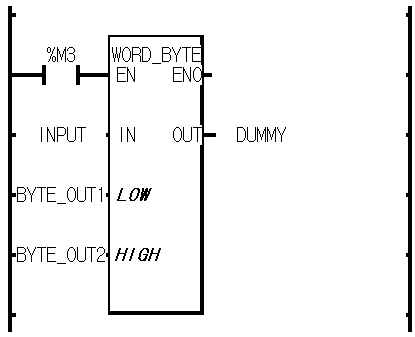
펑 션	설 명
	**입력** EN : 1일때 펑션 실행 IN : WORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : Dummy 출력 **입출력** LOW : 하위 BYTE 출력 HIGH : 상위 BYTE 출력

■ **기능**

▷ 하나의 워드를 2개의 바이트로 분산합니다.
LOW: 하위 바이트 출력, HIGH: 상위 바이트 출력

■ **프로그램 예**

LD

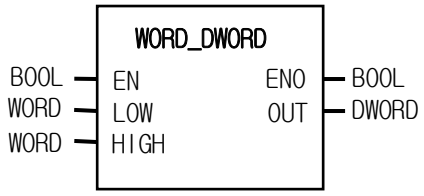


(1)실행조건(%M3)이 On 이면, WORD_BYTE 펑션이 실행됩니다.
(2)입력변수로 선언된 INPUT 값이 16#ABCD 일 때, 입출력 변수로 선언된 변수
BYTE_OUT1=16#CD
BYTE_OUT2=16#AB
값이 저장됩니다.

WORD_DWORD

2개의 WORD를 DWORD로 모음

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 LOW : 하위 WORD 입력 HIGH : 상위 WORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : DWORD 출력

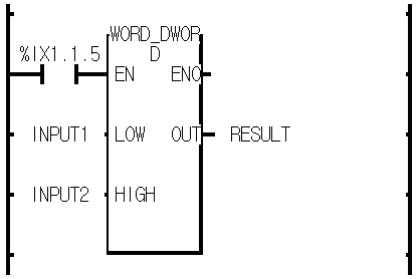
■ 기능

2개의 WORD를 하나의 DWORD로 조합합니다.

LOW: 하위 워드 입력, HIGH: 상위 워드 입력

■ 프로그램 예

LD



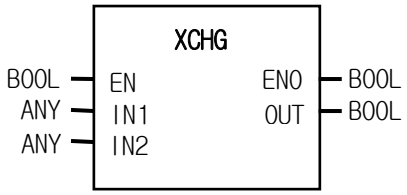
(1)실행조건(%IX1.1.5)이 On 되면, WORD_DWORD 펑션이 실행됩니다.

(2)입력변수로 선언된 INPUT1=16#10203040 이고 INPUT2=16#A0B0C0D0 일 때, 출력변수로 선언된 RESULT=16#A0B0C0D010203040 가 됩니다.

XCHG_ ***

2개의 입력변수 값을 서로 교환

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션	설 명
	**입력** EN : 1일때 펑션 실행 **출력** ENO : EN값을 그대로 출력 OUT : Dummy 출력 **입출력** IN1 : 입출력 1 IN2 : 입출력 2

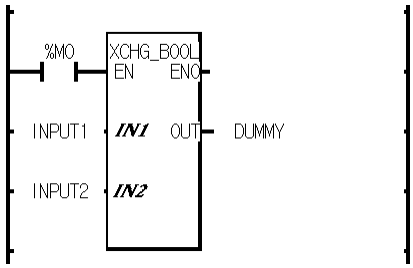
■ 기능

입력1과 입력2의 데이터를 교환합니다.

펑 션	입출력 타입	동작 설명
XCHG_BOOL	BOOL	두개의 BOOL입력 값을 서로 교환합니다.
XCHG_BYTE	BYTE	두개의 BYTE입력 값을 서로 교환합니다.
XCHG_WORD	WORD	두개의 WORD입력 값을 서로 교환합니다.
XCHG_DWORD	DWORD	두개의 DWORD입력 값을 서로 교환합니다.
XCHG_LWORD	LWORD	두개의 LWORD입력 값을 서로 교환합니다.
XCHG_SINT	SINT	두개의 SINT입력 값을 서로 교환합니다.
XCHG_INT	INT	두개의 INT입력 값을 서로 교환합니다.
XCHG_DINT	DINT	두개의 DINT입력 값을 서로 교환합니다.
XCHG_LINT	LINT	두개의 LINT입력 값을 서로 교환합니다.
XCHG_USINT	USINT	두개의 USINT입력 값을 서로 교환합니다.
XCHG_UINT	UINT	두개의 UINT입력 값을 서로 교환합니다.
XCHG_UDINT	UDINT	두개의 UDINT입력 값을 서로 교환합니다.
XCHG_ULINT	ULINT	두개의 ULINT입력 값을 서로 교환합니다.
XCHG_REAL	REAL	두개의 REAL입력 값을 서로 교환합니다.
XCHG_LREAL	LREAL	두개의 LREAL입력 값을 서로 교환합니다.
XCHG_TIME	TIME	두개의 TIME입력 값을 서로 교환합니다.
XCHG_DATE	DATE	두개의 DATE입력 값을 서로 교환합니다.
XCHG_TOD	TOD	두개의 TOD입력 값을 서로 교환합니다.
XCHG_DT	DT	두개의 DT입력 값을 서로 교환합니다.
XCHG_STRING	STRING	두개의 STRING입력 값을 서로 교환합니다.

■ 프로그램 예

LD



(1)실행조건(%M0)이 On 되면, XCHG_BOOL 펑션이 실행됩니다.

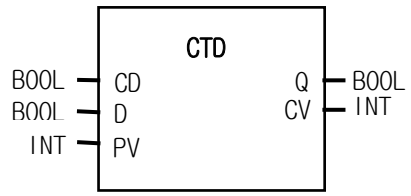
(2)펑션의 입력변수 INPUT1=0 이고 INPUT2=1 이면 펑션이 실행된 후 2 개의 입력값은 서로 바뀌어 출력 됩니다. 따라서, 펑션이 실행된 후 INPUT1=1 이고, INPUT2=0 이 됩니다.

2.3 기본 펑션블록 라이브러리

CTD

감산 카운터(펑션블록)

제 품 명	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CD : 다운_카운트(Down_Count) 펄스입력 LD : 설정값 입력(Load) PV : 설정값(Preset Value) **출력** Q : 카운트_다운(Count_Down) 출력 CV : 현재값(Current Value)

■ 기능

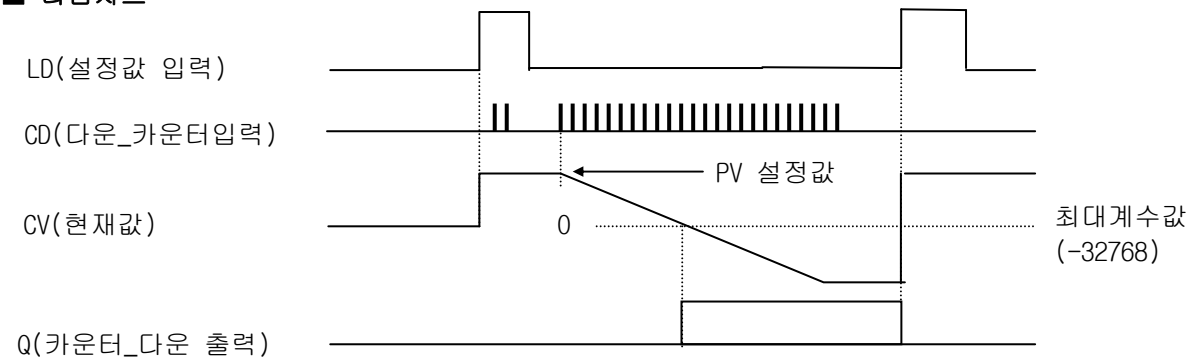
▷ 감산 카운터 펑션블록 CTD는 다운 카운터 펄스입력 CD가 0에서 1이되면, 현재값 CV가 이전값보다 1만큼 감소하는 카운터 입니다.

▷ 단, CV는 INT의 최소값인 -32768보다 클때만 감소하고, -32768이 되면 더이상 감소하지 않습니다.

▷ 설정값 입력 LD가 1이되면 현재값 CV에는 설정값 PV값이 로드 됩니다.(CV=PV)

▷ 출력 Q는 CV가 0이하 일때만 1이 됩니다.

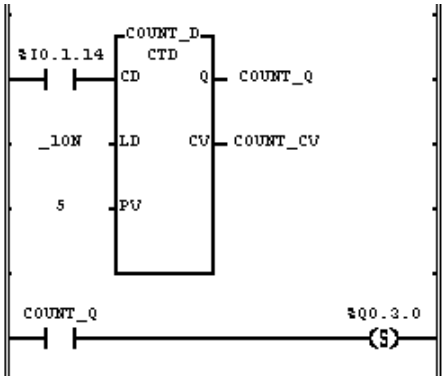
■ 타임차트



■ 프로그램 예

입력접점 %I0.1.14에 5번의 펄스가 들어오면, 출력접점 %Q0.3.0을 Set시키는 프로그램

LD

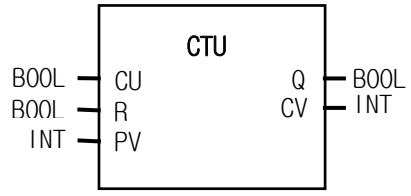


(1)CTD 펑션블록의 이름을 등록합니다.(COUNT_D)
(2)CD에 펄스입력이 들어올 접점 %I0.1.14를 입력합니다.
(3)PV를 CV로 로드시킬 사용자 플래그_1ON (첫스캔On)을 입력합니다.
(4)PV 값은 INT범위 (-32768~32767)내에서 5를 입력합니다.
(5)CV에는 임의의 출력변수 (COUNT_CV)를 설정하여 입력합니다.
(6)Q에는 임의의 출력변수 (COUNT_Q)를 설정 입력합니다.
(7)프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC로 쓰기를 합니다.
(8)쓰기를 완료하면 모드전환 (Stop →Run) 시킵니다.
(9)프로그램이 Run되면 PV값 5가 CV (Count_CV)로 로드됩니다.
(10)입력펄스가 입력접점 %I.0.1.14에 들어오면 현재값 CV (COUNT_CV)는 1만큼 줄어듭니다.
(11)입력접점에 5번의 펄스가 들어오면 현재값 CV는 0가 되고, Q (COUNT_Q)는 1이 됩니다.
(12)Q (COUNT_Q)가 1이되면 출력접점 %Q0.3.0이 Set 됩니다.

CTU

가산 카운터(펑션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CU : 업_카운트(Up_Count) 펄스입력 R : 리셋 입력(Reset) PV : 설정값 (Preset Value) **출력** Q : 업_카운트(Up_ Count) 출력 CV : 현재값(Current Value)

■ 기능

▷ 가산 카운터 펑션블록 CTU는 업 카운터 펄스입력 CU가 0에서 1이되면 현재값 CV가 이전값보다 1만큼 증가하는 카운터입니다.

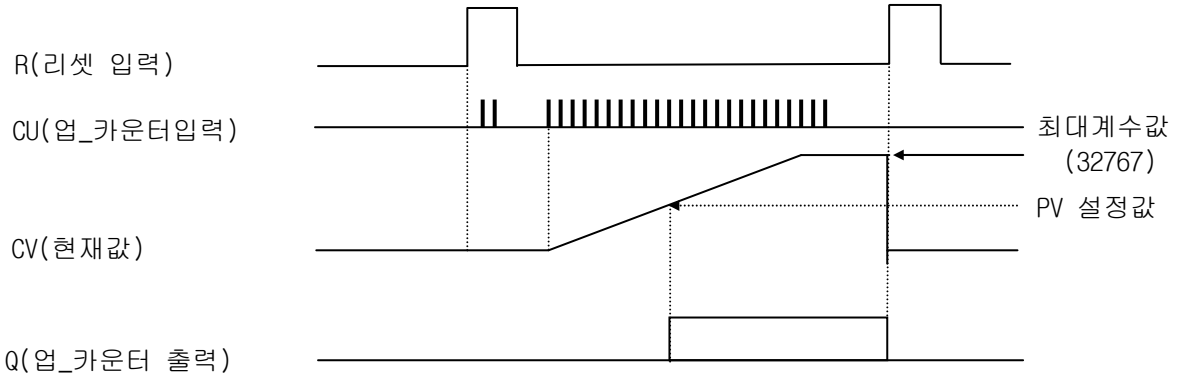
▷ 단, CV가 1NT의 최대값인 32767 미만일때만 증가하고, 32767이 되면 더이상 증가하지 않습니다.

▷ 리셋 입력 R이 1이되면 현재값 CV는 0으로 클리어 (Clear)됩니다.

▷ 출력 Q는 CV가 PV이상이 될때만 1이 됩니다.

▷ PV값은 CTU펑션블록을 수행시 설정값을 새롭게 가져와 연산합니다.

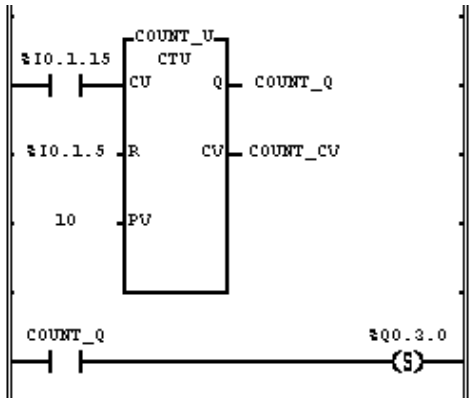
■ 타임차트



■ 프로그램 예

입력접점 %I0.1.15에 10번의 펄스가 들어오면, 출력접점 %Q0.3.1을 Set시키는 프로그램

LD

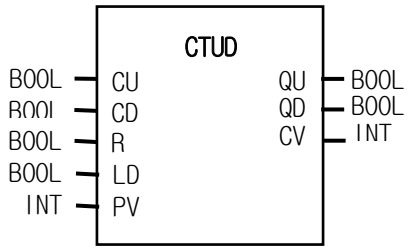


(1)CTU 펑션블록의 이름을 등록합니다.(COUNT_D)
(2)CU에 펄스 입력이 들어올 입력 접점 %I0.1.15를 입력합니다.
(3)PV에 10을 입력합니다.
(4)CV를 초기화시키는 R에는 임의의 입력 접점을 설정합니다.(%I0.1.5)
(5)CV에는 임의의 변수 (COUNT_CV)를 설정하여 입력합니다.
(6)Q에는 임의의 출력변수(COUNT_Q)를 설정 입력합니다.
(7)프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC로 쓰기를 합니다.
(8)쓰기를 완료하면 모드전환 (Stop →Run) 시킵니다.
(9)입력펄스가 입력접점 %I0.1.15에 들어오면 현재값 CV (COUNT_CV)는 1만큼 증가합니다.
(10)입력접점에 10번의 펄스가 들어오면 현재값 CV는 10이 되고, 설정값과 같게 되므로 출력Q(COUNT_Q)가 1이 됩니다.
(11)Q(COUNT_Q)가 1이되면 출력접점 %Q0.3.0이 Set 됩니다.

CTUD

가감산 카운터(펑션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CU : 업_카운트(Up_Count) 펄스입력 CD : 다운_카운트(Down_Count) 펄스입력 R : 리셋 입력(Reset) LD : 설정값 입력(Load) PV : 설정값(Preset Value) **출력** QU : 카운트_업(Count_UP) 출력 QD : 카운트_다운(Count_Down) 출력 CV : 현재값(Current Value)

■ 기능

▷ 가감산 카운터 펑션블록 CTUD는 업카운터 펄스 입력 CU가 0에서 1이 되면 현재값 CV가 이전값보다 1만큼 증가하고, 다운 카운터 펄스입력 CD가 0에서 1이되면 현재값 CV가 이전값보다 1만큼 감소하는 카운터 입니다.
단, CV가 1NT의 최소값인 -32768과 최대값인 32767 사이의 값을 가지며 최대, 최소값에 이르면 각각 더이상 증가, 감소하지 않습니다.

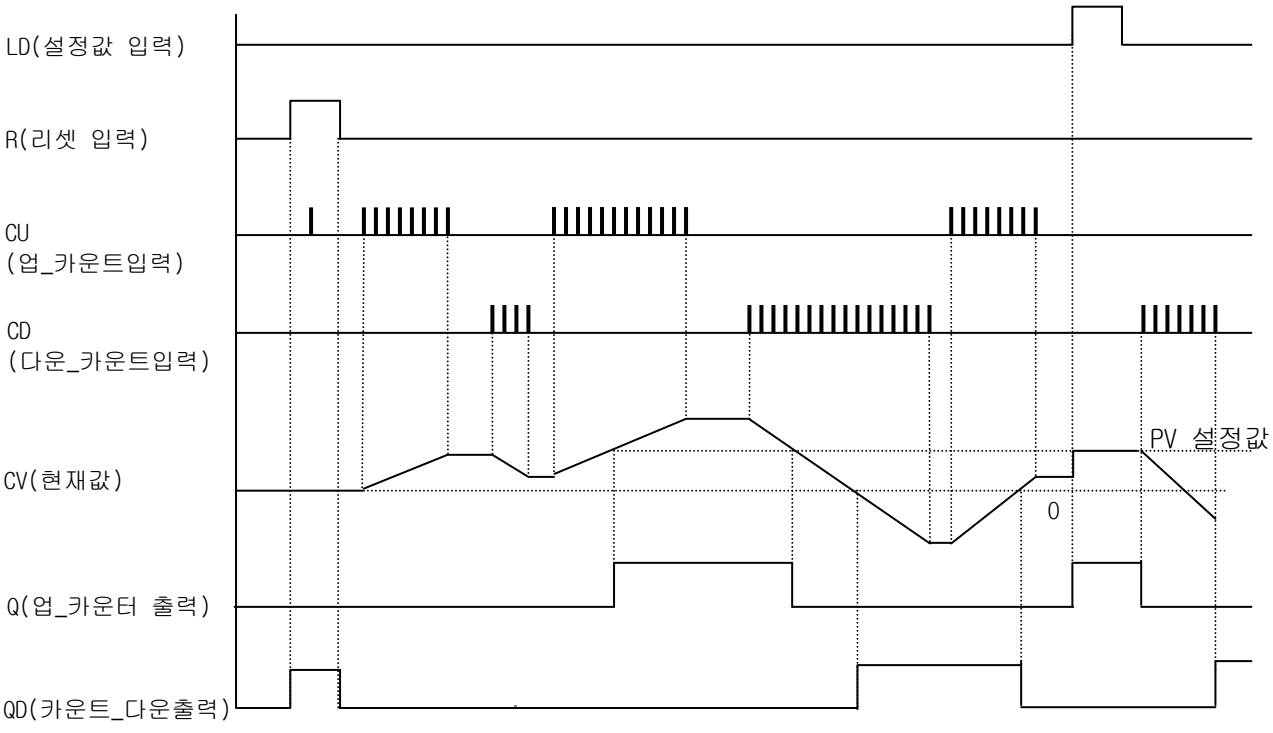
▷ 설정값 입력 LD가 1이되면 현재값 CV에는 설정값 PV값이 로드됩니다. (CV=PV)

▷ 설정값 입력 R이 1이되면 현재값 CV는 0으로 클리어(Clear) 됩니다. (CV=0)

▷ 출력 QV는 CV가 PV보다 크면 1이되고, QD는 CV가 0이하일때 1이 됩니다.

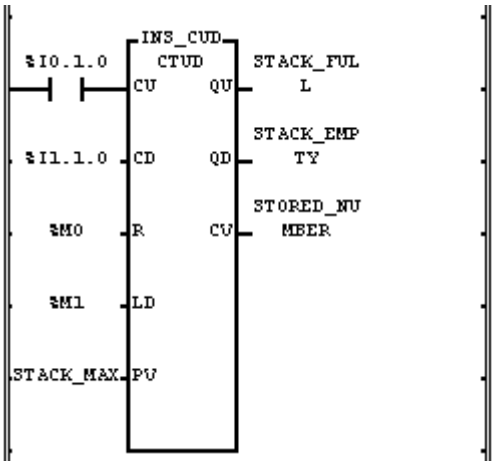
▷ 각 입력신호에 대해서 R > LD > CU > CD순으로 동작을 수행하며, 신호의 중복 발생시 우선 순위가 높은 동작 하나만 수행합니다.

■ 타임차트

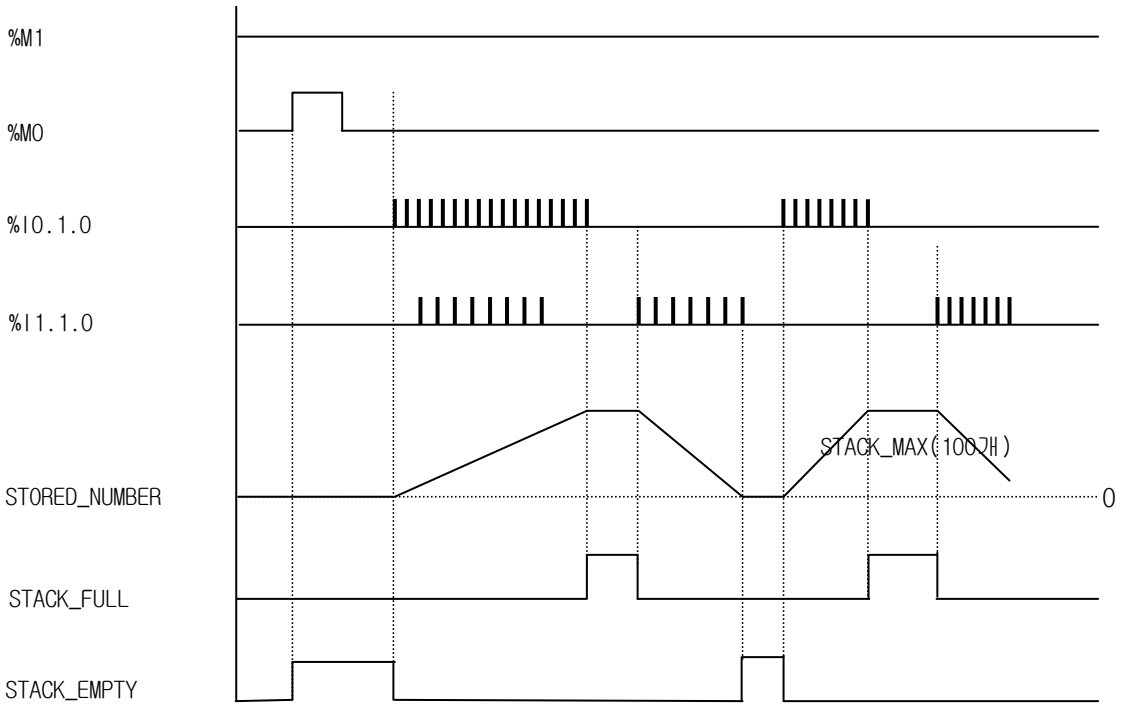


■ 프로그램 예

LD



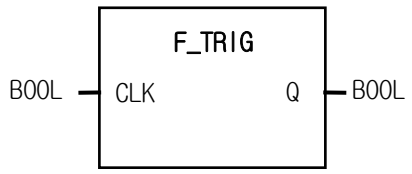
공정라인에서 중간 적재 완충부에 임시 저장될 수 있는 용량 STACK_MAX의 값이 100이고 적재부에 자재가 투입될 때마다 IN신호가 1이 되고, 적재부에서 자재가 빠져 나갈 때마다 OUT신호가 1이 되도록 되어 있다면, 운전을 시작하여 선 공정에서 완충부에 투입되는 속도가 다음 공정을 위해 빠져나가는 속도보다 빠를 때, 중간 적재부에 100개가 쌓이면 카운터의 상한 출력 QU가 1이 되어 STACK_FULL에 1이 출력됩니다. 반대로 중간 적재부에 남아 있는 것이 없으면 하한 출력 QD가 1이 되어 STACK_EMPTY에 1이 출력됩니다.
STORED_NUMBER에는 현재 적재부에 남아 있는 자재의 숫자가 표시됩니다.



F_TRIG

하강에지 검출(펑션블록)

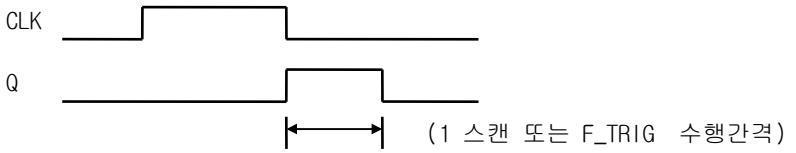
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CLK : 입력신호 **출력** Q : 하강 에지 검출결과

■ 기능

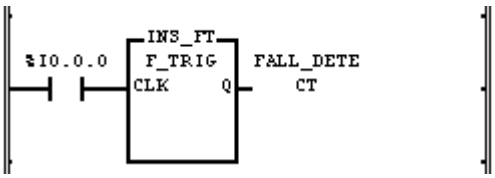
F_TRIG는 CLK에 연결된 입력의 상태가 1에서 0으로 변할 때 출력 Q를 1로 만들고 다음번 수행에서 0으로 만듭니다. 그 이외의 경우, 출력Q는 항상 0이 됩니다.

■ 타임 차트



■ 프로그램 예

LD

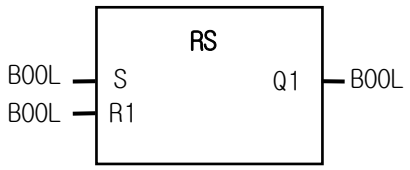


입력변수 %I0.0.0의 상태를 감시하여 1에서 0으로 변할 때에 출력변수FALL_DETECT에 1을 출력하고, 다음에 INS_FT수행시 FALL_DETECT에 0을 출력합니다.

