로보스타 로봇

N1 시리즈

프로그래밍 설명서(Rev 7)

- ☐ 취급 및 유지보수 설명서
- ☑ 조작 및 운용 설명서
- ☐ 프로그래밍 설명서
- ☐ 유니호스트 설명서
- ☐ GAIN 설정
- ☐ 알람코드 설명서

Robostar

www.robostar.co.kr

ROBOSTAR ROBOT

N1 Series

PROGRAMMING MANUAL(Rev 7)

- ☐ INSTRUCTION MANUAL
- ☑ OPERATION MANUAL
- ☐ PROGRAMMING MANUAL
- ☐ UNI-HOST MANUAL
- ☐ GAIN SETUP MANUAL
- ☐ ALARM CODE MANUAL

Robostar

www.robostar.co.kr

제품 보증에 관하여

㈜ 로보스타의 제품은 엄격한 품질 관리로 제조되고 있으며, 로보스타의 전 제품의 보증 기간은 제조일로부터 1년간입니다. 이 기간 내에 로보스타 측의 과실로 인한 기계의 고장 또는 정상적인 사용 중의 설계 및 제조상의 문제로 발생되는 고장에 한해서만, 무상으로 서비스를 합니다.

다음과 같은 경우에는 무상 서비스가 불가능합니다.

(1) 보증 기간이 만료된 이후
(2) 귀사 또는 제 3 자의 지시에 따른 부적당한 수리, 개조, 이동, 기타 취급 부주의로 인한 고장
(3) 부품 및 그리스 등 당사의 지정품 이외의 것의 사용으로 인한 고장
(4) 화재, 재해, 지진, 풍수해 기타 천재지변에 의한 사고로 발생되는 고장
(5) 분료 및 침수 등 당사의 제품 사양 외의 환경에서 사용함으로 인한 고장
(6) 소모 부품의 소모로 인한 고장
(7) 사용설명서 및 취급 설명서에 기재된 보수 점검 작업 내용대로 실시하지 않음으로 인해 발생되는 고장
(8) 로봇 수리에 드는 비용 이외의 손해

㈜ 로보스타 주소 및 연락처

- 본사 및 공장
  경기도 안산시 상록구 수인로 700
  700, Suin-ro, Sangnok-gu,Ansan-si,
  Gyeonggi-do, Republic of South Korea
  (15523)

- 제 2공장
  경기도 수원시 권선구 산업로 155 37,
  Saneop-ro, Gwonseon-gu, Suwon-si,
  Gyeonggi-do, Republic of South Korea
  (16648)

- 서비스요청 및 제품문의
  - 영업문의
    TEL. 031-400-3600
    FAX. 031-419-4249
  - 고객문의
    TEL. 1588-4428

Robostar
www.robostar.co.kr

사용 설명서의 구성

본 제품에 관한 사용 설명서는 다음과 같이 구성되어 있습니다. 본 제품을 처음 사용하는 경우 모든 설명서를 충분히 숙지하신 후 사용하시기 바랍니다.

■ **취급 설명서**
제어기의 전반적인 내용에 대하여 설명합니다. 제어기의 개요, 설치 및 외부 기기와의 인터페이스 방법에 대해 설명합니다.

■ **조작 및 운용 설명서**
제어기 사용의 전반적인 사용방법과 함께, 파라미터 설정, JOB 프로그램 편집, 로봇 구동 등에 대하여 설명합니다.

■ **프로그래밍 설명서**
로보스타 로봇 프로그램인 RRL (Robostar Robot Language)에 대하여, 그리고 RRL에 의한 로봇 프로그램 작성 방법에 대하여 설명합니다.

■ **유니호스트 설명서**
로보스타의 온라인 PC 프로그램인 '유니호스트'에 대하여 설명합니다.

■ **GAIN 설정 설명서**
시운전시 필요한 게인 설정 방법과 게인 값 변경에 따른 응답성에 대하여 설명합니다.

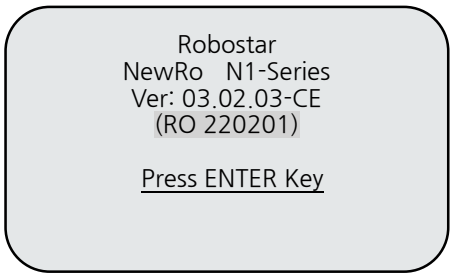
■ **알람코드 설명서**
제어기 운용 중 발생 할 수 있는 알람 상황에 대하여, 발생 원인 및 조치 사항에 대하여 설명합니다.

목차

● N1 제어기 버전

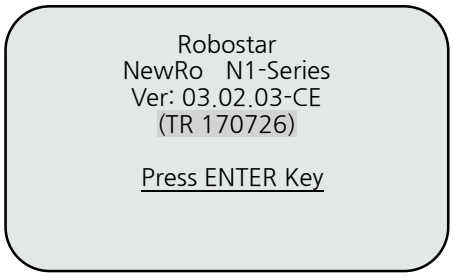
N1 제어기는 로봇 타입에 따라 2가지 운용 소프트웨어로 구분되며, 각각의 운용 소프트웨어는 제어기 전원 ON시 Teach Pendant 화면을 통해 확인 가능합니다.

1) 스카라, 직각, DeskTop, 동기로봇 운용 소프트웨어 T/P화면



스카라 로봇, 직각 로봇, DeskTop, 동기로봇 운용 소프트웨어(RO)

2) 반도체 이송용 로봇 운용 소프트웨어 T/P 화면



반도체 이송용 로봇 운용소프트웨어(TR)

● 제어기 버전에 따른 변경사항

	RO(Robot)	TR(Transfer Robot)
Global INT	500EA(0~499)	2000EA(0~1999)
Global FLOAT	500EA(0~499)	2000EA(0~1999)
Local Point	2000EA(0~1999)	2000EA(0~1999)
Global Point	1024EA(0~1023)	15000EA(0~14999)

본 매뉴얼에서 설명하고 있는 내용은 RO(Robot) 운용 소프트웨어 버전을 기준으로 설명 하고 있으며, TR(Transfer Robot) 운용 소프트웨어 버전 사용 시 변경사항은 위 표를 참고하시기 바랍니다.

CAUTION

▶ **로봇 구동전 반드시 운용소프트웨어와 로봇 타입을 확인 하시기 바랍니다.**
▶ **작업하는 로봇과 제어기 운용 소프트웨어가 일치되어야 정상적인 운용이 가능 합니다**

제1장 개요

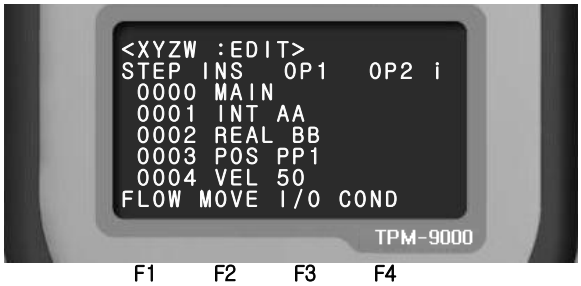
1.1 개요

1) 로봇언어는 Robostar에서 만든 전용 언어로 시스템 내에서 로봇 작업 프로그램을 작성할 때 사용 하는 명령어 입니다.
2) 로봇 언어는 크게 4개의 그룹으로 분류됩니다.
   가) FLOW 그룹
   프로그램의 조건 또는 무조건 분기, 반복 횟수, 부 프로그램 호출 등을 설정하는 명령어로 구성되어 있습니다.
   나) MOVE 그룹
   로봇의 동작 조건(PTP, 보간, Palletizing 등)을 설정하는 명령어로 구성되어 있습니다.
   다) I/O 그룹
   외부 입, 출력을 제어(16bit, 1bit)하는 명령어로 구성되어 있습니다.
   라) COND(=Condition) 그룹
   속도, 지연시간 등을 설정하는 명령어로 구성되어 있습니다.

3) 작업 용도에 맞는 적절한 명령어를 사용하여 원하는 작업 프로그램을 작성 하십시오.

제2장 명령어 일람

2.1 명령어 화면 표시



그룹	선택키	FLOW	MOVE	I/O	COND	특수 기호	삼각 함수
선택키		F1	F2	F3	F4	— X	— X — X
항목 이동 ▲▼	F1	MAIN	JMOV	OUT	VEL	+	ABS
	F2	FOR	LMOV	POUT	FOS	-	DEG
	F3	IF	CMOV	IN	DLAY	*	RAD
	F4	WHILE	AMOV	PIN	INT	/	RND
	F1	EOP	IMOV	CIN	ACC	%	POW
	F2	NEXT	PMOV	CWIN	PFOS	=	LOG
	F3	ELSE	HMOV	COUT	PLUP	<	SQRT
	F4	ENDWL		CWOUT	REAL	>	
	F1	SUBR	SVON	SYS	FORM	~	SIN
	F2	RET	SVOF		TOOL	!	COS
	F3	CALL	WITH		PCLR	&	TAN
	F4	ENDIF	ENDWT		POS	|	
	F1	LABL	MVR		LIMT	^	ASIN
	F2	GOTO	HERE		OFFS	(	ACOS
	F3	JCALL	PASS		CNT	)	ATAN
	F4		PCNT		INPOS	,	ATAN2
	F1	STOP			TMR	.	
	F2	EXIT			NO	'	
	F3				LEFT	\	
	F4				RIGHT	_	
	F1				FIX		
	F2				TRQ		
	F3				TQL		
	F4				XCHG		
	F1				DEC		

※ 특수 기호 입력 : T/P 하단에 FLOW, MOVE, I/O, COND 선택문구가 출력되어 있을 때, X키를 누른 후, 입력할 특수기호 선택

※ 삼각 함수 입력 : T/P 하단에 FLOW, MOVE, I/O, COND 선택문구가 출력되어 있을 때, X키를 두번 누른 후, 입력할 삼각함수 선택

2.2 명령어 일람표

2.2.1 프로그램 제어 관련 명령어

명령어	기 능	형 식	사용 예
MAIN	프로그램 시작	MAIN ⋮ EOP	MAIN VEL 200 JMOV P1 JMOV P2 EOP
EOP	프로그램 종료		
FOR	스텝반복수행	FOR <변수>=<초기값> TO <종료값> [BY<증분량>] ⋮ NEXT *** 변수는 정수형 변수만 사용가능**	⋮ FOR A=1 TO 5 BY 2 VEL 200 JMOV P1 NEXT ⋮
NEXT			
WHILE	조건반복수행	WHILE<조건식> ⋮ ENDWL	⋮ WHILE IN0==1 JMOV P1 JMOV P2 ENDWL ⋮
ENDWL			

IF	조건판단수행	IF<조건식> THEN
		⋮
		(ELSE)
		⋮
		ENDIF
			⋮
			IF IN3==1 THEN
			GOTO A0
			ELSE
			JMOV P1
			ENDIF
			⋮

ENDIF			
LABL	분기점 위치 지정	LABL <레이블명> ⋮ GOTO <레이블명>	⋮ LABL A1 JMOV P1 JMOV P2 GOTO A1 ⋮
GOTO	수행문 분기		

명령어	기 능	형 식	사용 예
SUBR	부실행문 시작	SUBR<부실행문명> ⋮ RET	⋮ SUBR HON GOTO A0 ELSE JMOV P1 RET ⋮
RET	부실행문 종료		
CALL	부실행문 호출	CALL <부실행문명>	
JCALL	JOB 파일 호출	JCALL<JOB 파일명>	
STOP	로봇 동작 정지	STOP	
EXIT	JOB 수행 정지	EXIT	

2.2.2 로봇 동작 관련 명령어

명령어	기 능	형 식	사용 예
JMOV	현 위치에서 목표점으로 PTP이동	JMOV <포인트 변수>	MAIN VEL 200 JMOV P1 LMOV P2 EOP
LMOV	현 위치에서 목표점으로 직선보간 이동	LMOV <포인트 변수>	
CMOV	현 위치에서 경유점1, 2를 잇는 원을 그리며 이동	CMOV <경유점1> <경유점2>	⋮ VEL 200 CMOV P1 P2 JMOV P3 AMOV P4 P5 ⋮
AMOV	현위치에서 경유점1과 목표점을 잇는 원호를 그리며 이동	AMOV <경유점1> <목표점>	
IMOV	현 위치에서 증분량만큼 PTP 이동	IMOV <포인트변수> <포인트변수> 값 → 축값 증분량	⋮ JMOV P1 IMOV P2 IMOV2 P3 ⋮
IMOV2	현 위치에서 증분량만큼 직선보간 이동	IMOV2 <포인트변수> <포인트변수> 값 → 축값 증분량	
PMOV	현 위치에서 지정된 Palletizing 작업 수행	PMOV <작업 pallet 번호> <작업 기준점> <작업 pallet 번호>의 데이터 값 → 파라미터에 설정	
HMOV	현위치에서 지정된 홈 위치로 이동	HMOV <HOME 위치 번호> <위치번호>의 데이터값 → ORG 파라미터에 설정	
JNTSYN	PTP 동기 모드 선택	JNTSYN <모드>	
EECH	말단 장치 선택	EECH <말단 장치>	
TIMOV	TOOL 좌표계 기준으로 증분이동량 만큼 직선보간 이동	TIMOV<위치형 변수>	

명령어	기능	형식	사용 예
SVON	서보 ON	SVON → 모든 축 해당 SVON<지정 축> → 1(X), 2(Y), 3(Z), 4(W), 5(E1), 6(E2)	SVON SVOF 2 SVON 2
SVOF	서보 OFF	SVOF → 모든 축 해당 SVOF<지정 축> → 1(X), 2(Y), 3(Z), 4(W), 5(E1), 6(E2)	

WITH	로봇 동작 중
	다음 수행문 열을
	동시처리
		WITH
		⋮
		ENDWT
			⋮
			WITH
			JMOV P1
			MVR=0
			WHILE MVR<60
			IF IN1==1 THEN
			OUT0=1
			ENDIF
			ENDWL
			ENDWT
			⋮

