

4. 모션 명령어

iM - Σ 시리즈의 모션 프로그램 명령어는 모션 조건을 지정하는 기능 명령, 프로그램 전체 흐름에 대한 제어 명령, 모션 동작 명령, 입출력 명령, 변수 지정 명령 및 내부 연산 명령으로 구성되어 있습니다.

- 1) 기능(Function) 명령 : 모션 동작의 속도, 지연 시간, 가감속 및 전체 좌표의 이동 등을 설정하는 명령어로 구성되어 있습니다.
- 2) 제어(Control) 명령 : 프로그램의 조건 분기, 무조건 분기, 반복 횟수 지정, 서브프로그램 호출, 다른 프로그램 호출 등 전체적인 프로그램 순서를 제어하는 명령어로 구성되어 있습니다.
- 3) 동작(MOVE) 명령 : 모션의 동작 형태(PTP 이동, 증분 이동...)를 설정하는 명령어로 구성되어 있습니다.
- 4) 입출력(I/O) 및 변수(Variable) 명령 : 외부 입력을 받거나 출력을 제어(1Bit, 8Bit)하는 명령어와 프로그램 내부의 정수형, 실수형, 위치형 변수 및 타이머 등에 대한 내부 시스템 변수를 설정할 수 있는 명령어로 구성되어 있습니다.
- 5) 내부 연산 명령 : 삼각함수연산, 지수함수연산, 비트 및 바이트 연산 등

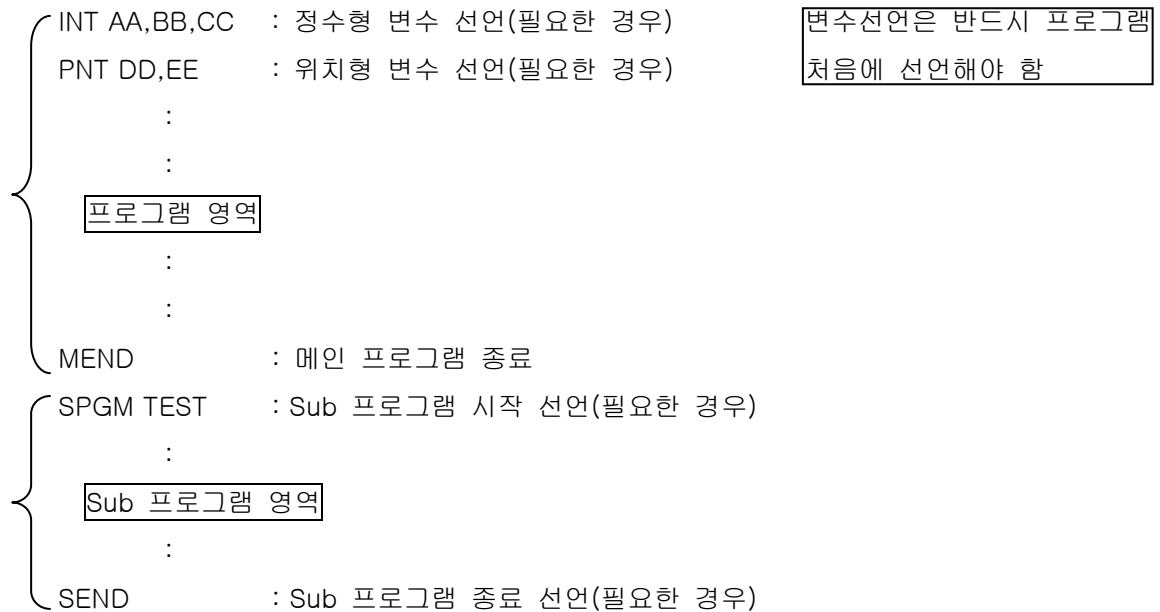
4-1 모션 명령어 화면 구성(Tree 구조)

다음은 모션 프로그램 작성 화면에서의 명령어 Tree 구조입니다.

Group	F1	F2	F3	F4	Menu 이동
FUNC	SPD	WAIT	STBY	PASS	- Group간 이동은 편집 초기 화면 에서 가능 - Group 내에서의 이동은 PgUp, PgDn Key 이용
	ARCH	SHFT	HAND	FIX	
	SRVO	FINE	SCRV	ACCR	
	DECR	XCH	SPDR	PSET	
	SRP0	SRP1	SRQ0	SRQ1	
	SROF	AOUT	AG	AOFS	
	AMAP	TLOF			
CTRL	TAG	GOTO	MEND	STOP	
	LOOP	ENDL	FOR	ENDF	
	IF	ELSE	ENDI	CALL	
	SPGM	SEND	LPMN	LPMI	
	SYNC	ACT	CONT	ENDS	
	LPTN	LPTI	TLMT	TMOD	
MOVE	MPTP	MLIN	MCIR	MARC	
	MPLT	MINC	MILR	MVIO	
	SPLA				
I/O	B	GPNT	GFLT	GINT	

	PNT	FLT	INT	TMR	
	CTR	PATH	PWRK	PCNT	
	PLTP	CURS	CURT0	CURT1	
	CURT2	CURT3	CURT4	CURT5	

4-2 모션 프로그램 작성 구조



4-3 모션 명령어 요약

다음에서 모션 프로그램의 명령어의 종류와 각 명령어의 기능에 대해 요약 설명합니다.

구 성	명 령 어	내 용
Function 명령	SPD	모션 이동의 속도 지정
	WAIT	모션 프로그램 운전 중 대기 시간 지정
	STBY	조건식이 참 일 때 까지 대기
	PASS	모션 이동 중 Inposition 전 다음 위치로 이동 조건 지정
	ARCH	모션 PTP 이동 전 Z축 우선 이동 높이 지정
	SHFT	현재 좌표계에 대한 지정한 값 만큼의 편차 보상
	HAND	SCARA 로봇에서 이동 전 Arm의 형상을 지정
	FIX	보간 이동 중 회전축 자세 고정 및 자세 해제 지정
	SRVO	지정 축 또는 모든 축의 강제 Servo On/Off 지정
	FINE	모션 이동 동작 완료 시 완료 시점의 편차 지정(Pulse)
	SCRV	가감속 시의 Smooth Curve 비율 지정
	ACCR	가속 시간 설정

구 성	명 령 어	내 용
Function 명령	DECR	감속 시간 설정
	XCH	TwinX 기구부 사용 시 축 전환
	SPDR	모션명령어가 실행되는 도중에 속도를 변경
	PSET	현재위치를 임의의 값으로 변경
	SRP0~1	좌표계 회전 시 기준 좌표 설정
	SRQ0~1	좌표계 회전 시 변경후 좌표 설정
	SROF	좌표계 회전 안함
	AOUT	아날로그 출력에 사용될 값의 종류 설정
	AG	아날로그 출력계산의 게인 설정
	AOFS	아날로그 출력계산의 옴셋 설정
	AMAP	아날로그 카드의 GINT 인덱스 설정
	TLOF	Tool 옴셋 설정
Control 명령	TAG	GOTO 문에 의해 분기될 꼬리표 지정
	GOTO	지정한 TAG 문으로 무조건 분기
	MEND	주 프로그램 종료
	STOP	동작 정지 또는 프로그램 정지 설정
	LOOP~ENDL	사용한 조건이 참일 때 무한 반복 실행
	FOR~ENDF	조건 반복 실행 설정
	IF~ENDI	조건 판단 실행을 설정
	CALL	지정한 서브 프로그램 호출
	SPGM~SEND	서브 프로그램 시작
	LPMN,LPMI	모션 프로그램 실행 중 다른 모션 프로그램 Load
	SYNC~ENDS	모션 이동 중 조건 판단의 시작을 설정
	ACT	모션 이동 중 조건 판단이 참일 경우의 실행문을 설정
	CONT	조건이 걸린 지점으로 복귀하여 이동 계속
	LPTN,LPTI	다른 포인트 파일에서 포인트 데이터를 Load
	TLMT	모션 이동 중 이동속의 토크 제한치 설정
	TMOD	모션 이동 중 이동속의 설정 토크 감지시 동작 설정
Motion 명령	MPTP	원점 기준 현재 위치에서 설정한 값만큼 PTP 이동
	MLIN	원점 기준 현재 위치에서 설정한 값만큼 직선 보간 이동
	MCIR	현재 위치에서 두 점의 설정값을 경유하는 원 보간 이동
	MARC	현재 위치에서 설정값을 경유, 목표값에 위치하는 원호 보간이동
	MPLT	기준 위치에서 설정된 Pallet에 의한 Palletizing 이동
	MINC	현재 위치에서 설정한 값만큼 PTP 증분 이동
	MILR	현재 위치에서 설정한 값만큼 직선 보간 증분 이동
	MVIO	외부 입력에 의한 포인트 No. 결정 이동
	SPLA	연속 원호보간 명령어

구 성	명 령 어	내 용
I/O 명령	B	접점 처리 명령
	GPNT	전역 위치형 변수
	GFLT	전역 실수형 변수
	GINT	전역 정수형 변수
	PNT	위치형 변수 선언
	FLT	실수형 변수 선언
	INT	정수형 변수 선언
	TMR	SEQ 프로그램에서 사용하는 타이머의 값
	CTR	SEQ 프로그램에서 사용하는 카운터의 값
	PATH	모션 명령어의 이동 비율을 저장하고 있는 읽기 전용 변수입니다.
	PLTP	직전 Pallet 동작의 위치 지령을 기억하고 있는 변수 입니다.
	PCNT	Pallet 동작을 위한 Pallet ID의 Count 설정
	PWRK	Pallet 동작을 위한 Pallet ID의 개수 설정
	CURS	현재의 이동 속도를 저장하고 있는 읽기 전용 변수 입니다.
	CURT0~5	현재의 축별 토크를 저장하고 있는 읽기 전용 변수 입니다.

참고 사항	각 채널의 모션 및 시퀀스 프로그램에서 공용으로 사용할 수 있는 전역변수
<p>1. 전역 위치형 변수 : GPNT(0) ~ GPNT(255)</p> <ul style="list-style-type: none"> - 각 변수당 6개의 실수형 변수 저장 : GPNT(0).1 ~ GPNT(0).6 - 값의 범위 : -99999.999 ~ 99999.999 - 읽기,쓰기,사칙 연산 및 내부 연산 명령에 사용 가능 - 채널 별 현재 위치 표시 <ul style="list-style-type: none"> ◆ GPNT(252) : 채널1의 현재 위치 ◆ GPNT(253) : 채널2의 현재 위치 ◆ GPNT(254) : 채널3의 현재 위치 ◆ GPNT(255) : 채널4의 현재 위치 - GPNT(252)~GPNT(255)는 각 채널의 현재 위치를 표시하므로 모션 프로그램에서는 읽기만 가능합니다. - GPNT(252).1 : 채널1의 첫 번째 축의 현재 위치를 나타냅니다. <p>2. 전역 실수형 변수 : GFLT(0) ~ GFLT(255)</p> <ul style="list-style-type: none"> - 값의 범위 : -99999.999 ~ 99999.999 - 읽기,쓰기,사칙 연산 및 내부 연산 명령에 사용 가능 <p>3. 전역 정수형 변수 : GINT(0) ~ GINT(255)</p> <ul style="list-style-type: none"> - 값의 범위 : 0 ~ 99999 	

- 읽기,쓰기,사칙 연산 및 내부 연산 명령에 사용 가능

4. 전역 점점 변수 : B(0) ~ B(255)

- 각 변수당 8bits로 구성 : B(0).0 ~ B(0).7
- 읽기,쓰기,Bit 및 Byte 연산 가능
- 상세 내용은 **6-1-1 입출력 점점 사용**을 참조

5. 각 변수의 불 휘발성 영역입니다.

- GPNT(148~249) : 102 개
- GFLT(200~249) : 50 개
- GINT(200~249) : 50 개
- B(210~219, 230~239) : 20 bytes

4-3 모션 명령어 설명

4-3-1 기능(Function) 명령

SPD

기능 설명

1. 기준 속도 대비 모든 축의 이동 속도를 설정합니다.
2. 이동 명령에 따라 참조하는 기준 속도가 다릅니다
 - 보관 동작(MLIN, MCIR)의 경우 : Channel - Common - **Basic Spd** 파라미터
 - PTP 동작(MPTP, MINC)의 경우 : Channel - Axis - **Ref RPM** 파라미터
3. 속도 데이터 입력 범위 : $1 \leq \text{설정값} \leq 10000$

입력 형식

SPD=5000 ; 이동 속도를 50%로 설정합니다.

SPD=A123 ; 프로그램 내에서 지정한 정수형 변수 A123의 값으로 속도를 설정합니다.

SPD=GINT(5) ; 전역 정수형 변수 GINT5에서 지정한 값으로 속도를 설정합니다.
; 전역 변수(Global 변수)는 모두 괄호 사용. 배열 취급. 따라서 GINT(A)사용

SPD=A123+100 ; A123에 100을 더한 값에 따라 속도를 설정합니다.

SPD=A123*1.1 ; A123의 10[%] 증가한 값으로 속도를 설정합니다.

SPD=SPD*2 ; 현재의 속도를 2배로 설정합니다.

Note

1. SPD=정수형 변수
 - 프로그램 내부에서 설정한 정수 변수에 있는 값이 이동 속도로 됩니다. 만약 사용하고 자 하는 정수형 변수가 프로그램 초기에 선언되어 있지 않으면 프로그램을 컴파일 할 때 문법 오류의 에러가 발생합니다.
 - 정수형 변수의 값이 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
2. SPD=정수형 전역 변수(GINT(xxx))
 - 전역 변수에서 설정한 GINT(xxx)의 값이 이동 속도로 지정됩니다.
 - 정수형 변수의 값이 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
3. SPD=연산식
 - 연산의 결과 값이 이동 속도로 지정됩니다. 연산의 결과가 실수이면 내부적으로 정수로 변환되어 사용됩니다.
 - 연산의 결과가 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.

WAIT

기능 설명

1. 프로그램 진행 중 설정시간 만큼 대기합니다.
2. 대기 시간은 “설정 시간 x 10[ms]”로 결정됩니다.
3. 시간 데이터 범위 : $1 \leq \text{설정값} \leq 10000$ [x 10ms]

입력 형식

WAIT=100	; 1[sec] 동안 대기합니다.
WAIT=ABCD	; 프로그램 내부에서 지정한 정수형 변수 ABCD에서 설정한 시간만큼 대기합니다.
WAIT=GINT(2)	; 전역 정수형 변수 GINT(2) 에서 설정한 시간만큼 대기합니다.
WAIT=GINT(2) * 10	; 전역 정수형 변수 GINT(2)에 10을 곱한 시간만큼 대기합니다.

Note

1. WAIT=상수
상수로 설정한 값 X 10[ms]의 시간 만큼 대기합니다.
2. WAIT=정수형 변수
 - 프로그램 내부에서 설정한 정수 변수에 있는 값이 대기 시간으로 지정됩니다. 만약 사용하고자 하는 정수형 변수가 프로그램 초기에 선언되어 있지 않으면 프로그램을 컴파일 할 때 문법 오류의 에러가 발생합니다.
 - 정수형 변수의 값이 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
3. WAIT=정수형 전역 변수(GINT(xxx))
 - 정수형 전역 변수에서 설정한 GINT(xxx)의 값이 대기 시간으로 지정됩니다.
 - 정수형 전역 변수의 값이 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
4. WAIT=연산식
 - 연산의 결과 값이 대기 시간으로 지정됩니다.
 - 연산의 결과가 범위(1~10000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.

STBY

기능 설명

프로그램 진행 중 조건이 참이 될 때 까지 대기합니다.