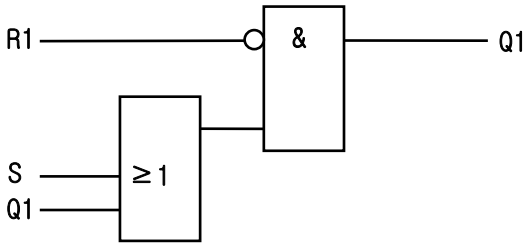
RS

Reset 우선 Bistable(평션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

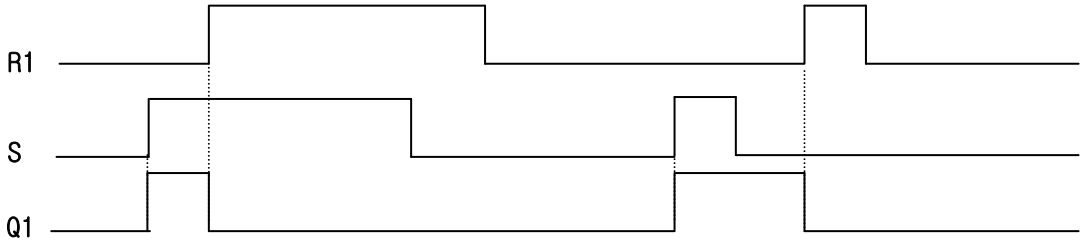
평 션 블 록	설 명
	**입력** R1 : Reset 조건 S : Set 조건 **출력** Q1 : 연산결과

■ 기능



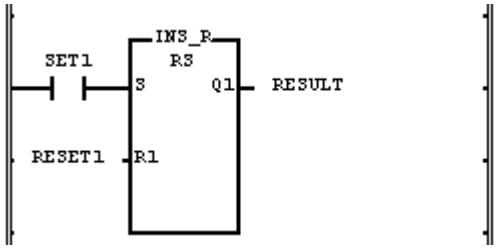
R1이 1이면, S와 관계없이 출력 Q1은 항상 0이 됩니다. 출력 Q1은 이전 상태를 유지하며, R1이 0이고 S가 1일 때 1이 됩니다. Q1의 초기상태는 0입니다.

■ 타임 차트



■프로그램 예

LD

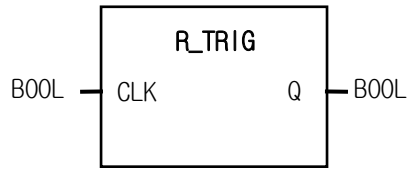


RESET1을 Reset조건으로, SET1을 Set조건으로 하여 연산 결과를 RESULT에 출력합니다.
동작 조건은 위의 타임 차트에서 R1을 RESET1, S를 SET1으로, Q1을 RESULT로 대치하여 보십시오.
(1) 입력 변수로 선언된 SET1과 RESET1이 동시에 On되면 출력 변수 RESULT는 1이 됩니다.
(2) 입력 변수로 선언된 RESET1이 On되면 RESULT로 선언된 출력 변수값은 0이 됩니다.
(3) 입력 변수로 선언된 SET1과 RESET1이 동시에 On되면 출력 변수 RESULT는 0가 됩니다.

R_TRIG

상승에지 검출(펑션블록)

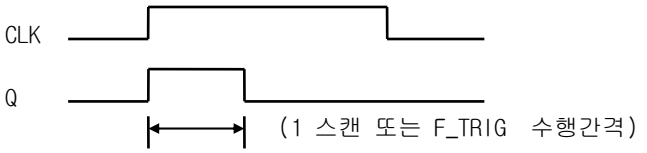
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CLK : 입력신호 **출력** Q : 상승에지 검출결과

■ 기능

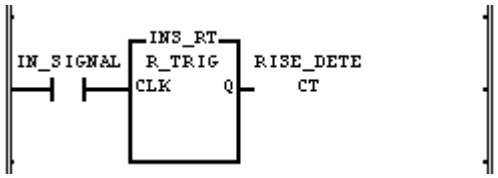
R_TRIG는 CLK에 연결된 입력의 상태가 0에서 1로 변할 때 출력Q를 1로 만들고 다음번 수행에서 0으로 만듭니다. 그 이외의 경우, 출력Q는 항상 0이 됩니다.

■ 타임 차트



■ 프로그램 예

LD

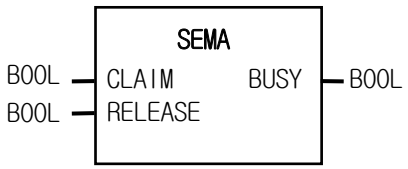


(1) 입력변수로 선언된IN_SIGNAL의 상태를 감시하여 0에서 1로 변할 때, RISE_DETECT에 1을 출력하고 다음에 INS_RT수행시 RISE_DETECT에 0을 출력합니다.

SEMA

시스템 자원 배분(펑션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CLAIM : 자원독점 요구신호 RELEASE : 해제신호 **출력** BUSY : 요구자원의 취득불가신호(대기)

■ 기능

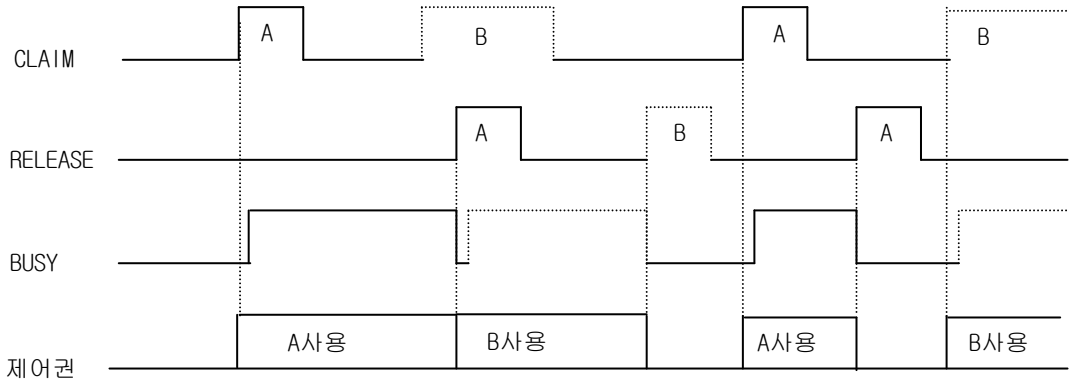
이 펑션 블록은 시스템 자원에 대한 독점적 제어권을 취득하는 데 사용합니다.
SEMA펑션을 수행했을 때 (CLAIM = 1 또는 0, RELEASE = 0 값으로) 타 프로그램에서 자원을 사용중이면 BUSY 가 1이 됩니다. 자원의 제어권을 획득하고자 할 때에는 CLAIM = 1, RELEASE = 0의 값으로 SEMA를 계속 기동 하여 BUSY가 0이 될 때를 기다립니다. BUSY가 0이 되면 해당 자원에 대하여 제어를 하고 제어 동작이 끝난 후에는 CLAIM = 0, RELEASE = 1의 값으로 SEMA를 한 번 수행하여 제어권을 양도합니다.
(이때 CLAIM = 0, RELEASE = 1의 값으로 SEMA를 수행하는 제어 양도 동작은 반드시 현재 제어권을 가지고 있는 프로그램에서만 수행해야 합니다.)

- SEMA의 인스턴스는 자원을 요구하는 프로그램에서 공통으로 액세스할 수 있도록 글로벌 영역에 설정하여야 합니다.
- 동일한 자원을 요구하는 각각의 프로그램은 우선 순위상 동일한 우선순위로 지정되어 있어야 합니다.
- GM1 멀티 CPU모듈 간에는 사용할 수 없습니다.
- SEMA펑션 블록의 내부 수행 구조

```
VAR  X : BOOL : = 0 ;  END_VAR
   BUSY : = X ;
   IF  CLAIM  THEN  X : = 1 ;
   ELSIF  RELEASE  THEN  BUSY : = 0; X : = 0 ;
   END_IF
```

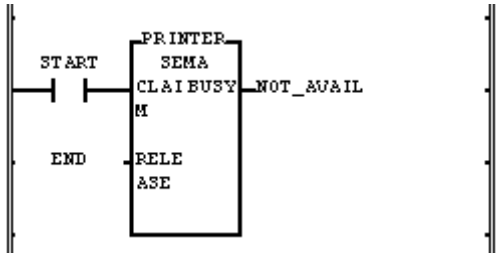
■ 타임차트

프로그램 블록 A와 프로그램 블록B에서 동일한 자원에 대한 액세스권을 주고 받는 경우,



■ 프로그램 예

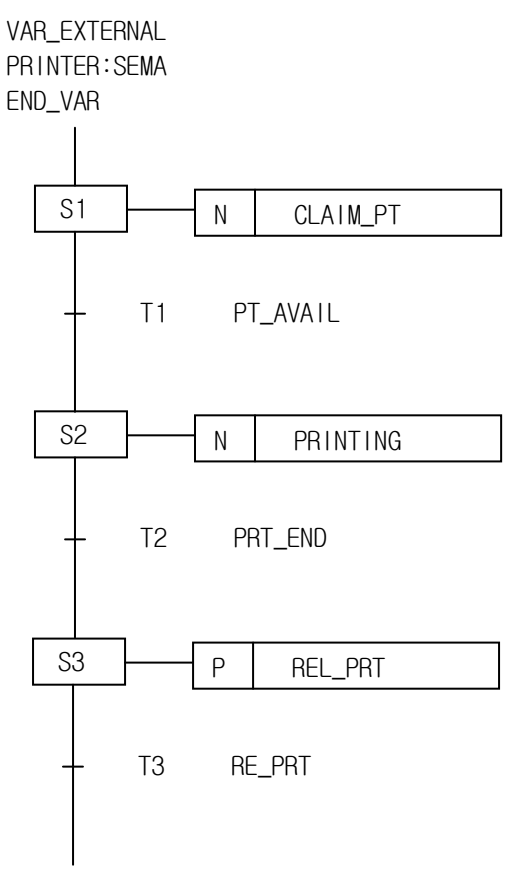
LD



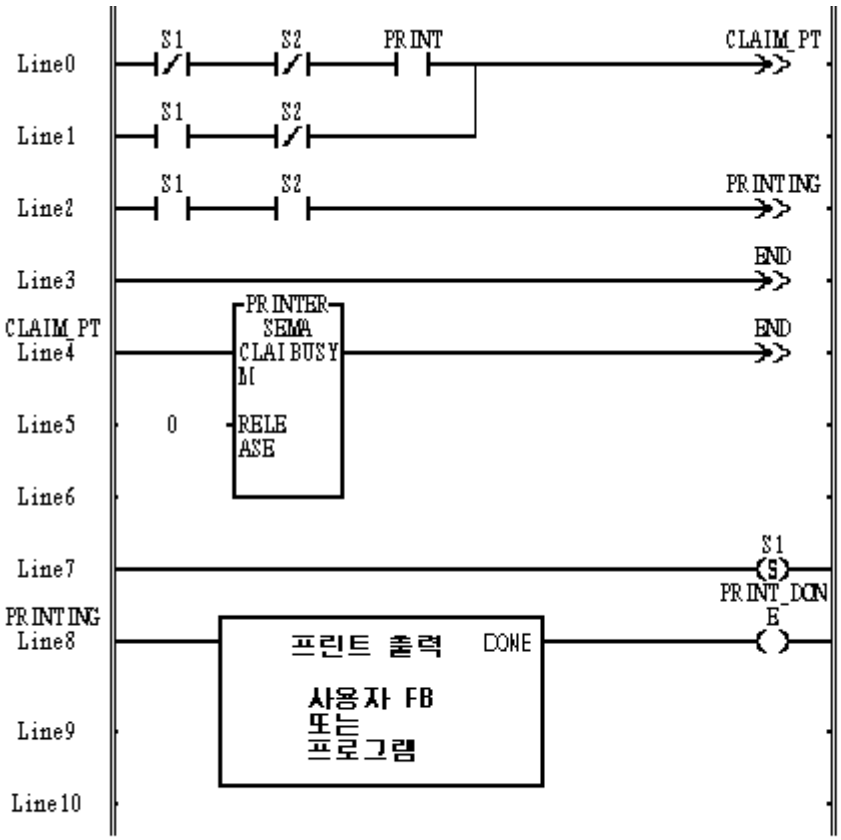
PLC시스템에 부착되어 있는 프린터로 서로 다른 프로그램 블록에서 출력을 내고자 할 때 인스턴스 'PRINTER'를 글로벌로 선언하고 각각의 프로그램에서 'PRINTER'로 명명된 SEMA펑션블록을 이용하여 프린터의 제어권을 쉽게 제어할 수 있습니다.

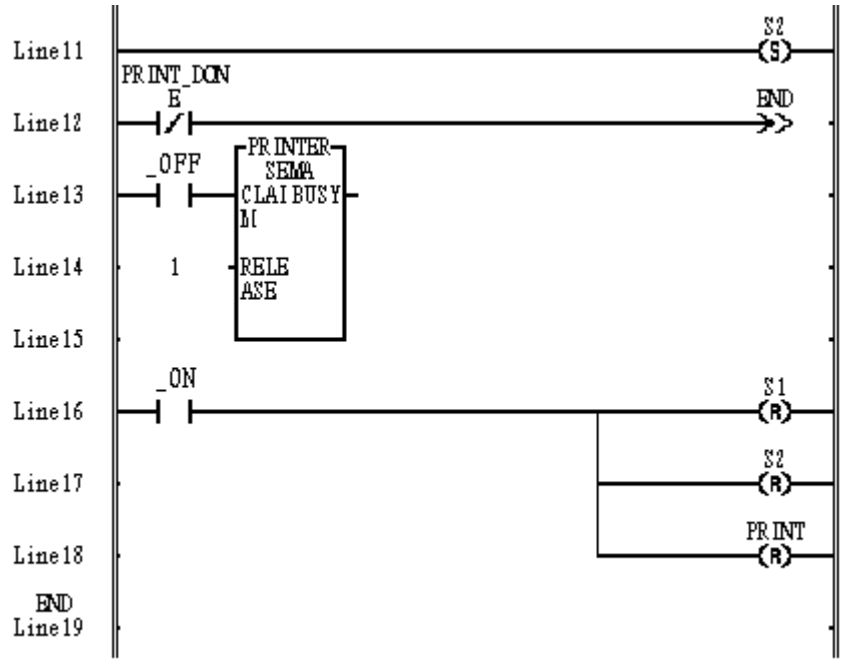
프린터에 출력이 필요한 시점에서 START는 1, END는 0인 상태에서 'PRINTER' SEMA를 수행하여 프린터의 제어권을 요구했을 때 다른 프로그램 블록에서 이미 프린터를 사용하고 있다면 BUSY신호가 1이 되어 NOT_AVAIL에 1이 출력됩니다. 만약 다른 블록에서 프린터를 사용하고 있지 않다면 BUSY는 0인 상태가 되어 그것을 신호로 프린터에 출력을 내는 프로그램을 기동시키면 됩니다.

프린트 동작이 모두 끝난 후에는 START는 0, END는 1인 상태로 'PRINTER' SEMA를 실행하여 타 부분에서 프린터의 제어권을 가져갈 수 있도록 해제하여 주어야 합니다.



S1	CLAIM_PT;프린터 제어권 요구
	CAL SEMA PRINTER CLAIM:= 1 RELEASE:= 0
T1	PT_AVAIL;프린터 제어권 획득 체크
	LDN PRINTER.BUSY ST TRANS
S2	PRINTING;프린터 출력
	프린터 제어 프로그램 프린터 완료시 PRINT_DONE:=1
T2	PRT_END;프린터 완료 체크
	LD PRINTER_DONE ST TRANS
S3	REL_PRT;프린터 제어권 양도
	CAL SEMA PRINTER CLAIM:= 0 RELEASE:= 1
T3	RE_PRT;프린터 재요구
	LD PRT_REQ ST TRANS

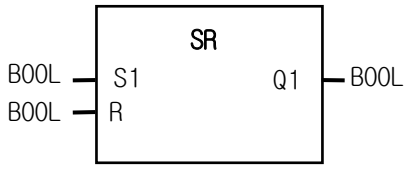

Line0
S1
S2
PRINT
CLAIM_PT
Line1
S1
S2
Line2
S1
S2
PRINTING
Line3
END
CLAIM_PT
Line4
PRINTER_SEMA
CLAIBUSY
M
END
Line5
0
RELEASE
Line6
Line7
S1
PRINTING
Line8
프린트 출력
DONE
PRINT_DONE
사용자 FB
또는
프로그램
Line9
Line10


Line11
S2
PRINT_DONE
Line12
END
_OFF
Line13
PRINTER_SEMA
CLAIBUSY
M
Line14
1
RELEASE
Line15
_ON
Line16
S1
Line17
S2
Line18
PRINT
END
Line19

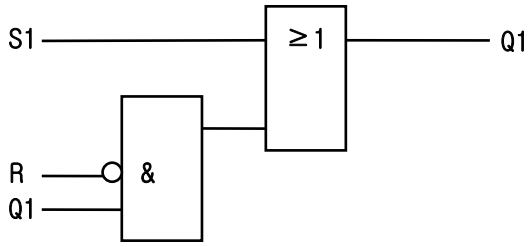
SR

Set우선 Bistable (펑션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

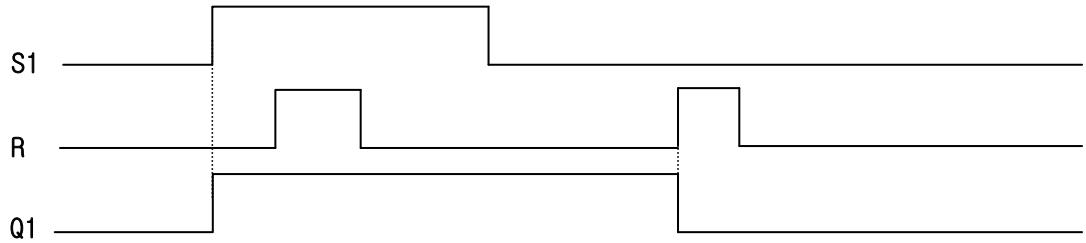
펑 션 블 록	설 명
	**입력** S1 : Set 조건 R : Reset 조건 **출력** Q1 : 연산결과

■ 기능



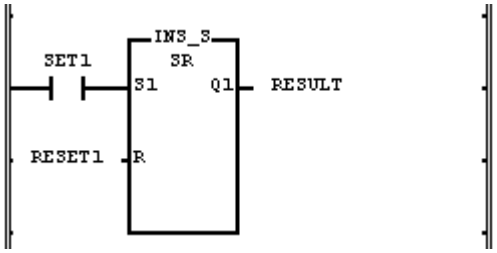
S1이 1이 되면, R과 관계없이 출력 Q1은 항상 1이 됩니다. 출력 Q1은 이전 상태를 유지하며, S1이 0이고 R이 1일 때 0이 됩니다. Q1의 초기상태는 0입니다.

■ 타임 차트



■ 프로그램 예

LD

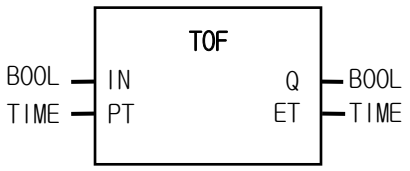


(1) 입력변수로 선언된 SET1이 On되면, 출력변수로 선언된 RESULT같이 1이 됩니다.
(2) 출력변수로 선언된 RESULT값이 0이 되도록 하려면, 입력변수로 선언된 SET1이 Off되고, RESELT1이 On되어야 합니다.

TOF

OFF 딜레이 타이머(펑션블록)

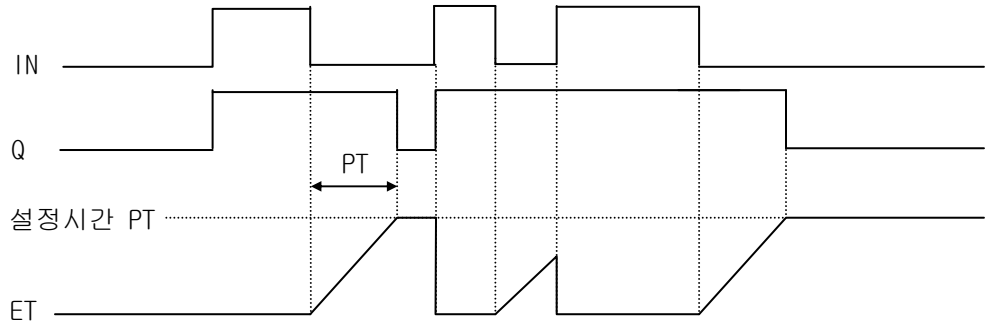
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

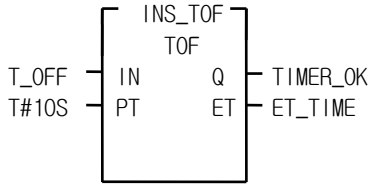
IN이 1이 되면, Q가 1이 되고, IN이 0이 된 후부터 PT에 의해서 지정된 설정시간이 경과한 후 Q가 0이 됩니다.
IN이 0이 된 후 경과시간이 ET로 출력됩니다.
만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 1이 되면, 경과시간은 다시 0으로 됩니다.

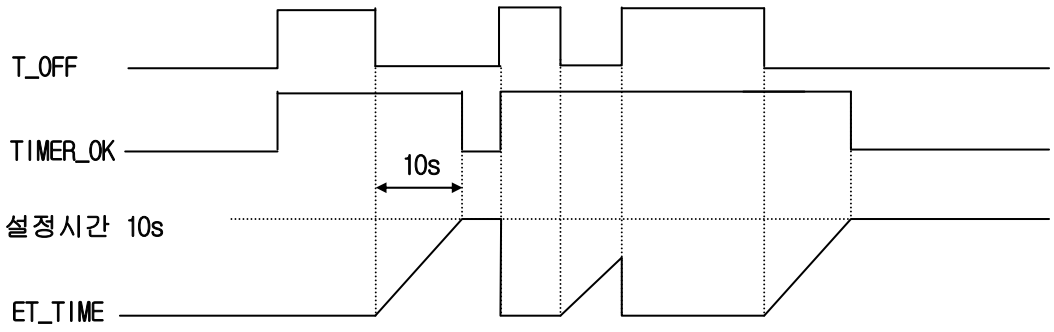
■ 타임 차트



■ 프로그램 예

LD



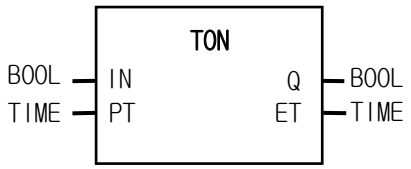


입력변수로 설정된 T_OFF가 1이 되면, 출력변수 TIMER_OK에 1이 출력되고 T_OFF가 0이 된 후 10초 후에 TIMER_OK가 0이 됩니다. T_OFF가 0이 된 후 10초 이내에 다시 1이 되면 타이머는 다시 초기상태가 됩니다. 타이머의 시간 측정값은 ET_TIME로 출력됩니다.

TON

ON 딜레이 타이머(펑션블록)

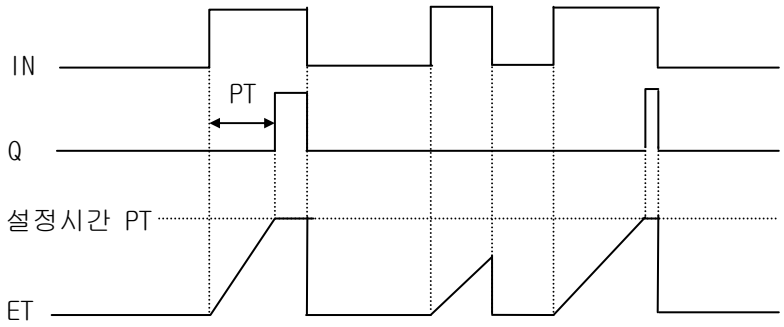
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

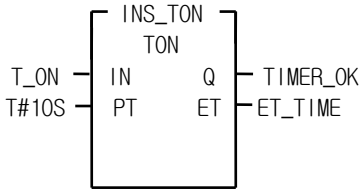
IN이 1이 된 후 경과시간이 ET로 출력됩니다.
만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 0이 되면, 경과시간은 0으로 됩니다.
Q가 1이 된 후 IN이 0이 되면, Q는 0이 됩니다.

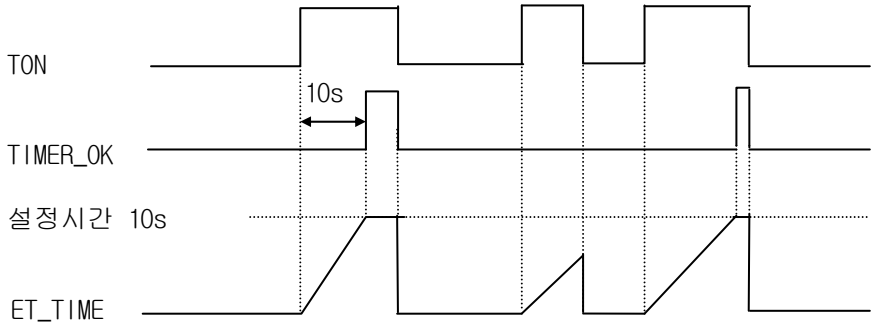
■ 타임 차트



■ 프로그램 예

LD





(1)입력변수 T_ON이 1이 된 후 10초가 경과한 후 출력 변수TIMER_OK가 1이 됩니다.
(2)입력변수T_ON이 1이 된 후 경과 시간이 출력 변수ET_TIME로 출력됩니다.
(3)만일 경과시간 ET_TIME이 설정시간 10초에 도달하기 전에 T_ON이 0이 되면, 경과 시간 ET_TIME은 0으로 됩니다.
(4)TIMER_OK가 1이 된 후 T_ON이 0이 되면, TIMER_OK는 0이되고 경과 시간 ET_TIME도 0으로 됩니다.