ENDWT			
PASS	Pallet 작업 통과 지정	PASS <pallet NO> <통과할 작업물 순번>	PASS 1 3 PASS 2 1

2.2.3 입출력 관련 명령어

명령어	기능	형식	사용 예
OUT	1비트 단위로, 지정된 비트번호의 출력을 ON(=1), OFF(=0)	OUT<출력비트번호>=<0/1> [펄스유효시간] [→] [펄스유효시간] 단위 : 10ms [펄스유효시간]이 없으면 : 출력신호 계속 유효 [펄스유효시간]이 지나면 이전상태로 복귀 [→] : 펄스유효시간을 주기로 갖는 주기파형 출력	OUT0=1 OUT0=1 100 OUT0=1 100 →
BOUT WOUT POUT DOUT	포트 단위로, 지정된 포트로 지정된 값을 출력합니다.	BOUT<출력포트번호>=<출력지정 값> WOUT<출력포트번호>=<출력지정 값> POUT<출력포트번호>=<출력지정 값> DOUT<출력포트번호>=<출력지정 값>	BOUT0=16 WOUT0=0H000F POUT1=0H000F DOUT1=0H000F000F
IN	1비트 단위로, 지정된 비트 번호의 ON(=1), OFF(=0) 값을 읽는다.	IN<입력비트번호>=<0/1> -지정된 비트 입력 조건이 만족 될 때까지 대기 <변수>=IN<입력비트 번호> -지정된 비트 입력 상태 값을 변수에 저장	INT AA IN10=1 AA=IN0
BIN WIN PIN DIN	포트 단위로, 지정된 입력 포트의 값을 읽는다.	<변수>=BIN<입력포트 번호> <변수>=WIN<입력포트 번호> <변수>=PIN<입력포트 번호> <변수>=DIN<입력포트 번호>	INT AA AA=BIN0 AA=WIN0 AA=PIN0 AA=DIN0

명령어	기능	형식	사용 예
CIN	1비트 단위로, 지정된 필드버스비트 입력번호의 ON(=1), OFF(=0) 값을 읽는다.	<변수>=CIN<입력비트번호> -지정된 비트 입력 상태 값을 변수에 저장	INT AA AA=CIN0
CBIN CWIN CDIN CFIN	포트 단위로, 지정된 필드버스 입력 포트의 값을 읽는다.	<변수>=CWIN<입력포트 번호>	INT AA AA=CWIN0
COUT	1비트 단위로, 지정된 필드버스 출력을 ON(=1), OFF(=0)	COUT<출력비트 번호>=<0/1>	COUT=1
CBOUT CWOUT CDOUT CFOUT	포트 단위로, 지정된 필드버스 입력 포트의 값을 읽는다.	CWOUT<출력포트번호>=<출력지정 값>	CWOUT3=0HFF0FFFF
SYS	시스템 제어 명령	예약된 시스템 명령어 수행	
MOINIT	이동 중 접점 출력 기능 초기화	<변수>=MOINIT<인덱스><채널><입력포트><출력시간>	I1=MOINIT(1,1,0,100)
MORNG	이동 중 접점 출력을 할 위치 및 위치편차 설정	<변수>=MORNG<인덱스><좌표계><포인트변수><편차>	I2=MORNG(1,0,XP,OFF)
MOCLR	이동 중 접점 출력기능 해제	<변수>=MOCLR<인덱스>	I0=MOCLR(0)

2.2.4 로봇 동작 조건 관련 명령어

명령어	기 능	형 식	사용 예
VEL	축 이동 속도의 천분율(%)을 설정	VEL <천분율 값(%)> 축 이동속도= 초기속도 X 정격속도 X 천분율 값X 0.001 ※ 정격속도는 MOTION 파라미터에서 설정	MAIN VEL 200 ACC 70 DEC 70 JMOV P1 VEL 1000 LMOV P2 ACC 100 DEC 100 EOP
ACC	가속시간의 백분율(%)을 설정	ACC <백분율 값(%)> 가속시간 = 정격가속시간 X 백분율 값 X 0.01 ※ 정격가속시간은 MOTION 파라미터에서 설정	
DEC	감속시간의 백분율(%)을 설정	DEC <백분율 값(%)> 감속시간 = 정격감속시간 X 백분율 값 X 0.01 ※ 정격감속시간은 MOTION 파라미터에서 설정	
FOS	축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적변경 함.	FOS <거리비율(%)> 거리비율은 전체 이동거리의 백분율(%) JMOV, LMOV, AMOV에 적용	FOS 5 JMOV P1 FOS 0(해제)
PFOS	축 선단이 목표점에 도달 전, 다음 목표점으로 궤적변경 함.	PFOS <거리(mm)> LMOV에 적용	PFOS 5 LMOV P1 PFOS 0(해제)

명령어	기 능	형 식	사용 예
SPOS	축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적변경 함	SFOS<거리>,<속도레벧>,<원호 각도> LMOV, AMOV, CMOV에 적용	SFOS 10 0 0 LMOV P0 LMOV P1 SFOS 0 0 0(해제)
DLAY	지연시간 설정	DLAY <지연시간> 지연시간 단위는 10ms (지연시간이 500이면 5초 지연됨)	JMOV P1 DLAY 20 JMOV P2
OFFS	목표점을 지정된 값만큼 이동시킴.	OFFS <편차 값> **※ 편차 값은 MDI모드로 입력** (예) P100 → X:0, Y:-100, Z:10, W:0, E1:20, E2:30	JMOV P1 OFFS P100 JMOV P2
USEROFFS	User좌표계 기준으로 목표점을 지정된 값만큼 이동시킴	<변환값>=USEROFFS<User좌표><목표위치><편차값>	XP=USEROFFS(0,P5,OFF)
SETUSER	User좌표계 생성	<변환 결과>= SETUSER(<사용자 좌표계번호>, <UP0>, <UP1>, <UP2>)	I12= SETUSER(3,GP1,GP2,GP3)
CALLUSER	User좌표계 오프셋 확인	<POS변수>=CALLUSER(<사용자 좌표계번호>)	AP=CALLUSER(3)
ADJUSER	User좌표계 오프셋 조정	<변환 결과>= ADJUSER(<사용자 좌표계번호>, <오프셋 변화값>)	AP.1 = AP.1+0.5 I12=ADJUSER(3,AP)
LIMT	각 축의 이동범위를 제한합니다.	LIMT <축 위치 값(최소)> <축 위치 값(최대)>	JMOV P1 LIMT P3 P4 JMOV P2
PLUP	PULL UP 동작을 위한 Z축 위치값 설정	PLUP <Z축 위치값> Z축에만 적용 됨	PLUP 5 JMOV P1 PLUP 0(해제)
APLUP	Pull up 동작을 위한 Z축 및 위치값 설정	APLUP <축 번호> <Z축값> Z축에만 적용 됨	APLUP 3 50 JMOV P1 APLUP 3 0(해제)
FORM	로봇 동작시 로봇 암의 형상을 지정 함.	FORM <형상> 수평 다관절 로봇에만 적용 됨. <형상> LEFT, RIGHT, NO 중 선댁	FORM RIGHT JMOV P1 FORM NO(해제)
TOOL	로봇에 부착된 사용 툴을 선댁	TOOL <툴번호> ※ 툴번호에 해당하는 데이터는 파라미터에 설정 <툴번호> : 0 ~ 3중 선댁	TOOL 1 JMOV P1 TOOL 0 JMOV P2
JTOU	Joint 좌표를 User 좌표로 변환	<변환값>=JTOU<User좌표계 번호><포인트 변수>	POS XP XP=JTOU(0,P5)
UTOJ	User 좌표를 Joint 좌표로 변환	<변환값>=UTOJ<User좌표계 번호><포인트 변수>	POS XP XP=UTOJ(0,P5)

명령어	기 능	형 식	사용 예
PCLR	PMOV의 Palletizing 카운터를 초기화	PCLR <pallet 번호> Pallet 번호 : 0~99까지 사용 가능. 클리어 시 카운트(XCNT, YCNT,ZCNT)는 1로 변경 됨.	MAIN PCLR 1 PCLR 2 PMOV P1 P10
INPOS	로봇 동작 시 목표점 도달 정도를 설정	INPOS <목표점 도달 정도> 도달 정도는 Pulse 값으로 설정.	INPOS 10 JMOV P2 INPOS 0(해제) JMOV P3
FIX	보간 동작에서 W축 동작선댁	FIX (0/1) 0 : W축 자세 회전, 1 : W축 자세 고정. 생략시 초기 값은 1로 고정 임.	FIX 0 LOMV P1 FIX 1
TRQ	충돌 감지시 토크 리밋 알람 발생	TRQ <축번호> <토크제한 값> 각축의 토크제한 값 이상일 때 알람 발생.	TRQ 1 50 LMOV P1 LMOV P2
TQL	각축의 최대 추력 토크값 제한	TQL <축번호> <토크제한 값> 각축을 제한 토크로 구동.	TQL 1 50 LMOV P1 LMOV P2
MSTART	MAPPING 기능 시작	MSTART<축><GP Index><센서 입력핀>	MSTART 3 GP10 3
MREAD	MAPPING 데이터 갱신	MREAD<제한 시간>	MREAD 500
NINIT	LATCH 기능 초기화	NINIT<센서 타입>	NINIT 1
NSTART	LATCH 기능 시작	NSTART<센서 입력핀><축><GP Index>	NSTART 3 3 GP10
NREAD	LATCH 데이터 갱신	NREAD<제한 시간>	NREAD 500

2.2.1 **Safety 관련 명령어**

명령어	기 능	형 식	사용 예
STEMP	제어기 내부 온도값 읽기	<실수형 변수>=STEMP<모드> <인덱스>	REAL TEMP TEMP = STEMP(1,0)
SUIN	Safety관련 접점 상태 읽기	<정수형 변수>=SUIN<모드> <인덱스>	I0=SUIN(1,1) I1=SUIN(2,1)
SUOUT	지정된 포트로 Safety User 포트 출력	<정수형 변수>=SUOUT<인덱스> <1 또는 0>	INT RETVAL RETVAL = SUOUT(0,1)

2.2.2 **변수 선언**

명령어	기 능	형 식	사용 예
INT	정수형 변수를 선언	INT <변수명>, REAL <변수명>, POS <변수명>, DEFSTR <변수명> 변수명은 6자 이내. 변수명은 대문자 알파벳과 숫자로 구성. 변수명은 숫자로 시작할 수 없음.	MAIN INT N REAL A,B POS MM, XA DEFSTR PACKET
REAL	실수형 변수를 선언		
POS	위치형 변수를 선언		
DEFSTR	문자열 변수를 선언		

2.2.3 프로그램 제어 관련 명령어

기 능	형 식	사용 예
포인트형 변수는 포인트 파일작성 시 정의됨	변수명은 POINT 명령 참조.	3.45 POSITION 변수 참조

2.2.4 시스템 변수

명령어	기 능	형 식	사용 예
CNTn TMR0 TMR1	시스템 정의변수 (카운터변수, 타이머 변수)	CNT<펄스입력포트번호>=<초기값> 카운터변수는 펄스입력포트 번호를 입력하는 순간부터 값이 할당되고 그 후 매 펄스입력을 카운트 합니다. TMR0=<초기값> 타이머 변수는 정수 값을 입력하는 순간부터 값이할당되어 시스템 파라미터에 정의된 시간 간격으로 1씩 증가합니다.	CNT0=2 TMR0=0 TMR1=-50
MVR	MOVE RATE	MVR <로봇이동구간의 백분율>	IF MVR<50 THEN
HERE	현재축 각도 값 변수	AP=HERE 현재축 각도 값을 저장하는 변수	AP=HERE
RSTATE	로봇 상태정보 읽기	<정수형 변수>=RSTATE(채널, STATE INDEX)	TEMP=RSTATE(1,6)
RERROR	알람 코드 확인	<정수형 변수>=RERROR(PAGE,INDEX)	ERR=RERROR(1,1)
REMCMD	시스템 명령어	REMCMD<채널><명령어>	REMCMD 1 3
GFTOGP	GF값을 GP로 저장	GFTOGP<Gloabal Float Index><Global Point Index>	GFTOGP F10 GP50

2.2.5 상수

기 능	형 식	사용 예
정수, 실수, 2진 정수, 16진 정수를 표시	<숫자or실수> : 10진수 [0B]<숫자> : 2진수 [0H]<숫자> : 16진수	POUT1=0H000F POUT2=0B00100100

2.2.6 연산자

명령어	기 능	형 식	사용 예
+, -, *, /, %	이항연산자 (더하기, 빼기, 곱하기, 나누기, 나머지)		A = B*C
( )	우선 연산자		A = B/(A+C)
=	대입연산자		A = B

명령어	기 능	형 식	사용 예
&,|, ~, ^, <<, >>	비트연산자 (BAND, BOR, Complement, BXOR, left shift, right shift)		I0 = PIN0&0H01
&&, ||, ^^, !	논리연산자 (AND, OR, XOR, Negation)		IF IN0==1 && IN1==0 THEN
>, <, >=, <=, !=, ==	비교연산자(초과, 미만, 이상, 이하, 상이, 상동)		IF CIN0<1 THEN
ABS	절대값을 취함.	ABS(-10.5)는 10.5	
DEG	라디안 값을 각도 값으로 변환.	DEG(3.1416)는 180.0	
RAD	각도 값을 라디안 값으로 변환.	RAD(180.0)는 3.1416	
POW	지수함수	POW(2,4)는 16	
RND	소수부를 반올림하여 실수값을 정수 값으로 변환	RND(14.8)은 15	
LOG	상용지수함수	LOG(20)은 2.996	
SQRT	평방제곱근을 구함	SQRT(16)은 4	
SIN	사인 함수	SIN(RAD(30))는 0.5	
ASIN	아크 사인 함수	ASIN(0.5)는 0.4794	
COS	코사인 함수	COS(0)은 1.0	
ACOS	아크 코사인 함수	ACOS(0.5)은 1.0472	
TAN	탄젠트함수	TAN(RAD(45))는 1.0	
ATAN	아크탄젠트 함수	ATAN(-1.0)은 -0.7854	
ATAN2	제2 아크탄젠트 함수(ATAN2(Y,X))	ATAN2(1,-1)은 2.3562	
SIN	사인 함수	SIN(RAD(30))는 0.5	
ASIN	아크 사인 함수	ASIN(0.5)는 0.4794	
COS	코사인 함수	COS(0)은 1.0	
ACOS	아크 코사인 함수	ACOS(0.5)은 1.0472	
TAN	탄젠트함수	TAN(RAD(45))는 1.0	
ATAN	아크탄젠트 함수	ATAN(-1.0)은 -0.7854	
ATAN2	제2 아크탄젠트 함수(ATAN2(Y,X))	ATAN2(1,-1)은 2.3562	