입력 형식

STBY B(0).1==1	; B(0).1이 1(On)이 될 때까지 대기합니다.
STBY B(1)==0B1010-----	; Port 010이 1010-----이 될 때까지 대기합니다. 1이 MSB(B(1).7)이고, ‘-’ 는 don’t care 비트입니다.
STBY B(1)==0XA-	; 0B1010----- 를 16진수로 표시한 경우 입니다.
STBY GINT(3)!=GINT(5)	; 전역 정수형 변수 GINT(3)과 GINT(5)의 값이 다를 때까지 대기합니다.
STBY GINT(5)>10	; 전역 정수형 변수 GINT(5)가 10보다 클 때까지 대기합니다.
STBY AB.0>10.123	; 위치형 변수 AB의 첫 번째 인자의 값이 10.123보다 클 때까지 대기합니다.
STBY GPNT(252).0 > 12.345	; 채널1의 첫 번째 축의 위치가 12.345보다 클 때까지 대기합니다.

Note

1. STBY 접점 변수 조건
 - 비트 단위(B(0).0)나 바이트 단위(B(1))를 모두 사용할 수 있습니다.
 - 2진수(0B10101010)나 16진수(0XAA)를 모두 사용할 수 있습니다.
 - 조건으로 대기하지 않는 접점은 don't care(-)를 사용하면 됩니다.
2. STBY 비교 부호 사용
 - 비교 부호는 ==(EQ), !=(NE), >(GT), >=(GE), <=(LE), <(LT)를 사용할 수 있습니다.
3. STBY 에서 사용한 조건이 항상 거짓이면 무한히 대기하므로 주의해서 사용하십시오.
4. STBY 에서 사용한 조건이 일정 시간이 지나도 만족하지 않을 때 STBY에서 빠져 나오려면 SYNC와 SEQ 프로그램에서 TMR 명령을 사용하면 됩니다.

예) 모션 프로그램

B(10).0 = 1	: SEQ 프로그램 타이머 시작 접점
SYNC	: SYNC 블록의 시작
STBY B(0).0 == 1	: B(0).0 이 1(ON) 될 때 까지 대기
ACT TMR(0) > 1000	: 대기 시간이 5초(1000*5ms)가 지나면
GOTO EXIT	: STBY 조건에서 빠져나감
ENDS	

예) SEQ 프로그램

LOAD B(10).0

LOAD B(10).0

TMR(0) B(10).1 <D> 2000

PASS

기능 설명

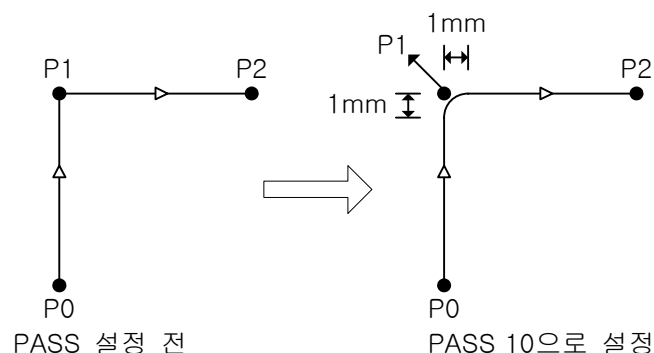
1. 비율로 설정할 경우 이동 중 목표점의 비율까지 도달하면 다음 목표점으로 이동합니다.
2. 거리로 설정할 경우 이동 중 목표점의 거리까지 도달하면 다음 목표점으로 이동합니다.
3. 비율 및 거리의 기준은 끝점을 기준으로 합니다.
4. 비율 데이터 입력 범위 : $1 \leq \text{설정값} \leq 99$ [%]
5. 거리 데이터 입력 범위 : $1 \leq \text{설정값} \leq 999$ [0.1mm]

입력 형식

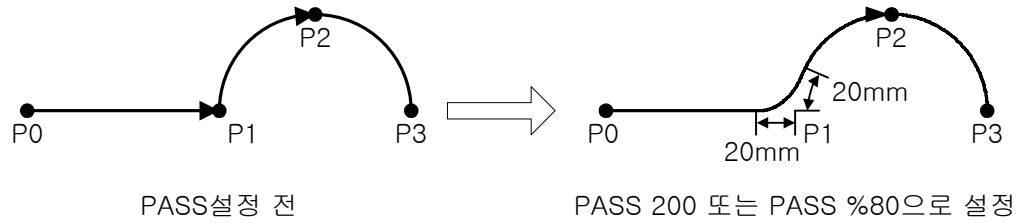
PASS 10	; 목표점 기준 1mm 전에서 다음 목표점으로 이동합니다.
PASS %90	; 목표점 기준 90%까지 진행한 후 다음 목표점으로 이동합니다.
PASS AB	; 정수형 변수로 선언된 AB 값에 의해 거리 값을 설정합니다.
PASS %AB	; 정수형 변수로 선언된 AB 값에 의해 거리 비율을 설정합니다.
PASS OFF	; 설정된 PASS 값을 해제합니다.

Note

1. PASS 명령어 사용에서 설정값에 %가 있을 경우에는 비율로 적용되고 없을 경우에는 거리로 적용됩니다. %로 설정할 경우 최소값은 50입니다.
2. PASS에 대한 설정값은 다른 PASS 명령을 만나기 전까지 유지됩니다.
3. 비율로 설정할 경우에는 MPTP일 경우에만 적용되고, 다른 이동 명령에서는 적용되지 않습니다.
4. MPTP는 MPTP와만 PASS연결됩니다. MPTP와 MLIN, MCIR, MARC가 연결 될 때에는 MPTP모션이 완전히 정지한 후 다음 모션을 시작합니다.
5. MLIN-MCIR, MCIR-MLIN, MLIN-MARC, MARC-MLIN 연결 시에는 SPLINE 궤적으로 움직입니다. MLIN-MLIN 연결 시에는 ARC 궤적으로 이동합니다. SPLINE은 PASS시점과 종점에서 속도 벡터가 연속이 되도록 하여 부드럽게 이동하는 방법입니다.
6. 직선 보간 이동을 할 때 PASS 10으로 설정될 경우 ARC궤적을 따라 다음과 같이 이동합니다.



7. MLIN-MCIR, MCIR-MLIN, MLIN-MARC, MARC-MLIN 연결 시에는 3차 spline곡선 궤적을 따라 이동합니다.



8. 궤적의 변화 없이 끊어짐 없는 연속된 모션을 구현하기 위해서는 DECR명령을 사용하여 감속 시간을 최소로 줄여줍니다. 자세한 내용은 DECR명령을 참조하십시오.

ARCH

기능 설명

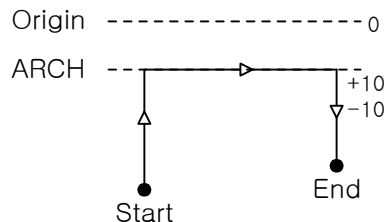
1. 현재 위치에서 다음 목표점으로 이동 전에 원점을 기준으로 Z축이 설정값 만큼 상승한 후 이동합니다.
2. ARCH 데이터 입력 범위 : $1 \leq \text{설정값} \leq \text{Z축 이동 범위 [mm]}$

입력 형식

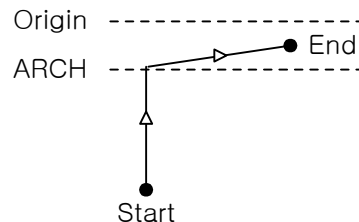
ARCH=10 ; 다음 위치 이동 전 Z축 상승을 원점 기준 10mm까지 상승 후 이동합니다.
 ARCH=AB ; 정수형 변수로 선언된 AB 값에 의해 ARCH 값을 설정합니다.
 ARCH=0 ; 설정된 ARCH 값을 해제합니다.
 ARCH=ARCH*2 ; 현재 설정의 2배로 ARCH 값을 설정합니다.

Note

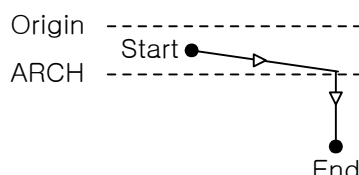
1. ARCH 설정값은 다음 ARCH 명령을 만나기 전까지 유지됩니다.
2. PTP 모션을 하는 경우에만 적용되고 보간 이동 명령에서는 무시됩니다.
(MPTP, MPLT, MINC 모션에 적용됨)
3. Z축으로 설정되어 있는 경우에만 동작합니다.
4. 원점 기준 이동 시작점과 이동 목표점의 Z축 위치가 ARCH 설정값보다 클 경우 [그림 1]처럼 동작합니다.
5. 원점 기준 이동 목표점의 Z축 위치가 ARCH 설정값보다 작은 경우 [그림 2]처럼 동작합니다.
6. 원점 기준 이동 시작점의 Z축 위치가 ARCH 설정값보다 작은 경우 [그림 3]처럼 동작합니다.
7. 원점 기준 이동 시작점과 이동 목표점의 Z축 위치가 ARCH 설정값보다 작은 경우 [그림 4]처럼 동작합니다.



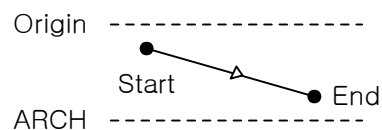
[그림 1]



[그림 2]

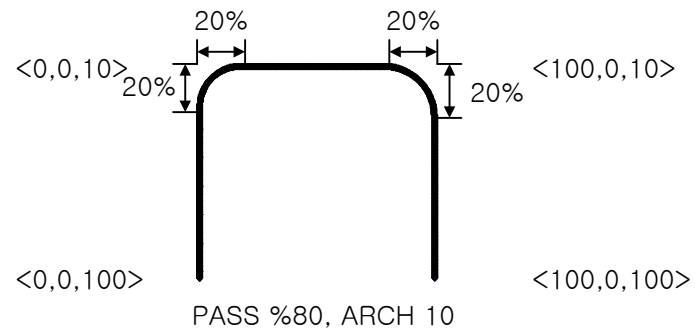


[그림 3]



[그림 4]

8. PASS와 같이 사용하면 아래 그림과 같이 움직이게 되므로 이동 시간을 단축할 수 있습니다.



SHFT

기능 설명

1. 모션 이동할 때 좌표계의 위치를 편차 보상하여 이동할 경우 사용합니다.
2. SHFT 데이터로는 포인트, 전역 위치형 변수, 위치형 변수, 실수를 사용할 수 있습니다.

입력 형식

SHFT=P10 ; 포인트 P10 에 저장되어 위치 만큼 편차 보상 이동합니다.
 SHFT=AB ; 위치형 변수 AB로 선언된 위치에 지정한 값 만큼 편차 보상 이동합니다.
 SHFT=<10.0,5.0> ; 1축은 10.0, 2축은 5.0 만큼 편차 보상 이동합니다.
 SHFT=SHFT*2 ; 현재 SHFT 값의 2배로 위치 편차보상을 설정합니다.
 SHFT=<0> ; SHFT 명령에서 사용된 보상 위치를 초기 값으로 반환합니다.(해제)

Note

1. 임의의 포인트 No.(0 ≤ 포인트 No. ≤ 999)에 편차를 설정하고 SHFT 명령을 사용할 경우 SHFT 명령 다음의 이동은 저장된 편차 값만큼 보상하여 이동합니다.
2. SHFT 명령에 의해 편차 보상하는 값은 해당 프로그램에서만 유효합니다.
3. SHFT 명령에 의해 편차 보상되는 모션 이동은 모든 이동 명령에 유효합니다.
4. Twin X 일 경우 X1 축 상의 작업 위치만을 Teaching한 후 X2 축 상의 위치는 SHFT 명령을 사용하여 프로그램을 간략화 할 수 있습니다.
5. SHFT 명령을 사용한 위치 변화의 예는 다음과 같습니다.

예) 편차 보상할 위치를 P100 에 저장하고 있다고 가정할 때

P100 = X : 0.65	Y : 100.12	Z : -0.13
P5 = X : 10.00	Y : 15.00	Z : 16.00
P6 = X : 14.03	Y : 18.12	Z : 5.43

프로그램 예	이동 후 위치
:	:
L008 MPTP P5	/* 10.00, 15.00, 16.00 */
L009 MPTP P6	/* 14.03, 18.12, 5.43 */
L010 SHFT=P100	
L011 MPTP P5	/* 10.65, 115.12, 15.87 */
L012 MPTP P6	/* 14.68, 118.24, 5.30 */
:	

FIX

기능 설명

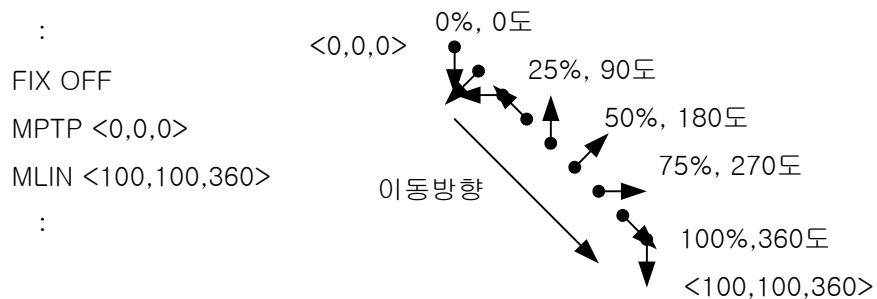
1. 보간 동작할 경우 도구 축의 자세 고정 여부를 선택합니다.
2. FIX 데이터 입력 범위 : ON 또는 OFF
3. 기본값 : ON

입력 형식

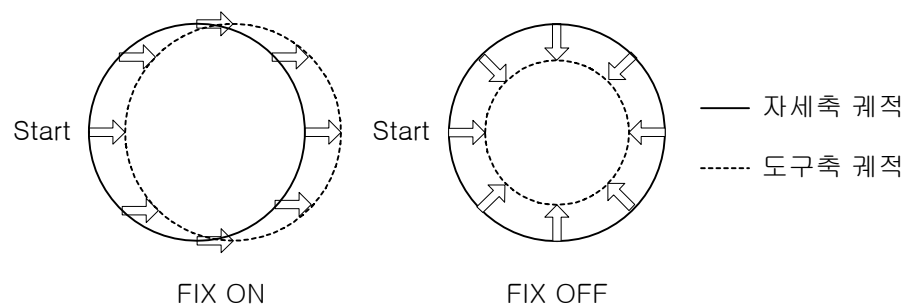
FIX ON ; 보간 동작할 때 도구 축의 자세를 고정합니다.
 FIX OFF ; 보간 동작할 때 도구 축의 자세 고정을 해제합니다.

Note

1. 로봇의 축은 자세 축과 도구 축으로 분류되며 모든 회전축과 4축 이상의 기계에서 4번째 이상의 축은 도구 축으로 취급됩니다. XYZT와 TWINX로봇에서 4번째 축도 도구 축으로 취급됩니다. CP(MLIN,MCIR,MARC) 이동 중 도구 축의 이동을 ON/OFF하며 PTP 이동에서는 적용되지 않습니다.
2. FIX OFF시 도구 축은 자세 축의 이동 비율에 따라 이동합니다.
 예) XYR기구부에서 다음 모션 프로그램은 아래 그림과 같이 동작합니다.



3. MCIR의 경우에는 회전축의 경우 도구 축의 목표점이 현재 좌표에서 360도를 더한 양으로 정해지고 직선 축의 경우 2번째 경유 점의 좌표를 목표 좌표 값으로 갖습니다. 이를 이용하면 아래 그림과 같이 응용할 수 있습니다.