TP

펄스 타이머(펑션블록)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

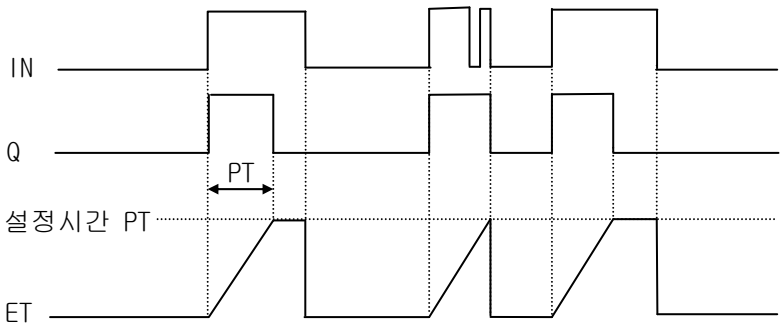
펑 션 블 록	설 명
	**입력** IN : 타이머 기동조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간 (Elapsed Time)

■ 기능

IN이 1이 되면 PT에 의해서 지정된 설정시간 동안만 Q가 1이 되고, ET가 PT에 도달하면 자동으로 0이 됩니다.

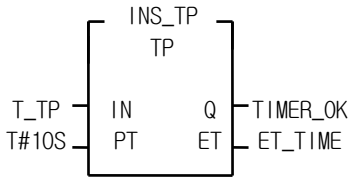
경과시간 ET는 IN이 1이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 0이 될 때 0의 값이 됩니다. ET가 증가할 동안은 IN이 0이 되거나 재차 1이 되어도 영향이 없습니다.

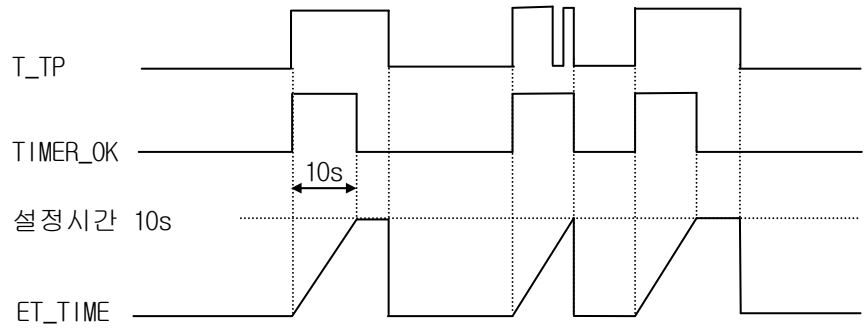
■ 타임 차트



■ 프로그램 예

LD





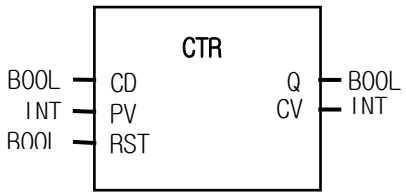
(1) 입력변수 T_TP가 0에서 1이 된 후 10초 동안 TIMER_OK는 1이 됩니다. 일단 타이머가 가동된 후에는 10초 동안 T_TP신호의 변화는 무시됩니다.
(2) ET_TIME값은 증가 후 T#10S에서 멈춥니다. 그리고 T_TP가 0이 될 때 0으로 됩니다.

2.4 응용 펑션블록 라이브러리

CTR

Ring 카운터

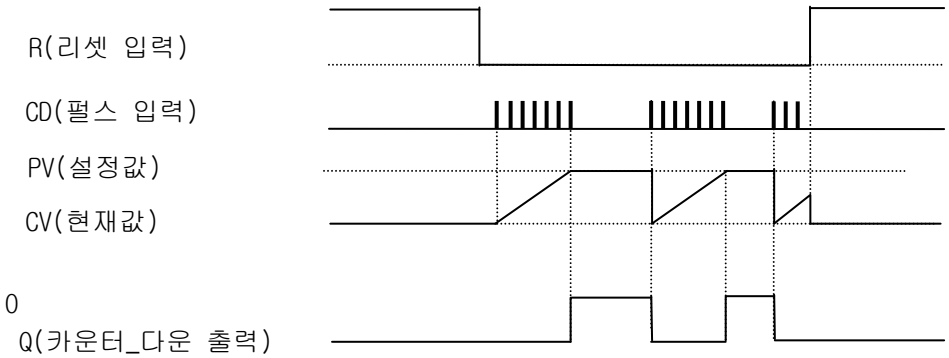
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** CD : 링 카운트(Ring Count) 펄스 입력 PV : 설정값(Preset Value) RST : 리셋 입력(Reset) **출력** Q : 링 카운트(Ring Count) 출력 CV : 현재값(Current Value)

■ 기능

▷ 링 카운터 CTR 펑션블록은 펄스 입력 CD가 0에서 1이 될 때마다 현재값 CV를 +1하고 현재값 CV가 설정값 PV에 도달한 후 CD가 다시 0에서 1로 되면 현재값은 1로 됩니다.
▷ 현재값이 설정값에 도달하면 출력은 1이 됩니다.
▷ 현재치가 설정치 미만이거나 Reset 조건이 1이 되면 출력은 0이 됩니다.

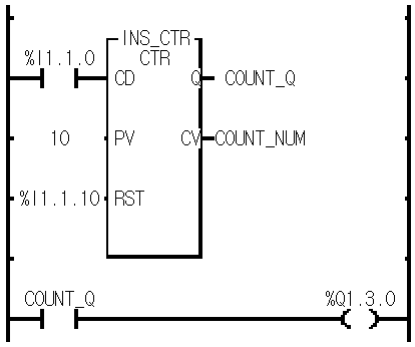
■ 타임 차트



■ 프로그램 예

입력 접점 %I1.1.0에 10번의 펄스가 들어올 때마다, 출력접점 %Q1.3.1을 ON시키는 프로그램

LD

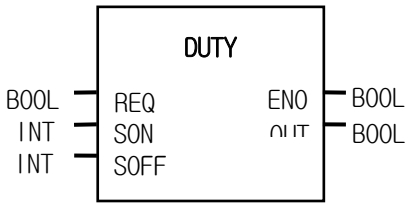


(1)CTR 펑션 블록의 이름을 등록합니다(INS_CTR)
(2)CD에 펄스 입력이 들어올 입력 접점 %I1.1.0을 입력합니다.
(3)PV에 10을 입력합니다.
(4)CV를 초기화 시키는 RST에는 임의의 입력 접점을 설정합니다.(%I1.1.10)
(5)CV에는 임의의 변수(COUNT_NUM)를 설정하여 입력합니다.
(6)Q에는 임의의 출력변수(COUNT_Q)를 설정 입력합니다.
(7)프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC로 쓰기를 합니다.
(8)쓰기를 완료하면 모드전환(Stop → Run)시킵니다.
(9)입력펄스가 입력접점 %I1.1.0에 들어오면 현재값 CV(COUNT_NUM)은 1만큼 증가합니다.
(10)입력접점에10번의 펄스가 들어오면 현재값 CV는10이 되고,설정값과 같게 되므로 출력Q(COUNT_Q)가 1이 됩니다.
(11)Q(COUNT_Q)가 1이 되면 출력 접점 %Q1.3.0은 ON됩니다.
(12)다시 한번의 입력 펄스가 입력 접점 %1.1.0에 들어오면 Q(COUNT_Q)는 0이 되고 출력 접점인 %Q1.3.0은 OFF됩니다.

DUTY

스캔 지정 On/Off

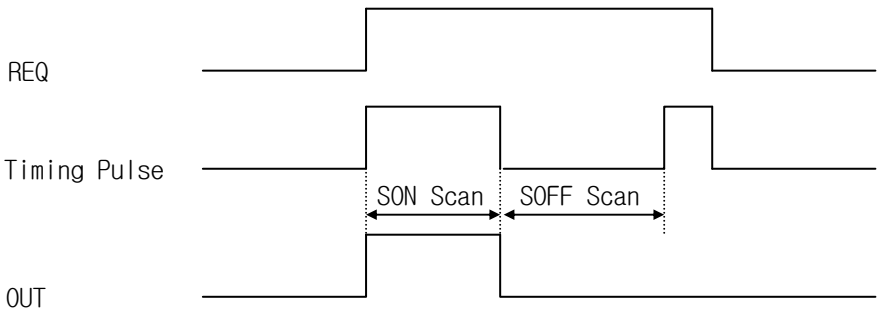
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** REQ : 펑션 블록 실행 요구 SON : On될 Scan 수 SOFF : Off될 Scan 수 **출력** DONE : REQ가 On이고 SON,SOFF두 입력변수가 0보다 작지 않으면 1을 출력 OUT : On Scan Time동안 1을 출력

■ 기능

▷ DUTY FB는 REQ가 ON된 시점부터 SON Scan동안 On, SOFF Scan동안 Off하는 펄스를 발생시킵니다.
▷ SON = 0이면 출력은 항상 OFF가 됩니다.
▷ SON > 0, SOFF = 0이면 출력은 항상 ON이 됩니다.
▷ REQ가 OFF면 출력은 OFF됩니다.
▷ SON < 0이거나 SOFF < 0이면, DONE은 OFF되고 OUT = 0이 됩니다.

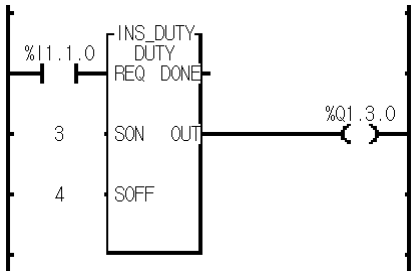
■ 타임 차트



■ **프로그램 예**

입력접점 %I1.1.0이 SET되어 있으면, 3번의 스캔 타임 동안 출력 접점 %Q1.3.0을 ON시키고, 4번의 스캔타임 동안 출력 접점 %Q1.3.0을 OFF시키는 프로그램

LD

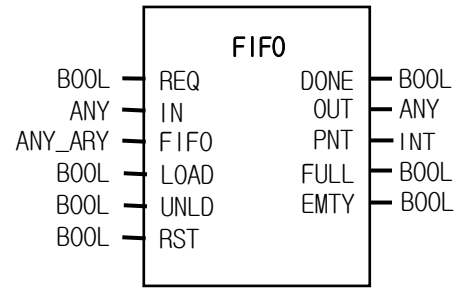


(1)DUTY 펑션 블록의 이름을 등록합니다(DUTY_C)
(2)REQ에 펑션 블록을 실행시킬 입력 접점 %I1.1.0을 입력합니다.
(3)SON에 3을 입력합니다.
(4)SOFF에 4를 입력합니다.
(5)OUT에 출력 접점 %Q1.3.0을 입력합니다.
(6)프로그램의 작성이 완료되면, 컴파일을 실행하고 PLC로 쓰기를 수행합니다.
(7)쓰기를 완료하면 모드전환(Stop → Run) 시킵니다.
(8)프로그램이 실행되면 3번의 스캔 타임동안 출력접점 %Q1.3.0이 ON되고, 4번의 스캔 타임동안에는 출력접점 %Q1.3.0이 OFF되는 동작을 반복합니다.

FIFO_***

FIFO 스택에 값을 Load/Unload
(선입 선출)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

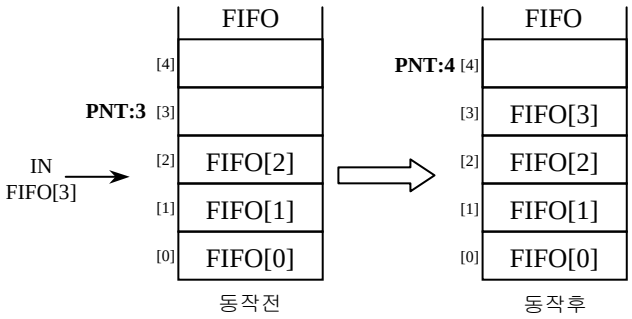
펑 션 블 록	설 명
	**입력** REQ : 펑션 블록 실행 요구 IN : FIFO 스택에 저장할 변수 또는 상수값 LOAD : ON이면 입력 모드 UNLD : ON이면 출력 모드 RST : POINTER VALUE 리셋 **출력** DONE : 최초 동작 후 1을 유지 OUT : 출력 모드일 경우 FIFO 스택으로부터 나온 값을 출력 PNT : FIFO 스택에 입력된 값에 대한 Pointer FULL : FIFO 스택이 가득차면 1을 출력 EMTY : FIFO 스택에 아무런 값도 저장되어 있지 않으면 1을 출력 **입출력** FIFO : FIFO 스택으로 사용되는 어레이

■ 기능

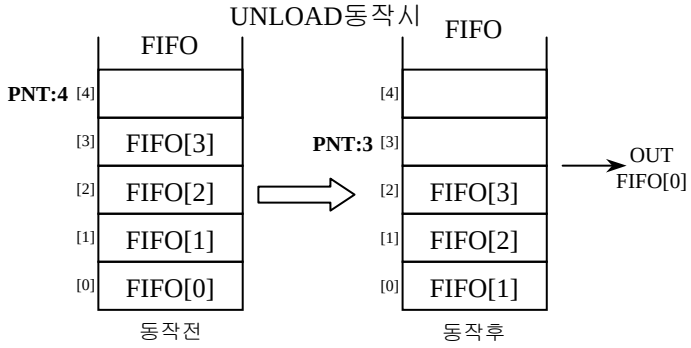
▷ FIFO 펑션블록은 IN값을 FIFO에 Load 또는 FIFO로부터 값을 Unload합니다.
▷ 입력모드 지정과 출력모드 지정이 동시에 ON되면 입출력을 동시에 수행합니다.
▷ FIFO로부터 값을 Unload하면 스택의 최하위 원소가 출력되고, 나머지 값들은 Shift되며, PNT값이 하나 줄고, PNT가 위치한 곳은 0으로 클리어(clear)됩니다.
▷ RST가 입력되면 PNT는 0으로 초기화되고, EMTY 플래그가 ON되며, FIFO 스택의 모든 값은 0으로 클리어(clear)됩니다.
▷ 스택의 개수는 입출력 변수 FIFO에 지령되는 입력시 어레이의 개수가 됩니다.
▷ 정전시 또는 전원 Off시에도 입력된 값들을 유지하기 위해서는 FIFO 어레이 변수와 FIFO 펑션블록 인스턴스를 모두 RETAIN으로 설정하여야 합니다.
▷ 리셋 동작은 REQ 요구가 없어도 동작이 가능합니다.
▷ PNT는 다음번 Load 동작시 IN값이 입력될 위치를 나타냅니다. 또는 Load되어 있는 전체 개수를 나타낸다고 볼 수 있습니다.
▷ 입력 모드일 경우 OUT 출력은 0이 됩니다.

펑 션	FIFO 변수 타입	동작 설명
FIFO_Q	BOOL	BOOL 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_B	BYTE	BYTE 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_W	WORD	WORD 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_DW	DWORD	DWORD 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_LW	LWORD	LWORD 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_SI	SINT	SINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_I	INT	INT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_DI	DINT	DINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_LI	LINT	LINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_USI	USINT	USINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_UI	UINT	UINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_UDI	UDINT	UDINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_ULI	ULINT	ULINT 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_R	REAL	REAL 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_LR	LREAL	LREAL 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_TM	TIME	TIME 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_DAT	DATE	DATE 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_TOD	TOD	TOD 타입 DATA에 대한 FIFO동작을 수행합니다.
FIFO_DT	DT	DT 타입 DATA에 대한 FIFO동작을 수행합니다.

LOAD동작시

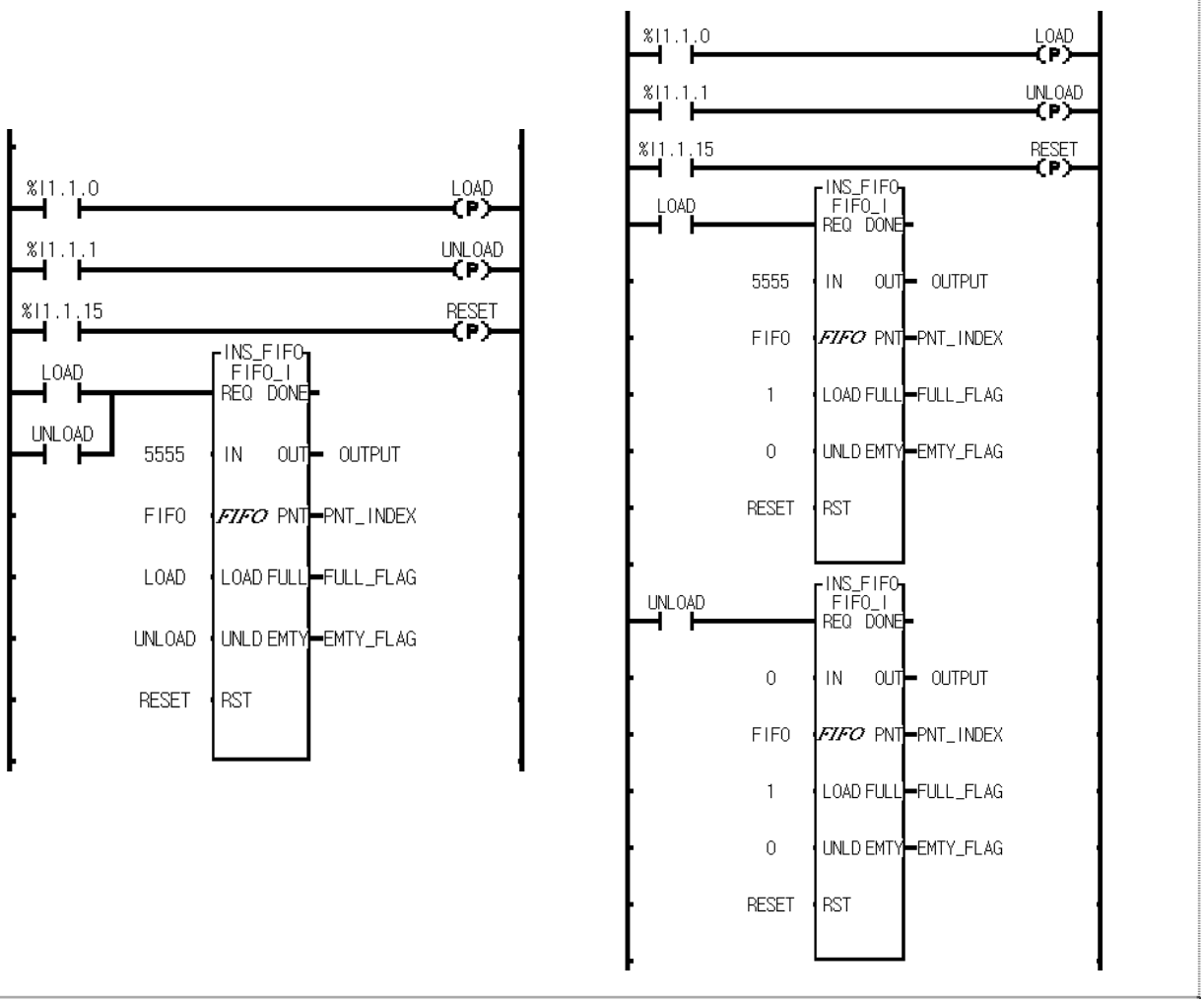


UNLOAD동작시



■ 프로그램 예

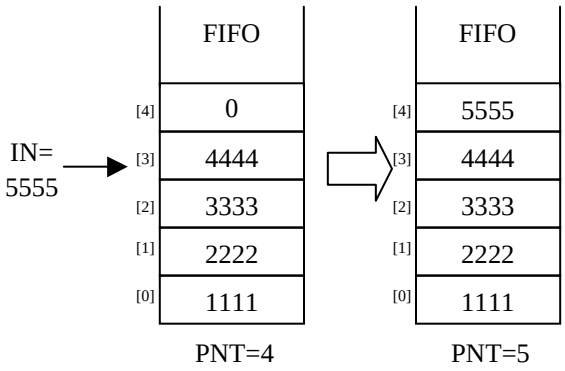
LD



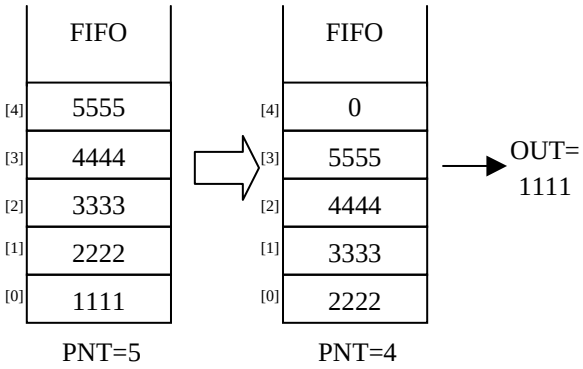
FIFO_*** 펑션 블록은 위의 그림과 같이 2가지 방법으로 사용이 가능합니다. 위에서 예를 든 2가지 방법은 동일한 동작을 수행합니다. 왼쪽의 예제는 하나의 펑션블록 만을 사용하여 입력과 출력을 동시에 수행할 수 있도록 한 프로그램이고, 오른쪽의 예제는 입력동작을 위한 펑션블록과 출력동작을 위한 펑션블록을 따로 작성하여 입출력 동작을 다른 위치에서 동작하도록 작성한 프로그램입니다. 단 이때 주의할 점은 인스턴스 이름이 같도록 설정해야 하는 점입니다.

(1)입력 조건(%I1.1.0, %I1.1.1, %1.1.1.15)이 성립되면 FIFO_INT가 실행됩니다.
(2)입력접점 %I1.1.00이 On되면 Load동작을 수행합니다. 5555가 FIFO스택에 입력되고 PNT_INDEX가 1 증가합니다.
(3)입력접점 %I1.1.10이 On되면 Unload동작을 수행합니다. FIFO 스택으로부터 1111이 출력되고 PNT_INDEX가 1 감소합니다.
(4)입력접점 %I1.1.15가 On되면 Reset 동작을 수행합니다. FIFO 스택의 모든 값이 0으로 클리어(Clear)되고, PNT_INDEX가 0으로 초기화되며, EMTY_FLAG가 On됩니다.

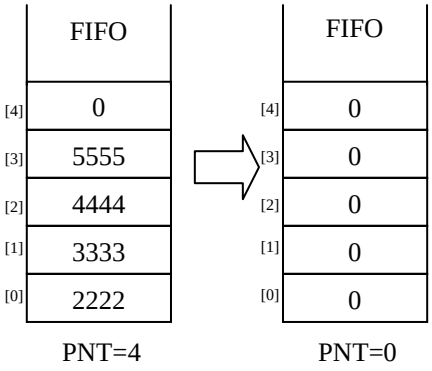
LOAD동작시(%I1.1.0이 On되면)



UNLOAD동작시(%I1.1.1이 On되면)



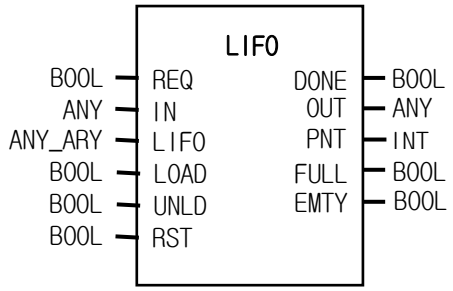
RESET동작시(%I1.1.15가 On되면)



LIFO_***

LIFO 스택에 값을 Load/Unload
(후입 선출)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

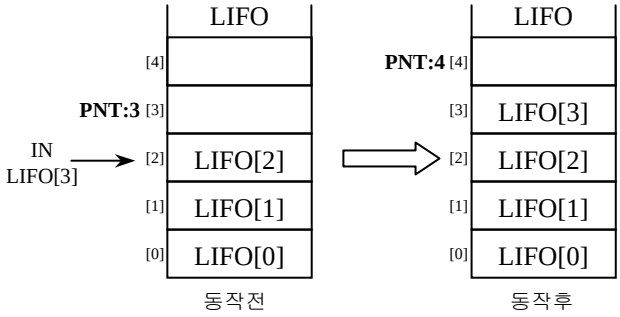
펑 션 블 록	설 명
	**입력** REQ : 펑션 블록 실행 요구 IN : LIFO 스택에 저장할 변수 또는 상수값 LOAD : ON이면 입력 모드 UNLD : ON이면 출력 모드 RST : POINTER VALUE 리셋 **출력** DONE : 최초 동작후 1을 유지 OUT : 출력 모드일 경우 LIFO 스택으로부터 나온 값을 출력 PNT : LIFO 스택에 입력된 값에 대한 Pointer FULL : LIFO 스택이 가득차면 1을 출력 EMTY : LIFO 스택에 아무런 값도 저장되어 있지 않으면 1을 출력 **입출력** LIFO : LIFO 스택으로 사용되는 어레이

■ 기능

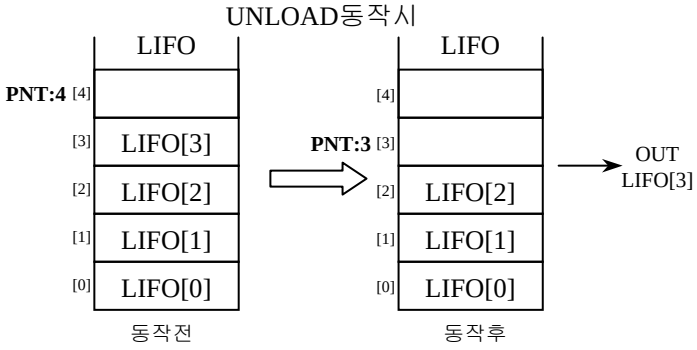
▷ LIFO 펑션블록은 IN값을 LIFO에 Load 또는 LIFO로부터 값을 Unload합니다.
▷ 입력모드 지정과 출력모드 지정이 동시에 ON되면 입력된 값이 바로 출력됩니다.
▷ LIFO로부터 Unload 동작이 수행되면 Unload된 값은 출력된 후 스택으로부터 삭제되고 0으로 초기화 됩니다.
▷ RST가 입력되면 PNT는 0으로 초기화되고, EMTY 플래그가 On되며, LIFO 스택의 모근 값은 0으로 클리어(clear)됩니다.
▷ 스택의 개수는 LIFO에 지정되는 어레이의 개수가 됩니다.
▷ 정전시 또는 전원 Off시에도 입력된 값들을 유지하기 위해서는 LIFO 어레이 변수와 LIFO 펑션블록 Instance를 모두 RETAIN으로 설정하여야 합니다.
▷ 리셋 동작은 REQ 요구가 없어도 동작이 가능합니다.
▷ PNT는 다음번 Load동작시 IN값이 입력될 위치를 나타냅니다. 또는 Load되어 있는 전체 개수를 나타낸다고 볼 수 있습니다.
▷ 입력모드일 경우 OUT출력은 0이 됩니다.
▷ Load 및 Unload신호가 동시에 입력되면 IN값이 그대로 OUT 으로 출력됩니다.