2.2.7 문자열

명령어	기 능	형 식	사용 예
ASC	문자열 중 첫문자를 정수로 변환	정수형 변수=ASC(문자열)	INT AA AA=ASC("ABC") ▶문자열 "ABC"중 'A'문자를 아스키 번호 65로 반환
CHR	정수를 문자로 변환	문자열 변수=CHR(정수)	DEFSTR AA AA=CHR(65) ▶정수 65를 아스키 코드 'A'문자로 변환
FLUSH	입력, 출력 버퍼 클리어	FLUSH<클리어 버퍼 선택> <클리어 버퍼 선택>: 1~3 1: 입력 버퍼 클리어 2: 출력 버퍼 클리어 3: 입력, 출력 버퍼 클리어	FLUSH 3 ▶입력, 출력 버퍼 클리어
FTOS	정수나 실수를 문자열로 변환	문자열 변수=FTOS(정수or실수)	DEFSTR AA AA=FTOS(1234) ▶정수 1234를 문자열 "1234"로 변환
HTOS	정수를 16진수의 문자열로 변환	문자열 변수=HTOS(정수)	DEFSTR AA AA=HTOS(10) ▶정수 10을 16진수 문자열 'A'로 변환
SLEFT	좌측부분 문자열 추출	문자열 변수=SLEFT(문자열, 숫자)	DEFSTR AA AA=SLEFT("ABCDEF",3) ▶문자열 "ABCDEF"중 세 번째 문자 'C'를 문자열 변수에 저장
SLEN	문자열 길이를 반환	정수형 변수=SLEN(문자열)	INT LEN LEN=SLEN("ABCDE") ▶문자열 "ABCDE"의 길이를 정수 5로 반환
SMID	문자열 지정위치부터 자릿수 만큼 문자열 추출	문자열 변수=SMID(문자열, 지정위치, 추출문자수)	DEFSTR AA AA=SMID("ABCDEF", 2, 3) ▶문자열 위치는 맨 처음이 0부터 시작
SPOS	문자열1 내에서 문자열2가 들어가 있는 위치 반환	정수형 변수= SPOS(문자열1,문자열2)	INT PP PP=SPOS("ABCDEF", "B") ▶"B"의 위치 1반환
SRIGHT	우측부분 문자열 추출	문자열 변수=SRIGHT(문자열, 숫자)	DEFSTR AA AA=SRIGHT("ABCDEF", 3) ▶문자열 "ABCDE"중 세 번째 문자 'D'를 문자열 변수에 저장

명령어	기 능	형 식	사용 예
STROUT	문자열 출력	정수형 변수= STROUT(문자열)	INT RT RT=STROUT("ABCDEF") ▶시리얼을 통하여 문자열 "ABCDEF"를 출력
SVAL	문자열을 숫자로 변환	변수=SVAL(문자열)	INT VA REAL VB VA=SVAL("1234") VB=SVAL("0.123") ▶문자열 "1234", "0.123"을 숫자로 변환
STRIN	구분자까지 문자열 입력	문자열 변수=STRIN(타임아웃시간) 단위 : ms	DEFSTR AA AA=STRIN(1000) ▶시리얼을 통하여 입력 된 문자열 읽기

제3장 명령어 해설

3.1 MAIN, EOP(프로그램 시작/종료)

기 능 프로그램 시작 및 종료를 나타냅니다.

형 식
```
MAIN
⋮
EOP
```

설 명
1) **MAIN - EOP는 반드시 같이 사용**하여 주 프로그램 블록을 형성하며 한 개의 JOB 파일에서는 한 개의 주 프로그램 블록만 작성 할 수 있습니다.
2) EOP(End Of Program)는 주 프로그램 블록의 끝을 의미하며 JOB 수행 시 이 명령을 만나면 **JOB 수행이 종료**됩니다.
   단, JCALL에 의해 수행된 JOB은 그 JOB의 수행을 종료하고 **호출한 JOB으로 복귀** 합니다.
3) **EOP 다음 라인**에 필요한 **부 프로그램 블록(SUBR ~ RET)을 계속 작성**할 수 있습니다.

3.1.1 프로그램 사용 예제

MAIN	.....	**주 Program 시작**
VEL 100	.....	축 이동 속도
WHILE 1	.....	조건반복수행 시작
JMOV P0	.....	P0로 이동
JMOV P1	.....	P1으로 이동
CALL ON	.....	부 프로그램 호출
ENDWL	.....	조건반복수행 종료
EOP	.....	**주 Program 종료**
SUBR ON	.....	부 프로그램 지정
OUT0=1 200	.....	1 비트 단위로, 지정된 비트에 출력
OUT1=1 200	.....	1 비트 단위로, 지정된 비트에 출력
OUT2=1 200	.....	1 비트 단위로, 지정된 비트에 출력
RET	.....	부 프로그램 복귀

3.2 FOR, NEXT(반복 수행문)

기 능 변수 값이 종료값을 만족할 때 까지 블록을 반복 수행합니다.

형 식 FOR <변수>=<초기값> TO <종료값> (BY <증분량>)
⋮
NEXT

용 어 <변수> : 정수형으로 선언된 변수이름을 사용합니다. 예) INT A,B,C,AA,...
<초기값> : FOR 블록을 반복 수행하기 직전 변수에 설정되는 정수 값입니다.
<종료값> : FOR 블록의 반복 수행 횟수를 제한하기 위한 정수 값입니다.
<증분량> : 변수의 값을 규칙적으로 일정하게 증가시키기 위해 사용되는 정수값입니다.

설 명
1) FOR – NEXT 는 반드시 같이 사용되어야 합니다.
2) 정수형 변수에 초기값이 설정된 후 종료값을 만족할때까지 FOR 블록을 반복 수행합니다.
3) FOR 블럭을 반복 수행할때마다 정수형 변수의 값은 증분값에 의해 증가되고, 증분값이 생략되었을 경우에는 자동적으로 1씩 증가합니다.
4) 종료값은 초기값보다 항상 큰 값이어야 하며 증분값은 1이상의 값이어야 합니다.
(초기값≤종료값, 증분값≥1)

3.2.1 프로그램 사용 예제

1) FOR 블록을 이용하여, 1부터 10까지 홀수의 합을 연산하여 정수형 변수 SUM에 저장.

```
MAIN
INT AA, SUM ..... 정수변수 AA, SUM 선언
SUM=0 ..... 정수변수 초기화
FOR AA=1 TO 10 BY 2 ..... AA=1부터 +2씩 증가 정수 10까지 5회 수행
SUM=SUM+AA ..... 5회 수행 후 NEXT 다음 이동
NEXT
EOP
```

2) FOR 블록을 이용하여, 포인트 P1 ↔ P2 간의 20회 왕복운동

```
MAIN
INT J ..... 정수변수 J선언
J=0 ..... 정수변수 초기화
FOR J=1 TO 20 ..... FOR문 20회 반복동작 후 NEXT 다음 이동
JMOV P1 ..... JMOV P1
JMOV P2 ..... JMOV P2
NEXT
EOP
```

3) Position 변수에 티칭포인트를 저장한 후 이동

```
MAIN
INT A ..... 정수변수 A선언
POS AP(11) ..... 위치변수 AP(11)선언 → AP(0),AP(1) ~ AP(10)
FOR A=1 TO 10 ..... FOR문 10회 반복동작 후 NEXT 다음 이동
AP(A)=P(A) ..... 포인트 변수 AP(A)에 P(1) ~ P(10) 저장
NEXT
EOP
```

4) Position 변수에 티칭포인트를 저장한 후 이동

```
MAIN
INT J,K ..... 정수변수 J,K선언
POS TMP, AP(11) ..... 위치변수 AP(11)선언 → AP(0),AP(1) ~ AP(10)
FOR K=1 TO 10 ..... FOR문 10회 반복동작 후 NEXT 다음 이동
AP(K)=P(K) ..... 포인트 변수 AP(K)에 P(1) ~ P(10) 저장
NEXT
IF P100.3==1.0 THEN
FOR J=1 TO 10 FOR문 10회 반복동작 후 NEXT 다음 이동
TMP=AP(J) ..... AP(J)=AP(I)=P(1) ~ P(10)
TMP.3=TMP.3+10 위치변수 TMP의 Z축 값에 10을 더함.
AP(J)=TMP
NEXT
ENDIF
FOR K=1 TO 10 ..... FOR문 10회 반복동작 후 NEXT 다음 이동
JMOV AP(K) 저장된 AP(1) ~ AP(10)으로 PTP 이동
DLAY 100
NEXT
EOP
```

3.3 WHILE, ENDWL(조건 반복수행 명령문)

기 능 조건식이 만족 되는 동안만 블록을 반복 수행합니다.

형 식
```
WHILE <조건식>
  ⋮
ENDWL
```

용 어 <조건식> : 수식의 진위를 판단할 수 있는 논리연산식이나 비교연산식을 의미합니다.

설 명
1) WHILE 문은 **조건이 만족(조건식의 결과가 참, 또는 0이외의 값)되는 동안** WHILE 블록을 **반복 수행** 합니다.
2) **WHILE – ENDWL은 반드시 같이** 사용되어야 합니다.
3) 조건식의 결과가 **항상 참인 경우에는 무한 반복 수행**을 하게 됩니다.

3.3.1 프로그램 사용 예제

1) 포인트 P0 ↔ P1을 반복이동

```
MAIN
VEL 100
WHILE 1          ..........   블록 무한 반복 실행
JMOV P0          ..........   P0 ↔ P1 반복이동
JMOV P1          ..........
ENDWL
EOP
```

CAUTION

▶ **명령어 입력시 주의 사항**

(O) WHILE ((IN0==1)&&(IN1==0))
(O) WHILE IN0==1&&IN1==0
(O) WHILE IN0==1
    &&IN1==0

3.3.2 조건식 사용 예제

WHILE(조건식)		설 명
상수	WHILE 1	WHILE ~ ENDWL 블록을 무한 반복 실행
입력	WHILE IN0==1	입력신호 IN0=1로 입력되는 동안만 블록 반복 실행
	WHILE ((IN0==1)&&(IN1==0))	입력신호 IN0=1, IN1=0 두 가지 조건이 동시에 만족하는 동안만 블록을 반복실행
	WHILE ((IN0==1)||(IN1==0))	입력신호 IN0=1, IN1=0 두 가지 조건 중 한가지만 만족하는 동안만 블록을 반복실행
	WHILE PIN0==0H000F	입력PORT 0번의 IN0, IN1, IN2, IN3이 모두 "1"인 동안만 블록을 반복 실행
	WHILE PIN0==32	입력 PORT 0번의 입력신호상태가 IN4가 "1"인 동안만 블록을 반복 실행
출력	WHILE OUT0==1	출력 OUT0이 "1"인 동안만 블록을 반복실행
	WHILE POUT0==0H000F	출력 POUT0번의 OUT0, OUT1, OUT2, OUT3이 모두 " 1"인 동안만 블록을 반복 실행
	WHILE POUT0== 0B0000000011111111	출력 POUT0번의 OUT0, OUT1, OUT2, OUT3, OUT4, OUT5, OUT6, OUT7이 모두 "1"인 동안만 블록을 반복 실행
변수	WHILE TMP==1	변수 TMP가 "1"인 동안만 블록을 반복 실행
	WHILE MVR>10	MVR변수 값이 "10"보다 큰 동안만 블록을 반복 실행
	WHILE TMR>10	TMR(TIMER)변수값이 "10"보다 큰 동안만 블록을 반복 실행
	WHILE CNT>10	CNT(COUNT)변수값이 "10"보다 큰 동안만 블록을 반복 실행
	WHILE AP10.1==1.0	POSITION 변수 AP값의 1축 값이 "1.0"과 같은 동안만 블록을 반복 실행

3.4 IF, ENDIF(조건 분기 명령어)

기 능 조건식의 조건 판단 후 분기문을 수행합니다.

형 식
```
IF <조건식> THEN
 ⋮
ELSEIF <조건식> THEN          (조건 분기가 3개 이상인 경우 사용)
 ⋮
ELSE                          (조건 분기가 2개인 경우 사용)
 ⋮
ENDIF
```

용 어 <조건식> : 수식의 진위를 판단할 수 있는 논리연산식이나 비교연산식을 의미합니다.

설 명
1) **IF – ENDIF 는 반드시 같이** 사용하여야 합니다.
2) 조건식의 연산 결과가 **참(또는 0이외의 값)이면 THEN 이하의 문장**들이 실행되고, **거짓(또는 0)이면 ELSE 다음의 문장**들이 실행됩니다.
3) ELSE문은 프로그램 작성자가 **선택적으로 사용**할 수 있습니다.
4) IF 블록 내부에서 다시 IF 블록을 사용하는 중첩 IF 블록을 사용할 수 있습니다. 이때 주의할 것은 IF와 ENDIF가 논리적으로 쌍을 이룰수 있도록 작성해야 합니다.