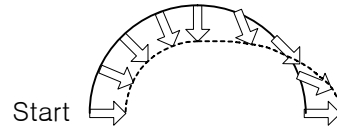


4. MARC에서는 도구 축이 경유 점을 거쳐서 이동합니다.

예) XYR기구부에서 다음 모션 프로그램 실행 시 아래 그림과 같이 이동합니다.

```

:
FIX OFF
MPTP <0,0,0>
MARC <50,50,90>,<100,0,0>
:
    
```

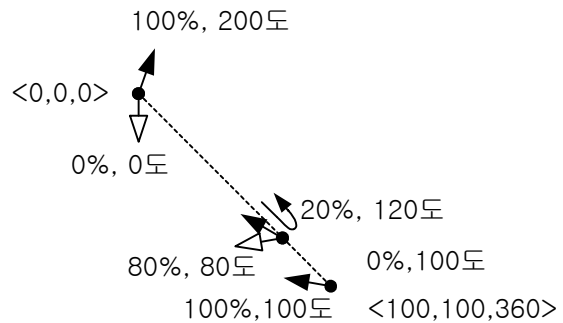


5. 이동 비율이 급격히 바뀌는 경우, 도구 축의 이동 속도가 매우 빠르게 되므로 Following에러가 발생할 수 있습니다. 예를 들어 자세 축은 고정하고 도구 축만 움직이고 싶은 경우 MLIN명령을 사용하면 이동 비율이 0%에서 바로 100%로 바뀌므로 R축이 바로 목표점으로 이동하게 되므로 Following에러가 발생합니다. 이 경우, MPTP명령을 사용하여 주십시오. PASS명령을 사용하는 경우에도 Following에러가 발생할 수 있는데, 그 예는 다음과 같습니다.

예) XYR기구부에서 다음 프로그램 실행시 R축이 80도에서 120도로 순간적으로 움직이게 되므로 Following에러가 발생합니다.

```

:
MPTP <0,0,0>
PASS %80
MLIN <100,100,100>
MLIN <0,0,200>
:
    
```



SRVO

기능 설명

1. 모션 동작 중 해당 축을 강제로 Servo On/Off 합니다.
2. SRVO 데이터 입력 범위 : 0,1

입력 형식

SRVO (1,0,1,1) ; 1, 3, 4축을 강제 Servo On, 2축을 강제 Servo Off 합니다.
 SRVO (,0,,0) ; 2, 4축을 강제 Servo Off, 1,3축은 현재 상태를 유지 합니다.

Note

1. 모션 동작 중 SRVO 명령에 의하여 해당 축이 Servo Off가 된 후, 다음의 모션 이동 명령을 만날 경우 “Motion Error”가 발생합니다. 이럴 경우 다시 SRVO 명령에 의하여 해당 축을 Servo On 한 후 이동 명령을 사용해야 합니다.
2. 상태 변경을 원하지 않는 축은 공백으로 하면 됩니다.

FINE

기능 설명

1. 모션 이동 동작 완료 시 완료 시점의 위치 편차 정도를 지정합니다.
2. FINE 데이터 입력 범위 : $1 \leq \text{설정값} \leq 5000[\text{pulse}]$

입력 형식

FINE 10 ; 완료 시점의 위치 편차를 10[pulse] 이내로 지정합니다.
 FINE K ; 완료 시점의 위치 편차를 정수형 변수 K 값에 의해 지정합니다.
 FINE OFF ; 편차량을 기본값(파라미터에서 설정한 값)으로 되돌립니다.

Note

1. 모션 이동 후 목표점에 설정 값 이내로 도달하는 정도를 설정합니다.
2. 로봇 이동 명령의 실행은 FINE 설정 값 이내로 Positioning된 후 다음 이동 명령을 수행하므로 허용 정도를 만족하지 않으면 다음 이동 명령을 수행하지 않습니다.
3. 설정 값을 작게 주면 줄수록 Cycle Time이 길어지므로 정밀한 작업이 아닌 경우에는 사용하지 마십시오.
4. 다음의 FINE 값을 만나기 전까지 유지 됩니다.

SCRV

기능 설명

1. 모션 이동에서 가감속 할 때 충격 완화 시간을 설정합니다.
2. SCRV 데이터 입력 범위 : $0 \leq \text{설정값} \leq 100$ (1일 때 5[ms]로 동작합니다.)

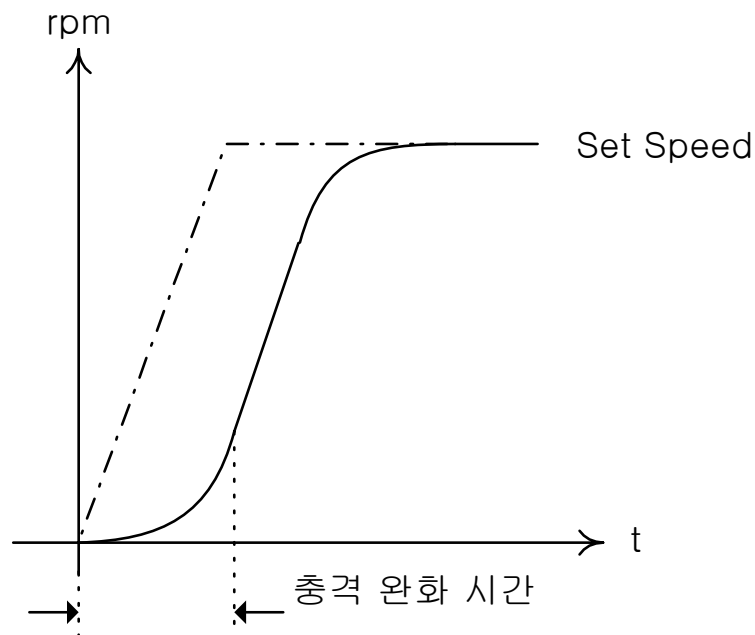
입력 형식

SCRV 5 ; 충격 완화 시간을 25[ms]로 설정합니다.

SCRV K ; 충격 완화 시간을 정수형 변수 K에 의해 지정된 값으로 설정합니다.

Note

1. 일반적인 모션의 형태는 가속, 등속, 감속의 사다리꼴 속도 패턴을 갖고 동작하는데, SCRV 명령을 사용하면 가속 및 감속시의 기울기를 조절할 수 있습니다.
2. 이 명령을 사용하지 않을 경우에는 기본적으로 사다리꼴 속도 패턴을 갖은 모션으로 동작합니다.
3. 사용에 대한 예는 다음과 같습니다.



4. 위 그림에서 보는 것처럼 충격 완화 시간을 설정할 경우에 사다리꼴 가속 패턴에서 S 자 형태의 가속 모션으로 동작합니다.

ACCR

기능 설명

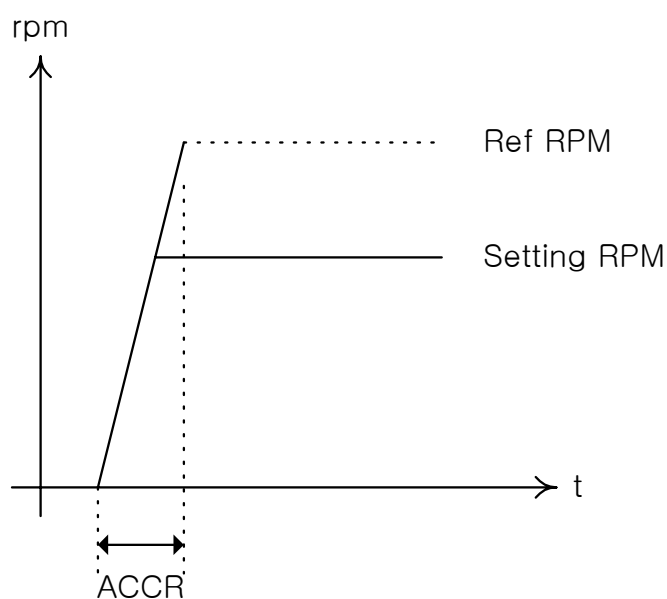
1. 모션 이동을 시작할 때 가속 시간을 설정합니다.
2. ACCR 데이터 입력 범위 : $1 \leq \text{설정값} \leq 5000$ [5ms]

입력 형식

ACCR 100 ; 가속 시간을 500[ms]로 설정합니다.
 ACCR ABC ; 가속 시간을 정수형 변수 ABC에 의해 지정된 값으로 설정합니다.
 ACCR GINT(5) ; 가속 시간을 전역 정수형 변수 GINT5에서 지정된 값으로 설정합니다.

Note

1. 가속 시간의 기준은 MPTP의 경우 Channel - Axis - P - RefRPM이고, CP(MLIN,MCIR,MARC)의 경우 Channel - Common - BasicSpd 에서 설정한 속도에 도달할 때까지 소요되는 시간입니다.(아래 그림 참조)
2. 변수에 의해 가속 시간을 설정할 경우, 변수에 저장된 값이 범위(1~5000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
3. 이 명령을 사용하지 않을 경우 모션 동작에서는 Channel - Common - MotionAcc 파라미터에서 설정한 값이 적용됩니다.
4. 이 명령은 해당 채널에서 사용하는 모든 축에 적용됩니다.
5. ACCR 명령은 다음 ACCR 명령을 만나기 전까지 유지됩니다.
6. ACCR 1로 설정한 경우, 한 sampling시간(5[ms])동안에 Setting속도로 가속합니다.



DECR

기능 설명

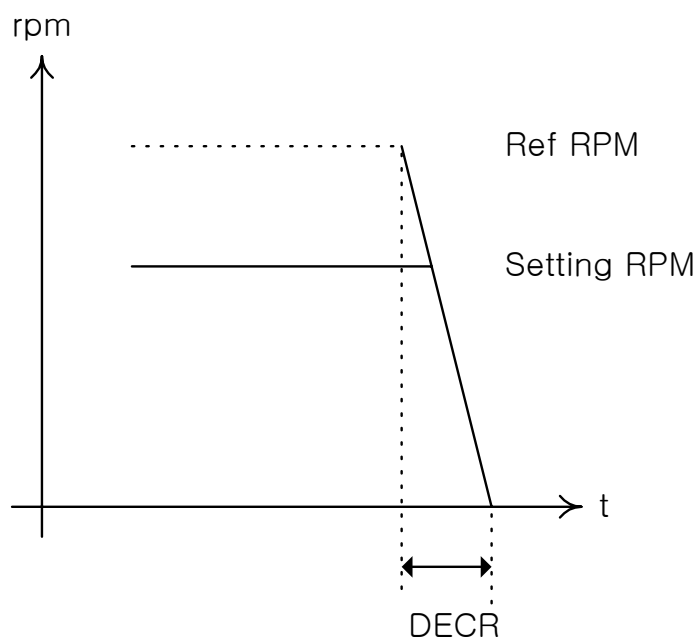
1. 모션 이동에서 정지할 때 감속 시간을 설정합니다.
2. DECR 데이터 입력 범위 : $1 \leq \text{설정값} \leq 5000$ [5ms]

입력 형식

DECR 100 ; 감속 시간을 500[ms]로 설정합니다.
 DECR ABC ; 감속 시간을 정수형 변수 ABC에 의해 지정된 값으로 설정합니다.
 DECR GINT(5) ; 감속 시간을 전역 정수형 변수 GINT(5)에서 지정된 값으로 설정합니다.

Note

1. 감속 시간의 기준은 MPTP의 경우 Channel - Axis - P - RefRPM이고, CP(MLIN,MCIR,MARC)의 경우 Channel - Common - BasicSpd 에서 설정한 속도에 도달할 때까지 소요되는 시간입니다.(아래 그림 참조)변수에 의해 가속 시간을 설정할 경우, 변수에 저장된 값이 범위(1~5000)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
2. 이 명령을 사용하지 않을 경우 모션 동작에서는 Channel - Common - MotionDec 파라미터에서 설정한 값이 적용됩니다.
3. 이 명령은 해당 채널에서 사용하는 모든 축에 적용됩니다.
4. DECR 명령은 다음 DECR 명령을 만나기 전까지 유지됩니다.
5. DECR 1로 설정할 경우, 한 sampling시간(5[ms])동안 한 번에 감속합니다. 다음에 모션 명령어가 올 경우 끊어짐이 없이 연속된 모션을 수행할 수 있습니다.



HAND

기능 설명

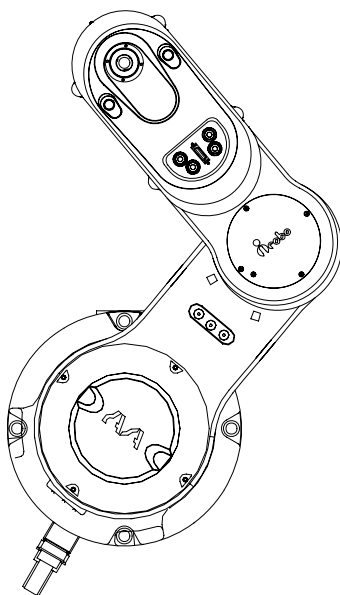
1. SCARA 로봇에서 이동 전 Arm의 형상을 지정합니다.
2. HAND 데이터 입력 범위 : LP 또는 RP 또는 OFF

입력 형식

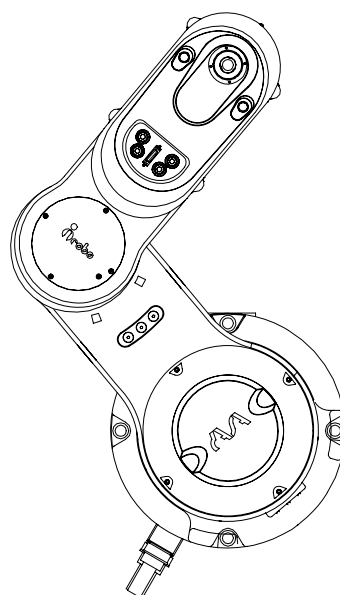
HAND LP ; 이동 전 Arm의 형상을 Left 위치로 합니다.
 HAND RP ; 이동 전 Arm의 형상을 Right 위치로 합니다.
 HAND OFF ; Arm 형상 지정을 해제합니다.

Note

1. SCARA 로봇의 원통 좌표계에는 ARM의 형태에 따라 두 가지의 형상(Left Arm / Right Arm)이 있습니다.
2. 이동 명령 수행에서 SCARA 로봇의 ARM 형태를 HAND 명령을 사용하여 임의로 지정할 수 있습니다.
3. 형상은 SCARA 로봇의 뒤에서 볼 때 사람의 팔 형상과 동일합니다.
4. SCARA 로봇을 Teaching할 때 Teaching 방법에 따라 다음과 같이 설정합니다.
 - 1) MDI(Manual Direct Input) Teaching : Arm의 형상을 입력해야 합니다.
 - 2) DTI(Direct Teaching Input) Teaching : Arm의 형상이 Teaching 위치에 따라 자동 입력 됩니다.
 - 3) 모션 프로그램에서 HAND 명령을 사용할 경우에는 기존 Teaching에서 지정되었던 Arm의 형상은 무시됩니다.



(Right Arm 형상)



(Left Arm 형상)

XCH

기능 설명

1. 기계 구조가 TwinX 이거나 TwinZ일 경우, 이동 축을 지정합니다.
2. XCH 데이터 입력 범위 : 0 or 1

입력 형식

XCH 0 ; TwinX인 경우 X1 축을, TwinZ인 경우 Z1축을 이동 축으로 지정합니다.
 XCH 1 ; TwinX인 경우 X2 축을, TwinZ인 경우 Z2축을 이동 축으로 지정합니다.

Note

1. 이 명령을 사용하는 기계 구조는 반드시 당사 제품의 TwinX 또는 TwinZ 로봇인 경우에만 적용됩니다.
2. 이 명령과 Servo Off 명령인 “**SRVO**”와 같이 사용할 경우 다음의 예와 같이 사용합니다.
 예) X2축에서 X1축으로 전환하여 작업하는 경우

:

SRVO(,,0) // X2또는 Z2축을 Servo Off 합니다.