펑 션	FIFO 변수 타입	동작 설명
LIFO_Q	BOOL	BOOL 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_B	BYTE	BYTE 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_W	WORD	WORD 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_DW	DWORD	DWORD 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_LW	LWORD	LWORD 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_SI	SINT	SINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_I	INT	INT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_DI	DINT	DINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_LI	LINT	LINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_USI	USINT	USINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_UI	UINT	UINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_UDI	UDINT	UDINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_ULI	ULINT	ULINT 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_R	REAL	REAL 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_LR	LREAL	LREAL 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_TM	TIME	TIME 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_DAT	DATE	DATE 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_TOD	TOD	TOD 타입 DATA에 대한 LIFO동작을 수행합니다.
LIFO_DT	DT	DT 타입 DATA에 대한 LIFO동작을 수행합니다.

LOAD동작시

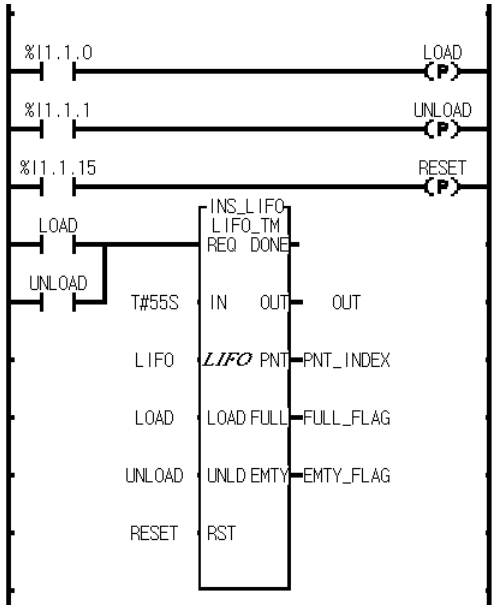


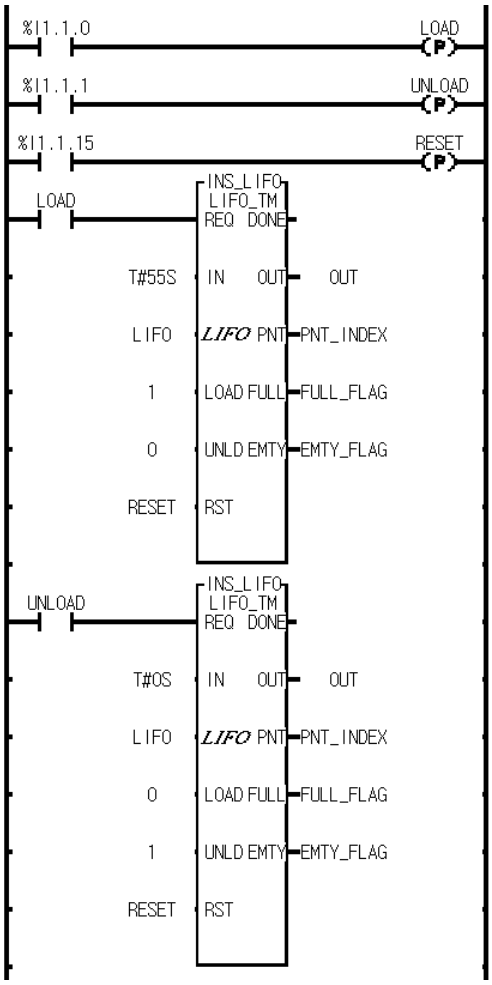
UNLOAD동작시



■ 프로그램 예

LD

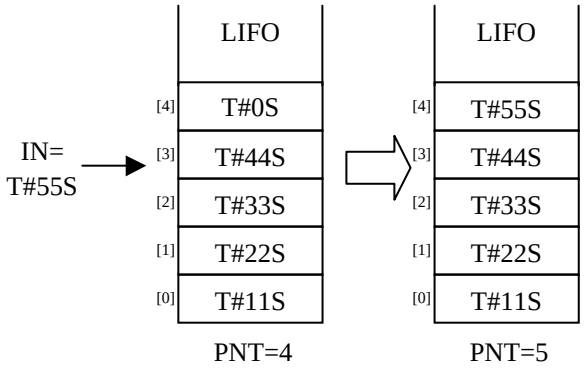




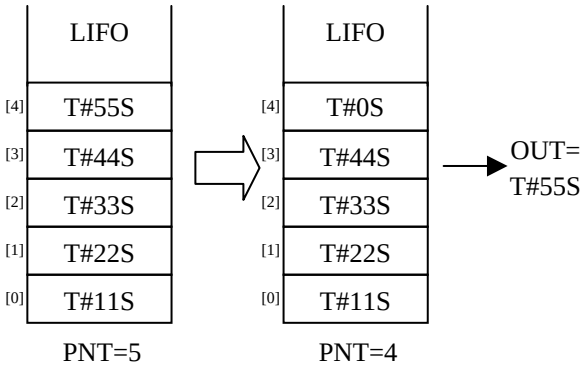
LIFO_*** 펑션 블록은 위의 그림과 같이 2가지 방법으로 사용이 가능합니다. 위에서 예를 든 2가지 방법은 동일한 동작을 수행합니다. 왼쪽의 예제는 하나의 펑션블록을 사용하여 입력과 출력을 동시에 수행할 수 있도록 한 프로그램이고, 오른쪽의 예제는 입력을 위한 펑션블록과 출력을 위한 펑션블록을 따로 작성하여 입출력 동작을 다른 위치에서 동작하도록 작성한 프로그램입니다. 단 이때 주의할 점은 인스턴스 이름이 같도록 설정해야 하는 점입니다.

(1)입력 조건(%I1.1.0, %I1.1.1, %1.1.1.15)이 성립되면 LIFO_TM이 실행됩니다.
(2)입력접점 %I1.1.0이 On되면 Load동작을 수행합니다. T#55S가 FIFO스택에 입력되고 PNT_INDEX가 1 증가합니다.
(3)입력접점 %I1.1.1이 On되면 Unload동작을 수행합니다. FIFO 스택으로부터 T#11S가 출력되고 PNT_INDEX가 1 감소합니다.
(4)입력접점 %I1.1.15가 On되면 Reset 동작을 수행합니다. FIFO 스택의 모든 값이 T#0S으로 클리어(Clear)되고, PNT_INDEX가 0으로 초기화되며, EMTY_FLAG가 On됩니다.

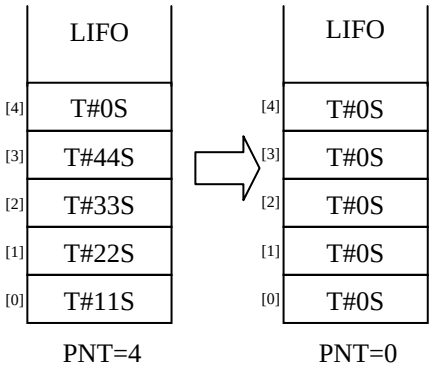
LOAD동작시(%I1.1.0이 On되면)



UNLOAD동작시(%I1.1.1이 On되면)



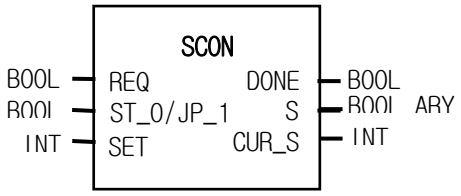
RESET동작시(%I1.1.15가 On되면)



SCON

스텝 컨트롤러(순차스텝 및 스텝점프)

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** REQ : 1일때 펑션 블록 실행 S/O : 0이면 SET 동작을 지정하고, 1이면 OUT동작을 지정 SET : 스텝의 번호(0~99) **출력** DONE : 펑션블록 실행이 에러없이 종료된 경우 ON되며, 에러가 발생하거나 펑션 블록 실행 요구가 없으면 OFF된다. S : Set된 bit array CUR_S : 현재 스텝 번호를 출력

■ 기능

▷ 순차작업 조의 설정

- 펑션블록의 인스턴스 이름이 하나의 순차작업 조의 이름이 됩니다.
  (펑션블록 선언 예: S00, G01, 제조1
  스텝 접점 예: S00.S[1], G01.S[1], 제조1.S[1])

▷ SET 동작일 경우(ST_0/JP_1 = 0)

- 동일 조 내에서 바로 이전의 스텝 번호가 On 되었을때 현재 스텝 번호가 On됩니다.
- 현재 스텝 번호가 On되면 자기 유지되어 입력 접점이 Off되어도 On되어진 상태를 유지합니다.
- 입력 조건 접점이 동시에 On되어도 한 조 내에서는 한 스텝 번호만이 On됩니다.
- Sxx.S[0]가 On되면 모든 SET 출력이 Clear됩니다.

▷ JUMP 동작일 경우(ST_0/JP_1 = 1)

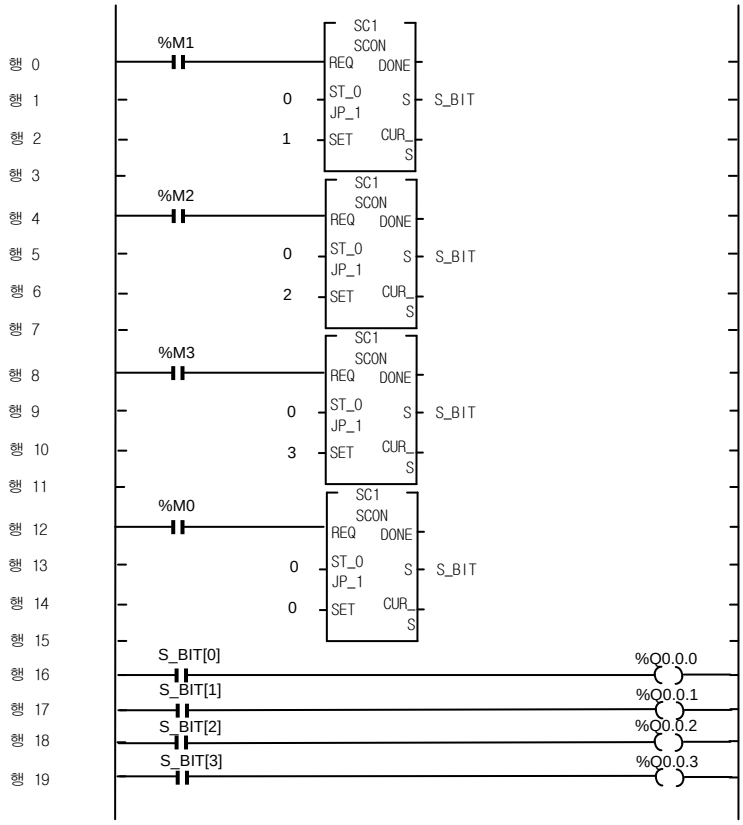
- 동일 조 내에서 입력조건 접점이 다수가 On 하여도 한 개의 스텝 번호만 On합니다.
- 입력 조건이 동시에 On하면 나중에 프로그램 된 것이 우선으로 출력 됩니다.
- 현재 스텝번호가 On되면 자기 유지되어 입력 조건이 Off되어도 On되어진 상태를 유지 합니다.
- Sxx.S[0]이 On되면 초기 스텝으로 복귀합니다.

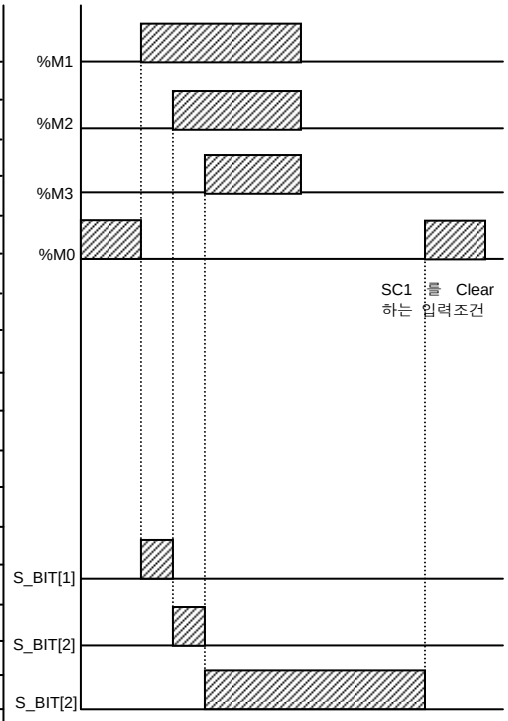
■ 에러

▷ 스텝지정(SET)이 범위(0~99)를 벗어나면 에러가 발생합니다.

▷ 에러 발생시에는 DONE이 OFF되고, 스텝출력은 이전 스텝을 유지합니다.

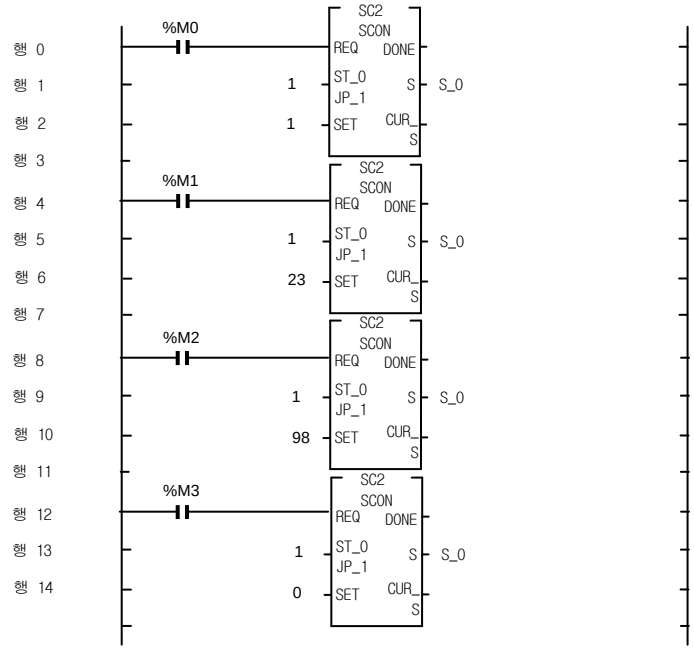
■ SET 동작일 경우(ST_0/JP_1 = 0)의 프로그램 설명
프로그램 SC1조를 이용한 순차제어 프로그램





순차제어는 바로 이전의 Step이 On되고
자신의 조건이 On되면 출력합니다.

■ JUMP 동작일 경우(ST_0/JP_1 = 1)의 프로그램 설명
프로그램 SC2조를 이용한 후입 우선 제어 프로그램

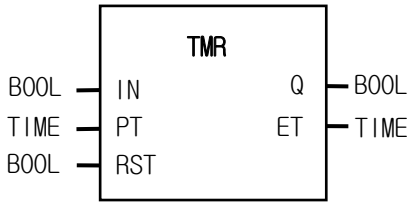


NO	%M1	%M2	%M3	%M4	S_0 [1]	S_0 [23]	S_0 [98]	S_0 [0]
1	On	Off	Off	Off	○			
2	On	On	Off	Off		○		
3	On	On	On	Off			○	
4	On	On	On	On				○

TMR

적산 타이머

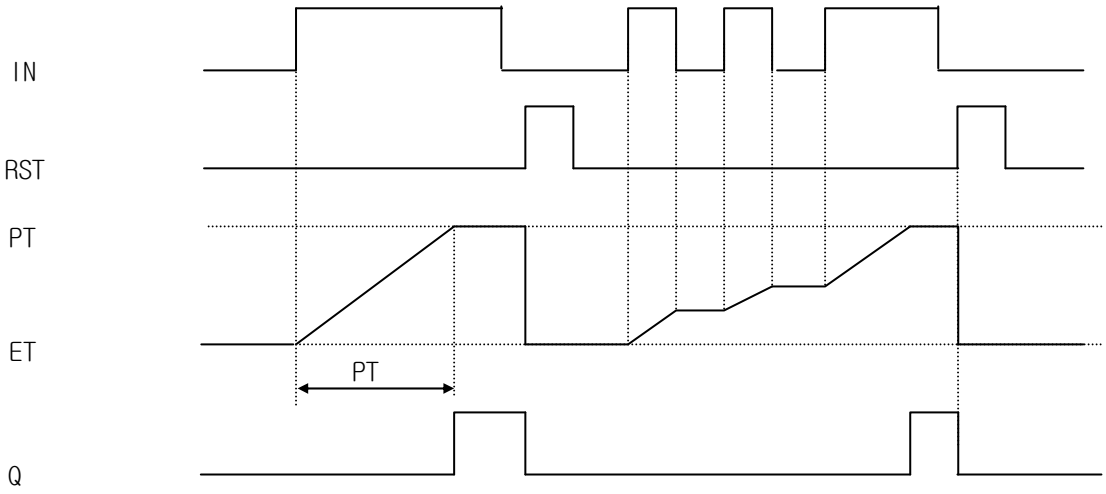
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

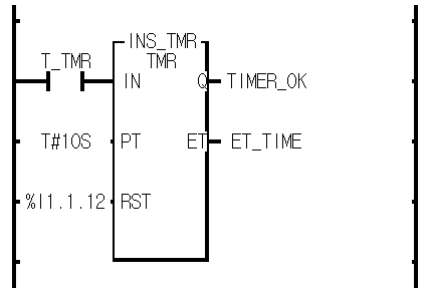
▷ TMR 펑션블록은 IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
▷ 경과 시간 ET가 설정시간에 도달하기 전에 IN이 0이 되어도 현재의 경과 시간을 유지하다가 IN이 다시 1이 되면 경과 시간을 다시 증가시킵니다.
▷ 설정 시간이 경과 시간에 도달하면 Q가 1이 됩니다..
▷ Reset 입력 조건이 성립되면 Q는 0이 되고 경과시간도 0이 됩니다.

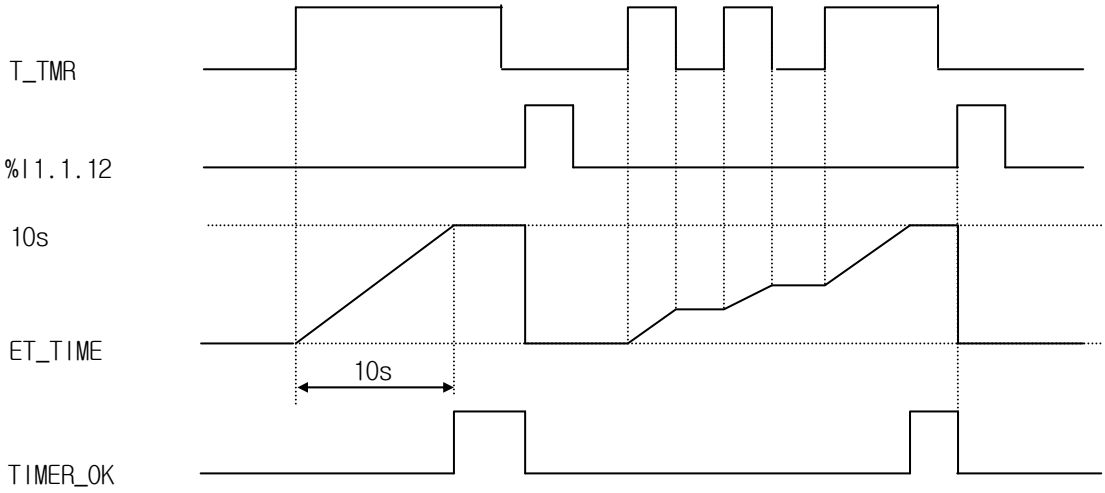
■ 타임 차트



■ 프로그램 예

LD



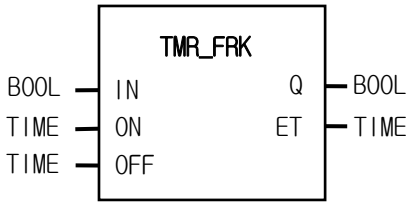


(1)입력변수 T_TMR이 1이 된 후 10초가 경과하면 출력 변수 TIMER_OK가 1이 됩니다.
(2)입력 변수 T_TMR이 1이 된 후 경과 시간이 출력변수 ET_TIME으로 출력됩니다.
(3)경과 시간 ET_TIME이 설정시간 10초에 도달하기 전에 T_TMR이 0이 되더라도 현재의 경과 시간을 유지합니다.
(4)입력 변수 T_TMR이 다시 1이 되면 멈추어 졌던 이전의 경과 시간부터 다시 시작합니다.
(5)입력 접점 %I1.1.12 가 1 이 되면 경과 시간 ET_TIME 및 출력 변수 TIMER_OK 모두 0 으로 클리어(Clear)됩니다.

TMR_FRK

점멸기능 타이머

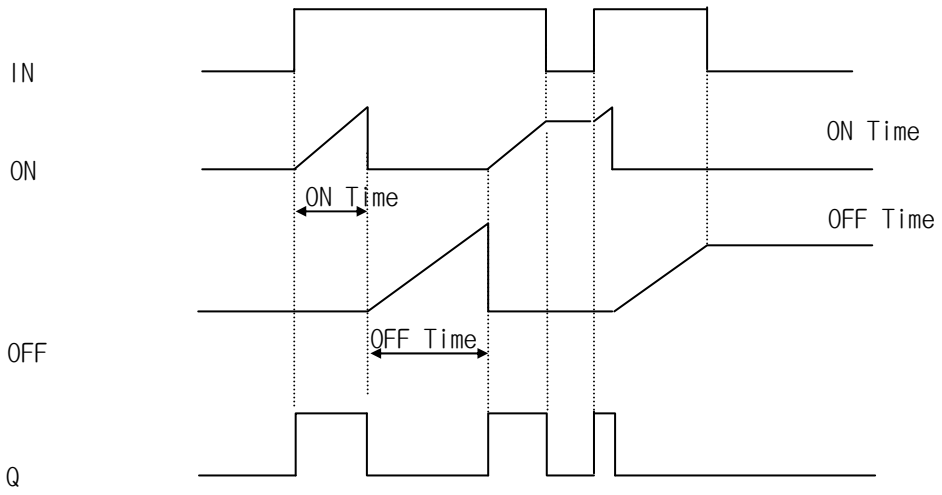
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 ON : ON 타이머 설정 시간 OFF : OFF 타이머 설정 시간 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

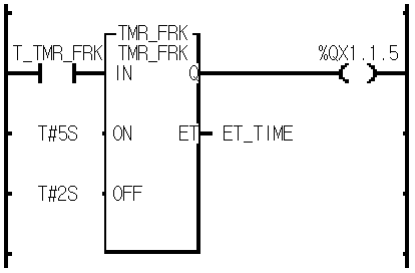
▷ TMR_FLK 펑션 블록은 IN이 1이 되는 순간 Q는 1이 되고, ON에서 지정된 시간만큼 Q는 1을 유지합니다.
▷ ON에서 지정된 시간이 경과하면 OFF에서 지정된 시간만큼 Q는 0이 됩니다.
▷ IN이 0이 되면 ON 또는 OFF 동작을 수행을 중지하고, IN이 0인 동안 중지된 시간을 유지하다가 IN이 다시 1이 되면 정지된 시간부터 다시 타이머가 동작합니다.
▷ IN이 0인 동안 출력 Q는 0이 됩니다.
▷ ON이 0이면 출력 Q는 항상 0이 됩니다.

■ 타임 차트



■ 프로그램 예

LD

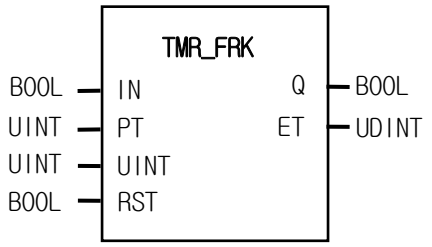


(1)입력변수 T_TMR_FRK가 0에서 1이 되면, TMR_FRK 펑션 블록이 동작을 시작합니다.
(2)입력변수 T_TMR_FRK가 1이 된 후 ON에서 지정된 5초 만큼 출력 접점 %QX1.1.5는 1이 됩니다.
(3)입력변수 T_TMK_FRK가 1이 된 후 ON에서 지정된 시간이 경과하면 OFF에서 지정된 2초만큼 출력 접점 %QX1.1.5는 0이 됩니다.
(3)입력변수 T_TMR_FRK가 1인 동안 출력 Q가 1인 동안의 경과 시간과 Q가 0인 동안의 경과 시간이 번갈아 ET_TIME으로 출력됩니다.
(4)입력변수 T_TMR_FRK가 0이 되면 동작 중인 시간을 유지하고 출력접점 %QX1.1.5는 0이 되며, T_TMR_FRK가 다시 1이 되면 정지되었던 시간부터 다시 동작합니다.