3.4.1 프로그램 사용 예제

1) 입력 IN0의 상태에 따라 이동할 포인트 결정

```
MAIN
VEL 100
IF IN0==1 THEN
JMOV P1 ..... 입력 IN0 가 1인 경우 포인트 P1로 PTP 이동
ELSE
JMOV P2 ..... 입력 IN0 가 0인 경우 포인트 P2로 PTP 이동
ENDIF
EOP
```

2) 입력 Port "1"(IN0 ~ 15), IN0의 상태 검사

```
MAIN
INT AA, BB
AA=PIN1 ..... Port1(IN0~15) 상태를 변수 AA에 저장
IF (AA==0HFFFF) THEN  AA값 판단 (IN0~15 모두 1인지 여부)
STOP ..... "참" 일 경우, 로봇 스톱
ELSE
BB=IN0 ..... "거짓" 일 경우, IN0의 상태를 BB에 저장
IF (BB==1) THEN  BB가 1인지 판단
DLAY 10 .....
JMOV P0 ..... 포인트 P0로 이동
ENDIF
ENDIF
EOP
```

3) 아래 프로그램은 동일한 내용이다.
(단, IN0~IN3중 동시에 가해지는 입력이 하나뿐이란 가정하에 동일 내용)

```
    ⋮
LABL A0
IF IN0==1 THEN
GOTO B0
ENDIF
IF IN1==1 THEN
GOTO C0
ENDIF
IF IN2==1 THEN
GOTO D0
ENDIF
IF IN3==1 THEN
GOTO A0
ENDIF
    ⋮
```

=

```
    ⋮
LABL A0
IF IN0==1 THEN
GOTO B0
ELSE
IF IN1==1 THEN
GOTO C0
ELSE
IF IN2==1 THEN
GOTO D0
ELSE
IF IN3==1 THEN
GOTO A0
ENDIF
ENDIF
ENDIF
ENDIF
    ⋮
```

CAUTION

- **명령어 입력시 주의 사항**
  (O) IF_((IN0==1)&&(IN1==0))
  (O) IF_IN0==1&&IN1==0
  (O) IF_IN0==1
  &&IN1==0
- **IF의 개수와 ENDIF의 개수가 다를경우 "Syntax Error"가 발생 됩니다.**
- **ELSE 또는 ELSEIF 개수는 최대 10개로 제한 됩니다**

3.4.2 IF(조건식) THEN 사용 예제

IF(조건식)		설 명
입력	IF_IN0==1_THEN	입력 IN0이 "1"인가?
	IF_((IN0==1)&&(IN1==0))_THEN	입력신호 IN0=1, IN1=0 두 가지 조건이 동시에 만족하는가?
	IF_((IN0==1)||(IN1==0))_THEN	입력신호 IN0=1, IN1=0 두 가지 조건 중 최소 한가지 만족하는가?
	IF_PIN0==0H000F_THEN	입력PORT0번의 IN0, IN1, IN2, IN3이 모두 "1"인가?
	IF_PIN0==32_THEN	입력 PORT0번의 입력신호상태가 IN4가 "1"인가?
출력	IF_OUT0==1_THEN	출력 OUT0이 "1"인가?
	IF_POUT0==0H000F_THEN	출력 POUT0번의 OUT0, OUT1, OUT2, OUT3이 모두 " 1"인가?
	IF_POUT0== 0B0000000011111111_THEN	출력 POUT0번의 OUT0, OUT1, OUT2, OUT3, OUT4, OUT5, OUT6, OUT7이 모두 "1"인가?
변수	IF_TMP==1_THEN	변수 TMP가 "1"인가?
	IF_MVR>10_THEN	MVR변수 값이 "10"보다 큰가?
	IF_TMR>10_THEN	TMR(TIMER)변수 값이 "10"보다 큰가?
	IF_CNT>10_THEN	CNT(COUNT)변수 값이 "10"보다 큰가?
	IF_AP10.1==1_THEN	POSITION 변수 값(1축 값) "1"과 같은가?

3.5 LABL, GOTO (분기 명령어)

기 능 분기점 위치 지정, 수행문 분기 이동 명령어 입니다.

형 식 LABL <레이블명>
GOTO <레이블명>

용 어 <레이블명> : 영문자와 숫자로 구성된 8자리의 문자 스트링입니다.
(단, 작업 포인트를 표시하는 "P0, P1, ... "는 사용할 수 없습니다.)
예)LABL UNCLAMP1 **(O)**, GOTO CHECK2 **(O)**
LABL P1 **(X)**, GOTO P100 **(X)**

설 명 1) GOTO명령은 **조건 없이 해당 레이블로 분기**하며 동일 블록 내에서만 가능합니다.
예를 들면, MAIN 블록 내에서 SUBR블록으로, SUBR블록 내에서 다른 SUBR블록으로 분기할 수 없습니다.
2) IF블록이나 FOR, WHILE블록을 중첩하여 사용할 때, 내부에서 외부로의 분기는 가능 하지만, **반대의 경우는 불가능**합니다.

3.5.1 프로그램 사용 예제

1) 두 포인트 P1, P2를 반복 이동

```
MAIN
VEL 100
LABL A0
JMOV P1
JMOV P2
GOTO A0        "LABL A0"로 Step 이동
EOP`
```

=

```
MAIN
VEL 100
WHILE 1
JMOV P1
JMOV P2
ENDWL
EOP
```

2) 입력 IN0의 상태에 따라 분기

```
MAIN
VEL 100
JMOV P0
IF IN0==1 THEN
GOTO A0 ..... 입력 IN0이 1이면  LABL A0로 무조건 점프
ELSE
GOTO A1 ..... 아니면 LABL A1로 무조건 점프
ENDIF
JMOV P1
LABL A0 ..... LABL A0 분기지점
JMOV P2
GOTO A2 ..... LABL A2로 무조건 점프
LABL A1 ..... LABL A1 분기지점
JMOV P3
LABL A2 ..... LABL A2 분기지점
EOP
```

CAUTION

▶ 명령어 입력시 주의 사항

```
MAIN
VEL 100
JMOV P0
IF IN0==1 THEN
GOTO A0
ELSE
GOTO A1
ENDIF
JMOV P1
LABL A0
JMOV P2
GOTO A2
LABL A1
JMOV P3
CALL D0
LABL A2
EOP
SUBR D0
JMOV P5
GOTO A0
RET
```

(O) (O)

(O)

(X) : GOTO 문으로 다른 루틴 진입불가

3.6 SUBR, RET(부실행문 명령어)

기 능 프로그램 중에서 특정 동작을 반복하는 부분을 따로 작성하여, 필요에 따라 호출이 가능합니다.

형 식
```
SUBR <부 실행문 명>
⋮
RET
```

용 어 <부실행문명> : 영문자와 숫자로 구성된 8자리의 문자 스트링입니다.
(단, 작업 포인트를 표시하는 "P0, P1, ... "는 사용할 수 없습니다.)

설 명
1) SUBR – RET 는 반드시 같이 사용하여야 합니다.
2) SUBR 블록은 CALL문에 의해서만 호출될 수 있습니다.
3) RET문은 SUBR블록의 실행을 종료하고, CALL 다음 STEP으로 복귀하도록 합니다.
4) CALL문은 MAIN이나 SUBR블록에서 모두 사용할 수 있으며 재귀호출(Recursive Call)도 가능합니다.
5) SUBR 블럭은 반드시 MAIN - EOP 블록 다음에 정의하여야 합니다.

3.6.1 프로그램 사용 예제

```
MAIN
VEL 100
JMOV P0
CALL A0 ..... 부실행문 "A0"을 CALL
CALL A1 ..... 부실행문 "A1"을 CALL
EOP

SUBR A0
JMOV P1
RET ..... SUBR A0 블럭의 실행을 종료하고,
          CALL 다음 스텝으로 복귀
SUBR A1
JMOV P2
RET ..... SUBR A1 블럭의 실행을 종료하고,
          CALL 다음 스텝으로 복귀
```

3.7 CALL, JCALL (호출 명령어)

기 능 부실행문, JOB 파일을 호출합니다.

형 식 CALL <부실행문명>
JCALL <JOB파일명>

용 어 <부실행문명> : 영문자와 숫자로 구성된 8자리의 문자열입니다.
(단, 작업 포인트를 표시하는 "P0, P1, ... "는 사용할 수 없습니다.)

<JOB파일명> : 호출할 JOB 프로그램의 확장자(.JOB)를 제외한 파일명입니다.

설 명 ■ **CALL 명령어**

1) CALL문을 사용하려면 반드시 **호출될 부실행문(SUBR)이 존재**해야 합니다.
2) CALL문의 **부실행문명과 부실행문이 동일**해야 합니다.
3) SUBR블록에서 RET를 만나면 **CALL 다음 STEP으로 복귀**합니다.

■ **JCALL**

1) JCALL은 JOB을 서브루틴으로 사용하는 것입니다.
2) 호출된 JOB에서 EOP를 만나면 호출한 프로그램의 JCALL문 다음으로 복귀합니다.
3) JCALL은 재귀호출이 되지 않도록 프로그램을 작성해야 합니다.

※ **재귀호출(Recursive Call)의 경우 아래와 같이 제한 횟수가 있습니다.**
주) CALL문 재귀 호출 **최대 8회** → 초과하면 "CALL DEPTH ERROR"
JCALL문 재귀 호출 **최대 3회** → 초과하면 "JCALL DEPTH ERROR"

CAUTION

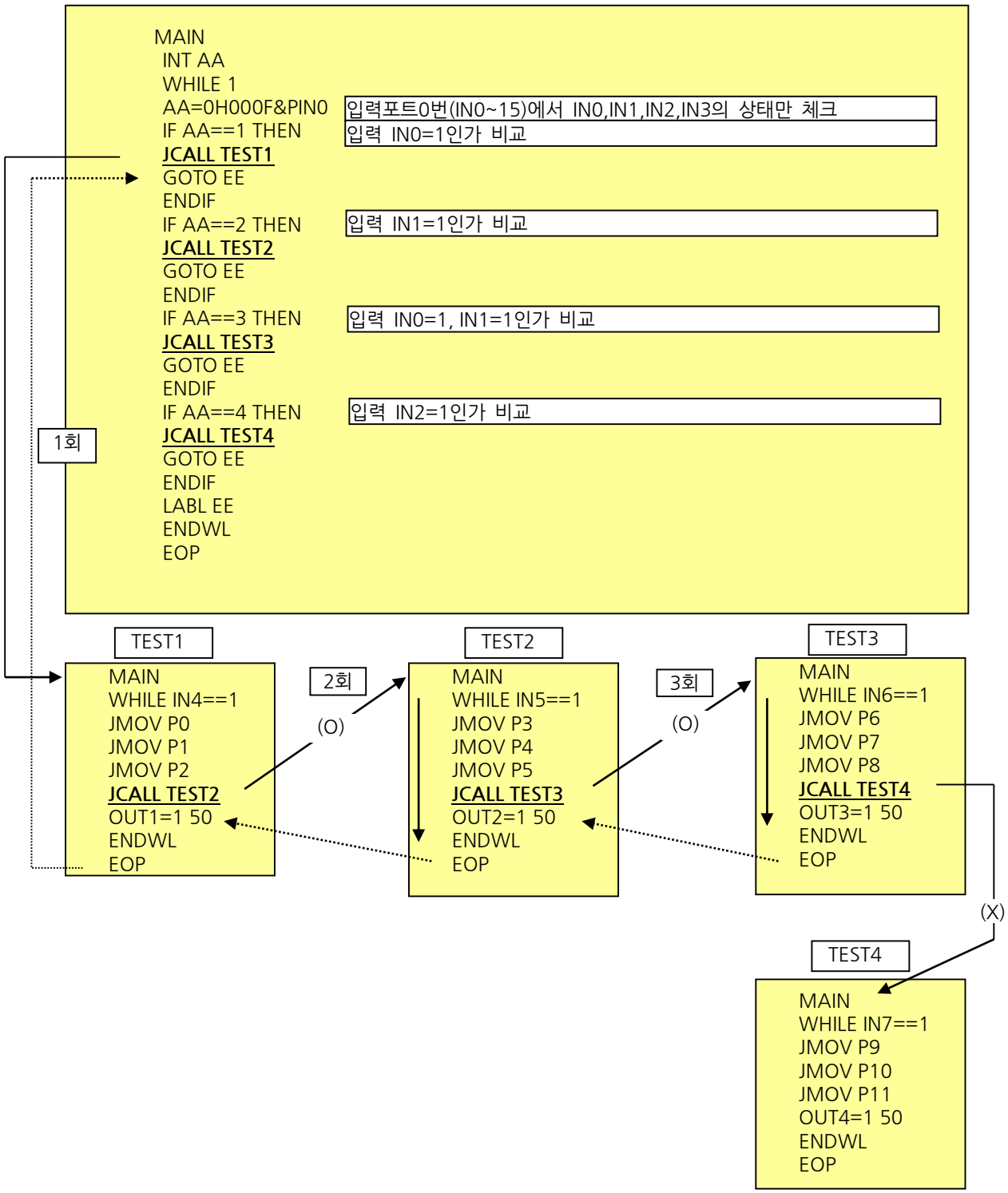
▶ **명령어 입력시 주의 사항**

```
MAIN
VEL 100
LABL A0
JMOV P1
JMOV P2
CALL A0
GOTO A0
EOP
SUBR A0
OUT1=1
IN1=1
RET
```

(X) LABL명과 SUBR명이 같으면 ERROR 발생 함.

3.7.1 프로그램 사용 예제

1) 입력포트 0번의 상태에 따라 JOB 파일명을 "TEST1"~"TEST4"중 하나를 JCALL 호출



3.8 STOP, EXIT(로봇 및 JOB 정지 명령어)

기 능 로봇 동작 정지(STOP), JOB 수행정지(EXIT)

형 식 STOP
EXIT

설 명
1) STOP 명령어는 **이동중인 로봇을 정지**시키고, 다음 STEP의 명령어를 처리합니다. 즉, **수행 중인 모션 정지 후 다음 JOB STEP을 계속 수행**합니다.
2) EXIT 명령어는 **JOB 수행을 중지**시킵니다. 이때 제어기는 **"EXIT Instruction"** 알람을 발생시킵니다.

3.8.1 프로그램 사용 예제

1) 포인트이동 후 출력 OUT1을 ON, OFF

```
MAIN
VEL 500
JMOV P1
MVR=0 ..... "이동구간 백분율"을 초기화
WITH .....
JMOV P2 ..... 포인트 P2로 이동(JMOV만 가능함)
WHILE MVR<60 ..... 이동구간의 60%미만 동안에
IF IN0==1 THEN ..... IN0=1이면 "LABL BB"로 점프
GOTO BB
ENDIF
ENDWL
OUT0=1
ENDWT

LABL BB
STOP ..... 로봇이동 동작 정지
ENDWT ..... WITH문 종료 (반드시 삽입)
JMOV P3
OUT1=1
EOP
```

3.9 JMOV(PTP 이동 명령어)

기 능 현 위치에서 목표점으로 축 보간 이동합니다.

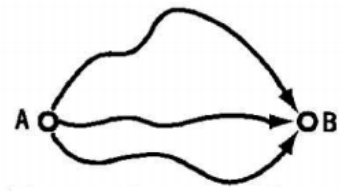
형 식 JMOV P<번호>
JMOV GP<번호>
JMOV <위치형변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P<번호> : JOB 별로 개별 사용 되는 LOCAL POINT
GP<번호> : 공통으로 사용 되는 GLOBAL POINT

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명
1) PTP(Point to Point)란 **점에서 점으로의 이동**을 의미합니다.
2) PTP 이동 경로는 로봇 자세에 의존하며 **직선동작에 한정되지는 않습니다**.



동작경로가 직선으로 한정되지 않는다.