WAIT=20 // 20[ms]을 대기합니다.

XCH 0 // X1또는 Z1축으로 이동 축을 지정합니다.

:

SPDR

기능 설명

1. 이동명령이 실행되는 중간에 바로 속도를 변경합니다.
2. 속도변경중에는 **ACCR, DECR**의 가감속 명령어가 적용됩니다.
3. 이동명령의 속도는 SPD명령, Override 1(TBOX및 터치판넬), Override 2(SPDR명령)등 3가지 값의 곱으로 결정됩니다.
4. 데이터 입력 범위 : $0.001 \leq \text{설정값} \leq 1.0$

입력 형식

SPDR=0.5 ; 50% 속도로 이동합니다.
 SPDR=F ; F 변수의 값을 적용합니다.
 SPDR=GFLT(0) ; GFLT(0)의 값을 적용합니다.
 SPDR=SPDR*2 ; 현재 속도의 2배로 이동 속도를 설정합니다.

Note

1. **SYNC** 명령어와 같이 사용하여 조건이 만족되면 속도를 바로 변경합니다.
 :

```

    SYNC          // SYNC명령을 사용하여 이동 중에 조건을 판단하도록 합니다.
    MPTP P0
    ACT B(0).0==0 // B(0).0이 0이 되면
    SPDR=1.0      // 100%속도로 이동합니다.
    ACT B(0).1==1 // B(0).0이 1이 되면
    SPDR=0.5      // 50%속도로 이동합니다.
    ENDS
    :
```

PSET

기능 설명

1. 현재의 위치를 변경합니다.
2. PSET 데이터 입력 범위 : 포인트 변수 또는 상수

입력 형식

PSET <100> ; 현재 위치의 좌표값을 100으로 설정합니다.
PSET P0 ; 현재 위치의 좌표값을 P0로 설정합니다.
PSET GPNT(0) ; 현재 위치의 좌표값을 GPNT(0)로 설정합니다.

Note

1. 기계 구조가 R일 경우에만 사용 가능합니다.
2. 다음의 예와 같이 사용합니다.

예) 한 방향으로 무한히 회전

:

TAG A

MPTP <100> // 0에서 100으로 이동합니다.

PSET <0> // 좌표 100을 0으로 변경합니다.

GOTO A

:

SRP0~1

SRQ0~1

SROF

기능 설명

1. 좌표계 회전 명령어 입니다.
2. SRP0 ~ 1 : 좌표계 변경시 의 기준 2포인트를 지정합니다.
3. SRQ0 ~ 1 : 좌표계 변경시 의 변경 후 2포인트를 지정합니다.
4. SROF : 회전 후 변경된 보상 위치를 초기 값으로 반환합니다.(해제)
5. 데이터 입력 범위 : 포인트 변수 또는 상수

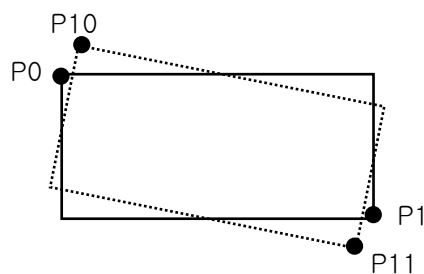
입력 형식

SRP0=P0 ; 좌표계 회전 시의 변경전 시작 기준 포인트를 P0 로 설정합니다.
 SRP1=P1 ; 좌표계 회전 시의 변경전 끝 기준 포인트를 P1 으로 설정합니다.
 SRQ0=P10 ; 좌표계 회전 시의 변경후 시작 기준 포인트를 P10 으로 설정합니다.
 SRQ1=P11 ; 좌표계 회전 시의 변경후 끝 기준 포인트를 P11 로 설정합니다.
 SROF ; 회전 후 변경된 보상 위치를 초기 값으로 반환합니다.(해제)
 SRP0=<0,0> ; 좌표계 회전 시의 변경전 시작 기준 포인트를 <0,0> 으로 설정합니다.
 SRP1=<100,100> ; 좌표계 회전 시의 변경전 끝 기준 포인트를 <100,100> 으로 설정합니다.

Note

1. 작업할 대상물의 위치가 변경(회전/이동) 되었을 때 좌표계 회전 명령어를 이용하여 변경된 좌표에 대한 새로운 티칭없이 작업이 가능합니다.
2. 다음의 예와 같이 사용합니다.(실선의 사각형 작업영역이 점선의 사각형 영역으로 회전)

SRP0=P0 // 변경전 기준점1
 SRP1=P1 // 변경전 기준점2
 SRQ0=P10 // 변경후 기준점1
 SRQ1=P11 // 변경후 기준점2



AOUT

AG

AOFS

AMAP

기능 설명

1. 아날로그 카드로 현재 속도와 토크 등을 출력하고자 할 때 사용합니다.
2. 아날로그 카드가 없는 경우에도 현재 속도 모니터링 기능으로 사용가능합니다.
3. 속도의 단위는 [mm/s]이고 토크의 단위는 정격대비 %입니다. 속도의 경우 PTP가 아닌 CP(직선,원,원호)보간인 경우에만 속도가 계산됩니다.
4. 모션 프로그램에서 한 번 설정된 경우 CP모션을 수행할 경우 모션 프로그램을 종료해도 계속 동작합니다. 서보 OFF시는 동작하지 않습니다.
5. AOUT은 출력값의 종류를 설정합니다.
 - 0 - 사용안함
 - 1 - 속도
 - 2 - 축0 토크
 - 3 - 축1 토크
 - 4 - 축2 토크
 - 5 - 축3 토크
 - 6 - 축4 토크
 - 7 - 축5 토크
6. 아래와 같은 식으로 계산됩니다.

$$GINT(AMAP) = \text{출력값} \times AG + AOFS$$

입력 형식

AG 1.0	; 계인을 1로 설정
AOFS 0.0	; 옴셋을 0으로 설정
AMAP 1	; 아날로그 카드의 출력에 사용되는 GINT번호를 설정
AOUT 1	; 속도출력으로 설정

Note

1. AOUT을 0이 아닌 값으로 설정하기 전에 AG,AOFS,AMAP의 값을 원하는 값으로 설정하여 주십시오.
2. AMAP에 사용하는 값을 결정하기 위해서는 아날로그 카드의 파라미터 D/A Out값을 참조하여 주십시오.

TLOF

기능 설명

1. Z축 끝단에 장착되어 있는 Tool 선택 명령어 입니다.
2. Tool 번호에 해당하는 Tool 의 형태는 파라미터 Channel - Common - ToolOfsX(Y,Z)0 ~ ToolOfsX(Y,Z)2 에서 설정합니다.
3. 데이터 입력 범위 : 0 ~ 2

입력 형식

- TLOF 1 ; 파라미터에 설정된 1 번 툴을 선택합니다.
 TLOF GINT(0) ; 전역정수형변수 GINT(0) 에 있는 번호의 툴을 선택합니다.

Note

1. 작업시 사용하는 툴의 형태를 선택합니다.
2. 툴의 형태는 파라미터 Channel - Common - ToolOfsX(Y/Z)0 ~ ToolOfsX(Y/Z)2 에서 설정합니다.
3. 툴 형상 설정에 관한 상세한 내용은 **7-4-2 Channel Group** 을 참조하십시오

4-3-2 제어(Control) 명령

TAG ~ GOTO

기능 설명

1. Tag(꼬리표)를 설정하고, GOTO문에 의해 해당 Tag(꼬리표)로 무조건 분기합니다.
2. TAG 명은 문자, 숫자 조합 8자 이내 입니다.(숫자만의 사용은 불가)
3. TAG는 SYNC~ENDS 안에서는 사용할 수 없습니다.(SYNC 안으로 GOTO 불가)

입력 형식

TAG ABCDED ; ABCDED 이름을 가진 Tag를 설정합니다.
 TAG G12 ; G12 이름을 가진 Tag를 설정합니다.
 GOTO ABCDED ; ABCDED 이름을 가진 Tag로 무조건 분기합니다.
 GOTO G12 ; G12 이름을 가진 Tag로 무조건 분기합니다.

Note

1. 한 개의 프로그램 내에서는 동일한 Tag명을 사용할 수 없습니다.
2. Main 프로그램에서 서브 프로그램으로 분기할 수 없습니다.
3. 서브 프로그램에서 Main 프로그램으로 분기할 수 없습니다.
4. 서브 프로그램에서 다른 서브프로그램으로 분기할 수 없습니다.
5. GOTO문에 의해 분기되는 Tag가 없을 경우 Compile시 에러가 발생합니다.
6. TAG ~ GOTO 문 사용에 대한 프로그램 구조는 다음과 같습니다.



MEND

기능 설명

1. 모션 프로그램 작성에서 Main 프로그램의 끝을 선언합니다.
2. 부 명령어 없이 단독으로 사용합니다.

입력 형식

MEND ; 모션 프로그램 작성에서 Main 프로그램 끝을 선언합니다.

Note

1. 모션 프로그램의 Main 프로그램 끝에서 반드시 선언하여야 합니다.
2. MEND 명령어는 컨트롤러에 Main 프로그램의 끝을 선언하고 서브 프로그램의 시작을 알립니다.(서브 프로그램이 있을 경우)
3. 서브 프로그램이 있을 경우 반드시 MEND 다음부터 작성해야 합니다.

LPTN, LPTI

기능 설명

1. 모션 프로그램 실행 중 다른 포인트 파일을 Load 합니다.
2. 사용할 수 있는 부 명령어는 이미 작성되어 있는 포인트 파일 이름입니다.(LPTN)
3. 사용할 수 있는 부 명령어는 이미 작성되어 있는 포인트 파일 번호입니다.(LPTI)

입력 형식

LPTN DASA ; 모션 프로그램 실행 중 DASA.PNT 파일을 Load 합니다.
 LPTI 5 ; 모션 프로그램 실행 중 파일 번호가 5인 파일을 Load 합니다.
 LPTI II ; 모션 프로그램 실행 중 정수형 변수 II의 값을 번호로 가지는 파일을 Load 합니다.

Note

1. LPTN,LPTI 명령 사용 이후의 이동은 Load된 포인트 파일에 의해 이동합니다.
2. 원 상태로 복귀하기 위해서는 현재 실행하고 있는 모션 프로그램의 파일명을 다시 Load 해야 합니다.
3. LPTN 사용시 파일명은 이미 저장되어 있는 파일로써, 첫째자리가 문자를 포함해서 8자 이내 입니다.
4. LPTI 명령어 사용시는 파일 번호로 숫자 또는 정수형 변수를 사용할 수 있고 값은 0~99 사이 이어야 합니다.
5. 사용중인 포인트(Pxxx가 Load된 포인트 파일에 저장되어 있지 않은 경우 모션 프로그램 운전 중 에러가 발생합니다.
6. 저장된 포인트의 개수가 많을수록 실행시간이 오래 걸리게 되고 이 때에는 RS232C 통신을 포함한 다른 Background작업이 중단됩니다. 따라서 터치 또는 PC와의 통신이 잠시 끊어질 수 있습니다. 100개 정도의 포인트를 포함하는 포인트 파일을 로드할 경우 0.2초 정도가 소요되고, 1000개의 포인트를 포함하는 포인트 파일을 로드할 경우 2~3초 정도가 소요됩니다.

LOOP ~ ENDL

기능 설명

1. LOOP 다음에 오는 조건이 참일 경우 ENDL까지 무한 반복 실행합니다.
2. 사용할 수 있는 조건은 상수, 논리 연산식 또는 비교 연산식입니다.

입력 형식

LOOP 1	; 무한 반복 실행합니다.
LOOP B(0).0==1	; 점점 B(0).0이 1이면 LOOP 실행
LOOP B(0).1==1&&B(5).0==1	; 점점 B(0).1과 B(5).0이 1이면 LOOP 실행
LOOP B(0).4==1 B(0).3==1	; 점점 B(0).4 또는 B(0).3이 1이면 LOOP 실행
LOOP B(0) == 0XA-	; 점점 B(0)의 상위 비트가 0XA 이면 LOOP 실행
LOOP GINT(5)==10	; 전역 정수형 변수 GINT(5)가 10이면 LOOP 실행
LOOP ABC==10	; 정수형 변수 ABC가 10이면 LOOP 실행
LOOP TMR(0)<200	; TMR(0)의 값이 200(200*5[ms])보다 작으면 LOOP 실행
LOOP CTR(0)>20	; CTR(0)의 값이 20보다 크면 LOOP 실행
ENDL	; 하나의 LOOP문을 종료합니다.

Note

1. 한 프로그램 내에서 여러 개의 LOOP 명령을 사용할 수 있습니다.
2. 조건이 참인 LOOP 문 내에서 IF 조건을 사용하여 LOOP문을 빠져 나올 수 있습니다.
3. 컨트롤러는 LOOP 조건 판단을 한 후, 조건이 거짓일 경우 가장 근접하게 사용된 ENDL 명령을 실행하고 다음 스텝을 실행합니다.
4. LOOP ~ ENDL 사용에 대한 프로그램 구조는 다음과 같습니다.

```

:
L005 LOOP 1
:
L020 ENDL
:
L030 LOOP B(0).1==1&&B(5).0==1
:
L035 LOOP B(0).3==1
:
L055 ENDL
L066 ENDL

```

무한 반복 실행(빠져 나오기 위해서 GOTO 명령 사용)

블록 #1

블록 #2

이 조건이 거짓일 경우 L066 라인 실행

FOR ~ ENDF

기능 설명

1. 시작 값부터 카운터하여 종료 값을 만족할 때까지 반복 실행 합니다.
2. 시작 값과 종료 값은 정수형 변수로 지정합니다.

입력 형식

FOR I=1 TO 10 ; FOR 루프를 1부터 시작하여 10회 실행합니다.
 FOR IS=1 TO IE ; FOR 루프를 1부터 시작하여 IE가 지정하는 수 만큼 실행합니다.
 FOR I2=1 TO GINT(3) ; 전역 정수형 변수 GINT(3)에서 지정하는 수 만큼 실행합니다.
 ENDF ; 하나의 FOR 문을 종료합니다.