TMR_UNIT

정수 설정 적산 타이머

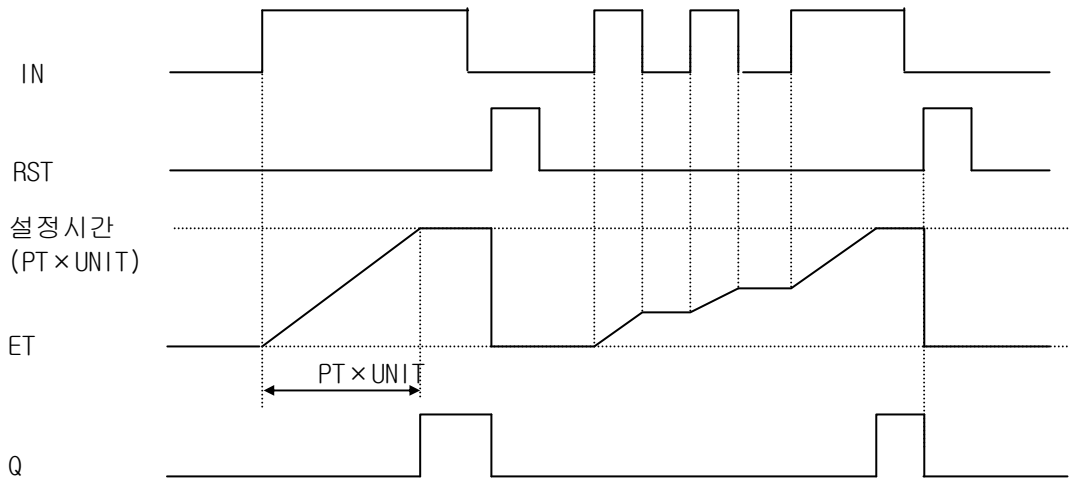
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

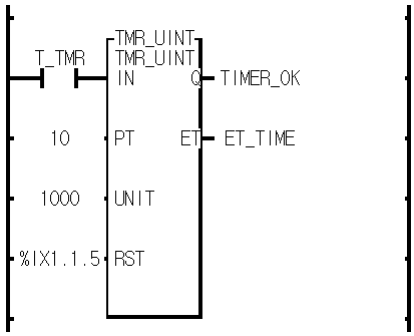
▷ TMR_UINT 펑션 블록은 IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
▷ 경과 시간 ET가 설정시간에 도달하기 전에 IN이 0이 되어도 현재의 경과 시간을 유지하다가 IN이 다시 1이 되면 경과 시간이 다시 증가됩니다.
▷ 설정 시간이 경과 시간에 도달하면 Q가 1이 됩니다..
▷ Reset 입력 조건이 성립되면 Q는 0이 되고 경과시간도 0이 됩니다.
▷ 설정시간은 PT*UNIT[mSec]입니다.

■ 타임 차트

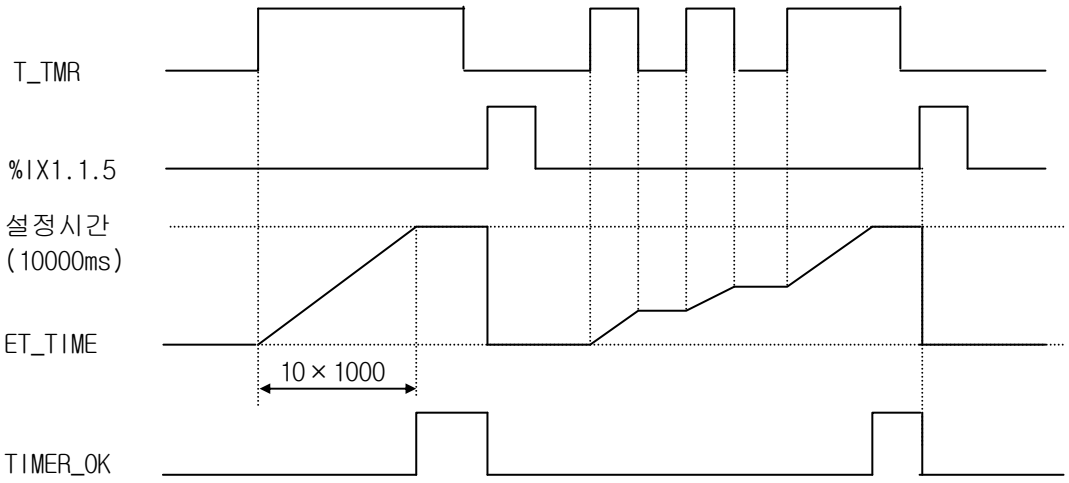


■ 프로그램 예

LD



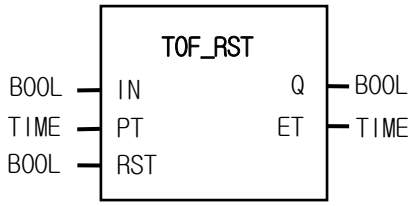
(1)설정 시간은 PT*UNIT[ms] = 10*1000[ms] = 10[s]가 됩니다.
(2)입력변수 T_TMR이 1이 된 후 10초가 경과하면 출력 변수 TIMER_OK가 1이 됩니다.
(3)입력 변수 T_TMR이 1이 된 후 경과 시간이 출력변수 ET_TIME으로 출력됩니다.
(4)경과 시간 ET_TIME이 설정시간 10초에 도달하기 전에 T_TMR이 0이 되더라도 현재의 경과 시간을 유지합니다.
(5)입력 변수 T_TMR이 다시 1이 되면 멈추어 졌던 이전의 경과 시간부터 다시 시작합니다.
(6)입력 접점 %IX1.1.5가 1이 되면 경과 시간 ET_TIME 및 출력 변수 TIMER_OK 모두 0으로 클리어(Clear)됩니다.



TOF_RST

동작중 출력 OFF가 가능한 딜레이 타이머

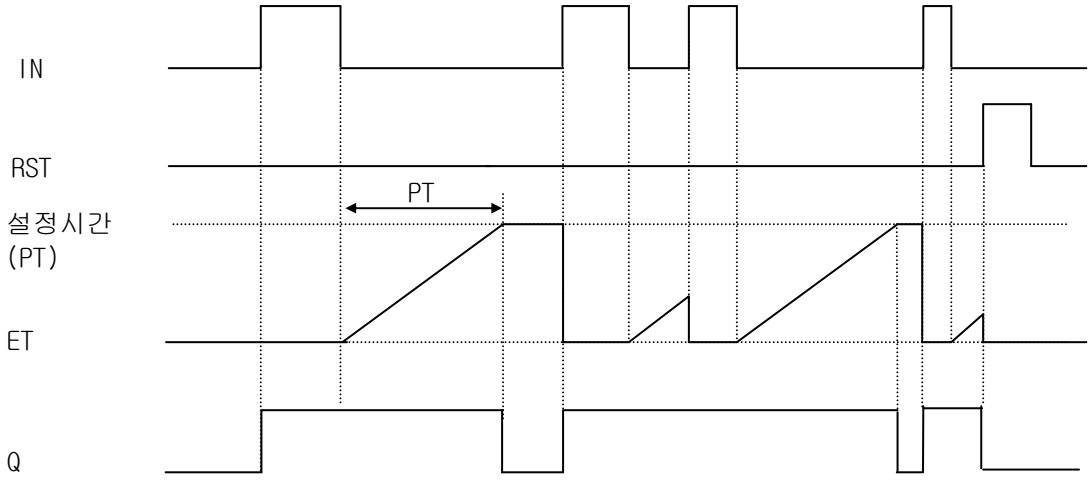
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

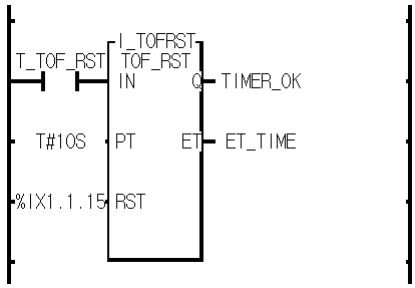
▷ TOF_RST 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, IN이 0이 된 후부터 PT에 의하여 지정된 설정시간이 경과한 후 Q가 0이 됩니다.
▷ IN이 0이 된 후 경과 시간이 ET로 출력됩니다.
▷ 만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 1이 되면, 경과 시간은 다시 0으로 됩니다.
▷ Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

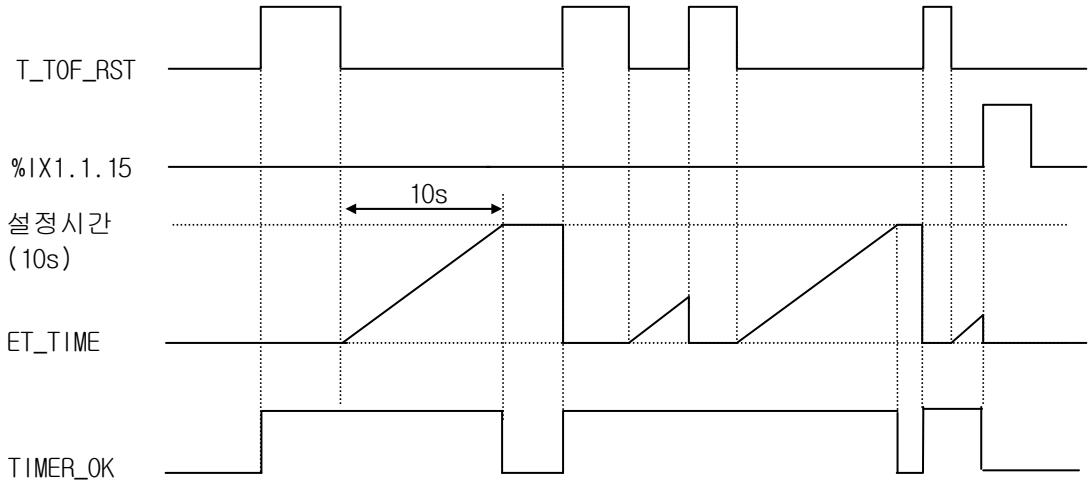
■ 타임 차트



■ 프로그램 예

LD



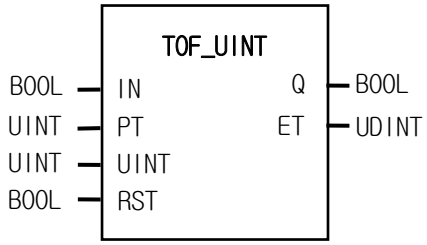


(1)입력변수로 설정된 T_TOF_RST가 1이 되면, 출력변수 TIMER_OK에 1이 출력되고 T_TOF_RST가 0이 된 후 10s 후에 TIMER_OK가 0이 됩니다.
(2)T_OF_RST가 0이 된 후 10초 이내에 다시 1이 되면 타이머는 다시 초기 상태가 됩니다.
(3)타이머의 시간 측정값은 ET_TIME로 출력됩니다.
(4)입력접점 %IX1.1.15가 1이 되면 TIMER_OK와 ET_TIME 모두 0으로 클리어(Clear)됩니다.

TOF_UINT

정수 설정 OFF 타이머

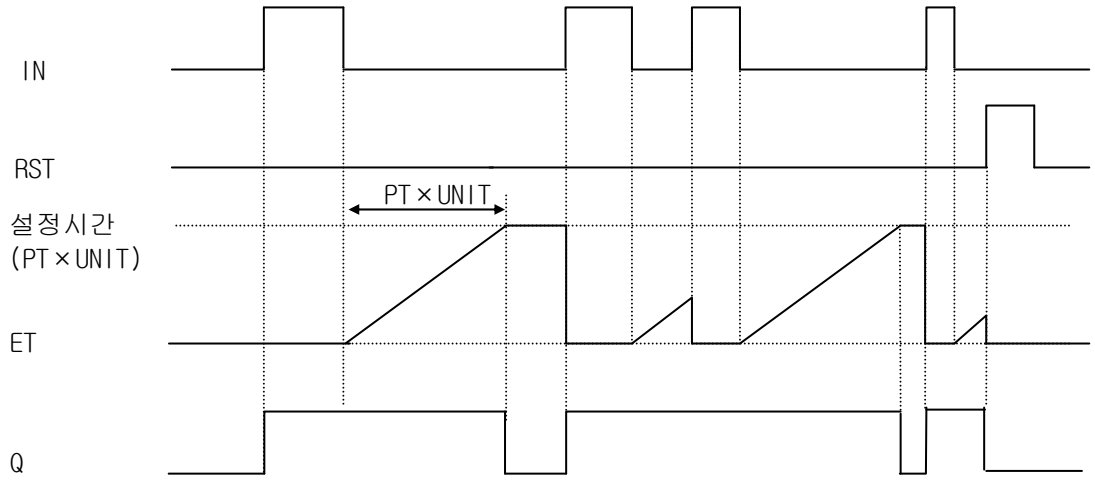
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

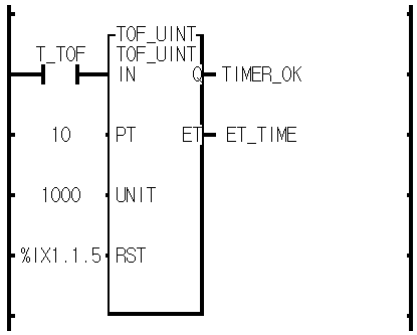
▷ TOF_UINT 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, IN이 0이 된 후부터 PT에 의하여 지정된 설정시간이 경과한 후 Q가 0이 됩니다.
▷ IN이 0이 된 후 경과 시간이 ET로 출력됩니다.
▷ 만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 1이 되면, 경과 시간은 다시 0으로 됩니다.
▷ Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.
▷ 설정시간은 PT*UNIT[ms]입니다.

■ 타임 차트

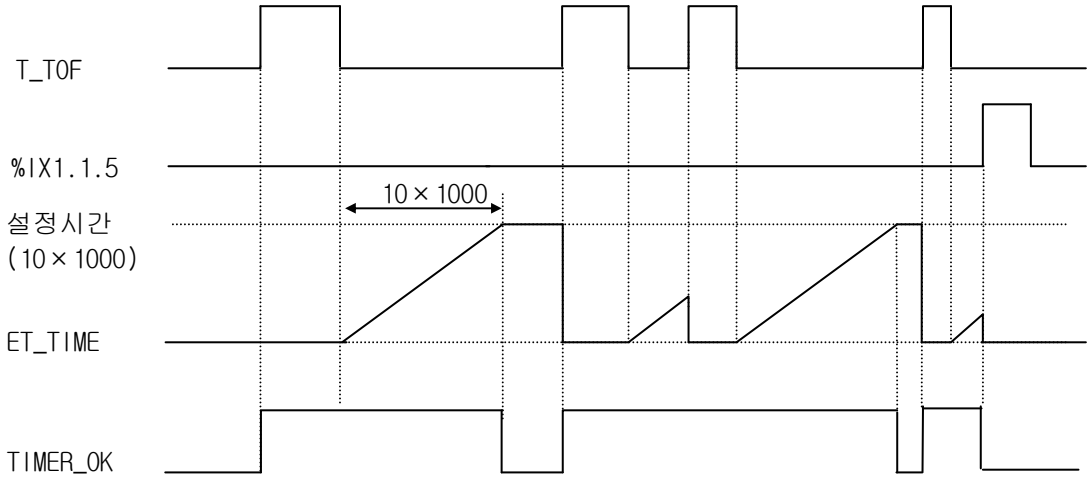


■ 프로그램 예

LD



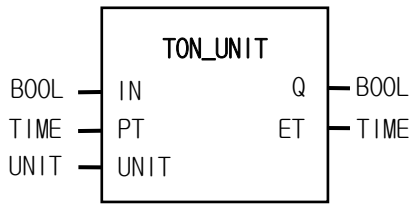
(1)설정 시간은 PT*UNIT[ms] = 10*1000[ms] = 10[s]가 됩니다.
(2)입력변수로 설정된 T_TOF가 1이 되면, 출력변수 TIMER_OK에 1이 출력되고 T_TOF가 0이 된 후 10초 후에 TIMER_OK가 0이 됩니다.
(3)T_TOF가 0이 된 후 10초 이내에 다시 1이 되면 타이머는 다시 초기 상태가 됩니다.
(4)타이머의 시간 측정값은 ET_TIME로 출력됩니다.
(5)입력접점 %IX1.1.5가 1이 되면 TIMER_OK와 ET_TIME 모두 0으로 클리어(Clear)됩니다.



TON_UINT

정수 설정 ON 타이머

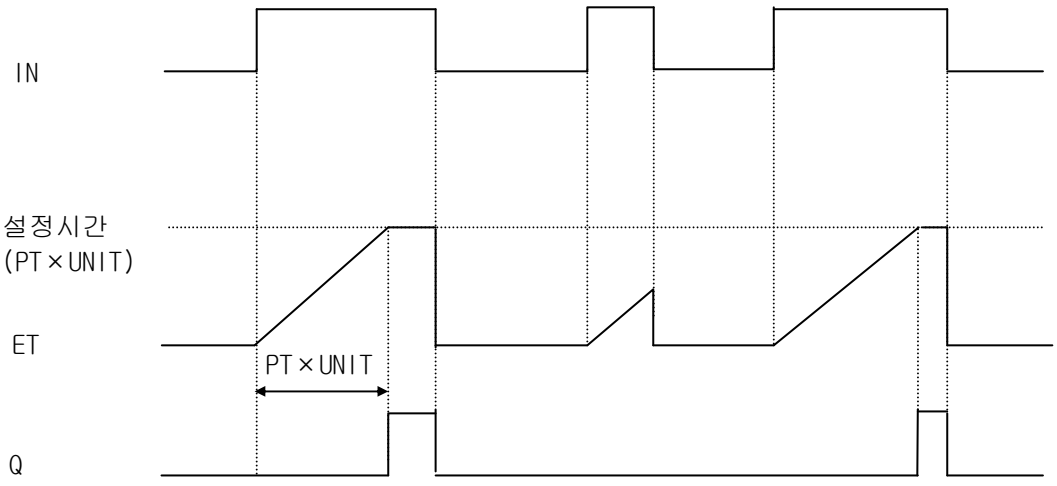
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

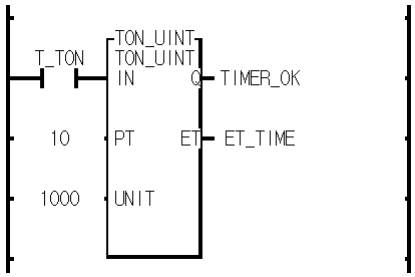
▷ TON_UINT 펑션 블록은 IN이 1이 된 후 경과시간이 ET로 출력됩니다.
▷ 만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 0이 되면, 경과 시간 ET는 0으로 됩니다.
▷ Q가 1이 된 후 IN이 0이 되면, Q는 0이 됩니다.
▷ 설정시간은 PT×UNIT[ms]입니다.

■ 타임 차트

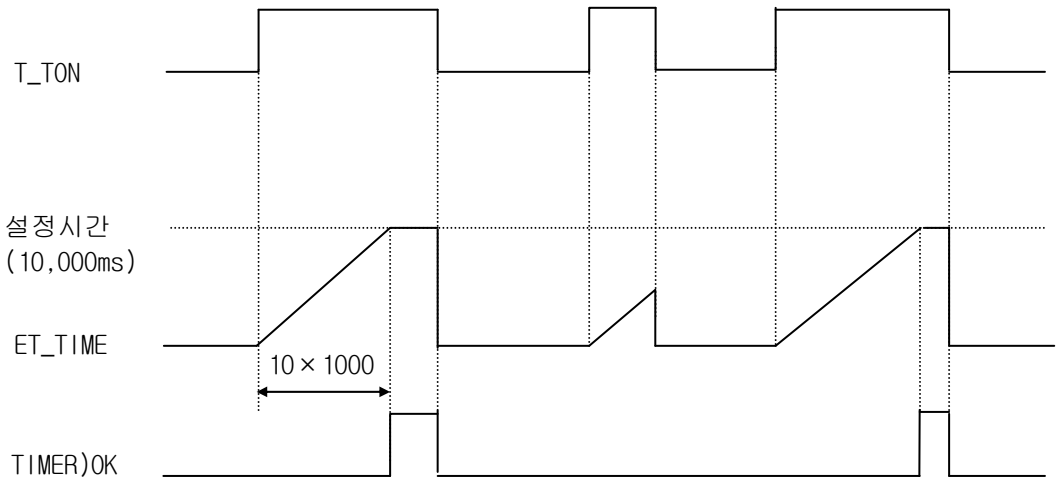


■ 프로그램 예

LD



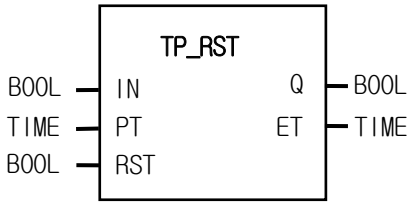
(1)설정 시간은 PT*UNIT[s] = 10*1000[s] = 10[s]가 됩니다.
(2)입력변수 T_TON이 On이 된 후, 10초가 경과한 후에 출력 변수 TIMER_OK가 1이 됩니다.
(3)입력변수 T_TON이 On된 후 경과 시간이 출력 변수 ET_TIME로 출력됩니다.
(4)만일 경과 시간 ET_TIME이 설정시간 10초에 도달하기 전에 T_TON이 0이 되면, 경과 시간 ET_TIME은 0으로 됩니다.
(5)TIMER_OK가 1이 된 후 T_TON이 0이 되면, TIMER_OK는 0이 되고 경과 시간 ET_TIME도 0이 됩니다.



TP_RST

펄스 타이머 출력 OFF가 가능한 모노스테이블 타이머

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

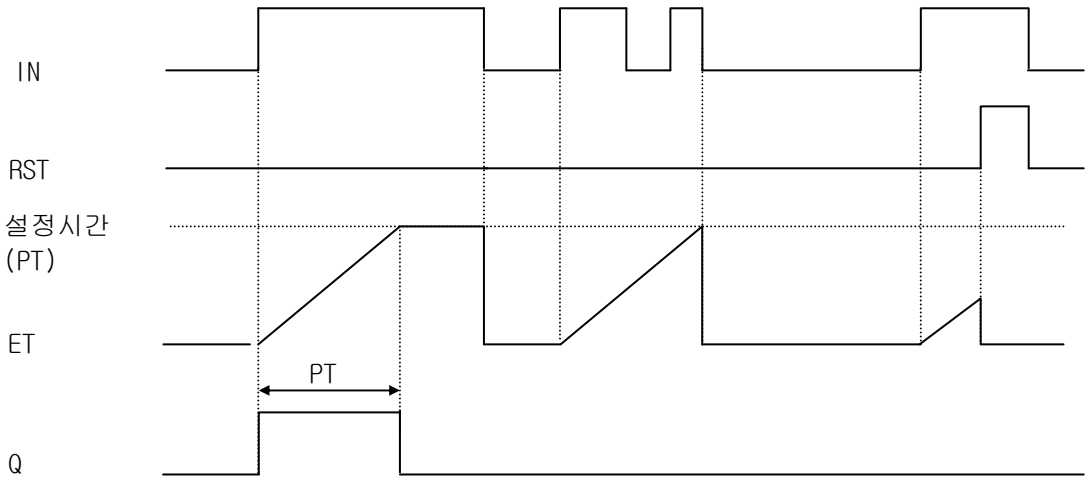
▷TP_RST펑션블록은 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.

▷ 경과 시간 ET는 IN이 1이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 0이 될 때 0으로 클리어(clear) 됩니다.

▷ 타이머 출력 Q가 1인 동안(펄스 출력 중)에는 타이머 기동조건 IN이 1,0변화를 하여도 무시합니다.

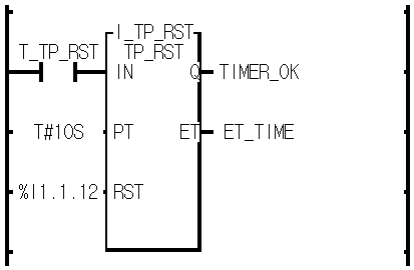
▷ Reset 입력 조건이 성립하면 펄스 출력 중에도 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

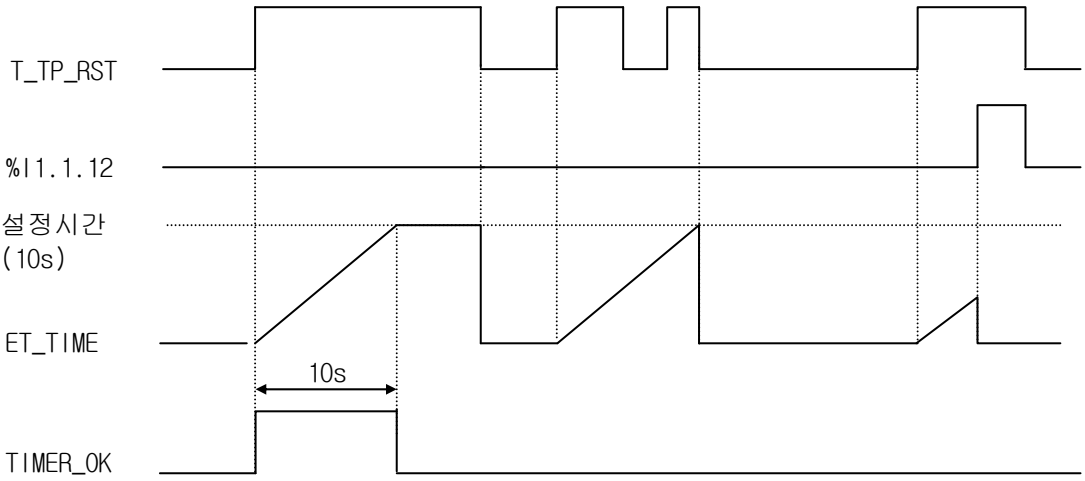
■ 타임 차트



■ 프로그램 예

LD



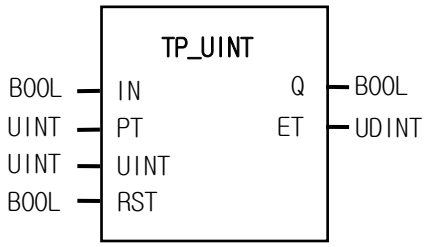


(1)입력변수 T_TP_RST가 1이 되면 출력변수 TIMER_OK가 1이 되고 10초가 경과하면 출력변수 TIMER_OK는 0이 됩니다.일단 타이머가 가동된 후에는 10초 동안 T_TP_RST신호는 무시됩니다.
(2)ET_TIME 값은 증가 후 10S 에서 멈춥니다. 그리고 T_TP_RST 가 0 이 될 때 0 으로 됩니다.
(3)입력 접점 %I1.1.12가 1이 되면 TIIMER_OK와 ET_TIME 모두 0으로 클리어(Clear)됩니다.