3.9.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P10          .....   포인트 P10 으로 PTP 이동
JMOV P11          .....   포인트 P11 으로 PTP 이동
JMOV P100         .....   포인트 P100 으로 PTP 이동
EOP
```

2) 위치형 변수를 이용한 이동(1)

```
MAIN
POS XA                    .....   포인트형 변수 XA 선언
XA=<400.0,50.0,10.4,10.0,0,0,1>   .....   XA의 속성값을 지정
                                    <X,Y,Z,W,E1,E2,FORM>
VEL 100
JMOV P0           .....   포인트 P10 으로 PTP 이동
JMOV XA           .....   XA 위치 좌표로 PTP 이동
JMOV P100         .....   포인트 P100 으로 PTP 이동
EOP
```

3) 위치형 변수를 이용한 이동(2)

```
MAIN
POS AP,AP1,AP2            .....   포인트형 변수 AP,AP1,AP2를 선언
AP=P10 +<10,10,10,10,0,0>     .....   포인트 P10의 각축값에 10을 더한 값을 저장
AP1=P11 -<10,10,10,10,0,0>    .....   포인트 P11의 각축값에 10을 뺀 값을 저장
AP2=P100                  .....   포인트 AP2에 P100의 값을 저장
AP2.3=P100.3+10           .....   포인트 P100의 Z축값에 10을 더한 값을 저장
JMOV AP           .....   AP 위치 좌표로 PTP 이동
JMOV AP1          .....   AP1 위치 좌표로 PTP 이동
JMOV AP2          .....   AP2 위치 좌표로 PTP 이동
EOP
```

3.10 LMOV(CP 이동 명령어)

기 능 현위치에서 목표점으로 직선보간 이동합니다.

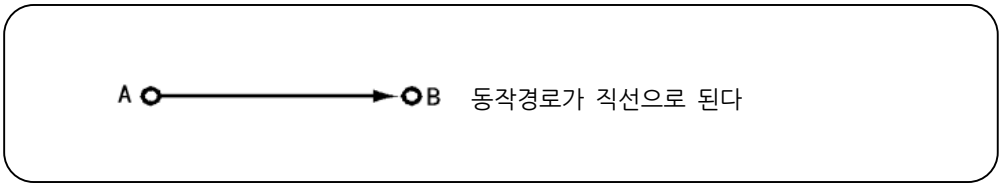
형 식 LMOV P<번호>
LMOV GP<번호>
LMOV <위치형변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P<번호> : JOB 별로 개별 사용 되는 LOCAL POINT
GP<번호> : 공통으로 사용 되는 GLOBAL POINT

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명 CP제어는 동작목표 위치에 도달하는 경로를 **직선이 되도록 보간제어**합니다.



CAUTION

- ▶ CP제어를 실시하면 로봇의 위치에 따라 동작할 수 없는 경우가 있습니다.
  이때 제어기는 "Unreachable Point" 또는 "Inverse Error" 등의 Alarm이 발생 됩니다.
- ▶ 특히 "FOS", "FIX", "TOOL" 명령어 사용 시 주의 하시기 바랍니다.
- ▶ SCARA Robot은 현재 ARM FORM을 유지하여 보간 동작을 수행합니다.
- ▶ 프로그램의 처음 동작명령에는 PTP 제어를 사용해야 합니다.
- ▶ SCARA Robot에서 W축을 보간 동작 할 때 유의사항
  1. 파라미터의 "OFFS", "LENG를 기계부의 Sticker에 명시된 값으로 정확하게 설정합니다.
  2. TOOL OFFSET이 있을 때 W축 보간 동작에는 두가지 방법이 있습니다.
     W축 자세고정 (로봇명령어 "FIX 1") W축 자세 회전 (로봇 명령어 "FIX 0")
     (3.35장 FIX 명령어를 참조바랍니다.)
  3. TOOL OFFSET이 있는 경우 작업 포인트 티칭에 주의 해야 합니다.
     W축 자세 고정 : TOOL 방향을 일정한 방향으로 유지 하면서 티칭
     W축 자세 회전 : TOOL 오프셋을 정확하게 설정 후 TOOL 선정 및 작업 포인트티칭
  4. 가급적 W축 자세를 고정하여 작업 할 수 있도록 TOOL 선정 및 작업 포인트 티칭을
     하십시오.
     (특히 W축 자세 회전은 연속된 보간 동작 중 원호를 그릴 때 (AMOV) 적용되지 않습니다)
  5. 로봇 기계부가 제어기의 지령에 추종 하지 못하는 경우(Unreachable Point)가
     발생 할 수 있습니다.

3.10.1 TOOL 방향을 일정 하게 유지 하면서 티칭(SCARA 로봇)

Step 1. PALLET 화면 이동

```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
A : 0.13          B : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

포인트 화면으로 이동 합니다.

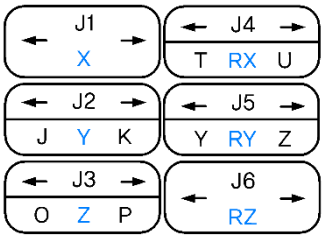
```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
A : 0.13          B : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

F2

"F2"키를 입력하여 좌표계를 원통 → 직교 로 변경 합니다.

```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
X : 595.32        Y : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

직교좌표계 화면에서 JOG 동작을 실시하면 W축자세를 유지한 상태로 이동 됩니다.




```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
X : 400.38        Y : 0.22
Z : 50            W : 65
EXCH   CORD   PJUMP   FWRD
```

CAUTION

▶ 티칭방법은 N1-OM "포인트 수정(위치 변경)"항목을 참고하시기 바랍니다.

3.10.2 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P10 ..... 포인트 P10 으로 PTP 이동
LMOV P11 ..... 포인트 P11 으로 CP 이동
LMOV P100 ..... 포인트 P100 으로 CP 이동
EOP
```

2) 위치형 변수를 이용한 이동(1)

```
MAIN
POS XA ..... 포인트형 변수 XA 선언
XA=<400.0,50.0,10.4,10.0,0,0,1> ..... XA의 속성값을 지정
                                      <X,Y,Z,W,E1,E2,FORM>
VEL 100
JMOV P10 ..... 포인트 P10 으로 PTP 이동
LMOV XA ..... XA 위치 좌표로 CP 이동
LMOV P100 ..... 포인트 P100 으로 CP 이동
EOP
```

3) 위치형 변수를 이용한 이동(2)

```
MAIN
POS AP,AP1,AP2 ..... 포인트형 변수 AP,AP1,AP2를 선언
AP=P10 +<10,10,10,10,0,0> ..... 포인트 P10의 각축값에 10을 더한 값을 저장
AP1=P11 -<10,10,10,10,0,0> ..... 포인트 P11의 각축값에 10을 뺀 값을 저장
AP2=P100 ..... 포인트 AP2에 P100의 값을 저장
AP2.3=P100.3+10 ..... 포인트 P100의 Z축값에 10을 더한 값을 저장
JMOV AP ..... AP 위치 좌표로 PTP 이동
LMOV AP1 ..... AP1 위치 좌표로 CP 이동
LMOV AP2 ..... AP2 위치 좌표로 CP 이동
EOP
```

3.11 CMOV(원형 보간 이동 명령어)

기 능 현위치에서 경유점1, 2를 잇는 원을 그리며 이동합니다.

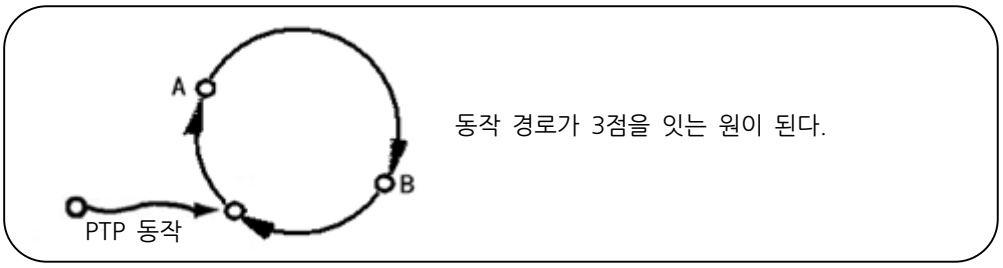
형 식 CMOV P<경유점1 번호> P<경유점2 번호>
CMOV GP<경유점1 번호> GP<경유점2 번호>
CMOV <경유점1 위치형 변수> <경유점2 위치형 변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P<번호> : JOB 별로 개별 사용 되는 LOCAL POINT
GP<번호> : 공통으로 사용 되는 GLOBAL POINT

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명 1) 동작 목표위치에 도달하는 경로를 **원을 그리도록 보간 제어**합니다.
2) 원형보간 동작을 할때에는 **3개의 포인트 좌표가 필요** 합니다.



3.11.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P1          .....   포인트 P1 으로 PTP 이동
CMOV P2 P3       .....   포인트 P2 P3를 경유하는 원형 보간 이동
EOP
```

CAUTION

- ▶ 잘못된 Point Teaching으로 Auto RUN을 실행시 동작할 수 없는 경우가 있습니다.
  이때 제어기는 "Unreachable Point" 또는 "Inverse Error" 등의 Alarm이 발생 됩니다.
- ▶ 특히 "FOS", "FIX", "TOOL" 명령어 사용 시 주의 하시기 바랍니다.
- ▶ W축의 보간 동작에 대하여는 "LMOV"명령어를 참조 하십시오.
- ▶ CMOV, AMOV는 단독으로 사용 할 수 없으며 예제와 같이 JMOV를 추가하십시오.

3.12 AMOV(원호 보간 이동 명령어)

기 능 현위치에서 경유점1, 2를 잇는 원호를 그리며 이동합니다.

형 식 AMOV P<경유점1 번호> P<경유점2 번호>
AMOV GP<경유점1 번호> GP<경유점2 번호>
AMOV <경유점1 위치형 변수> <경유점2 위치형 변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P<번호>: JOB 별로 개별 사용 되는 LOCAL POINT
GP<번호> : 공통으로 사용 되는 GLOBAL POINT

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명 1) 동작 목표위치에 도달하는 경로를 **원호를 그리도록 보간 제어**합니다.
2) 원호보간 동작을 할때에는 **3개의 포인트 좌표가 필요** 합니다.

동작 경로가 3점을 잇는 원호가 된다.

3.12.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P1          .....   포인트 P1 으로 PTP 이동
AMOV P2 P3       .....   포인트 P2 P3를 경유하는 원호 보간 이동
EOP
```

CAUTION

- ▶ 잘못된 Point Teaching으로 Auto RUN을 실행시 동작할 수 없는 경우가 있습니다.
  이때 제어기는 "Unreachable Point" 또는 "Inverse Error" 등의 Alarm이 발생 됩니다.
- ▶ 특히 "FOS", "FIX", "TOOL" 명령어 사용 시 주의 하시기 바랍니다.
- ▶ W축의 보간 동작에 대하여는 "LMOV"명령어를 참조 하십시오.
- ▶ CMOV, AMOV는 단독으로 사용 할 수 없으며 예제와 같이 JMOV를 추가하십시오.

3.13 IMOV, IMOV2 (증분 이동 명령어)

기 능 현위치에서 중분량으로 축보간 이동합니다.
IMOV – PTP 이동
IMOV2 – CP 이동

형 식 IMOV P<번호>
IMOV GP<번호>
IMOV <위치형변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P<번호> : JOB 별로 개별 사용 되는 LOCAL POINT
GP<번호>: 공통으로 사용 되는 GLOBAL POINT

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명 1) 지정된 증분량만큼 **상대 이동**합니다.
2) IMOV는 **현재위치를 기준**으로 하여 증분량만큼 이동하는 것입니다.

IMOV

동작경로가 직선으로 한정 되지 않는다.

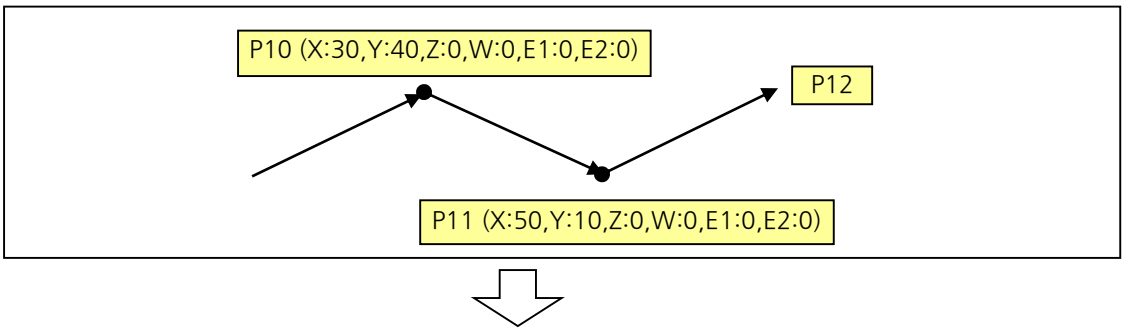
IMOV2

동작경로가 직선보간으로 이동합니다.

CAUTION

- IMOV를 위한 Point Teaching에서 위치좌표를 저장할 때 파라미터의 RANG"이내의 값으로 저장하지만 이 값은 원점(ORIGIN)을 기준으로 설정된 위치값입니다. 따라서 이전 Point를 기준으로 로봇이 이동(IMOV)할 때 "Range over error"가 발생할 수 있습니다. 파라미터 "RANG"설정을 조정하십시오.
- IMOV는 이전 위치를 기준으로 이동함으로 이전위치의 In Position정도에 따라 위치편차가 발생할 수 있습니다. 이때 작업프로그램에 "INPOS","DLAY"등을 추가 하여 로봇이 이전 위치에 정확하게 In Position 완료 후 IMOV가 이루어 지도록 프로그램을 수정하시기 바랍니다.

3.13.1 프로그램 사용 예제



1) IMOV 사용(포인트 P1 사용)

```
MAIN
VEL 100
JMOV P10 ..... 포인트 P10 으로 PTP 이동
JMOV P11 포인트 P11 으로 PTP 이동
INPOS 10 ..... INPOS 또는 DLAY 사용 하여 위치 편차 예방.
IMOV P1 ..... 포인트 P11을 기준으로
포인트 P1 만큼 PTP 이동
EOP
```

2) IMOV 사용(위치형변수 AP 사용)