Note

1. 다음과 같은 입력 형식에서 [조건 #1]은 시작 값을 지정하고, [조건 #2]는 종료 값을 지정합니다.
예) FOR [조건 #1] TO [조건 #2]
2. 정수 변수의 선언은 Main 프로그램의 변수 선언 위치(시작 위치)에서 합니다.(정수형 변수를 Main 프로그램 내부에서 지정할 경우)
3. 정수 변수는 1회 FOR ~ ENDF 문을 실행할 때마다 자동적으로 1씩 증가합니다.
4. 1 이외의 수로 증가를 원할 경우 변수 가산 명령을 사용하여 프로그램으로 처리해야 합니다.
5. 한 개의 프로그램 안에서 여러 개의 FOR ~ ENDF 문을 사용할 수 있습니다.
6. END ~ ENDF 문의 중첩은 3회까지 사용 가능합니다.
(Ver. 1.4.10.10부터 5회까지 사용가능)
7. FOR ~ ENDF 문에 대한 사용 예는 다음과 같습니다.

```

      :
L015  FOR I2=1 TO 10      }
      :                  } L025까지 10회 반복 실행
L025  ENDF
L026  FOR I3=1 TO J
      :
L030  FOR I4=1 TO 10      }
      :                  } L041까지 5회
L040  I4=I4+2             } 반복 실행
L041  ENDF
L042  ENDF
  
```

L042까지 J값 만큼 반복 실행

IF ~ { ELSE ~ } ENDI

기능 설명

1. IF 명령 다음에 오는 조건이 참이면 이하를, 거짓이면 ELSE 이하를 실행합니다.
2. 사용할 수 있는 조건은 논리 연산식 또는 비교 연산식입니다.

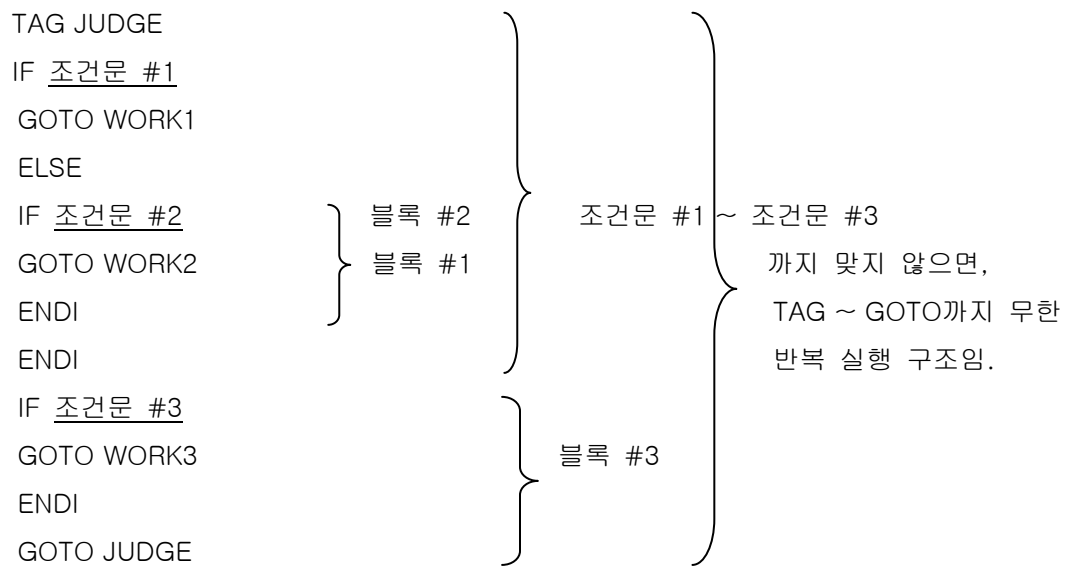
입력 형식

IF B(0).0==1 ; 접점 B(0).0이 1이면,
 IF (B(0).1==1)&&(B(5).0==1) ; 접점 B(0).1과 B(5).0이 1이면,
 IF (B(0).4==1)|| (B(0).3==1) ; 접점 B(0).4 또는 BIT(0).3이 1이면,
 IF B(0) == 0B1010---- ; 접점 B(0)의 상위 4비트가 1010 이면
 IF B(0) == 0XA- ; B(0) == 0B1010---- 의 16진수 표현
 IF GINT(5)==10 ; 전역 정수형 변수 GINT(5)가 10이면,
 IF ABC!=10 ; 정수형 변수 ABC가 10이 아니면,
 IF AB1.2>=50.00 ; 위치형 변수 AB1의 3번째 축이 50보다 같거나 크면,
 IF GPNT(252).0 > 123.456 ; 채널1의 1축의 위치가 123.456보다 크면,
 IF (GPNT(252).0 > 10.0) && ; 채널1의 1축의 위치가 10.0 ~ 50.0 이면,
 (GPNT(252).0 < 50.0)
 IF TMR(0)>=200 ; TMR(0)의 값이 200(200*5[ms])보다 크거나 같으면,
 IF CTR(0)<20 ; CTR(0)의 값이 20보다 작으면
 ENDI ; 하나의 IF 문을 종료합니다.

Note

1. 조건문의 연산 결과가 참이면 이하의 명령이 실행되고, 거짓이면 ELSE 이하의 명령이 실행됩니다.
2. 명령어 표기에서 { }안에 있는 문장은 생략 가능하며, 생략할 경우 조건문의 연산 결과가 만족하지 않으면 ENDI 다음 Line에 있는 명령이 실행됩니다.
3. IF문 안에 또 다른 IF문을 사용할 수 있습니다.

4. IF ~ ENDI 사용에 대한 프로그램 구조는 다음과 같습니다.



CALL

기능 설명

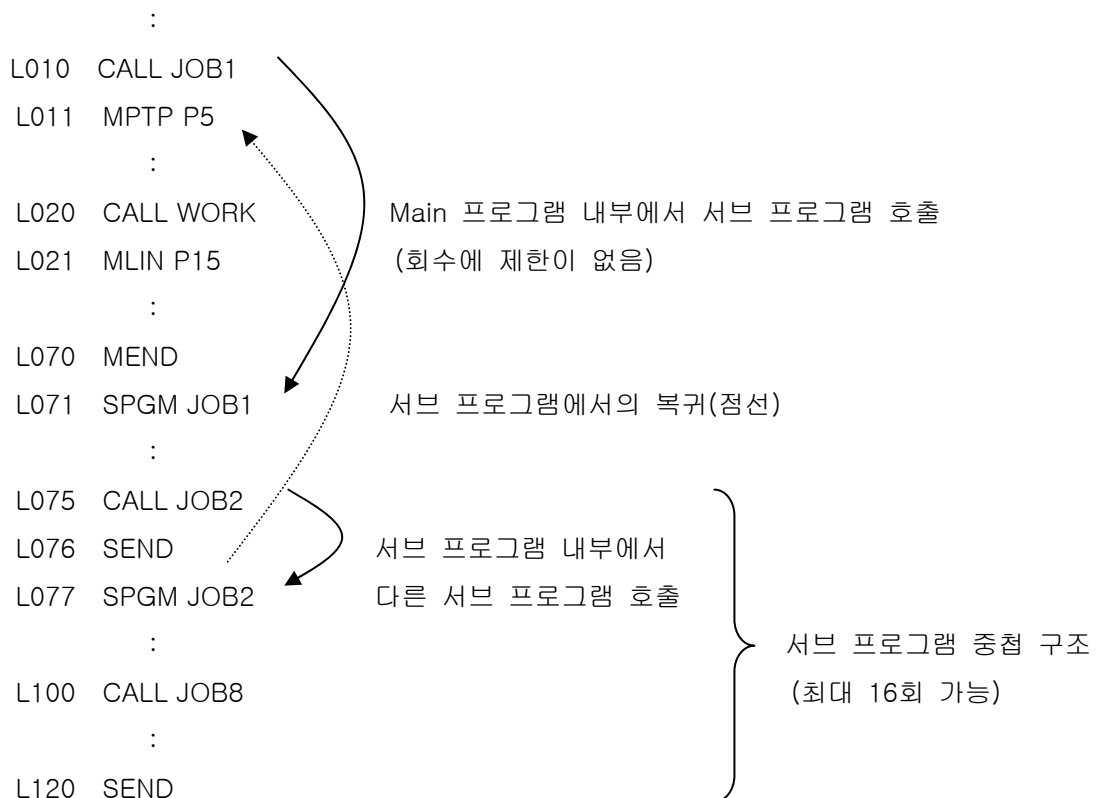
1. 서브 프로그램 명으로 지정된 서브 프로그램을 호출합니다.
2. 서브 프로그램 명은 문자, 숫자 조합 8자 이내 입니다.

입력 형식

CALL SULBI2 ; SULBI2의 이름을 가진 서브 프로그램을 호출합니다.
 CALL ABCD ; ABCD의 이름을 가진 서브 프로그램을 호출합니다.

Note

1. 모션 프로그램 진행 중 서브 프로그램을 호출합니다.
2. 서브 프로그램에서의 복귀는 SEND 명령을 만났을 때 이루어지며 CALL 명령을 사용한 다음 스텝으로 복귀합니다.
3. SPGM ~ SEND로 구성된 서브 프로그램 내부에서 다시 CALL 명령을 사용하여 다른 서브 프로그램을 호출할 수 있습니다. 이 경우 16회 사용이 가능합니다.
4. Main 프로그램 내에서 CALL 명령에 의한 호출 및 Main으로의 복귀하고, 다른 서브 프로그램 호출하는 프로그램을 작성할 경우에는 개수에 상관하지 않고 CALL 명령을 사용할 수 있습니다.
5. CALL 문에 대한 사용 예는 다음과 같습니다.



SPGM ~ SEND

기능 설명

1. 하나의 서브 프로그램 시작과 끝을 선언합니다.
2. 서브 프로그램 명은 문자, 숫자 조합 8자 이내 입니다.

입력 형식

SPGM SULBI2 ; SULBI2의 이름을 가진 서브 프로그램을 작성합니다.
 SPGM ABCD ; ABCD의 이름을 가진 서브 프로그램을 작성합니다.
 SEND ; 하나의 서브 프로그램을 종료합니다.

Note

1. Main 프로그램을 종료한 후 MEND 다음부터 작성합니다.
2. 서브 프로그램이 Main 프로그램 안으로 들어갈 경우에는 문법 에러(Compile Error)가 발생합니다.
3. Main 프로그램 또는 서브 프로그램 안에서 CALL 명령에 의해 호출됩니다.
4. SPGM ~ SEND 문 내부에 또 다른 SPGM ~ SEND 문을 작성할 수 없습니다. 다른 서브 프로그램을 작성할 경우에는 SEND로서 하나의 서브 프로그램을 종료한 후 작성해야 합니다.
5. SPGM ~ SEND 문에 대한 사용 예는 다음과 같습니다.

:	}	Main 프로그램 영역
MEND		
SPGM JOB1	}	서브 프로그램 블록 #1
:		
SEND	}	서브 프로그램 블록 #2
SPGM JOB2		
:	}	서브 프로그램 블록 #3
SEND		
:	}	서브 프로그램 블록 #3
:		
SPGM JOB8	}	서브 프로그램 블록 #3
:		
SEND		

LPMN, LPMI

기능 설명

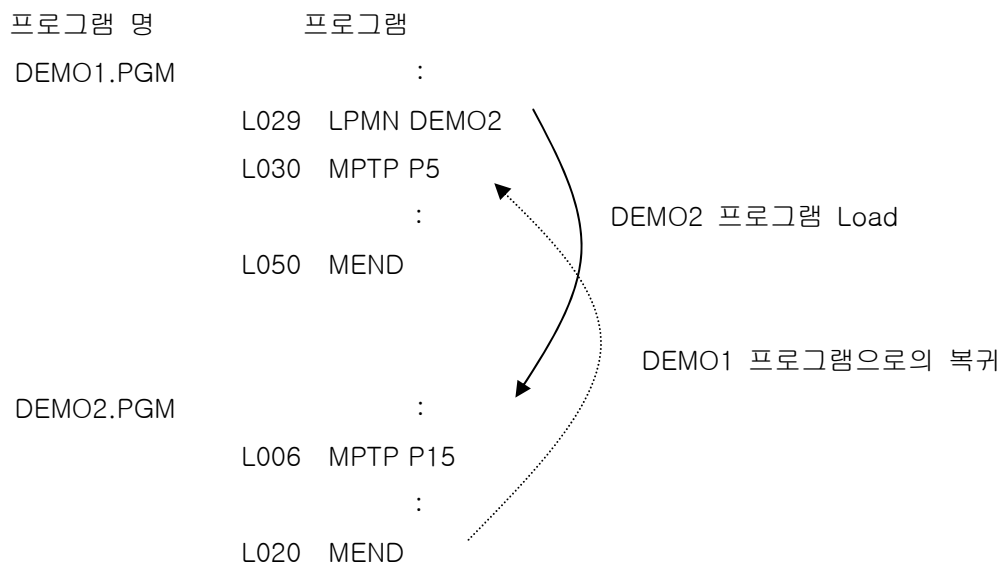
1. 모션 프로그램 실행 중 다른 모션 프로그램 파일을 Load 합니다.
2. 사용할 수 있는 부 명령어는 이미 작성되어 있는 모션 프로그램 파일 이름입니다.(LPMN)
3. 사용할 수 있는 부 명령어는 이미 작성되어 있는 모션 프로그램 번호 이름입니다.(LPMI)

입력 형식

LPMN DASA	;	모션 프로그램 실행 중 DASA.PGM 파일을 Load 합니다.
LPMI 5	;	모션 프로그램 실행 중 ID가 5인 파일을 Load 합니다.
LPMI II	;	모션 프로그램 실행 중 정수형 변수 II의 값을 ID로 가지는 파일을 Load 합니다.

Note

1. 모션 프로그램 실행 중 다른 모션 프로그램 파일을 Load합니다.
2. LPMN 명령어 사용시 파일명은 이미 저장되어 있는 파일로써 8자 이내 입니다.
3. LPMI 명령어 사용시는 파일 번호는 숫자 또는 정수형 변수를 사용할 수 있고 값은 0~99 사이 이어야 합니다.
4. LPMN,LPMI 명령어에 의해 Load된 프로그램에서의 복귀는 해당 프로그램에서 사용한 MEND 명령을 만났을 때 이루어지며 LPMN,LPMI한 다음 스텝으로 복귀합니다.
5. LPMN,LPMI 명령에 대한 사용 예는 다음과 같습니다.



6. 프로그램의 크기가 클수록 실행시간이 오래 걸리게 됩니다. 실행중에는 232통신과 같은 다른 배경작업이 중단되므로 유의하십시오.

STOP

기능 설명

1. 모션 이동 정지 및 프로그램 정지를 실행합니다.
2. 사용할 수 있는 부 명령어는 PGM 또는 MOVE 입니다.

입력 형식

STOP PGM ; 프로그램 실행을 완전히 종료합니다.
STOP MOVE ; 모션 이동을 정지합니다.

Note

1. SYNC 문을 사용하여 이동 중 조건을 만족하여 이동을 중지하고자 할 경우에 STOP MOVE 명령을 사용합니다.
2. STOP PGM 명령으로 프로그램 실행을 중지한 경우, 프로그램 실행을 재개하기 위해서는 Operating Loader로 Run을 하거나 I/O에서 Run 신호를 입력하여 프로그램을 재 실행할 수 있습니다.

SYNC ~ ACT ~ {CONT} ~ ENDS

기능 설명

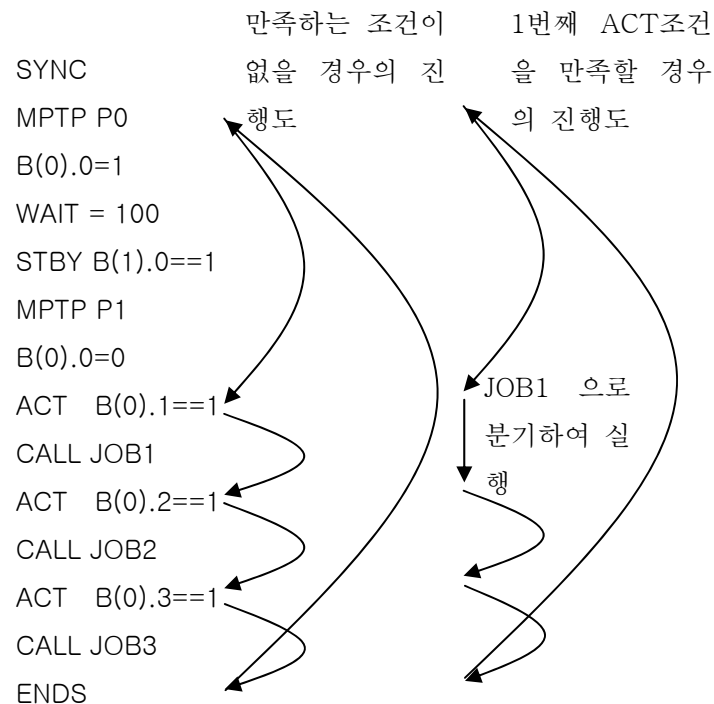
1. 이동 중(또는 WAIT, STBY 중) 조건 판단하여 설정한 조건이 맞으면 ACT 이하의 명령어를 처리합니다.
2. ACT 조건에 대한 설정은 IF 문과 동일합니다.