TP_UINT

정수 설정 펄스 타이머

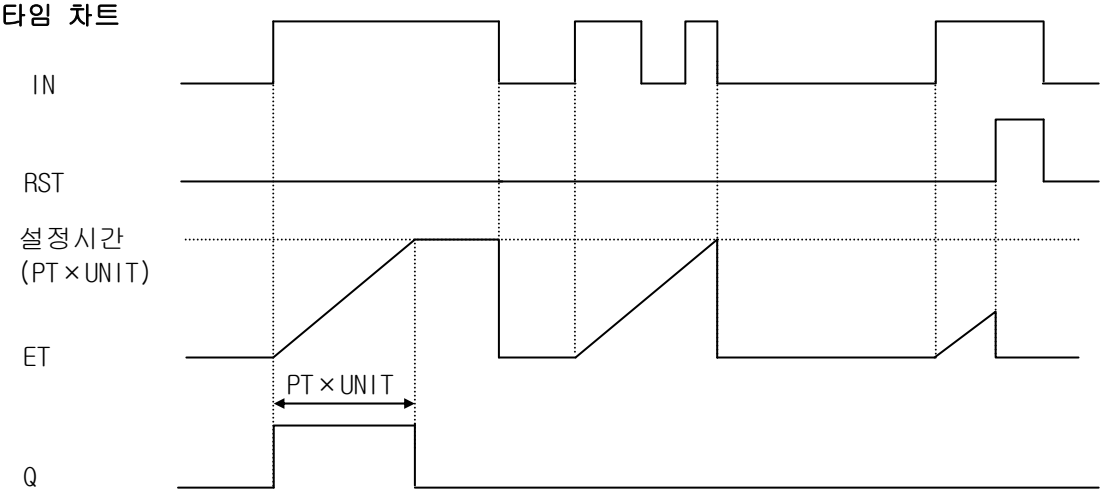
CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

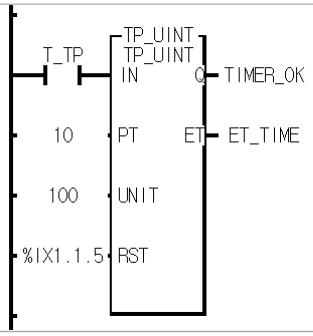
▷ TP_UINT펑션블록은 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.
▷ 경과 시간 ET는 IN이 1이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 0이 될 때 0으로 클리어(Clear) 됩니다.
▷ 타이머 출력 Q가 1인 동안(펄스 출력 중)에는 타이머 기동조건 IN이 1,0변화를 하여도 무시합니다.
▷ Reset 입력 조건이 성립하면 펄스 출력 중에도 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.
▷ 설정시간은 PT*UNIT[ms]입니다.

■ 타임 차트

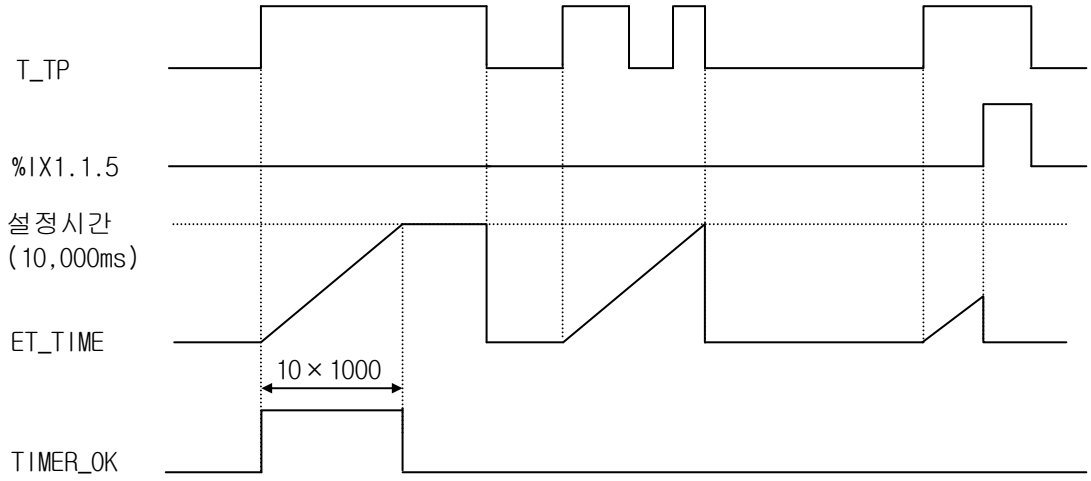


■ 프로그램 예

LD



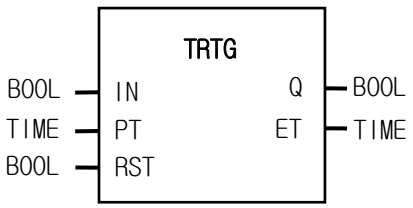
(1)설정 시간은 PT*UNIT[ms] = 10*1000[ms] = 10[s]가 됩니다.
(2)입력변수 T_TP가 1이 되면 출력변수 TIMER_OK가 1이 되고 10초가 경과하면 출력변수 TIMER_OK는 0이 됩니다.일단 타이머가 가동된 후에는 10초 동안 T_TP 신호는 무시됩니다.
(3)ET_TIME 값은 증가 후 10,000 에서 멈춥니다. 그리고 T_TP 가 0 이 될 때 0 으로 됩니다.
(4)입력 접점 %IX1.1.5가 1이 되면 TIIMER_OK와 ET_TIME 모두 0으로 클리어(Clear) 됩니다.



TRTG

리트리거블 타이머

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

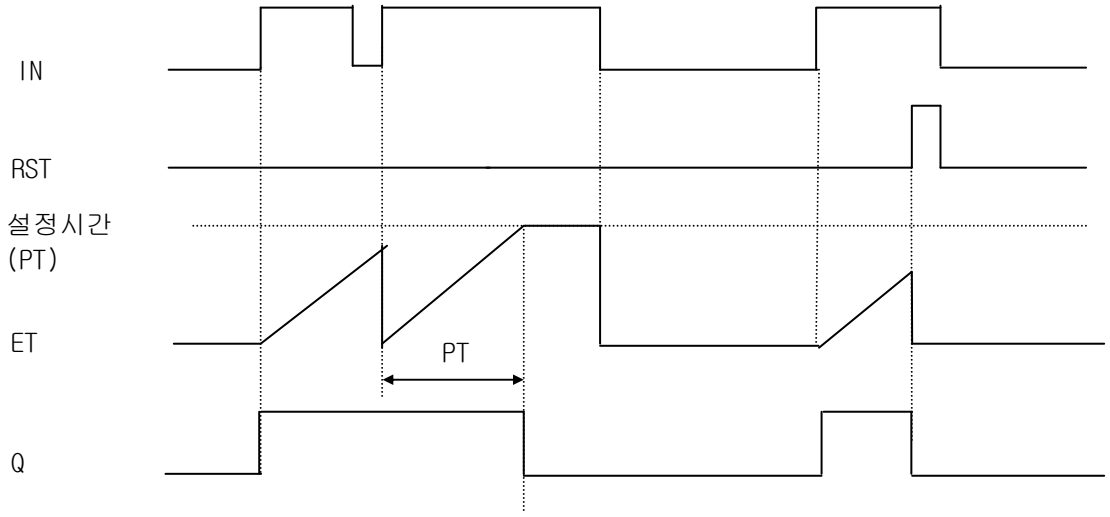
■ 기능

▷ TRTG 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 된다.

▷ 타이머 경과 시간이 0이 되기 전에 IN이 또 다시 0에서 1로 되면 경과 시간은 0으로 재설정 되고 다시 증가하여 설정시간 PT에 도달하면 Q는 0이 된다.

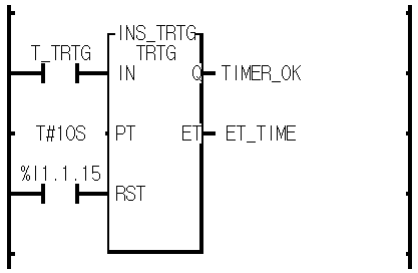
▷ Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 된다.

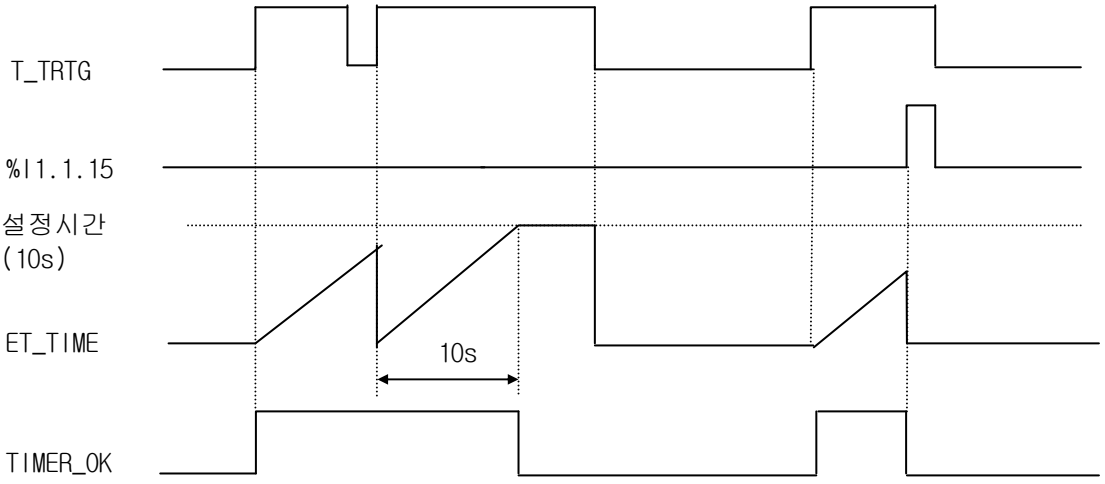
■ 타임 차트



■ 프로그램 예

LD



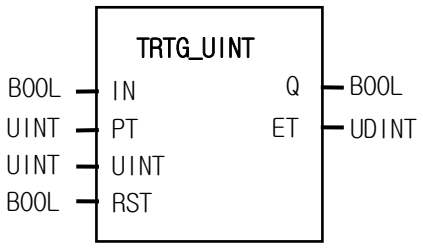


(1)입력변수 T_TRTG가 0에서 1이 된 후 10초 동안 TIMER_OK는 1이 됩니다. 타이머가 가동된 후 T_TRTG 가 다시 0에서 1이 되면 ET_TIME은 0부터 다시 시작됩니다.
(2)T_TRTG가 1에서 0이 되어도 10초 동안 TIMER_OK는 1이 됩니다.
(3)ET_TIME은 증가 후 T#10S에서 멈춥니다. 그리고 T_TRTG가 0이 될 때 0으로 됩니다.
(4)입력접점 %I1.1.15가 1이 되면 TIMER_OK와 ET_TIME모두 0으로 클리어(Clear) 됩니다.

TRTG_UINT

정수 설정 리트리거블 타이머

CPU 명	GMR	GM1	GM2	GM3	GM4	GM6	GM7
적용 가능	●	●	●	●	●	●	●

펑 션 블 록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

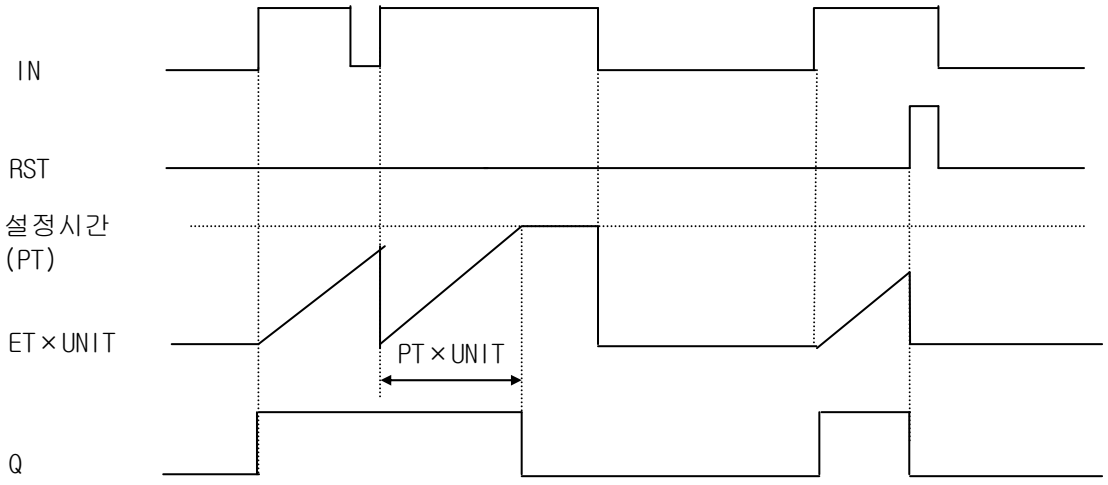
▷ TRTG_UINT 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.

▷ 타이머 경과 시간이 0이 되기 전에 IN이 또 다시 0에서 1로 되면 경과 시간은 0으로 재설정 되고 다시 증가하여 설정시간 PT에 도달하면 Q는 0이 됩니다.

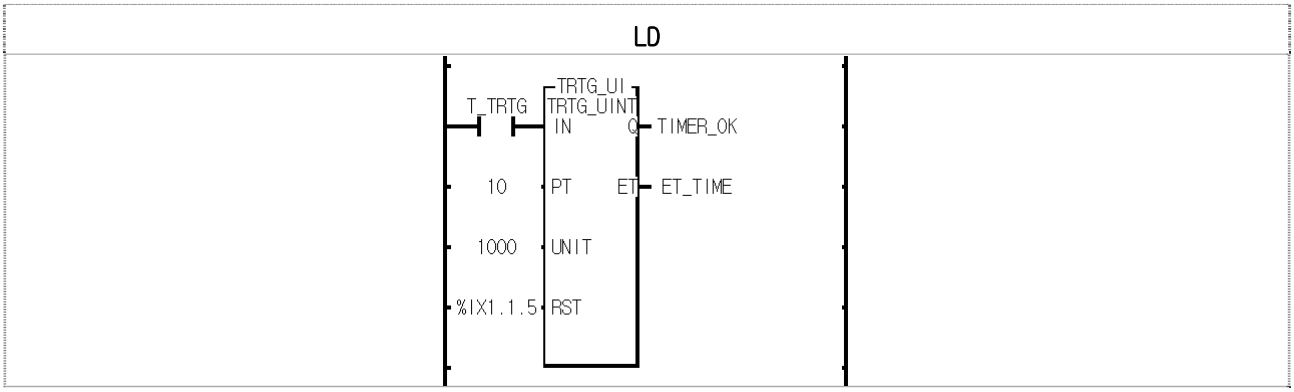
▷ Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

▷ 설정시간은 PT*UNIT[ms]입니다.

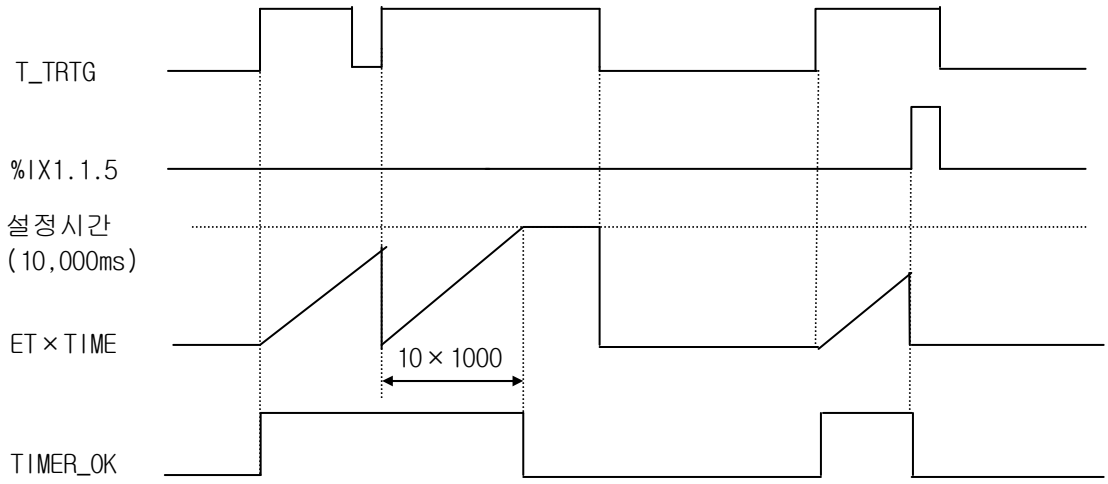
■ 타임 차트



■ 프로그램 예



(1)설정 시간은 PT*UNIT[ms] = 10*1000[ms] = 10[s]가 됩니다.
(2)입력변수 T_TRTG가 0에서 1이 된 후 10초 동안 TIMER_OK는 1이 됩니다. 타이머가 가동된 후 T_TRTG가 다시 0에서 1이 되면 ET_TIME은 0부터 다시 시작됩니다.
(3)T_TRTG가 1에서 0이 되어도 10초 동안 TIMER_OK는 1이 됩니다.
(4)ET_TIME은 증가 후 10,000에서 멈춥니다. 그리고 T_TRTG가 0이 될 때 0으로 됩니다.
(5)입력접점 %IX1.1.5가 1이 되면 TIMER_OK와 ET_TIME모두 0으로 클리어(Clear) 됩니다.



부록C. 태스크 프로그램

부록 C. 태스크 프로그램

1.1 태스크 프로그램의 종류 및 실행 방식

1.1.1 태스크 프로그램의 종류

연산자, 펑션, 펑션 블록 등을 사용하여 작성된 각각의 프로그램은 그 실행을 제어하기 위한 프로그램의 기동 조건인 태스크에 따라 다음과 같이 분류됩니다.

구분	태스그 종류	기동 조건	동작
시스템 태스크	초기화 프로그램	_INIT 태스크로 기동	기본 파라미터로 콜드/웜 리스타트 설정시 실행
		_H_INIT 태스크로 기동	기본 파라미터로 핫 리스타트 설정 시 실행
	에러처리 프로그램	_ERR_SYS 태스크로 기동	시스템 에러 발생시 실행
Event 태스크	정주기 태스크 프로그램	시간 간격 설정 태스크로 기동	설정 시간 마다 주기적으로 프로그램 실행
	외부 접점 태스크 프로그램	인터럽트 모듈 태스크로 기동	인터럽트 모듈 접점의 상승 에지 검출 시 프로그램 실행
	내부 접점 태스크 프로그램	내부 접점 태스크로 기동	내부 접점의 상승 에지 검출시 스캔 프로그램 실행 완료 후 프로그램 실행
스캔 태스크	스캔 프로그램	기동 조건 없음	매 스캔 마다 프로그램 실행

☞ 태스크 우선 순위 : 시스템 > 정주기, 외부 접점 > 내부 접점 > 스캔

☞ 시스템 태스크란 시스템에서 제공되는 태스크를 말합니다.
(기본적으로 등록되어 있음)

☞ Event 태스크는 사용자에 의해 등록이 필요합니다.

☞ 스캔 태스크란 특별한 기동 조건이 없는 태스크를 말합니다.
(등록이 필요 없음)

☞ 정주기(인터벌) 태스크란 시간 간격 설정에 의한 태스크를 말합니다.

☞ 외부 접점(인터럽트) 태스크란 인터럽트 모듈에 의한 태스크를 말합니다.

☞ 내부 접점(싱글) 태스크란 내부 접점에 의한 태스크를 말합니다.

☞ 시스템 태스크의 기동 조건 해제(종료)

_ 초기화 태스크 : _INIT_DONE 플래그 On시 해제

_ 시스템 에러 해제 플래그 On시 해제

태스크의 속성

종류 규격	정주기 태스크 (인터벌 태스크)	외부 접점 태스크 (인터럽트 태스크)	내부 접점 태스크 (싱글 태스크)
개수	32 개	16 개	16 개
기동 조건	정주기 (10ms ~ 4294967.29sec)	인터럽트모듈의 입력 접점 의 상승 또는 하강 에지	내부 메모리 데이터중 지 정한 불(Bool)변수 데이터 의 상승 에지 (0->1)
검출 실행	설정 시간 마다 주기적으 로 실행	인터럽트 모듈의 접점의 에지 발생시 즉시 실행	스캔 프로그램 실행 완료 후 에지를 검색하여 실행
검출 지연 시간	최대 5ms 지연	최대 5ms 지연 + 인터럽트 모듈 지연 (0.5ms 이내)	최대 스캔 타임 만큼 지연
실행 우선 순위	0 ~ 7 레벨 설정 (0 레벨이 가장 높음)	좌동	좌동

1.1.2 태스크 프로그램의 실행 방식

(가) 초기화 프로그램

- ♦ 콜드나 웜 리스타트 모드로 운전을 시작할 때 초기화 프로그램은 스캔 및 태스크 프로그램 실행 전에 시스템 초기화를 할 때 사용 됩니다.
  초기화 프로그램은 종료 조건이 성립될 때까지 반복 연산을 실행합니다.
  (초기화 프로그램에서 _INIT_DONE 플래그가 ON 될 때까지 반복 연산을 실행)
  초기화 프로그램 실행 중에도 입출력 리프레시를 실행 합니다

- ♦ 핫 리스타트 모드로 운전을 시작할 때 초기화 프로그램은 정전발생 시 정지된 스텝 부터 실행을 다시 시작하거나 특수모듈을 정전 이전의 상태로 복원할 필요가 있을 때 사용됩니다
  초기화 프로그램은 설정한 조건이 성립될 때까지 반복 연산을 실행합니다.
  (초기화 프로그램에서 _INIT_DONE 플래그가 ON 될 때까지 반복 연산을 실행)
  핫 리스타트 초기화 프로그램 실행 중 입출력 리프레시를 실행 하지 않습니다.
  (정전시 입출력 이미지 데이터를 그대로 유지)
  입출력 리프레시가 필요하면 입출력 펑션을 사용하십시오.
  핫 리스타트 초기화 프로그램 실행 완료 후 정지 스텝부터 프로그램을 실행하여 1스캔이 완료 되어도 출력 리프레시는 하지 않고 그 다음 스캔부터 정상적인 입출력 리프레시를 합니다.
- ♦ 초기화 프로그램 실행 중에는 _INIT_RUN 플래그가 ON 됩니다.

(나) 태스크 프로그램

- ♦ 정주기 태스크에 의해 지정된 프로그램은 태스크에 설정된 시간 간격으로 프로그램을 실행합니다.
- ♦ 외부 인터럽트 태스크에 의해 지정된 프로그램은 인터럽트 모듈에서 발생한 외부 인터럽트 접점 신호에 의해 프로그램을 실행합니다.
- ♦ 내부 접점 태스크에 의해 지정된 프로그램은 내부 접점의 상승 에지 발생시 스캔 프로그램의 처리한 다음 태스크 프로그램 실행합니다.

(다) 스캔 프로그램

사용자가 작성한 프로그램 중 초기화 프로그램이나 태스크 프로그램으로 지정하지 않은 프로그램들은 자동으로 스캔 프로그램이 됩니다. 스캔 프로그램은 태스크와 무관한 프로그램으로 실행의 우선순위가 가장 낮으며, 주기적인 반복 연산을 수행하는 프로그램으로서 등록 순서에 따라 처리 순서가 결정되어 순차적인 반복 연산을 실행합니다.

(라) 에러 처리 프로그램

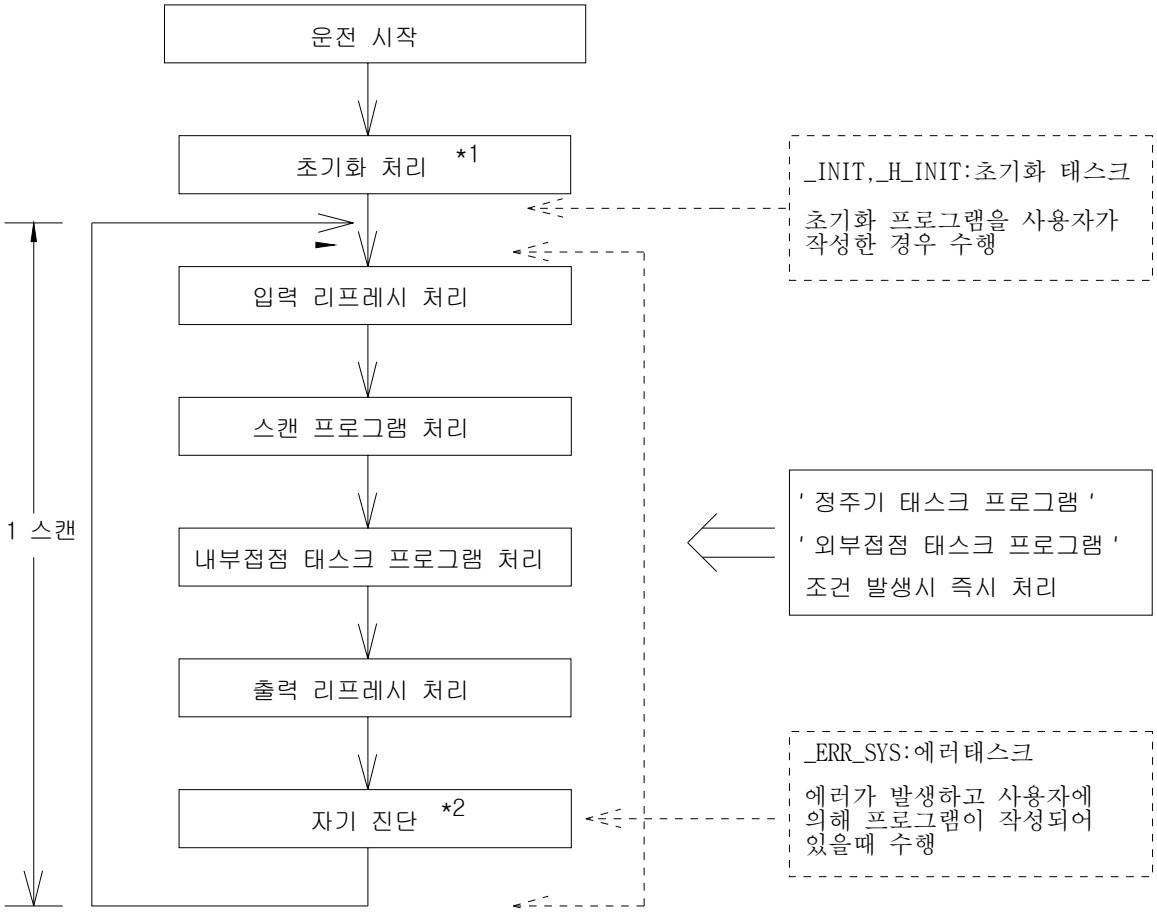
에러 태스크 프로그램은 사용자 프로그램의 실행 중 시스템의 에러가 발생할 경우 실행됩니다.