```
MAIN
POS AP ..... 포인트형 변수 AP 선언
AP=<20.0,-30.0,0,0.0,0,0,1> ..... AP의 속성값을 지정 <X,Y,Z,W,E1,E2,FORM>
VEL 100
JMOV P10 ..... 포인트 P10 으로 PTP 이동
IMOV AP ..... 포인트 P10을 기준으로 AP 값 만큼 PTP 이동
JMOV P100 ..... 포인트 P100 으로 PTP 이동
EOP
```

3) IMOV2 사용

```
MAIN
VEL 100
LMOV P10 ..... 포인트 P10 으로 CP 이동
LMOV P11 포인트 P11 으로 CP 이동
INPOS 10 ..... INPOS 또는 DLAY 사용 하여 위치 편차 예방.
IMOV2 P1 ..... 포인트 P11을 기준으로
포인트 P1 만큼 CP 이동
EOP
```

4) IMOV2 사용(위치형변수 AP 사용)

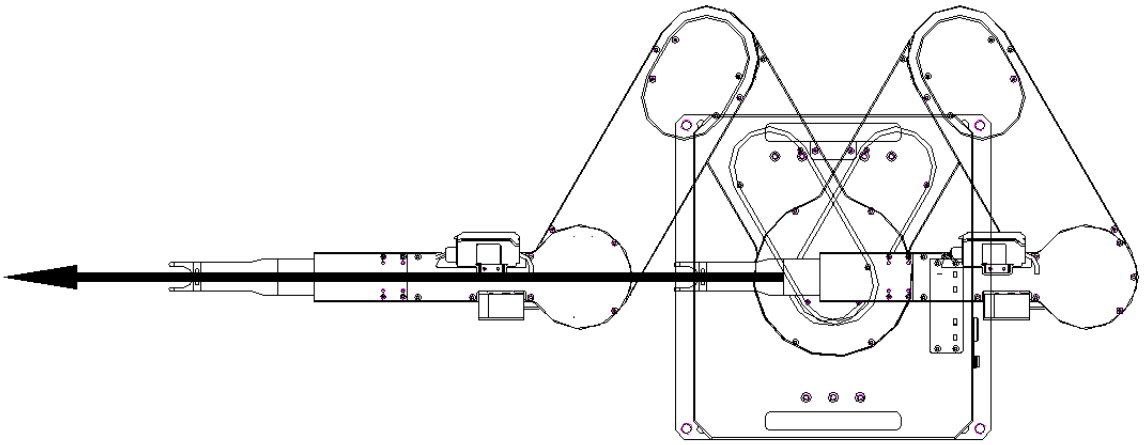
```
MAIN
POS AP ..... 포인트형 변수 AP 선언
AP=<20.0,-30.0,0,0.0,0,0,1> ..... AP의 속성값을 지정 <X,Y,Z,W,E1,E2,FORM>
VEL 100
LMOV P10 ..... 포인트 P10 으로 CP 이동
IMOV2 AP ..... 포인트 P10을 기준으로 AP 값 만큼 CP 이동
LMOV P100 ..... 포인트 P100 으로 CP 이동
EOP
```

3.14 JNTSYN (PTP 동기 모드 명령어)

기 능 PTP 동기 모드를 선택합니다.
이 명령어는 ROSEP Robot 및 Ver 03.02.08 이후 버전에서만 사용 가능 합니다

형 식 JNTSYN <형식>
JNTSYN 0 : PTP 동기 모드 사용을 안합니다.
JNTSYN 1 : PTP 동기 모드 사용을 합니다.

설 명
1) 특이점 근처에서 직선보간모션(LMOV)을 수행 할 경우, 특이점에서 급격한 속도 변화로 인해 Over Speed 알람이 발생 합니다.이러한 문제점을 해결 하기 위해 ROSEP로봇 타입의 경우 PTP 모션(JMOV)을 동기로 구동하여 특이점을 지나는 직선 모션이 가능 합니다.
2) "JNTSYN 0"으로 설정 되면, 일반적인 JMOV와 동일하게 동작 합니다.
3) "JNTSYN 1"으로 설정 되면 J2, J4, J5의 이동량을 J1의 이동량에 동기시켜 특이점을 통과하는 직선 모션을 합니다. (J1(A), J2(B), J3(Z), J4(W1), J5(W2))
4) 축별 이동거리를 비교하여 동기조건이 성립하는지 확인 합니다.
W1, W2가 동기조건을 만족하지 못하면 동기식에서 제외합니다.
A, B가 동기 조건을 만족하지 못하면 알람(1422: PTP Sched. Error)이 발생합니다.
예) P1 =<-25, 230, 0, -115, -115, 0>, P2 =<25, 130, 0, -65, -65, 0>
P2.1 - P1.1= 50 = delta_TH1
P2.2 - P1.2= -100 = -2 * delta_TH1
P2.4 - P1.4= 50 = delta_TH1
P2.5 - P1.5= 50 = delta_TH1



[ROSEP ROBOT TYPE의 특이점 통과]

3.14.1 프로그램 사용 예제

MAIN	.....	Program 시작
VEL 10	.....	축 이동 속도
JMOV P1	.....	P1=<-25, 230, 0, -115, -115, 0>로 이동
WHILE 1	.....	조건반복수행 시작
JNTSYN 1	.....	PTP 동기 모드 사용 함
JMOV P2	.....	P2=<25, 130, 0, -65, -65, 0>로 이동
JMOV P1	.....	P1=<-25, 230, 0, -115, -115, 0>로 이동
JNTSYN 0	.....	PTP 동기 모드 사용 안함
ENDWL	.....	조건반복문 종료
EOP	.....	Program 종료

CAUTION

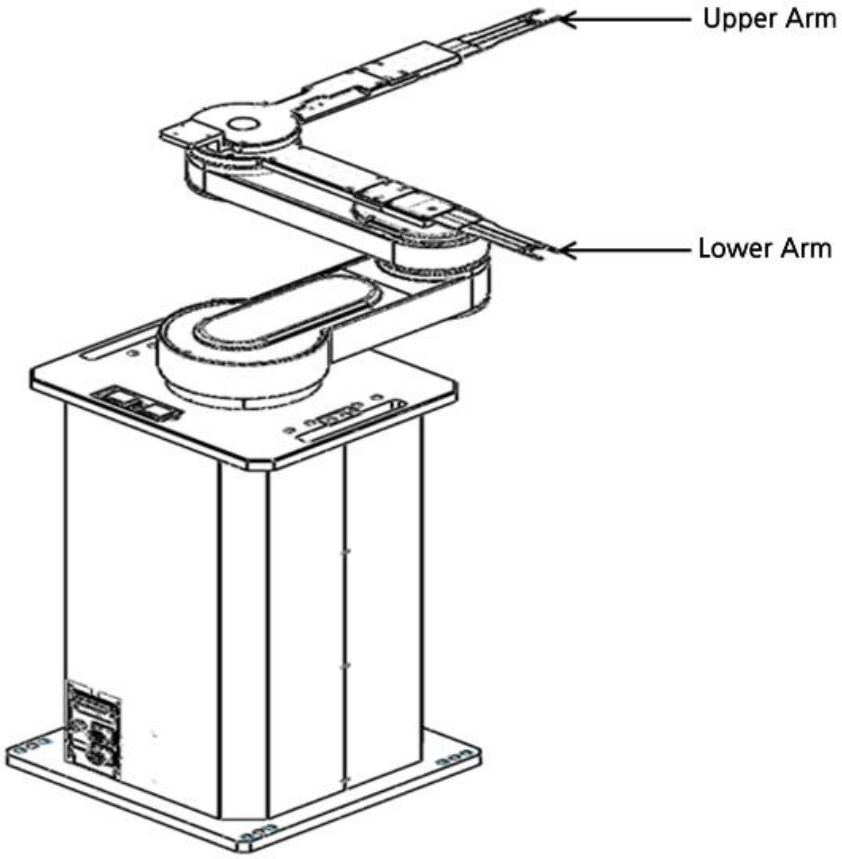
- ▶ 특이점 구간에서 직선모션을 하기 위해서 PTP동기 모드로 이동 할 때 J1(A) 이동량에 비해 J2(B) 이동량이 2배가 되어야 합니다.(J1×2 == J2)
  예) J1(A) 이동량이 50° 이면, J2(B) 이동량은 100° 입니다.
  ※ 이동량이 2배가 아닌경우 "PTP Sched. Error " 알람이 발생 합니다.
- ▶ J4(W1),J5(W2)의 이동량이 J1(A)이동량과 같지 않으면, PTP동기 모드가 아닌 개별축 이동(PTP) 모션으로 변경 됩니다.

3.15 EECH (말단 장치 선택 명령어)

기 능 말단 장치(W1, W2)를 선택합니다.
이 명령어는 ROSEP Robot 및 Ver 03.02.08 이후 버전에서만 사용 가능 합니다.

형 식 EECH <형식>
EECH 1: 첫 번째 W1축 및 TOOL 0번을 선택합니다.
EECH 2: 두 번째 W2축 및 TOOL 1번을 선택합니다.

설 명
1) ROSEP Robot의 경우 말단 장치가 2개(W1, W2)로 구성되어 있어 XY좌표 변환 시 적용할 말단 장치를 선택 하는 명령어 입니다.
2) "EECH 1"으로 설정 하면, 로봇의 W1축 (Lower Arm)과 말단 장치의 Tool 0번이 선택 됩니다.
3) "EECH 2"으로 선택이 되면 로봇의 W2축(Upper Arm)과 말단 장치의 Tool 1번이 선택 됩니다.



[ROSEP ROBOT Arm 선택]

3.15.1 프로그램 사용 예제

MAIN	.....	Program 시작
VEL 10	.....	이동 속도 설정
WHILE 1	.....	조건반복수행 시작
JMOV P0	.....	포인트 P0으로 PTP 이동
JMOV P1	.....	포인트 P1으로 PTP 이동
EECH 1	.....	로봇의 W1 말단 장치 선택
LMOV P2	.....	포인트 P2으로 CP 이동
LMOV P1	.....	포인트 P1으로 CP 이동
JMOV P3	.....	포인트 P3으로 PTP 이동
EECH 2	.....	로봇의 W2 말단 장치 선택
LMOV P4	.....	포인트 P4으로 CP 이동
LMOV P3	.....	포인트 P3으로 CP 이동
ENDWL	.....	조건반복문 종료
EOP	.....	Program 종료

3.16 TOOL(사용 툴 선택 명령어)

기 능　　로봇에 부착된 사용 툴을 선택합니다.

형 식　　TOOL <툴번호>

용 어　　<툴번호> : 파라미터에서 "TOOL" 옵셋값(dx, dy, dz)이 저장된 툴(TOOL)의 번호를 설정
(0 ≤ <툴번호> ≤ 7)

설 명　　1) 작업시 사용 하는 툴의 번호를 설정 합니다.
2) 툴의 형상은 시스템 파라미터에서 설정 합니다.

3.16.1 프로그램 사용 예제

1) 툴(TOOL)을 사용한 보간 이동

```
MAIN
VEL 300
TOOL 1          .....   툴(TOOL)1번 을 사용 하여 보간동작 시작
FIX 0
LMOV P1
LMOV P2
OUT0=0
FIX1
TOOL 0          .....   옵셋이 없는 툴 0번으로 변경
EOP
```

CAUTION

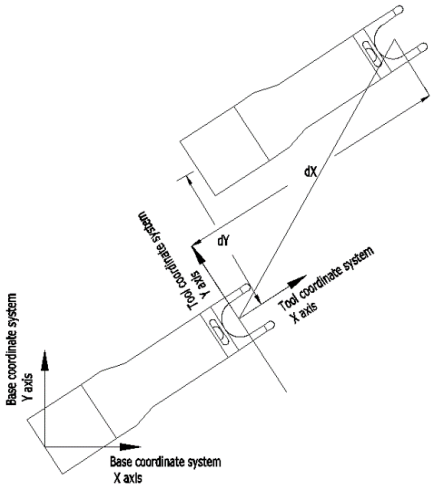
- ▶ 수평 다관절 로봇에서 TOOL 0의 옵셋 값은 티치펜던트 JOG의 기준이 됩니다.
- ▶ 작업 프로그램에서 TOOL 명령어를 사용하지 않으면 "TOOL 0" 옵셋값이 적용됩니다.
- ▶ 가급적 "TOOL 0"은 TOOL옵셋 초기화용으로 사용하고 TOOL 옵셋 적용은 1~7 중 하나를 사용 하십시오.
- ▶ 옵셋이 있는 TOOL을 사용하는 작업에서 작업 POINT 티칭
  1. 정확한 TOOL 옵셋 설정(파라미터 참조)
  2. W축을 고정(FIX 1) 또는 회전(FIX 0) 중 1가지를 설정 후 작업 포인트 티칭("LMOV 명령어" 설명 참조)
- ▶ 연속된 보간 동작이 있는 작업(Sealing 또는 Dispensing)에서는 툴의 끝단으로 작업 POINT를 티칭 할 때 A(X), B(Y) 축에 무리한 동작을 요구하은 티칭이 될 수 있습니다.
- ▶ 작업 POINT 티칭에 주의 하십시오.
- ▶ TOOL 제작 및 고정
  1. "LMOV 명령어"에서 설명한 W축 자세 고정이 가능하도록 TOOL 제작
  2. 툴 옵셋이 dx,dy 둘 중 한쪽 방향으로만 존재 하도록 툴 고정

3.17 TIMOV(TOOL 좌표계 기준 증분 이동 명령어)

기 능 현재 위치에서 TOOL 좌표계 기준으로 증분량만큼 직선 보간 이동합니다.

형 식 TIMOV <위치형 변수>
<위치형 변수> : POS형으로 선언된 변수 이름을 의미 합니다.
(XY 좌표를 기준으로 이동 하기 때문에 위치형 변수 이름은 X으로 시작 해야 합니다.)
예) POS XP, XCUR

설 명 1) TIMOV는 TOOL 좌표계 기준으로 현재 위치에서 지정된 증분량(dx, dy, dz)만큼 상대 이동 합니다.



3.17.1 프로그램 사용 예제