입력 형식

SYNC	; 이동 중 조건 판단 시작을 선언합니다.
MPTP P0	; P0 위치로 MPTP 이동 중 ACT의 조건을 검사합니다.
ACT B(0).1==1	; 판단할 조건을 설정합니다.
CONT	; 조건이 걸린 지점으로 복귀하여 이동을 계속 합니다.
ENDS	; SYNC 문을 종료합니다.

Note

1. SYNC ~ ACT 문 사이에 오는 이동 명령을 실행하면서 ACT 문에서 설정한 조건을 판단합니다. 다소 시간이 소요되는 이동명령어나 WAIT, STBY 명령어를 만날 경우에만 ACT로 분기하여 조건을 검사합니다. B(0).0=1과 같은 명령문은 ACT로 분기하지 않습니다.



2. SYNC ~ ACT 문 사이에는 여러 개의 이동 명령을 사용할 수 있으며, 각각의 이동 명령에서 조건 판단합니다.
3. 이동 중 목표점에 도달하지 않고 조건을 만족하여 다른 명령을 수행한 후에 다시 처음으로 복귀하여 이동을 계속하고자 하면, 조건을 처리한 후 CONT 명령을 사용합니다.

4. CONT 명령에 의해 복귀되는 점은 PTP 이동 및 직선 보간 이동을 하면서 조건이 걸린 경우에만 가능합니다.
5. CONT 명령은 ACT 블록 밖에서는 무의미합니다.(아무 일도 안 함)
6. CONT 명령은 ACT 블록 안에서도 STOP MOVE를 전에 사용하지 않는 경우 동작 안 합니다.

```

SYNC
MPTP P0
ACT 1           // 항상 참이므로 다음 명령을 수행합니다.
MPTP P1         /* P0 까지 완전히 간 후 P1 처리됩니다.*/
CONT            /* 의미 없습니다.*/
ENDS

```

7. SYNC ~ ENDS 사이에 여러 개의 ACT 조건을 사용할 수 있습니다.
8. SYNC ~ ACT ~ ENDS 명령에 대한 사용 예는 다음과 같습니다.

```

:
L020 SYNC
L021 MPTP P5
L022 MPTP P6
L023 MPTP P7
L024 MLIN P8
L025 MLIN P9
L026 ACT B(0).2==1
L027 STOP MOVE      /* 조건 판단한 후 조건이 걸린 지점에서 정지합니다.*/
L028 MPTP P10
L029 B(3).0=1
L030 CONT           /* 조건이 걸린 점으로 복귀하여 이동을 계속합니다.*/
L031 ENDS
L032 B(3).1=1       /* P9점까지 이동한 후 L031 스텝을 수행합니다.*/
:

```

} P5 ~ P9점을 이동하면서 Bit002가 On이 되는지 조건 판단합니다.

} P5 ~ P9점을 이동하면서 조건이 걸린 경우 L027 ~ L028를 실행합니다.

TLMT

기능 설명

1. 모션 이동 중 이동축의 토크 제한치를 설정합니다.
2. TLMT 데이터 입력 범위 : $1 \leq \text{설정값} \leq 100$ [%]

입력 형식

TLMT (20,20,20,20) ; 1, 2, 3, 4축의 토크 제한치를 20%로 설정합니다.
 TLMT (I,J,K,L) ; 토크 제한치를 정수형 변수로 지정된 I,J,K,L로 설정합니다.
 TLMT OFF ; 토크 제한을 해제합니다.

Note

1. 이동 중 해당 축의 토크 제한을 설정하며, 토크 제한의 기준의 해당 축 모터의 정격 전류를 기준으로 합니다.
2. 토크 제한치가 초기 마찰력보다 낮게 설정이 될 경우 움직이지 못하고 알람이 발생할 수 있습니다.
3. 이동 중 설정된 토크가 감지되면 모션 명령어 TMOD 에서 설정한 모드에 따라 동작합니다.

TMOD

기능 설명

1. TLMT 명령어로 토크 제한치를 설정한 축의 현재 토크가 제한치를 초과할 때의 동작을 설정합니다.

입력 형식

TMOD 0 ; 토크 감지 후 동작 모드를 0으로 설정합니다.
 TMOD K ; 토크 감지 후 동작 모드를 정수형 변수로 K에 저장된 값으로 설정합니다.
 TMOD GINT(10) ; 토크 감지 후 동작 모드를 전역정수형 변수 GINT(10)에 저장된 값으로 설정합니다.

Note

1. TMOD 명령어로 설정할 수 있는 동작 모드는 0 ~ 3 까지 입니다.
2. 설정한 동작 모드가 범위(0 ~ 3)를 벗어나면 프로그램 실행 시 에러가 발생합니다.
3. 설정한 모드에 따른 동작은 다음과 같습니다.
 TMOD 0 : 특정 동작 없음
 TMOD 1 : 설정 토크 초과 감지 후 모션을 정지하고 현재 모션 프로그램 진행스텝에서 대기합니다.
 TMOD 2 : 설정 토크 초과 감지 후 모션을 정지하고 모션 프로그램의 다음 스텝을 수행합니다.
 TMOD 3 : 설정 토크 초과 감지 후 에러를 발생 시킵니다.
4. 프로그램 예
 TLMT (100,100,20) : 3축의 토크 제한치를 20%로 설정합니다.
 TMOD 2 : 설정 토크 초과 감지 시 모션 프로그램의 다음 스텝 수행으로 설정
 MPTP P0 : P0 로 이동 중 3축의 현재 토크가 20% 이상이 되면 다음 스텝진행
 GOTO AA : AA 로 지정한 TAG 로 분기합니다.

4-3-3 이동(MOVE) 명령

MPTP

기능 설명

1. 원점 기준 현재 위치에서 설정 값(목표 위치)으로 PTP 이동합니다.
2. 부 명령어의 사용은 포인트 No. , 위치형 변수 또는 위치 값을 사용합니다.

입력 형식

MPTP P10	; P10의 위치로 PTP 이동합니다.
MPTP GPNT(5)	; 전역 위치형 변수 GPNT(5)의 위치로 PTP 이동합니다.
MPTP AB	; 위치형 변수 AB의 위치로 PTP 이동합니다.
MPTP K(I)	; 위치형 Array 변수 K(I)의 위치로 PTP 이동합니다.
MPTP <10,0,200,50>	; 이동 할 위치를 바로 지정하는 것이 가능합니다.
MPTP P(0..10)	; P0~P10까지 연속 이동합니다.
MPTP P0,P10	; P0로 이동 후 P10으로 이동 합니다.

Note

1. 원점 기준 현재 위치에서 설정한 목표 위치까지 PTP(Point To Point) 이동 합니다.
2. 이동 방식은 축 조합 상태 및 축별 속도를 내부적으로 연산하여 시작 포인트에서 끝 포인트까지 최단 거리로 이동합니다.

MLIN

기능 설명

1. 원점 기준 현재 위치에서 설정 값(목표 위치)으로 직선보간 이동합니다.
2. 부 명령어의 사용은 포인트 No. , 위치형 변수 또는 위치 값을 사용합니다.

입력 형식

MLIN P10	; P10 의 위치로 직선보간 이동합니다.
MLIN GPNT(5)	; 전역 위치형 변수 GPNT(5)의 위치로 직선보간 이동합니다.
MLIN AB	; 위치형 변수 AB의 위치로 직선보간 이동합니다.
MLIN K(I)	; 위치형 Array 변수 K(I)의 위치로 직선보간 이동합니다.
MLIN <10,0,200,50>	; 이동 할 위치를 바로 지정하는 것이 가능합니다.
MLIN P(0..10)	; P0~P10까지 연속 이동합니다.
MLIN P0,P10	; P0로 이동 후 P10으로 이동 합니다.

Note

1. 원점 기준 현재 위치에서 설정한 목표 위치까지 직선 보간 이동 합니다.
2. 이동 방식은 축 조합 상태 및 축별 속도를 내부적으로 연산하여 시작 포인트에서 끝 포인트까지 직선으로 이동합니다.
3. 모든 축이 동시에 연산 되어 직선 이동하므로 지정 속도가 빠르거나, 지정 포인트의 위치가 상이하면 추종 에러가 발생합니다.
4. 이 명령의 실행에서 ARCH 명령은 동작하지 않습니다.

MCIR

기능 설명

1. 시작점(현재 위치)을 기준으로 경유점 1, 경유점 2를 통과하는 원 보간 이동을 합니다.
2. 부 명령어의 사용은 포인트 No. 또는 위치형 변수를 사용합니다.

입력 형식

MCIR P1,P2	; P1과 P2를 경유하는 원 보간 이동합니다.
MCIR GPNT(2),GPNT(3)	; 전역 위치형 변수 GPNT(2)와 GPNT(3)를 경유하는 원 보간 이동합니다.
MCIR AB,CD	; 위치형 변수 AB와 CD를 경유하는 원 보간 이동합니다.
MCIR K(I),L(J)	; 위치형 Array 변수 K(I)와 L(J)를 경유하는 원 보간 이동합니다.
MCIR P1,AB	; P1과 위치형 변수 AB를 경유하는 원 보간 이동합니다.
MCIR AB,P2	; 위치형 변수 AB와 P2 를 경유하는 원 보간 이동합니다.
MCIR GPNT(1),P(1)	; 전역 위치형 변수 GPNT(1)과 P1 을 경유하는 원 보간 이동합니다.
MCIR AB,GPNT(2)	; 변수 AB와 전역 위치형 변수 GPNT(2)를 경유하는 원 보간 이동합니다.

Note

1. 위치 값을 가진 변수를 이동 포인트로 사용할 경우 원통 값으로 연산됩니다.
2. 원 보간 이동을 할 경우 속도가 빠를 경우 Channel - Axis - Mech Mov/Mot Rev 파라미터 값에 따라 원의 지름이 틀려질 수 있습니다.
3. 경유점 1과 경유점 2의 위치가 동일할 경우 프로그램 실행 중 에러가 발생합니다.
4. 경유점 1과 경유점 2의 위치가 반드시 지정이 되어야 합니다.
5. 시작점, 경유점 1 및 경유점 2는 반드시 이동하고자 하는 원주 상에 위치하여야 하며, 실제 이동은 시작점에서부터 경유점 1과 경유점 2를 통과하는 원의 지름은 내부 연산하여 이동합니다.
6. 이 명령의 실행에서 ARCH 명령은 동작하지 않습니다.
7. 시작점과 경유점1, 경유점2가 거의 직선과 유사하여 원의 반지름이 매우 크게 될 경우, 3점 중 어느 두 점이 원의 중심과 이루는 각도가 2.5도 이하인 경우 에러가 발생합니다.

MARC

기능 설명

1. 시작점(현재 위치)을 기준으로 경유점을 통과하여 목표점까지 원호 보간 이동을 합니다.
2. 부 명령어의 사용은 포인트 No. 또는 위치형 변수를 사용합니다.

입력 형식

MARC P1,P2	; P1 을 경유, P2 까지 원호 보간 이동합니다.
MARC GPNT(2),GPNT(3)	; 전역 위치형 변수 GPNT(2)을 경유, GPNT(3)까지 원호 보간 이동합니다.
MARC AB,CD	; 위치형 변수 AB를 경유, CD까지 원호 보간 이동합니다.
MARC K(I),L(J)	; 위치형 Array 변수 K(I)를 경유, L(J)까지 원호 보간 이동합니다.
MARC P1,AB	; P1 을 경유, 위치형 변수 AB까지 원호 보간 이동합니다.
MARC AB,P2	; 위치형 변수 AB를 경유, P2 까지 원호 보간 이동합니다.
MARC GPNT(1),P1	; 전역 위치형 변수 GPNT(1)를 경유, P1 까지 원호 보간 이동합니다.
MARC AB,GPNT(2)	; 변수 AB를 경유, 전역 위치형 변수 GPNT(2)까지 원호 보간 이동합니다.

Note

1. Z축의 위치 값을 가진 변수를 이동 포인트로 사용할 경우 원통 값으로 연산됩니다.
2. 경유점과 목표점의 위치가 동일할 경우 프로그램 실행 중 에러가 발생합니다.
3. 경유점과 목표점의 위치가 반드시 지정이 되어야 합니다.
4. 시작점, 경유점 및 목표점 반드시 이동하고자 하는 원호 상에 위치하여야 하며, 실제 이동은 시작점에서부터 경유점을 통과하여 목표점까지의 경로는 내부 연산하여 이동합니다.
5. 이 명령의 실행에서 ARCH 명령은 동작하지 않습니다.
6. 시작점과 경유점1, 경유점2가 거의 직선과 유사하여 원의 반지름이 매우 크게 될 경우, 3점 중 어느 두 점이 원의 중심과 이루는 각도가 2.5도 이하인 경우 에러가 발생합니다.

MPLT

기능 설명

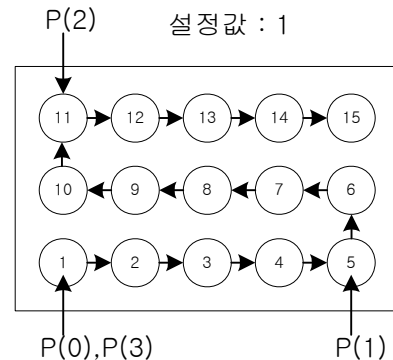
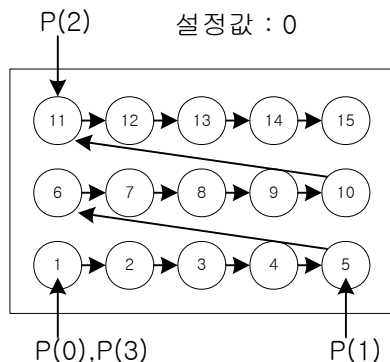
1. 기준 위치에서 설정된 Pallet에 의한 Palletizing 이동합니다.
2. 부 명령어의 사용은 팔레트 패턴 ID, Point No. 또는 위치형 Array 변수를 사용합니다.

입력 형식

MPLT<0,P1,0> ; 최초에 초기 위치 P1번으로 이동하여 0번 패턴으로 이동 시작합니다.
 MPLT<1,A(K),1> ; 최초에 위치형 Array 변수로 지정된 초기 위치 A(K)번으로 이동하여 1번 패턴으로 이동 시작합니다.

Note

1. 입력 형식에 대한 순서는 다음과 같습니다.
 MPLT <Pallet ID, 초기 이동 위치, 이동 패턴>
2. Pallet Pattern은 설정 값에 따라 다음과 같이 동작합니다.



3. 이동 포인트에 대한 설정은 반드시 지정되어 있어야 하며, 이 명령을 만나게 되면 명령어에서 지정된 초기 이동 위치로 PTP 이동하여 작업 Pattern에 따라 Palletizing 이동을 시작합니다.
4. 이동 포인트를 설정할 때, 초기 이동 위치를 시작으로 연속된 4개의 포인트(초기 이동 위치, 최초 진행 방향의 끝, Y(X) 방향의 끝, Z 방향의 끝)가 있어야 합니다.
5. 위치형 Array 변수를 사용할 경우에는 초기 대입 위치에서부터 자동으로 1씩 증가하여 4개의 위치까지 Pallet 포인트로 인식됩니다.
 예) 위치형 Array 변수 A(K)가 선언되어 있고 Pallet 이동의 최초 위치가 A(2)인 경우
 A(2) ; 초기 이동 위치(시작점), A(3) ; 최초 진행 방향의 끝, A(4) ; Y(X) 방향의 끝
 A(5) ; Z 방향의 끝
6. 프로그램 실행 중 이 명령을 만나기 전 반드시 PCNT 및 PWRK에 의한 Pallet No.가 설정되어 있어야 합니다.(PCNT 및 PWRK명 참조)
7. Palletizing 이동이 시작되면 PCNT에서 설정한 각 방향 카운터 값은 자동으로 1씩 증가하며, 모니터링 화면에서 증가되는 카운터 값이 자동으로 표시됩니다.