사용자는 시스템의 에러에 대응하는 프로그램을 작성한 후 _ERR_SYS 태스크를 조건으로 하여 프로그램을 기동합니다. 에러 태스크 프로그램에서 시스템 에러 해제 플래그를 제어하면 시스템의 운전 정지를 막을 수 있습니다.

* 에러 종류 : - 모듈 착탈 에러
- FUSE 단선 에러
- 입출력 모듈 읽기/쓰기 에러(고장)
- 특수, 통신 모듈 인터페이스 에러(고장)
- 외부 기기의 중 고장 검출 에러

1.1.3 태스크 프로그램의 실행도

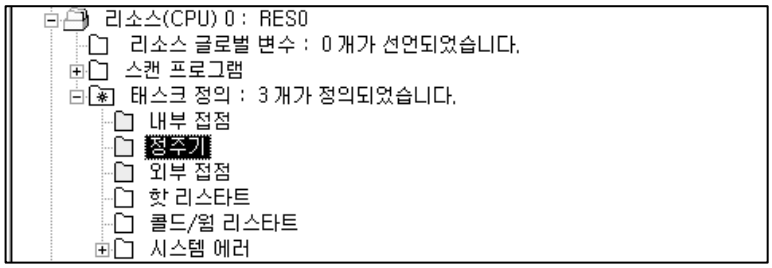
스캔 프로그램은 작성된 순서대로 처음부터 마지막까지 실행됩니다.
이와 같이 스캔 프로그램은 반복 연산 방식으로 수행되고 이 한차례의 과정을 '1 스캔'
이라고 합니다.
스캔 프로그램 실행 중에 정주기 또는 외부 접점 태스크 프로그램의 실행 조건이 발생하면 현재 실행 중인 프로그램의 실행을 일시 중지하고 해당하는 태스크 프로그램을 실행합니다.
스캔 프로그램의 실행이 완료되면, 내부 접점 태스크 프로그램의 실행 조건을 체크하여 실행 조건이 발생한 태스크 프로그램을 실행합니다.



1.2 태스크 추가하기

태스크는 프로그램의 실행 조건을 정의하는 것입니다. 그 종류로는 싱글, 인터벌, 인터럽트 등이 있습니다.

1) 태스크 추가하기



◆ 프로젝트 창 내의 목록 중 편집하려는 태스크를 선택합니다.

◆ 메뉴 **[프로젝트]-[프로젝트 항목 추가]-[태스크]**를 선택하거나 팝업 메뉴에서 **[프로젝트]-[프로젝트 항목 추가]-[테스크]**를 선택합니다.

◆ 태스크 이름 입력란에 태스크 이름을 입력합니다.

◆ 태스크 번호 입력란에 수행 조건에 따라 아래 번호를 입력합니다.

수행 조건	태스크 번호
정주기	0 ~31
외부접점	32 ~47 (GM4:32 ~39)
내부접점	48 ~63

단 인터럽트의 경우는 인터럽트 번호와 해당 태스크 번호가 지정되어 있으므로 자동으로 설정됩니다.

GM6,7 의 태스크 번호는 우선순위에 따라 자동으로 설정됩니다.

GM6,7 의 태스크는 정주기, 외부접점, 내부접점을 모두 합하여 8 개 입니다.

내부 접점인 경우에는 변수 이름을, 정주기인 경우에는 경과 시간을, 외부 접점인 경우에는 인터럽트 카드의 입력 위치를 번호로 표현합니다. 다음은 각 경우에 가능한 입력의 예를 듭니다.

(1) 내부 접점인 경우

리소스 글로벌 변수 중 BOOL 인 것	A, VALVE1 등
직접 변수 중 BOOL 인 것	%IX0.0.0, %QX0.1.1, %MX10 등

(2) 정주기인 경우

경과 시간을 나타내는 상수 값	T#10S, T#1H10M10S10MS 등

(3) 외부 접점인 경우

Interrupt 번호(GM1~3:0~15, GM4: 0~7 까지 가능)	0, 1, 2 .. 15
(GM6~7 : 직접변수 %I)	%IX0.0.0

- ◆ 우선순위 목록상자에서 우선 순위를 선택합니다. 우선 순위는 0~7 까지 가능하며, 작은 숫자가 높은 우선 순위를 갖습니다.
- ◆ **[확인]**을 클릭 합니다.

2) 태스크 삭제하기

- ◆ 태스크 정의 목록상자에서 삭제할 항목을 선택합니다.
- ◆ 팝업 메뉴의 **[삭제]** 또는 Del 키를 누릅니다.

3) 태스크 수정하기

- ◆ 태스크 정의 목록상자에서 수정할 항목을 선택합니다.
- ◆ 수정하려는 태스크를 마우스로 더블클릭 하거나, 팝업 메뉴에서 **[등록정보]**를 선택합니다.
- ◆ 태스크 추가 시와 같은 방법으로 입력한 후 **[확인]**을 클릭 합니다.
- ◆ 하나씩 이동시킵니다.

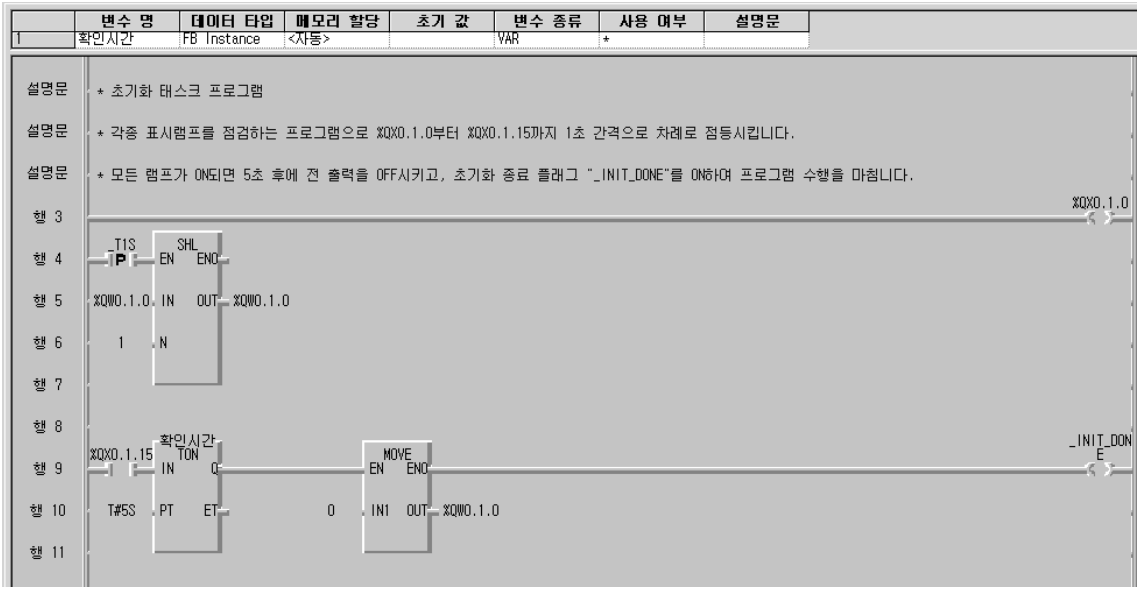
1.3 태스크 프로그램

1.3.1 초기화 태스크 프로그램

(1) 초기화 태스크 프로그램 작성

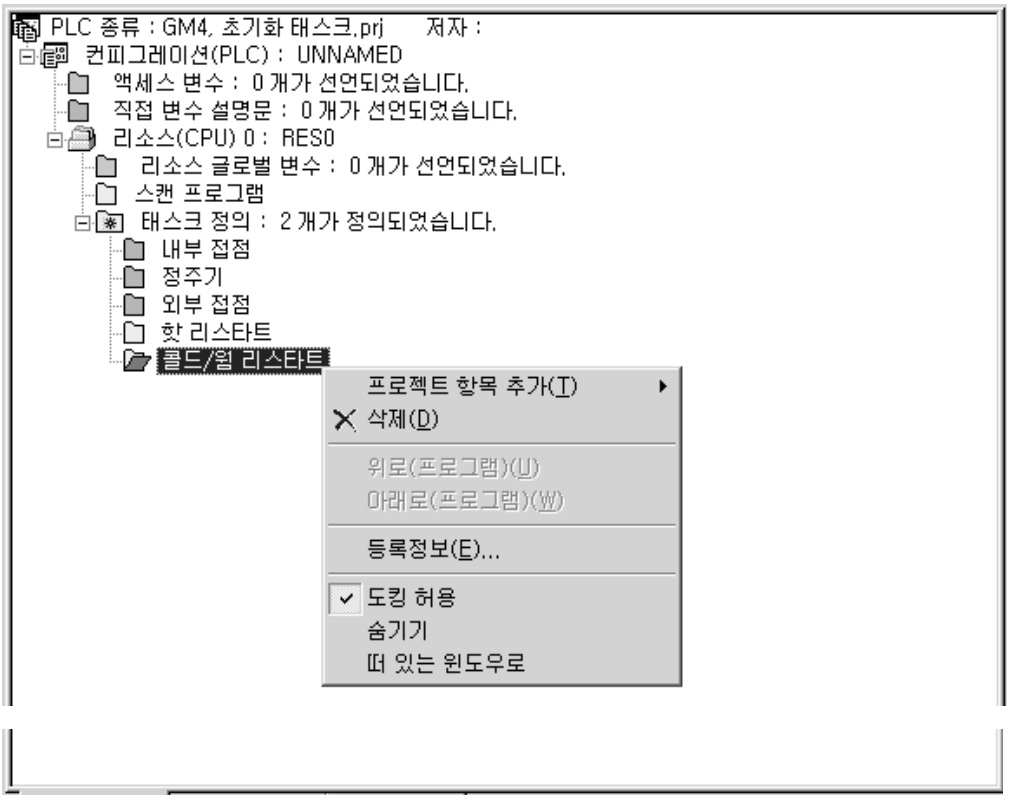
초기화 태스크 프로그램(초기화 태스크.src)을 작성합니다.

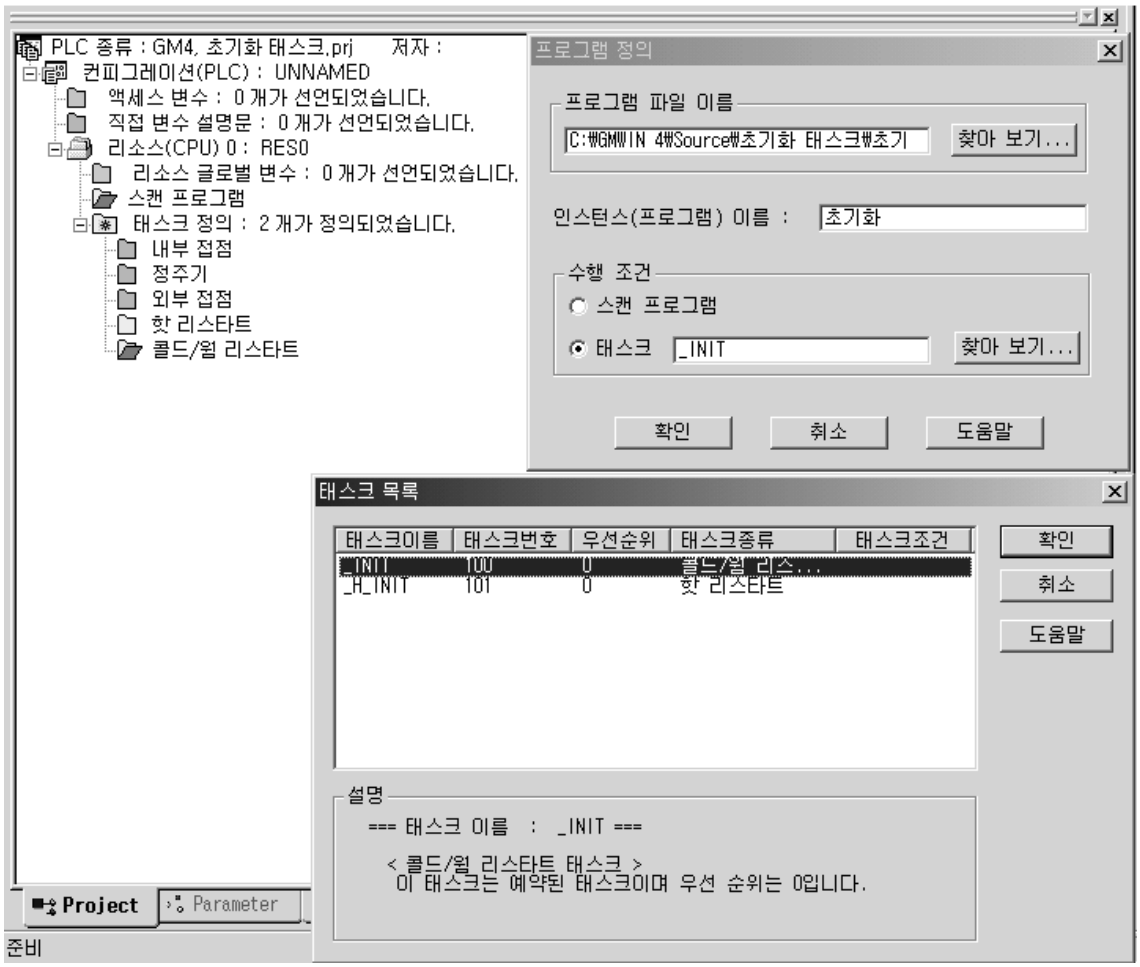
'_INIT_DONE'은 '초기화 프로그램 완료' 플래그입니다.

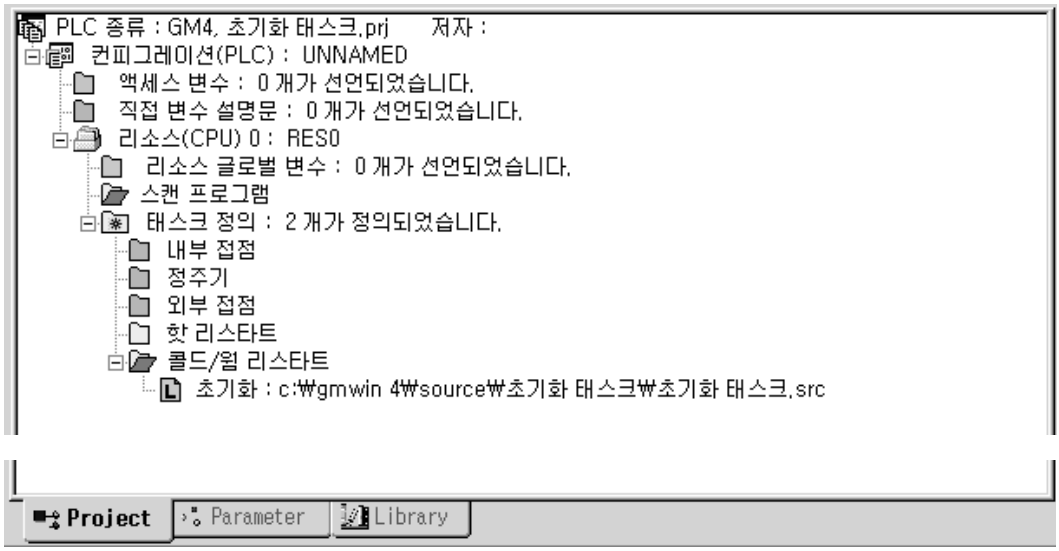


(2) 초기화 태스크 설정

예약된(이미 정의되어 있는) 초기화 태스크(_INIT)를 '프로젝트 항목 추가'를 통해 설정합니다.







(3) 초기화 태스크 프로그램 모니터

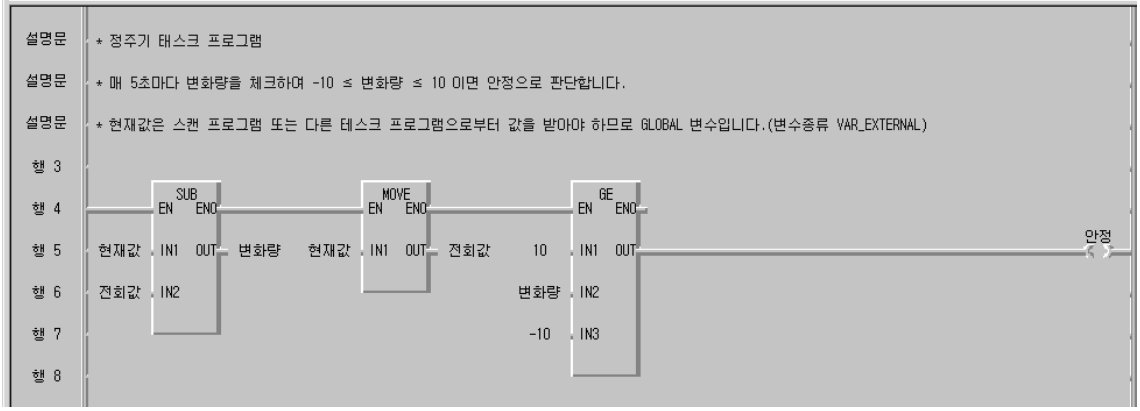
초기화 태스크 프로그램을 실행시켜 동작을 확인합니다.

1.3.2 정주기 태스크 프로그램

(1) 정주기 태스크 프로그램 작성

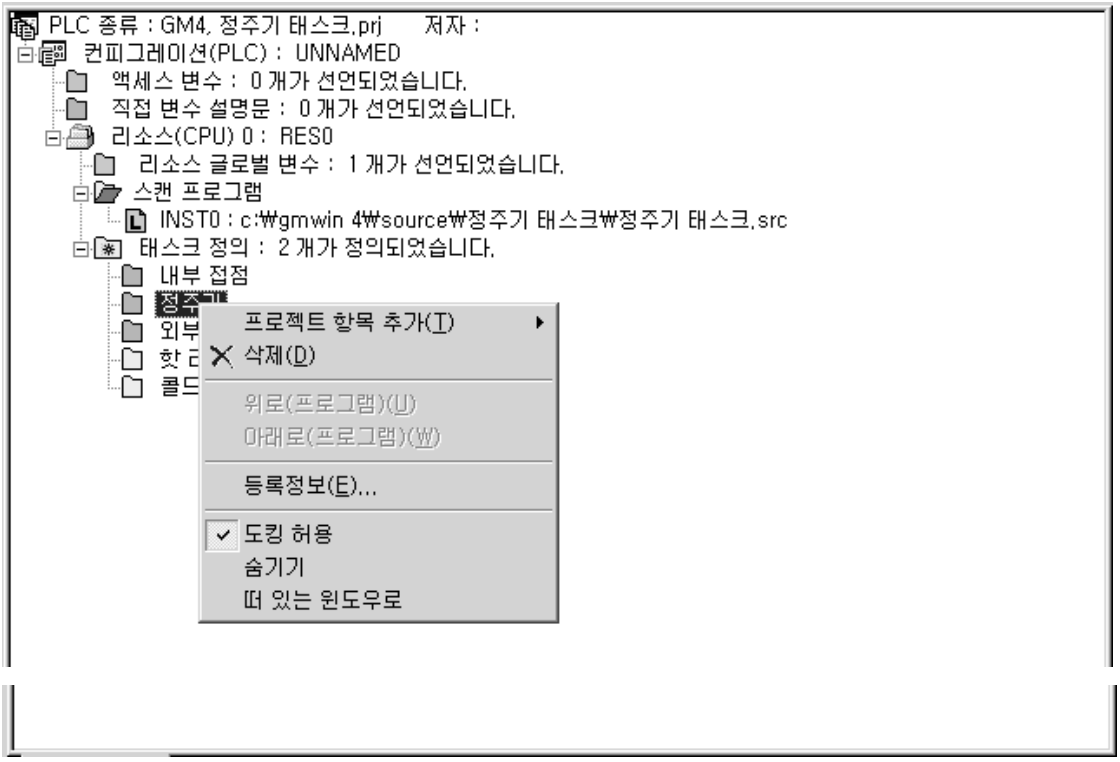
정주기 태스크 프로그램을 작성합니다.

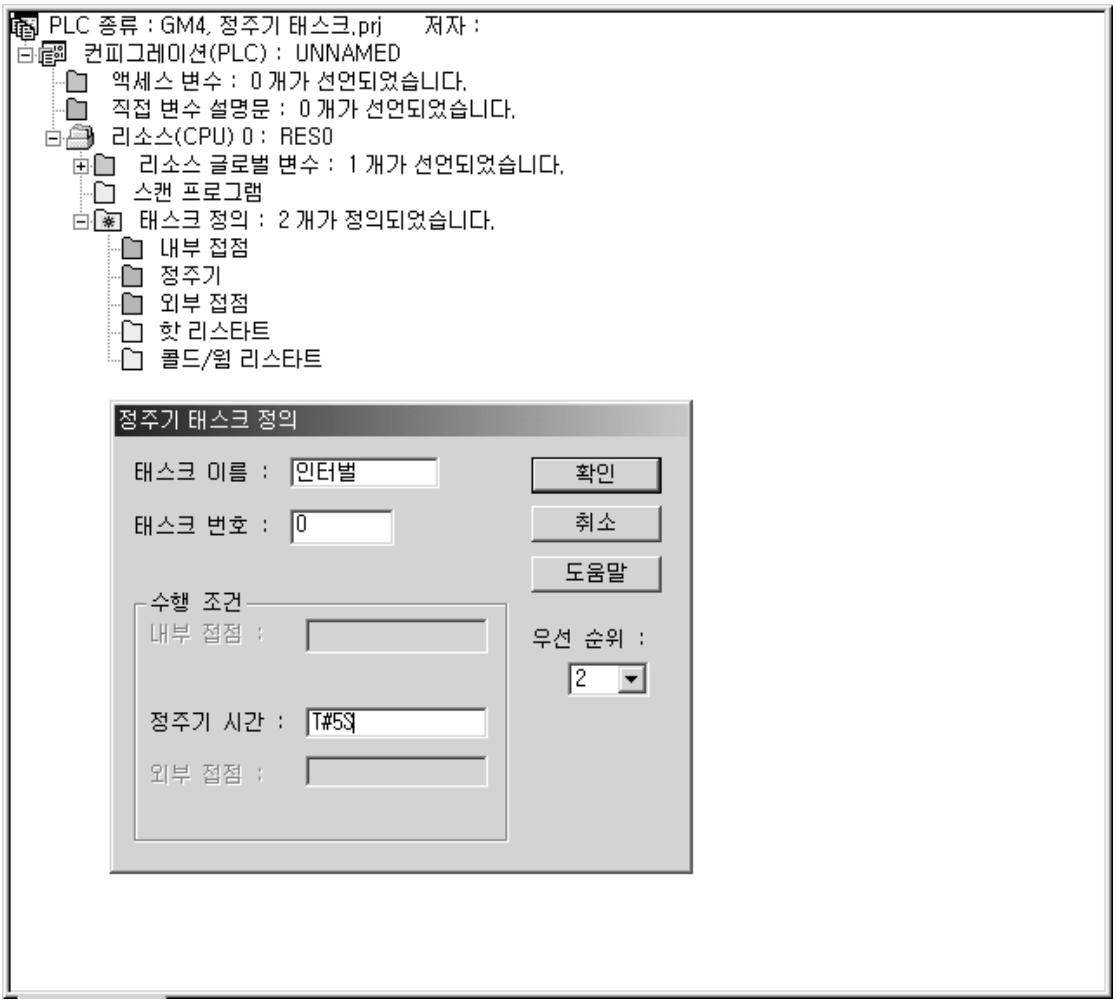
	변수 명	데이터 타입	메모리 할당	초기 값	변수 종류	사용 여부	설명문
1	변화량	INT	<자동>		VAR	*	
2	안정	BOOL	<자동>		VAR	*	
3	전회값	INT	<자동>		VAR	*	
4	현재값	INT	<자동>		VAR_EXTERNAL	*	



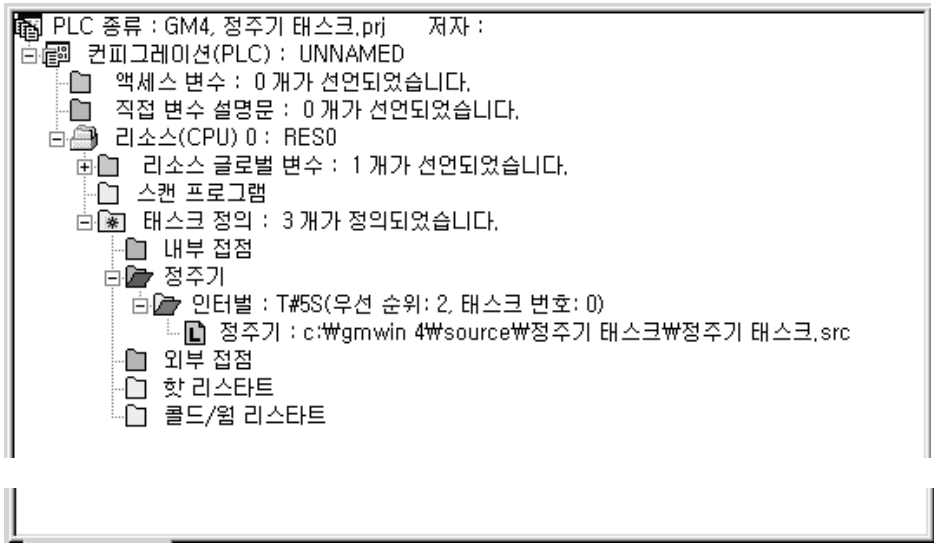
(2) 정주기 태스크 설정

정주기 태스크를 정의하고(선언하고), 기동 조건을 부여합니다.