```
MAIN
VEL 300
TOOL 1          .....   툴(TOOL)1번 을 사용 하여 보간동작 시작
FIX 0
JMOV P0         .....   대기 위치 P0 이동
TIMOV P1        .....   X방향으로 50mm 이동 <50 0 0 0 0 0>
TIMOV P2        .....   Y방향으로 -50mm 이동 <0 -50 0 0 0 0>
TOOL 0                  옵셋이 없는 툴 0번으로 변경
EOP
```

CAUTION

- TIMOV를 위한 Point Teaching에서 위치좌표를 저장할 때 파라미터의 RANG"이내의 값으로 저장하지만 이 값은 원점(ORIGIN)을 기준으로 설정된 위치값입니다. 따라서 이전 Point를 기준으로 로봇이 이동(TIMOV)할 때 "Range over error"가 발생할 수 있습니다. 파라미터 "RANG"설정을 조정하십시오.
- TIMOV는 이전 위치를 기준으로 이동함으로 이전위치의 In Position정도에 따라 위치편차가 발생할 수 있습니다. 이때 작업프로그램에 "INPOS","DLAY"등을 추가 하여 로봇이 이전 위치에 정확하게 In Position 완료 후 TIMOV가 이루어 지도록 프로그램을 수정하시기 바랍니다.
- 이 명령어는 ROSEP Robot 및 Ver 03.02.08 이후 버전에서만 사용 가능 합니다.

3.18 HMOV (원점 이동 명령어)

기 능 현위치에서 지정된 홈 위치로 이동

형 식 HMOV <정수형 변수 또는 번호>

용 어 <번호> : 파라미터에서 설정된 원점 좌표값 번호 입니다.
(0 ≤ 번호 ≤ 3)

<정수형 변수> : INT 형으로 선언된 정수형 변수의 이름을 의미 합니다.

설 명 1) ORG 파라미터의 HOME 에서 설정한 **원점 좌표 값으로 로봇을 PTP이동** 합니다.
2) 원점 좌표값 번호는 0번 ~ 3번 (HMOV 0, HMOV 1, HMOV 2, HMOV 3) 까지 4개의 값을 가집니다.

3.18.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P1          .....   포인트 P1 으로 PTP 이동
HMOV 0           .....   원점 수행 SEQ로 HOME 0으로 이동
HMOV 1           .....   HOME 1 으로 PTP 이동
EOP
```

CAUTION

- ▶ HMOV 0로 이동 하는 경우에는 **ORG 파라미터의 SEQ에서 설정**한 원점 순서에 따라 이동합니다.
- ▶ HMOV1 ~ HMOV 3으로 이동 하는 경우에는 **SEQ와 무관하게 PTP 이동** 합니다.

3.19 PMOV(Palletizing 이동)

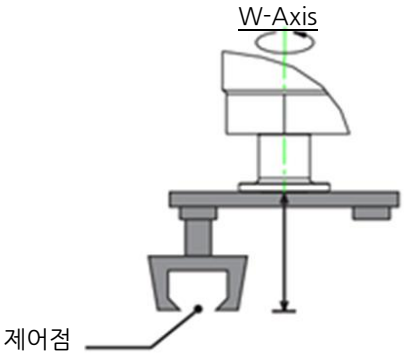
기 능　　현위치에서 지정된 Palletizing 작업 수행을 합니다.

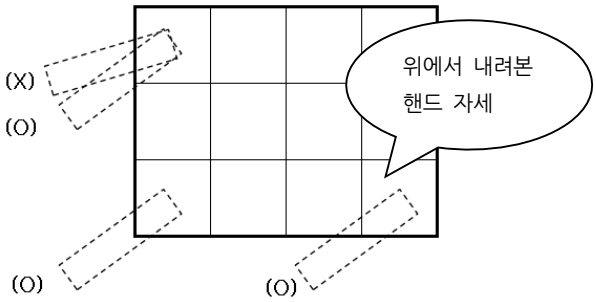
형 식　　PMOV <작업 Pallet 번호> <작업 기준점>

용 어　　<작업 Pallet 번호> : 파라미터에서 설정한 Pallet 번호를 입력 합니다.
(0 ≤ 번호 ≤ 99)

<작업 기준점> : Pallet 위치 4개의 티칭 포인트 중 처음 포인트 번호를 입력 합니다.
(0 ≤ 로컬 포인트 번호 ≤ 1996)
(0 ≤ 글로벌 포인트 번호 ≤ 1020)
예) PMOV P0 P1 인 경우: Pallet번호(P0),기준점 (P1), X방향(P2), Y방향(P3), Z방향(P4)

설 명　　1) PTP 동작으로 Palletizing 작업 위치로 이동 합니다.
2) PMOV 수행은 파라미터의 카운터(CNT)에서 지정한 Pallet상의 위치로 이동 하는 것 입니다.
3) Palletizing 작업을 수행하려면 두 가지 내용이 설정 되어 있어야 합니다.
가) 파라미터 모드에서 팔레트 정보 입력
(최초 정보 입력하기 전에 Pallet 데이터를 초기화 하십시오.)
나) Pallet 상의 4개의 포인트를 티칭 하십시오





CAUTION

- ▶ 기계부가 수평 다관절 로봇(SCARA 로봇) 일때는 A, B arm의 "OFFSET" 값을 정확하게 설정 하십시오.
- ▶ Pallet 작업순서는 파라미터에서의 TYPE에 따라 결정됩니다. (취급 설명서를 참조 바랍니다.)
- ▶ 회전 축 (W축)이 있는 로봇에서 Z축 끝단(End effect)에 핸드(HAND)를 부착 하였을 경우 팔레트 티칭 방법은 아래와 그림과 같이 핸드자세를 동일한 상태 유지한채 티칭해야 합니다.

3.19.1 TOOL 방향을 일정 하게 유지 하면서 티칭(SCARA 로봇)

Step 1. PALLET 화면 이동

```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
A : 0.13          B : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

포인트 화면으로 이동 합니다.

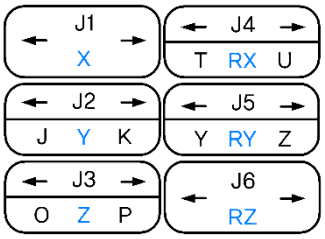
```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
A : 0.13          B : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

"F2"키를 입력하여 좌표계를 원통 → 직교 로 변경 합니다.

F2

```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
X : 595.32        Y : 0.12
Z : 0.11          W : 65
EXCH   CORD   PJUMP   FWRD
```

직교좌표계 화면에서 JOG 동작을 실시하면 W축자세를 유지한 상태로 이동 됩니다.




```
< RBSA804A : EDIT>  V:     50
F:TEST   P0          NW   B   L
X : 400.38        Y : 11.77
Z : 51            W : 65
EXCH   CORD   PJUMP   FWRD
```

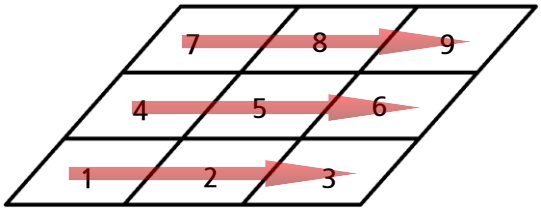
CAUTION

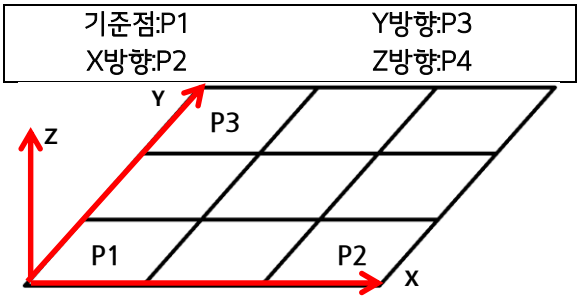
▶ 티칭방법은 N1-OM "포인트 수정(위치 변경)"항목을 참고하시기 바랍니다.

3.19.2 Pallet 작업 예

1) 예제 1(PMOV P1 P1)

가) 작업순서→ 포인트 티칭





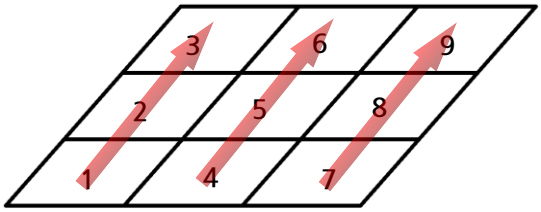
나) DATA 설정

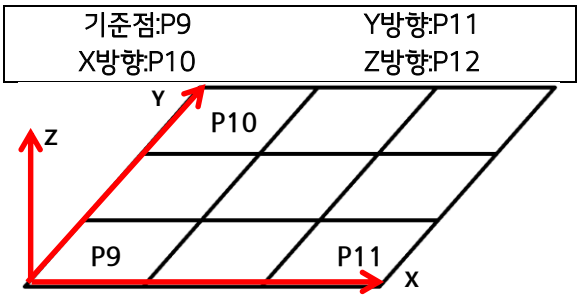
```
<PALLET-DATA>
    Pallet NO=01
1. XWM=03
2. YWM=03
3. ZWM=01

Update OK? (ENT/ESC)
```

P4 포인트 티칭은 1단인 경우 P4=P1
2단 인 경우 P4는 P1의 Z축값만 다릅니다.

2) 예제 2(PMOV P2 P9)

가) 작업순서→ 포인트 티칭





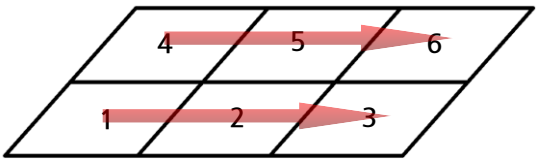
나) DATA 설정

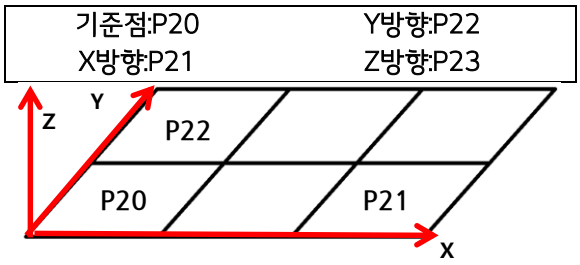
```
<PALLET-DATA>
    Pallet NO=02
1. XWM=03
2. YWM=03
3. ZWM=01

Update OK? (ENT/ESC)
```

P12 포인트 티칭은 1단인 경우 P12=P1
2단 인 경우 P12는 P9의 Z축값만 다릅니다.

3) 예제 3(PMOV P3 P20)

가) 작업순서→ 포인트 티칭





나) DATA 설정

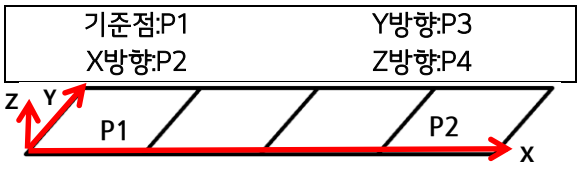
<PALLET-DATA>
Pallet NO=03
1. XWM=03
2. YWM=02
3. ZWM=01

Update OK? (ENT/ESC)

P23 포인트 티칭은 1단인 경우 P23=P20
2단 인 경우 P23는 P20의 Z축값만 다릅니다.

4) 예제 4(PMOV P4 P1)

가) 작업순서→ 포인트 티칭



나) DATA 설정

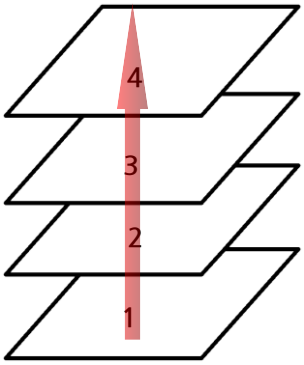
<PALLET-DATA>
Pallet NO=04
1. XWM=04
2. YWM=01
3. ZWM=01

Update OK? (ENT/ESC)

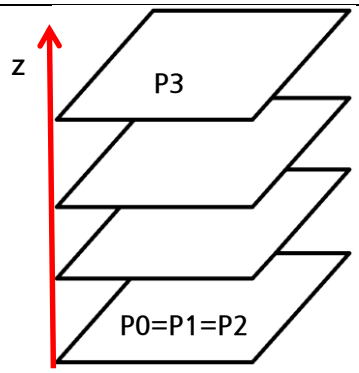
P4 포인트 티칭은 1단인 경우 P4=P1
2단 인 경우 P4는 P1의 Z축값만 다릅니다.

5) 예제 5PMOV P0P0)

가) 작업순서→ 포인트 티칭



기준점:P0	Y방향:P2
X방향:P1	Z방향:P3



나) DATA 설정

<PALLET-DATA>
Pallet NO=00
1. XWM=01
2. YWM=01
3. ZWM=04

Update OK? (ENT/ESC)

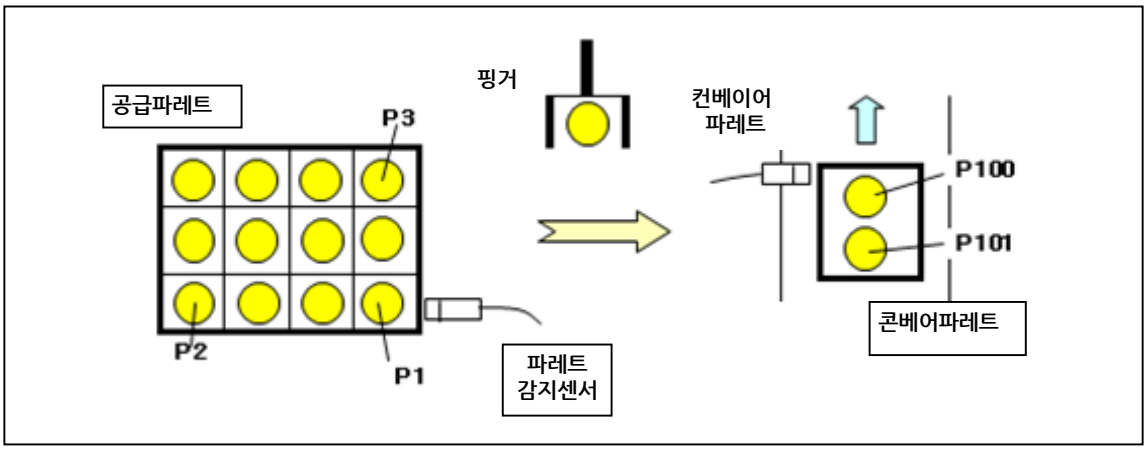
CAUTION

▶ Pallet 작업패턴 NO. 1 ~ 3의 내용은 N1-OM "PALLET(Palletizing)"항목을 참고하시기 바랍니다.