MINC

기능 설명

1. 현재 위치에서 설정 값(목표 위치) 위치를 더하여 PTP 증분 이동합니다.
2. 부 명령어의 사용은 포인트 No. , 위치형 변수 또는 위치 값을 사용합니다.

입력 형식

MINC P5	; 현재 위치에서 P5 위치를 더하여 PTP 증분 이동합니다.
MINC GPNT(10)	; 현재 위치에서 전역 변수 GPNT(10) 위치를 더하여 PTP 증분 이동합니다.
MINC ABC	; 현재 위치에서 위치형 변수 ABC 위치를 더하여 PTP 증분 이동합니다.
MINC K(I)	; 현재 위치에서 Array 변수 K(I) 위치를 더하여 PTP 증분 이동합니다.

Note

1. 현재 위치에서 설정한 목표 위치를 더하여 PTP(Point To Point) 이동 합니다.
 예)

	1축	2축	3축	4축
현재 위치 :	10.02,	15.33,	20.11,	-36.13
설정 위치 :	5.13,	-5.56,	10..11,	36.13
이동 위치 :	15.15,	9.77,	30.22,	0.00
2. 이동 방식은 축 조합 상태 및 축별 속도를 내부적으로 연산하여 시작 포인트에서 끝 포인트까지 최단 거리로 이동합니다.

MILR

기능 설명

1. 현재 위치에서 설정 값(목표 위치) 위치를 더하여 직선 보간으로 증분 이동합니다.
2. 부 명령어의 사용은 포인트 No. , 위치형 변수 또는 위치 값을 사용합니다.

입력 형식

MILR P1	; 현재 위치에서 P1 위치를 더하여 직선보간 증분 이동합니다.
MILR GPNT(11)	; 현재 위치에서 변수 GPNT(11) 위치를 더하여 직선보간 증분 이동합니다.
MILR ABC	; 현재 위치에서 변수 ABC 위치를 더하여 직선보간 증분 이동합니다.
MILR K(I)	; 현재 위치에서 Array 변수 K(I) 위치를 더하여 직선보간 증분 이동합니다.

Note

1. 현재 위치 기준 설정한 목표 위치를 더하여 직선 보간으로 증분 이동 합니다.
2. 이동 방식은 축 조합 상태 및 축별 속도를 내부적으로 연산하여 시작 포인트에서 끝 포인트까지 직선으로 이동합니다.
3. 모든 축이 동시에 연산 되어 직선 이동하므로 지정 속도가 빠르거나, 지정 포인트의 위치가 상이하면 추종 에러가 발생합니다.
4. 이 명령의 실행에서 ARCH 명령은 동작하지 않습니다.

MVIO

기능 설명

1. 현재 위치에서 접점에서 설정한 포인트의 위치로 이동합니다.
2. 부 명령어 없이 사용합니다.
3. MVIOStart 접점 rising edge에서 이동 시작

입력 형식

MVIO ; 외부 포인트 No. 입력에 의해 이동합니다.

Note

1. 모션 프로그램 실행 중 이 명령을 만나면 입력접점 InputPnt0~InputPnt7의 입력에 의해 설정된 포인트의 위치로 PTP 이동합니다.
2. 입력접점 InputPnt0~InputPnt7의 입력에 의해 설정된 포인트 No.에 위치가 저장되어 있지 않을 경우 에러가 발생합니다.
3. **MVIOStart 접점의 rising edge가 검출되지 않으면 무한히 대기하므로 주의 바랍니다.**
4. InputPnt0 ~ InputPnt7 로 설정할 수 있는 포인트 No.는 **P(0) ~ P(255)**까지 입니다.
5. 설정되는 포인트는 현재 실행 중인 파일에 Loading된 포인트 파일의 포인트 입니다.
6. InputPnt0 ~ InputPnt7 이 나타내는 포인트 No.는 다음과 같습니다.
 InputPnt0.....InpitPnt7
 0 0 0 0 0 0 0 0 : P(0)
 1 0 0 0 0 0 0 0 : P(1)
 0 1 0 0 0 0 0 0 : P(2)
 ...
 0 1 1 1 1 1 1 1 : P(254)
 1 1 1 1 1 1 1 1 : P(255)
7. H/W Input 에서 InputPnt0~ InputPnt7, MVIOStart 로의 맵핑은 사용자가 시스템 시퀀스 프로그램을 작성하여야 합니다.
8. 시스템 입력접점의 상세 내용은 6-1-3 시스템 입력 영역을 참조 하십시오.

SPLA

기능 설명

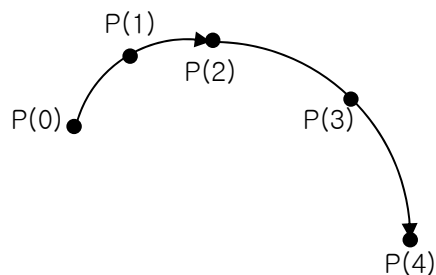
1. 시작점(현재 위치)을 기준으로 연속된 포인트를 원호 보간 이동을 합니다.
2. 부 명령어의 사용은 포인트 No. 를 사용합니다.
3. 연속된 포인트는 각 원호의 경유점과 목표점으로 사용됩니다.
4. 부 명령어에 설정된 포인트 No. 의 차이는 반드시 홀수 이어야 합니다.

입력 형식

SPLA P(1..4) ; P1 을 경유, P2 까지 원호 보간 이동하고 연속해서 P3 을 경유 P4 까지 원호 보간 이동합니다.

Note

1. MARC(원호보간) 명령어를 연속으로 사용한 것과 같은 이동 형태입니다.
2. SPLA P(1..4) 의 형태로 명령할 경우 P(1), P(2)가 첫번째 원호의 경유점과 목표점이 되고 P(3),P(4)가 두번째 원호의 경유점과 목표점이 됩니다.
3. 경유점과 목표점의 위치가 동일할 경우 프로그램 실행 중 에러가 발생합니다.
4. 시작점, 경유점 및 목표점 반드시 이동하고자 하는 원호 상에 위치하여야 하며, 실제 이동은 시작점에서부터 경유점을 통과하여 목표점까지의 경로는 내부 연산하여 이동합니다.
5. 경유점과 목표점의 위치가 반드시 지정이 되어야 합니다.
6. 이 명령의 실행에서 ARCH 명령은 동작하지 않습니다.
7. SPLA 명령어에 인자로 입력되는 포인트 번호의 차이는 반드시 홀수 이어야 합니다. 예를 들어 SPLA P(1..5) 인경우는 포인트 번호의 차이가 짝수(4) 이므로 프로그램 실행 시 에러를 발생합니다.
8. SPLA 명령어에 인자로 입력되는 포인트 번호는 상수 입력만 가능합니다.
SPLA P(GINT(1)..GINT(2)) 의 형태는 가능하지 않습니다.
9. 사용 예(현재위치가 P(0) 일 때)
SPLA P(1..4) : P(0)→P(1)→P(2) 로 원호보간하고 이어서 P(2)→P(3)→P(4) 원호보간



4-3-4 입출력 및 변수 선언(IO & Variable) 명령

B

기능 설명

1. Bit 및 Byte 단위로 접점을 처리합니다.
2. 파라미터에서 입력으로 설정된 접점은 출력으로 사용할 수 없습니다.
3. 내부 접점 중 시스템 영역으로 지정된 접점은 사용할 수 없습니다.
(상세한 접점의 범위는 매뉴얼 “6-1-1 입출력 접점 사용”을 참조 하십시오.)

입력 형식

$B(4).0=1$; 접점 B(4).0 을 On 합니다.
$B(4).0=0$; 접점 B(4).0 을 Off 합니다.
$B(4).0 = \sim B(0).1$; 접점 B(0).1를 반전 시켜 접점 B(4).0 에 출력합니다.
$B(4).0=B(4).1 \& B(4).2$; 접점 B(4).1과 B(4).2를 AND 하여 B(4).0 에 출력합니다.
$B(4).0=B(4).1 B(4).2$; 접점 B(4).1과 B(4).2를 OR 하여 B(4).0 에 출력합니다.
$B(4).0=B(4).1 \wedge B(4).2$; 접점 B(4).1과 B(4).2를 XOR 하여 B(4).0 에 출력합니다.
$B(4) = 0B1010----$; 접점 B(4) 포트의 상위 4Bits를 1010으로 출력하고 하위 4Bits는 변경하지 않습니다.(B(4).7=1, B(4).6=0, B(4).5=1, B(4).4=0)
$B(4) = 0XA-$; B(4) = 0B1010---- 의 16진수 표현 입니다.
$B(4) = B(5) \& B(6)$; 접점 B(5)포트와 B(6)포트를 AND 하여 B(4) 포트에 출력합니다.
$B(4) = B(5) B(6)$; 접점 B(5)포트와 B(6)포트를 OR 하여 B(4) 포트에 출력합니다.
$B(4) = B(5) \wedge B(6)$; 접점 B(5)포트와 B(6)포트를 XOR 하여 B(4) 포트에 출력합니다.
$GINT(0) = B(4)$; 접점 B(4) 포트의 값을 전역 정수형 변수 GINT(0)에 저장 합니다. ; B(4)=01010101(0x55) 이면 GINT(0)에는 85가 저장 됩니다.
$GINT(0) = B(5) \& B(6)$; 접점 B(5)포트와 B(6)포트를 AND 한 값을 GINT(0)에 저장 합니다.
$GINT(0) = B(5) + B(6)$; 접점 B(5)포트와 B(6)포트를 더한 값을 GINT(0)에 저장 합니다.

Note

1. 범용 출력, 내부 접점, 확장 출력 접점을 Bit 및 Byte 단위로 On/Off 합니다.
2. 모션 프로그램을 시퀀스 프로그램과 동시에 실행할 경우 시퀀스 프로그램에서 사용하는 출력(내부) 접점을 모션 프로그램도 제어를 하면 이상 출력을 하므로 주의 바랍니다.
3. 프로그램 작성시 사용자 입력 접점으로 설정된 접점에 출력하면 프로그램 운전 중 에러가 발생합니다.

GPNT

기능 설명

1. 전역 위치형 변수를 표시합니다.
2. GPNT(0) ~ GPNT(255) 까지 사용 가능합니다.
3. 전 채널에서 공용으로 사용할 수 있습니다.
4. 채널 간 Interface 용으로 사용할 수 있습니다.

입력 형식

GPNT(0)=GPNT(1)+GPNT(2) ; 전역 위치형 변수의 덧셈 연산 입니다.

GPNT(0)=GPNT(1)*2-
GPNT(2)/10.0 ; 전역 위치형 변수의 사칙 연산 입니다.

GPNT(0).1 = 12.345 ; 전역 위치형 변수의 1번째 위치에 값을 저장합니다.

GPNT(0).1 = AB ; 전역 위치형 변수의 1번째 위치에 실수형 변수를 대입합니다

GPNT(0).1= 2.345+GPNT(1).1 ; 전역 위치형 변수의 1번째 위치에 연산 결과를 대입합니다.

Note

1. 각 변수는 6개의 실수형 변수로 구성됩니다 : GPNT(0).1 ~ GPNT(0).6
2. 값의 범위는 -99999.999 ~ 99999.999 입니다.
3. GPNT(252) ~ GPNT(255)는 각 채널의 현재 위치를 항상 저장하므로 프로그램에서는 읽기만 가능합니다.
 - ◆ GPNT(252) : 채널1의 현재 위치
 - ◆ GPNT(253) : 채널2의 현재 위치
 - ◆ GPNT(254) : 채널3의 현재 위치
 - ◆ GPNT(255) : 채널4의 현재 위치
4. GPNT(252).1 : 채널1의 1축의 현재 위치를 나타냅니다.
5. 포인트 연산에 대하여 가감승제 부호(+, -, *, /)로 모션 프로그램 내에서 사칙 연산을 할 수 있습니다.
6. 채널간 Interface 용으로 사용

예) 채널2의 모션 프로그램에서 채널1의 위치를 이용

STBY GPNT(252).0 > 100.00 ; 채널1의 1축의 위치가 100보다 크면

MPTP P1

GFLT, GINT

기능 설명

1. 전역 실수형 변수를 표시합니다.
2. GFLT(0) ~ GFLT(255) 까지 사용 가능합니다.
3. GINT(0) ~ GINT(255) 까지 사용 가능합니다.
4. 전 채널에서 공용으로 사용할 수 있습니다.
5. 채널 간 Interface 용으로 사용할 수 있습니다.

입력 형식

GFLT(0)= GFLT (1)+ GFLT (2) ; 전역 실수형 변수의 덧셈 연산 입니다.
 GFLT(0)= GFLT(1)*2- ; 전역 실수형 변수의 사칙 연산 입니다.
 GFLT(2)/10.0
 GFLT(0) = 12.345 ; 전역 실수형 변수에 값을 저장합니다.
 GFLT(0) = AB ; 전역 실수형 변수에 실수형 변수를 대입합니다
 GFLT(0)= 2.345+GFLT(1) ; 전역 실수형 변수에 연산 결과를 대입합니다.
 GFLT(0)= 2.345+GPNT(1).0 ; 전역 실수형 변수에 연산 결과를 대입합니다.

 GINT(0)= GINT(1)+ GINT(2) ; 전역 정수형 변수의 덧셈 연산 입니다.
 GINT(0)= GINT(1)*2- ; 전역 정수형 변수의 사칙 연산 입니다.
 GINT(2)/10
 GINT(0) = 12 ; 전역 정수형 변수에 값을 저장합니다.
 GINT(0) = CD ; 전역 정수형 변수에 실수형 변수를 대입합니다

Note

1. 전역 실수형 변수(GFLT)의 값의 범위는 -99999.999 ~ 99999.999 입니다.
2. 전역 정수형 변수(GFLT)의 값의 범위는 0 ~ 4294967296(2^{32}) 입니다.
3. 가감승제 부호(+, -, *, /)로 모션 프로그램 내에서 사칙 연산을 할 수 있습니다.
4. 채널간 Interface 용으로 사용
 예) 채널1 에서 이동 이 끝날 때 마다 GINT(0)을 1 증가 시키고 채널2에서 GINT(0)을
 비교문에 이용
 IF GINT(0) > 10 ; 채널1의 이동이 10회를 넘으면
 MPTP P1

PNT

기능 설명

1. 위치형 변수를 선언합니다.
2. 변수명은 문자, 숫자 조합 8자 이내 입니다.
3. 한 프로그램에서 선언한 변수는 다른 프로그램이나 다른 채널에서 참조할 수 없습니다.

입력 형식

PNT AA, AB ; 위치형 변수 AA, AB를 선언합니다.

PNT AB(10) ; 위치형 Array 변수 AB를 선언합니다.