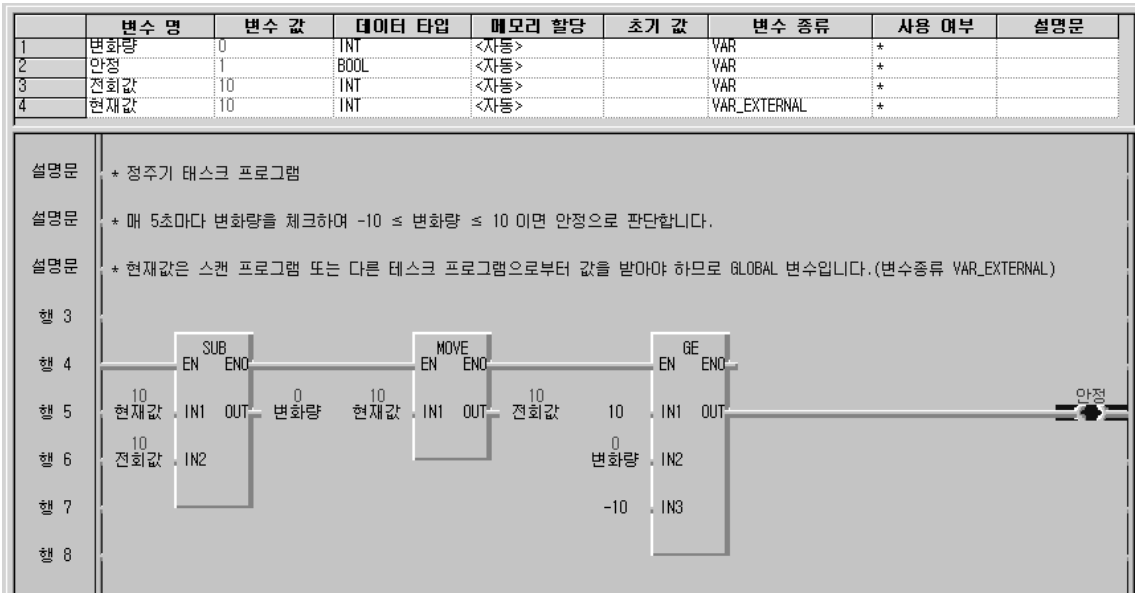

PLC 종류 : GM4, 정주기 태스크.prj 저자 :
컨피그레이션(PLC) : UNNAMED
액세스 변수 : 0 개가 선언되었습니다.
직접 변수 설명문 : 0 개가 선언되었습니다.
리소스(CPU) 0 : RES0
리소스 글로벌 변수 : 1 개가 선언되었습니다.
스캔 프로그램
태스크 정의 : 2 개가 정의되었습니다.
내부 접점
정주기
외부 접점
핫 리스타트
콜드/웜 리스타트
정주기 태스크 정의
태스크 이름 : 인터벌
태스크 번호 : 0
확인
취소
도움말
수행 조건
내부 접점 :
우선 순위 :
2
정주기 시간 : T#5S
외부 접점 :


PLC 종류 : GM4, 정주기 태스크.prj 저자 :
컨피그레이션(PLC) : UNNAMED
액세스 변수 : 0 개가 선언되었습니
직접 변수 설명문 : 0 개가 선언되
리소스(CPU) 0 : RES0
리소스 글로벌 변수 : 1 개가 선
스캔 프로그램
태스크 정의 : 3 개가 정의되었
내부 접점
정주기
인터벌 : T#5S(우선 순위: 2
외부 접점
핫 리스타트
콜드/웜 리스타트
프로그램 정의
프로그램 파일 이름
C:₩GMWIN 4₩Source₩정주기 태스크₩정주
찾아 보기...
인스턴스(프로그램) 이름 :
정주기
수행 조건
스캔 프로그램
태스크
찾아 보기...
확인
취소
도움말
태스크 목록
태스크이름
태스크번호
우선순위
태스크종류
태스크조건
_H_INIT
101
0
핫 리스타트
_INIT
100
0
콜드/웜 리스...
인터벌
0
2
정주기
T#5S
확인
취소
도움말
설명
=== 태스크 이름 : 인터벌 ===
정주기 := T#5S
우선 순위 := 2


PLC 종류 : GM4, 정주기 태스크.prj 저자 :
컨피그레이션(PLC) : UNNAMED
액세스 변수 : 0 개가 선언되었습니다.
직접 변수 설명문 : 0 개가 선언되었습니다.
리소스(CPU) 0 : RES0
리소스 글로벌 변수 : 1 개가 선언되었습니다.
스캔 프로그램
태스크 정의 : 3 개가 정의되었습니다.
내부 접점
정주기
인터벌 : T#5S(우선 순위: 2, 태스크 번호: 0)
정주기 : c:₩gmwin 4₩source₩정주기 태스크₩정주기 태스크.src
외부 접점
핫 리스타트
콜드/웜 리스타트

(3) 정주기 태스크 프로그램 모니터

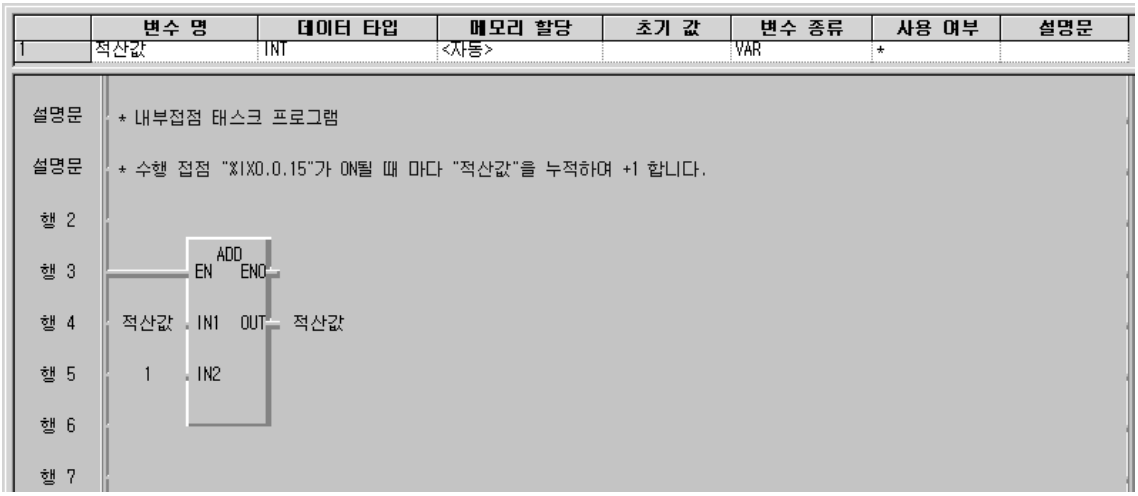
정주기 태스크 프로그램을 실행시켜 동작을 확인합니다.



1.3.3 내부 접점 태스크 프로그램

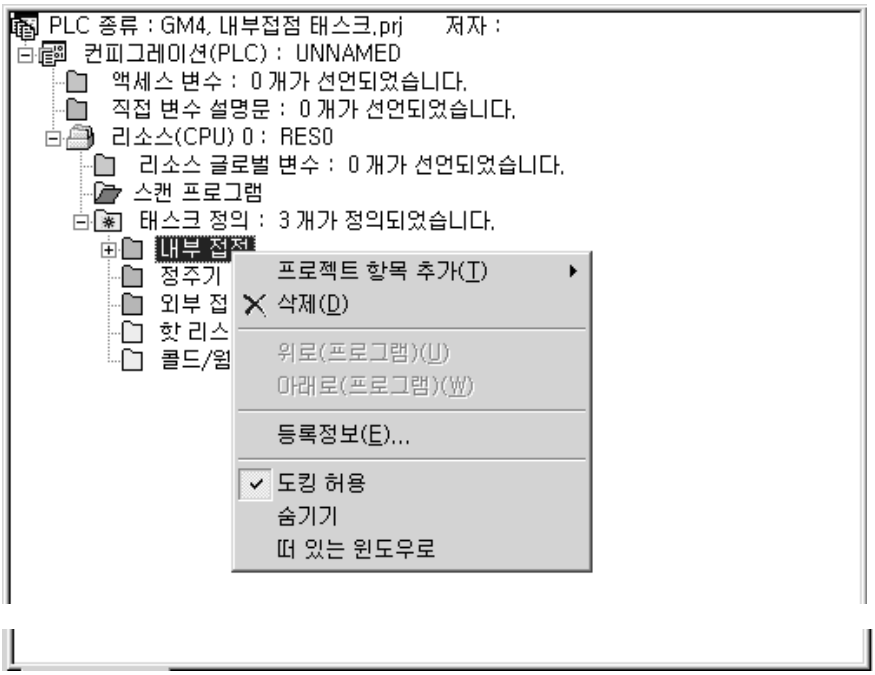
(1) 내부 접점 태스크 프로그램 작성

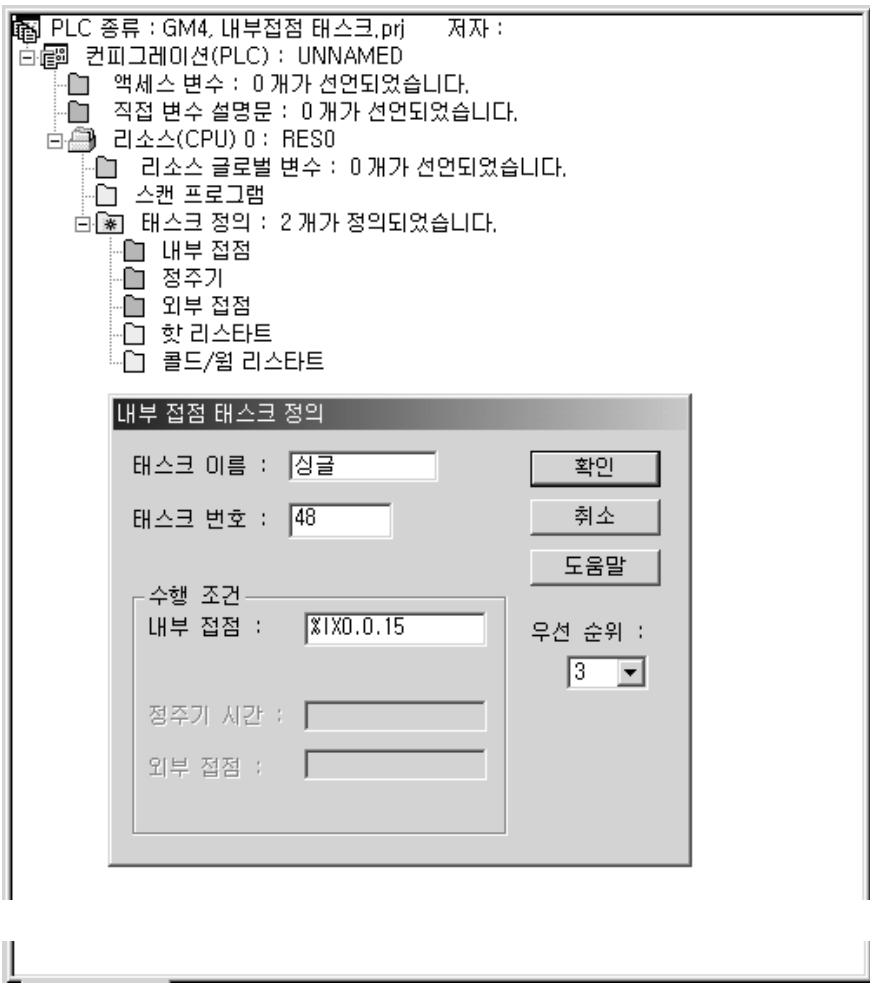
내부 접점 태스크 프로그램을 작성합니다.

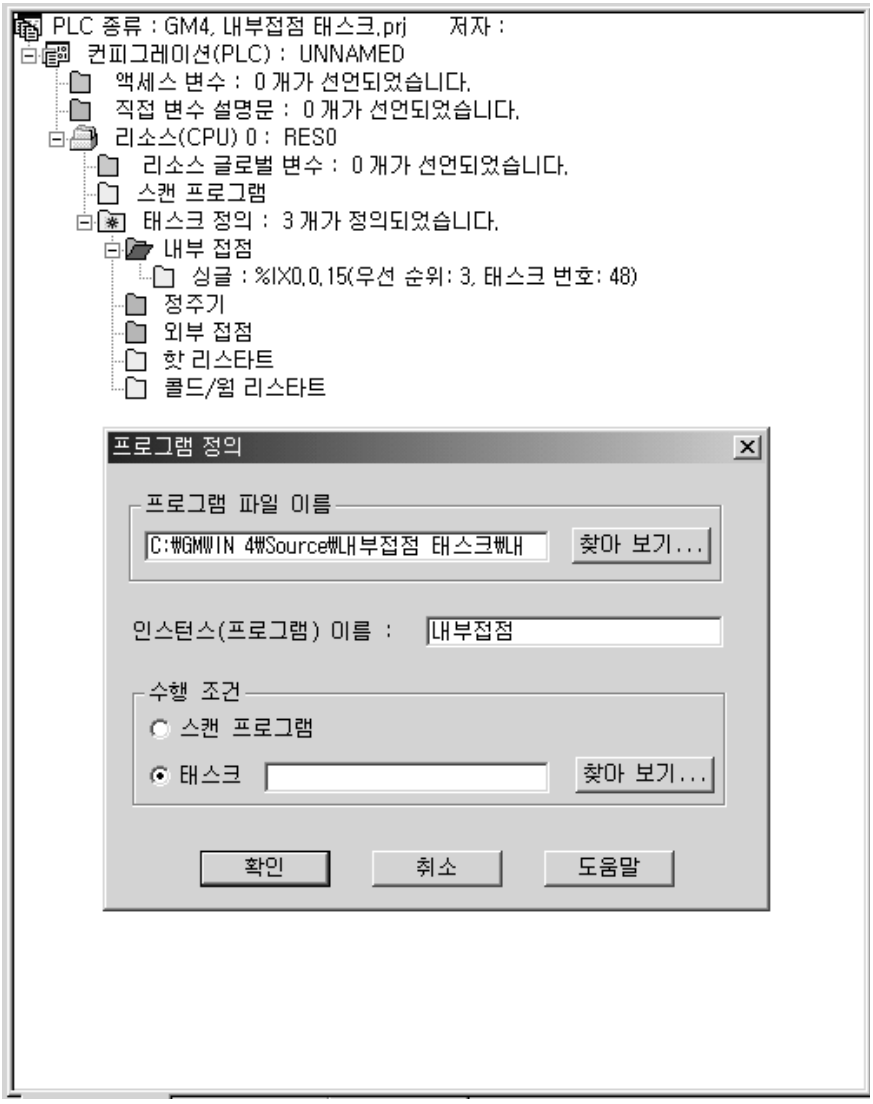


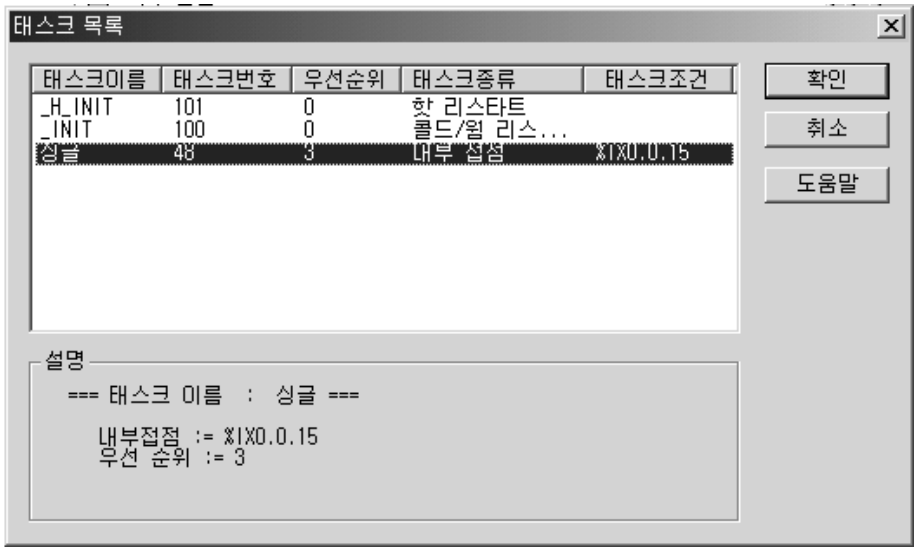
(2) 정주기 태스크 설정

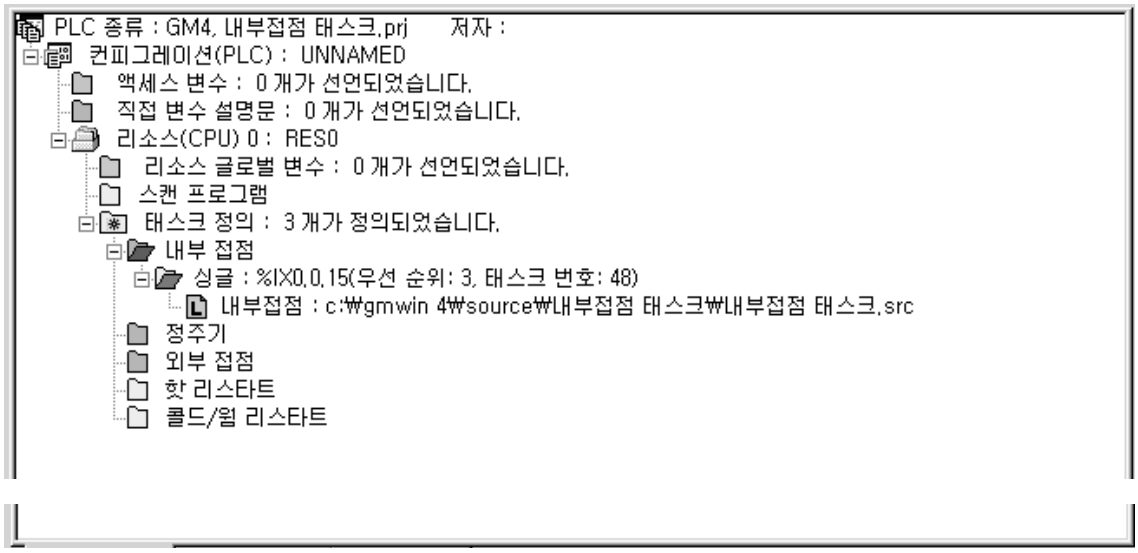
정주기 태스크를 정의하고(선언하고), 기동 조건을 부여합니다.






PLC 종류 : GM4, 내부접점 태스크.prj 저자 :
컨피그레이션(PLC) : UNNAMED
액세스 변수 : 0 개가 선언되었습니다.
직접 변수 설명문 : 0 개가 선언되었습니다.
리소스(CPU) 0 : RES0
리소스 글로벌 변수 : 0 개가 선언되었습니다.
스캔 프로그램
태스크 정의 : 3 개가 정의되었습니다.
내부 접점
싱글 : %IX0.0.15(우선 순위: 3, 태스크 번호: 48)
정주기
외부 접점
핫 리스타트
콜드/웜 리스타트
프로그램 정의
프로그램 파일 이름
C:₩GMWIN 4₩Source₩내부접점 태스크₩내
찾아 보기...
인스턴스(프로그램) 이름 :
내부접점
수행 조건
스캔 프로그램
태스크
찾아 보기...
확인
취소
도움말

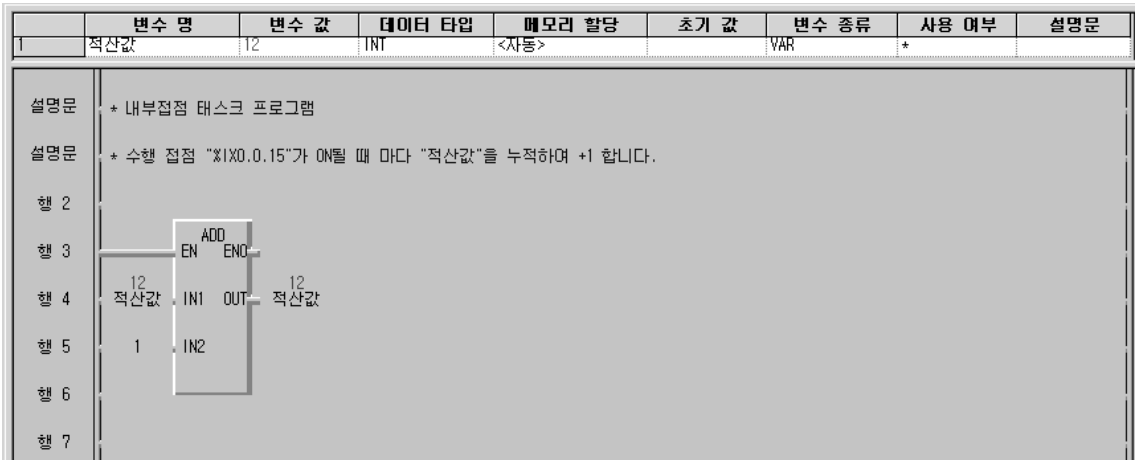

태스크 목록
태스크이름
태스크번호
우선순위
태스크종류
태스크조건
_H_INIT 101 0 핫 리스타트
_INIT 100 0 콜드/웜 리스...
싱글 48 3 내부 접점 %IX0.0.15
확인
취소
도움말
설명
=== 태스크 이름 : 싱글 ===
내부접점 := %IX0.0.15
우선 순위 := 3



(3) 내부 접점 태스크 프로그램 모니터

스캔 프로그램에서 수행 조건 접점 %IX0.0.15 가 ON 되면(상승 에지 검출),

내부 접점 태스크 프로그램이 1 스캔 동작되어 "적산값"이 +1 되는지 확인합니다.



㈜ 내부 접점 태스크에서 수행 조건은 직접 변수 또는 GLOBAL 변수(변수 종류 VAR_EXTERNAL)로 지정해야 합니다. 또한, 데이터 타입은 모두 BOOL 이어야 합니다.

LS PLC 제품 Line-Up

	형명	외형	최대 I/O점수 (리포트 I/O)	처리속도/Step	프로그램 메모리	특 징
XGT Series	XGK-CPUH		6,144점 (32,000점)	0.028μs	64K Step	XGT Series 초고속, Compact, Open Network Solution을 지향하는 신개념의 차세대 PLC • 전용 MPU탑재로 업계 최고 CPU 처리속도 실현 (0.028μs/Step) • 경쟁사 대비 동급 최소 Size • 통합프로그래밍 툴 지원 (XG5000, XG-PD, XG-APM) • Open Network 기반의 System 통합 • 다양한 Network 진단 및 Monitoring 기능 • Device(R, U, Z) 추가 및 Memory 용량 증대 • 구조화 및 Task 프로그래밍 구현 (총 256개) • 최대 16축 Motion 제어
	XGK-CPUH0		3,072점 (32,000점)	0.028μs	32K Step	
	XGK-CPUS		3,072점 (32,000점)	0.084μs	32K Step	
	XGK-CPUS0		1,536점 (32,000점)	0.084μs	16K Step	
MASTER-K Series	K1000S		1,024	0.2μs	30K Step	
	K300S		1,024	0.2μs	15K Step	
	K200S		384	0.5μs	7K Step	MASTER-K Series • 국내 최대의 적용실적 • 간단하고 쉬운 전용명령 지원 (Mnemonic, Ladder) • 다양한 Open Network 지원 (Ethernet, Profibus, DeviceNet) • Windows 환경의 손쉬운 프로그래밍 툴 지원 (KGLWIN)
	K120S		120	0.1μs	10K Step	
	K80S		80	0.5μs	7K Step	
GLOFA-GM Series	GMR		7,680	0.12μs	2M Byte	
	GM1		16,000	0.12μs	512K Byte	
	GM2		4,096	0.12μs	512K Byte	
	GM3		2,048	0.2μs	256K Byte	GLOFA-GM Series • IEC61131-3 국제표준언어 지원 (LD, IL, SFC) • 다양한 Open Network 지원 (Ethernet, Profibus, DeviceNet) • Windows 환경의 손쉬운 프로그래밍 툴 지원 (GMWIN) • 이중화 (GMR CPU) 및 Multi-CPU (GM1 CPU) 기능으로 완벽한 신뢰성 구현
	GM4 *GM4-CPUC기준		3,584	0.12μs	1M Byte	
	GM6		384	0.5μs	68K Byte	
	GM7U		120	0.1μs	132K Byte	
	GM7		80	0.5μs	68K Byte	

LS산전

Leader in Electrics & Automation