Note

1. 위치형 변수명은 숫자, 문자 포함 8자까지 가능합니다.(첫 자리는 반드시 문자)
2. 모션 프로그램에서 사용하는 명령어는 변수명으로 사용할 수 없습니다.
사용 불가 예) PNT GPNT, GINT, IF : 명령어를 변수명에 잘못 사용
3. 여러 개의 변수명을 동시에 선언할 경우 1줄에 명령어 포함 50자까지 가능합니다.
4. 위치형 Array 변수로 선언할 경우 다음과 같은 위치형 변수로 구성됩니다.
예) PNT DASA(4)로 선언할 경우
- DASA(0), DASA(1), DASA(2), DASA(3)의 형태를 가집니다. 개수 설정에 유의하시기 바랍니다.
5. 정수 및 실수형 변수와 마찬가지로 프로그램의 시작 라인부에 선언을 해야 합니다.
6. 모든 Teaching된 포인트 좌표들은 PNT 변수에 대입 및 연산이 가능합니다.
7. 포인트 연산에 대하여 가감승제 부호(+, -, *, /)로 모션 프로그램 내에서 사칙 연산을 할 수 있습니다.

예) 위치형 변수 PI, DD, BB 및 위치형 Array 변수 PIC와 정수형 변수 K가 정의되어 있다고 가정할 때,

PIC(K)=P(K) ; Array 변수에 위치 데이터 바로 대입

PI.1=DD.1+20.0 ; 변수에 실수 값 가산

PI.3=BB.3-K ; 변수에 실수 값 감산

PI=PI*2 ; 변수에 정수 값 곱하여 자기 자신에 저장

DD=DD/1.5 ; 변수에 실수 값 나누어 자기 자신에 저장

☞ 참고

기 Teaching된 포인트 좌표의 축별 연산이 가능합니다.

- P(0) 포인트의 데이터가 1축 : 200.0, 2축 : 300.0, 3축 : 100.0 일 경우

P(0).1 - P(0) 포인트의 1축 데이터를 나타냄(P(0).1 = 200.0)

P(0).2 - P(0) 포인트의 2축 데이터를 나타냄(P(0).2 = 300.0)

P(0).3 - P(0) 포인트의 3축 데이터를 나타냄(P(0).3 = 100.0)

FLT, INT

기능 설명

1. 실수형 또는 정수형 변수를 선언합니다.
2. 변수명은 문자, 숫자 조합 8자 이내 입니다.
3. 한 프로그램에서 선언한 변수는 다른 프로그램이나 다른 채널에서 참조할 수 없습니다.

입력 형식

FLT B1,B2 ; 실수형 변수 B1, B2를 선언합니다.

INT AB,AC ; 정수형 변수 AB, AC를 선언합니다.

Note

1. 정수 및 실수형 변수명은 숫자, 문자 포함 8자까지 가능합니다.(첫 자리는 반드시 문자)
2. 모션 프로그램에서 사용하는 명령어는 변수명으로 사용할 수 없습니다.
사용 불가 예) INT GPNT,GINT,IF : 명령어를 변수명에 잘못 사용
3. 여러 개의 변수명을 동시에 선언할 경우 1줄에 명령어 포함 50자까지 가능합니다.
(프로그램 작성 시 한 라인에 50자 이상(주석문 제외)의 글자가 있으면 컴파일시 에러가 발생)
4. 실수형 Array 변수로 선언할 경우 다음과 같은 실수형 변수로 구성됩니다.
예) FLT DASA(4)로 선언할 경우
- DASA(0), DASA(1), DASA(2), DASA(3)의 형태를 가집니다. 개수 설정에 유의하시기 바랍니다.
5. 정수 및 실수형 변수와 마찬가지로 프로그램의 시작 라인부에 선언을 해야 합니다.
6. 모든 Teaching된 포인트 좌표들은 FLT 변수에 대입 및 연산이 가능합니다.
7. 가감승제 부호(+, -, *, /)로 모션 프로그램 내에서 사칙 연산을 할 수 있습니다.

TMR, CTR

기능 설명

1. SEQ 프로그램에서 사용하는 TMR 과 CTR의 값을 참조 합니다.
2. SEQ 프로그램에서 사용되지 않는 TMR 과 CTR의 값은 0 입니다.

입력 형식

IF TMR(0) > 100 ; SEQ에서 사용하는 TMR(0)의 값이 100보다 크면,
 IF CTR(1) > 10 ; SEQ에서 사용하는 CTR(1)의 값이 10보다 크면,

Note

1. 타이머는 가산 타이머로 동작하며 단위는 5[ms]를 기준으로 합니다.
2. 카운터는 가산 카운터로 동작합니다.
3. 비교 연산식(=, <, >, !=)을 사용할 수 있습니다.

PATH

기능 설명

1. 이동 구간에 대한 현재 진행 비율을 저장하고 있는 변수입니다.
2. 값의 직접대입은 할 수 없고, 조건문에서 사용합니다.

입력 형식

IF PATH > 50 ; 이동 진행이 목표점 기준 50% 이상이면,
 STBY PATH <= 90 ; 이동 진행이 목표점 기준 90%이하이면 대기

Note

이동 중 거리에 대하여 조건 판단할 때 SYNC ~ENDS 명령과 같이 사용합니다.

PCNT()

기능 설명

1. Pallet 동작을 위한 Pallet ID의 Count를 설정합니다.
2. PCNT(ID) : ID 입력 범위($0 \leq \text{설정값} \leq 49$)

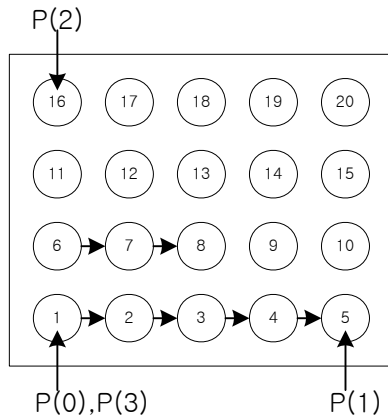
입력 형식

PCNT(5)=(1,1,1) ; Pallet No. 5번의 시작 카운트를 (1,1,1)로 설정합니다.
 PCNT(10)=(I,J,K) ; Pallet No. 10번의 시작 카운트를 정수형 변수(I,J,K)로 설정합니다.
 PCNT(11)=(GINT(200), ; Pallet No. 10번의 시작 카운트를 전역 정수형 변수(GINT(200),
 GINT(201),GINT(202)) ; GINT(201),GINT(202))로 설정합니다.

Note

1. MPLT 이동을 하기 위해서는 해당 Pallet ID로 반드시 선언되어 있어야 합니다.
2. 입력 형식에 대한 설명은 다음과 같습니다.

PCNT(Pallet ID)=< 최초 진행 방향의 시작 카운터, Y(X) 방향의 시작 카운터, Z 방향의 시작 카운터>



위의 그림에서 작업을 8까지 진행했으면 PCNT(Pallet ID) = (3,2,1)

PWRK()

기능 설명

1. Pallet 동작을 위한 각 방향 분할(또는 포인트)의 개수를 설정합니다.
2. PWRK(ID) : ID 입력 범위($0 \leq \text{설정값} \leq 49$)

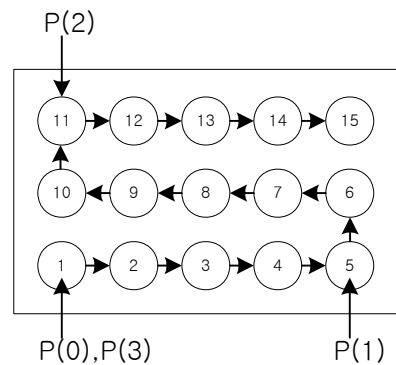
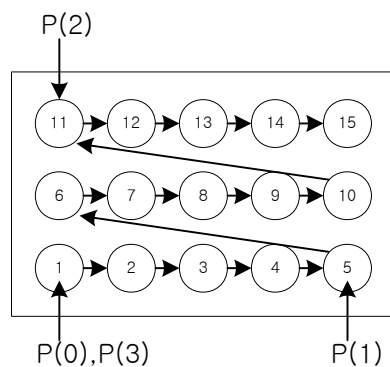
입력 형식

PWRK(5)=(10,10,2) ; Pallet No. 5번의 개수를 (10,10,2)로, 이동 패턴은 0으로 설정합니다.
 PWRK(10)=(I,J,K) ; Pallet No. 10번의 개수를 정수형 변수(I,J,K)로 설정합니다.
 PWRK(11)=(GINT(200), ; Pallet No. 10번의 개수를 전역 정수형 변수(GINT(200),
 GINT(201),GINT(202)) ; GINT(201), GINT(202))로 설정합니다.

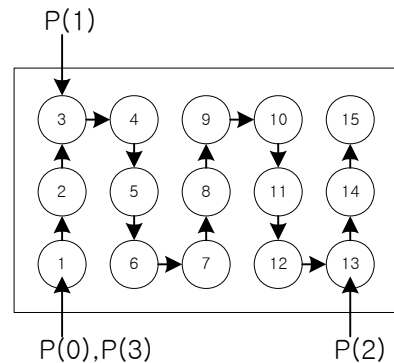
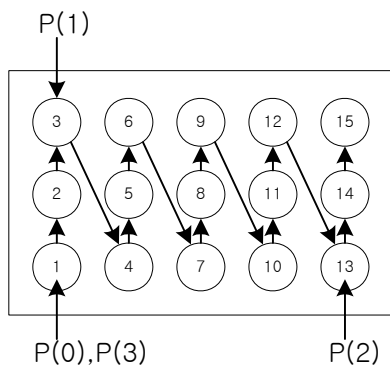
Note

1. MPLT 이동을 하기 위해서는 해당 Pallet ID로 반드시 선언되어 있어야 합니다.
2. 입력 형식에 대한 설명은 다음과 같습니다.
 PWRK(Pallet ID)=<최초 진행 방향의 분할개수, Y(X) 방향의 분할개수, Z 방향의 분할개수>
3. 포인트를 티칭하는 순서에 따라 이동하는 방향이 결정 됩니다.
4. 이동 패턴(0~1)은 다음과 같습니다.

◆ 이동 패턴 0(X방향 우선진행): PWRK(5,3,1) ◆ 이동 패턴 1 : PWRK(5,3,1)



◆ 이동 패턴 0(Y방향 우선진행): PWRK(3,5,1) ◆ 이동 패턴 1 : PWRK(3,5,1)



PLTP

기능 설명

1. 가장 최근의 Pallet이동 명령의 위치를 Load 합니다.
2. 부 명령어는 동작 명령 또는 위치형 변수와 같이 사용합니다.

입력 형식

MPTP PLTP ; 가장 최근의 Pallet이동 명령의 위치로 PTP 이동합니다.
AB = PLTP ; 가장 최근의 Pallet이동 명령의 위치를 위치형 변수 AB에 임시 저장합니다.
GPNT(5) = PLTP ; 가장 최근의 Pallet이동 명령의 위치를 전역 위치형 변수 GPNT(5)에 임시 저장합니다.

Note

1. 모션 프로그램을 실행할 때 이 명령어를 만나면 가장 최근의 Pallet이동 명령의 위치가 참조됩니다.
2. 위치형 변수를 이용할 경우 가장 최근의 Pallet이동 명령의 위치가 변수에 임시 저장됩니다.
3. 동작 명령과 같이 사용할 경우 동작 명령(MPTP, MLIN, MINC, MILR)에 따라 가장 최근의 Pallet이동 명령의 위치로 이동합니다.

CURS

기능 설명

1. 현재의 선속도를 저장하고 있는 읽기형 변수입니다.
2. 이동 형태가 보간(Linear, Circle)일 경우의 선속도값 입니다.
3. 이동 형태가 PTP 일경우의 값은 0 입니다.
3. 출력값의 단위는 [mm/s] 입니다.

입력 형식

GINT(1)=CURS ; 현재의 선속도를 전역 정수형 변수 GINT(1)에 출력합니다.
 GINT(1)=CURS*2 ; 현재의 선속도의 두배값을 전역 정수형 변수 GINT(1)에 출력합니다.

Note

1. MPG(Analog) Card 가 장착되어 있는 경우에 현재의 이동 속도를 아나로그 값으로 출력하여 외부의 아나로그 장치와 인터페이스 할 수 있습니다.
2. SYNC ~ ENDS 명령어와 같이 사용하여 이동 속도를 출력할 수 있습니다.
3. 프로그램 예


```
SPD=1000
SYNC
MLIN P0
ACT 1
GINT(1)=CURS // 현재의 속도를 GINT(1) 에 출력함
SEND
```

CURT0 ~ 5

기능 설명

1. 각 축의 현재 출력 토크를 저장하고 있는 읽기형 변수입니다.
2. 출력값의 단위는 [%] 입니다.

입력 형식

GINT(1)=CURT0 ; 첫번째 축의 현재 출력 토크를 전역 위치형 변수 GINT(1)에 출력합니다.
GINT(1)=CURT1*2 ; 두번째 축의 현재 출력 토크의 두배값을 전역 위치형 변수 GINT(1)에 출력합니다.

Note

1. MPG(Analog) Card 가 장착되어 있는 경우에 현재의 출력 토크를 아나로그 값으로 출력하여 외부의 아나로그 장치와 인터페이스 할 수 있습니다.
2. SYNC ~ ENDS 명령어와 같이 사용하여 각 축의 토크를 출력할 수 있습니다.
3. 프로그램 예
SPD=1000
SYNC
MLIN P0
ACT 1
GINT(1)=CURT0 // 첫번째 축의 현재 토크를 GINT(1) 에 출력함
SEND

4-3-5 연산 명령어 및 연산자

기능 설명	
삼각함수, 지수함수 및 기타 연산을 수행합니다.	
입력 형식	
<연산함수> GFLT(0) = ABS(-12.3) GFLT(0) = DEG(3.14) GFLT(0) = RAD(180.0) GINT(1) = POW(2,4) GINT(1) = RND(14.5) GFLT(0) = LOG(100) GFLT(0) = SQRT(4) GFLT(0) = SIN(45) GFLT(0) = ASIN(0.3) GFLT(0) = COS(45) GFLT(0) = ACOS(0.3) GFLT(0) = TAN(45) GFLT(0) = ATAN(TAN(45)) GFLT(0) = ATAN2(1,1) GFLT(0) = EXP(1) GFLT(0) = LN(1)	<연산함수> 절대치 연산입니다. Radian 값을 Angle 값으로 바꿔주는 연산입니다. Angle값을 Radian 값으로 바꿔주는 연산입니다. 누승 연산입니다. POW(밑,지수) : $2^4 \rightarrow 16$ 반올림 연산입니다. 상용 Log 연산 입니다. 제곱근 연산 입니다. SQRT(4) $\rightarrow 2$ Sine 연산입니다. Arc Sine 연산입니다. Cosine 연산입니다. Arc Cosine 연산입니다. Tangent 연산입니다. Arc Tangent 연산입니다. Secondary Arc Tangent 연산입니다. ATAN2(y값,x값) Exponent 연산입니다. 자연 Log 연산 입니다.
<연산자> << >> % & ^ ~ && > < >= <= == !=	<연산자> Shift Left 연산자 입니다. Shift Right 연산자 입니다. 나머지(Modulus) 연산자 입니다. 비트곱(Bitwise AND) 연산자 입니다. 비트합(Bitwise OR) 연산자 입니다. Bitwise XOR(Exclusive OR) 연산자 입니다. NOT(1st Complement) 연산자 입니다. (~0B11110000 \rightarrow 0B00001111) 논리곱(Logical AND) 연산자 입니다. 논리합(Logical OR) 연산자 입니다. “크다” 를 나타내는 비교연산자 입니다. “작다” 를 나타내는 비교연산자 입니다. “크거나 같다” 를 나타내는 비교연산자 입니다. “작거나 같다” 를 나타내는 비교연산자 입니다. “같다” 를 나타내는 비교연산자 입니다. “다르다” 를 나타내는 비교연산자 입니다.