3.19.3 프로그램 사용 예제(1)

1) 작업내용

가) 로봇이 부품공급 Pallet의 부품을 집어, 콘베어 파레트에 부품을 공급합니다.



2) 로봇 사용자(USER) I/O

IN		OUT	
0	공급 Pallet 감지	0	핑거 클램프 시작
1	콘베어 Pallet 감지	1	핑거 언클램프 시작
2	핑거 클램프 감지	2	Pallet 작업완료
3	핑거 언클램프 감지	3	1 싸이클 완료
4		4	

3) Pallet 데이터 파라미터 설정

DATA

```
<PALLET-DATA>
   Pallet NO=01
1.XWM=04
2.YWM=03
3.ZWM=01

DATA   CNT   OUT   PATTERN
```

→

OUT

```
<PALLET-OUT>
   Pallet NO=01
1. Port= 02
2. Time=500
3. Type=04

DATA   CNT   OUT   PATTERN
```

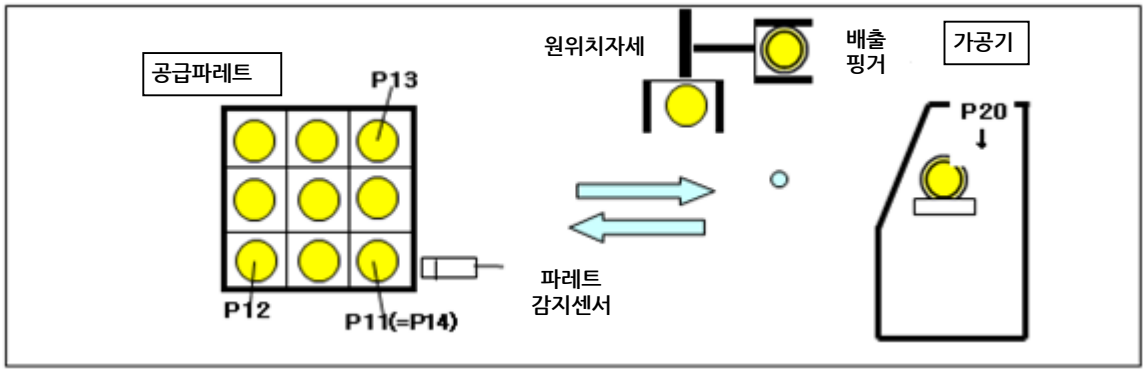
4) 프로그램 작성

```
MAIN
VEL 500 ..... 속도 50% 설정
PLUP 5 ..... Z축 풀업 5mm로 설정
POUT0=0 ..... 출력 ALL OFF
CALL HOFF ..... 핑거 언클램프
LABL A0
IN0=1 ..... Pallet가 도착할 때 까지 대기
CALL PALT ..... Pallet작업 서브루틴 호출
IN1=1 ..... 콘베어Pallet가 도착할 때 까지 대기
JMOV P100 ..... 콘베어 위치 P100로 이동
CALL HOFF ..... 핑거 언클램프
CALL PALT ..... Pallet 작업 서브루틴 호출
JMOV P101 ..... 콘베어 위치 P101로 이동
CALL HOFF ..... 핑거 언클램프
OUT3=1 100 ..... 1싸이클 작업완료
GOTO A0 ..... LABL A0로 점프
EOP

SUBR PALT ..... Pallet에 소재집는 작업
PMOV P1 P1
CALL HON
DLAY 100 ..... 핑거 언클램프
RET
SUBR HOFF
OUT0=0
OUT1=1
IN3=1 ..... 핑거 클램프
RET
SUBR HON
OUT1=0
OUT0=1
RET
```

3.19.4 프로그램 사용 예제(2)

1) 작업내용

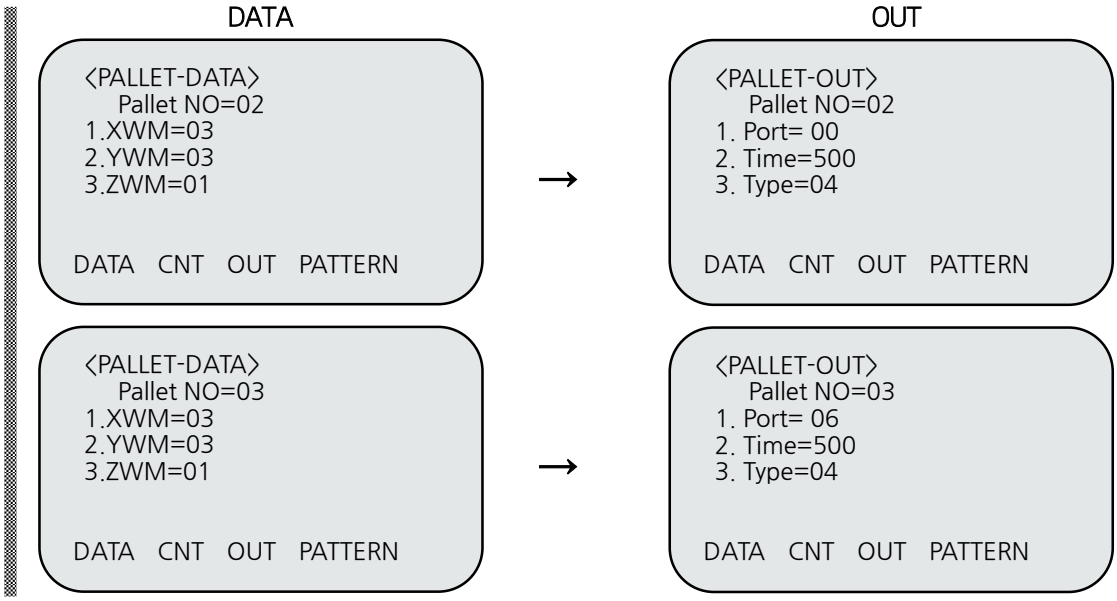
가) 로봇이 부품공급 Pallet의 부품을 집어 가공기에 공급 후, 가공완료된 부품을 다시 공급 Pallet에 적재하는 작업방식입니다.



2) 로봇 사용자(USER) I/O

IN		OUT	
0	공급 Pallet 작업시작	0	Pallet 소재배출 작업완료
1	가공기 작업시작	1	가공기 소재공급 완료(가공시작)
2	공급핑거 클램프 감지	2	공급핑거 클램프 시작
3	공급핑거 언클램프 감지	3	공급 핑거 언클램프 시작
4	배출핑거 클램프 감지	4	배출핑거 클램프 시작
5	배출핑거 언클램프 감지	5	배출핑거 언클램프 시작
6	핸드 원위치 감지	6	Pallet 소재공급 작업완료
7	핸드 180 도 회전 감지	7	핸드 원위치 시작
8		8	핸드 180 도 회전 시작

3) Pallet 데이터 설정



4) 프로그램 작성

```
MAIN
POS AP                  .....   POS변수 선언 AP
AP=<0,0,-100,0,0,0>
VEL 500                 .....   속도 50% 설정
PLUP 5                  .....   Z축 풀업 5mm 설정
JMOV P1                 .....   대기위치 이동
POUT0=0                 .....   출력신호 전체리셋
CALL H0                 .....   로봇핸드 원위치(0도)
CALL HOFF1              .....   로봇핸드 공급핑거 언클램프
CALL HOFF2              .....   로봇핸드 배출핑거 언클램프

LABL A0                 .....   레이블 A0설정
IN0=1                   .....   Pallet 작업시작 신호대기
CALL DEPALT             .....   부실행문 "DEPALT" 호출
JMOV P1                 .....   대기위치 이동
PLUP 0                  .....   Z축 풀업 해제
IN1=1                   .....   가공기 작업시작 신호대기
CALL H180               .....   부실행문 "H180" 호출
JMOV P20                .....   가공기측 포인트 P20→021로 이동
JMOV P21                .....
CALL HON2               .....   로봇핸드 배출핑거 클램프
JMOV P20                .....   가공기측 상단포인트 P20로 이동
CALL H0                 .....   로봇핸드 원위치(0도)
JMOV P21                .....   가공기측 하단포인트 P21로 이동
CALL HOFF1              .....   로봇핸드 공급핑거 언클램프
JMOV P20                .....   포인트 P20→P1로 이동
JMOV P1                 .....
```

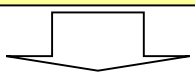
```
OUT1=1 100          ..............  가공시작
CALL H180           ..............  부실행문 "H180" 호출
CALL PALT           ..............  부실행문 "PALT" 호출
IMOV AP             ..............  상대이동 (Z축 100mm 상승)
CALL H0             ..............  로봇핸드 원위치
GOTO A0             ..............  레이블 A0FH 점프
EOP                 ..............  "Palletizing 작업" 부실행문

SUBR DEPALT
PLUP 5
PMOV P2 P11
CALL HON1
DLAY 100
RET

SUBR PALT           ..............  "Palletizing 작업" 부실행문
PLUP 5
PMOV P3 P11
CALL HOFF2
DLAY 100
RET

SUBR HOFF1          ..............  "로봇핸드 공급핑거 언클램프 작업" 부실행문
OUT2=0
OUT3=1
IN3=1
RET

SUBR HON1           ..............  "로봇핸드 공급핑거 클램프 작업" 부실행문
OUT3=0
OUT2=1
IN2=1
RET

SUBR HOFF2          ..............  "로봇핸드 배출핑거 언클램프 작업" 부실행문
OUT4=0
OUT5=1
IN5=1
RET
SUBR HON2           ..............  "로봇핸드 배출핑거 클램프 작업" 부실행문
OUT5=0
OUT4=1
IN4=1
RET
```

```
SUBR H0                 ..............    "로봇핸드 원위치(0도) 작업" 부실행문
OUT8=0
OUT7=1
IN6=1
RET

SUBR H180               ..............    "로봇핸드 회전(180도) 작업" 부실행문
OUT7=0
OUT8=1
IN7=1
RET
```

3.19.5 프로그램 사용 예제(3)

1) 작업내용

복수개의 Pallet 작업시, 전원이 OFF/ON 해도 작업중이던 Pallet넘버를 기억하여 계속 작업 가능합니다.



프로그램 작성시 정수변수(INT 변수이름)를 선언해 주고 이 변수명에 PIN9 값을 읽어들여 기억시키면 이전에 진행 중이던 팔레트 넘버가 기억됩니다.
프로그램내에서 이 변수에 설정된 팔레트 넘버를 비교하여 프로그램 작성합니다.

2) 예제) 2개의 파레트를 이용하는 작업

```
MAIN
INT PALNO ..... 정수변수선언 "PALNO"
PLUP 5
VEL 300
PALNO=PIN9 ..... PIN9의 값을 PALNO에 저장
IF PALNO==1 THEN
GOTO PAL1                    } PALNO의 값이 "1"이면 NO.1 Pallet 작업중
ELSE
IF PALNO==2 THEN
GOTO PAL2                    } PALNO의 값이 "2"이면 NO.2 Pallet 작업중
ENDIF
ENDIF

LABL LOOP
LABL PAL1
PMOV P1 P10
IF OUT1==0 THEN              } NO.1 Pallet 작업완료가 아니면
GOTO PAL1                    } NO.1 Pallet 계속작업
ELSE
GOTO PAL2
ENDIF

LABL PAL2
PMOV P2 P20
IF OUT2==0 THEN              } NO.2 Pallet 작업완료가 아니면
GOTO PAL2                    } NO.2 Pallet 계속작업
ELSE
GOTO LOOP
ENDIF
EOP
```

3) Pallet 데이터 설정

NO 01, NO02 데이터 입력

<PALLET-OUT>
Pallet NO=01
1. Port= 01
2. Time=200
3. Type=04

DATA CNT OUT PATTERN

NO 01 DATA을 입력합니다.

<PALLET-OUT>
Pallet NO=02
1. Port= 02
2. Time=200
3. Type=04

DATA CNT OUT PATTERN

NO 02 DATA을 입력합니다.

4) 전원이 꺼졌다 켜진 후에도 위 프로그램과 같이 정수변수 선언 후 이곳에 PIN9 값을 저장하면 이전 작업중이던 Pallet 번호를 기억할 수 있습니다.

3.20 PASS(팔레트 작업 중 지정한 작업물 통과 명령어)

기 능 팔레트 작업 중 지정한 번호의 작업물(적재 칸)을 통과합니다.

형 식 PASS <Pallet NO> <통과할 작업물 순번>

용 어 <Pallet NO> : 통과할 적재칸이 있는 Pallet 번호입니다.

<통과할 작업물 순번> : 통과할 적재칸의 번호입니다

설 명
1) 관련 명령어: PMOV
2) 1개의 Pallet당 PASS 사용개수: 20개
(20개 초과 사용하면 "E1217(Passing PLT over)"가 발생합니다.)
3) 한 개의 JOB에 사용할 수 있는 PASS 명령어를 사용하는 PALLET 수는 최대 5개까지 가능함
4) (프로그램 예제(2) 내용을 참조바랍니다.)

3.20.1 프로그램 사용 예제(1)

1) 작업내용

PASS 명령어 사용

1	2	3
4	5	6
7	8	9

표시부분만 작업합니다.

2) 프로그램 작성

```
MAIN
VEL 200
PASS 1 1
PASS 1 3
PASS 1 7
PASS 1 9
LABL A0
PLUP 10
PMOV P1 P10
DLAY 10
GOTO A0
EOP
```

Pallet NO.1의 1번, 3번, 7번, 9번 작업물 작업않고 통과

3.20.2 프로그램 사용 예제(2)

1) 한 개의 JOB에 PASS 명령어와 관련된 PALLET 수는 5개까지 사용할 수 있습니다.

```
MAIN
VEL 200
PLUP 10
LABL A0
PASS 1 1
PASS 1 3                    } Pallet NO.1의 1번, 3번 작업물 작업않고 통과
PMOV P1 P10
DLAY 10
PASS 2 2
PASS 2 4                    } Pallet NO.2의 2번, 4번 작업물 작업않고 통과
PMOV P2 P10
DLAY 10
PASS 3 5
PASS 3 6                    } Pallet NO.3의 5번, 6번 작업물 작업않고 통과
PMOV P3 P10
DLAY 10
PASS 4 1
PASS 4 4                    } Pallet NO.4의 1번, 4번 작업물 작업않고 통과
PMOV P4 P10
DLAY 10
PASS 5 2
PASS 5 3                    } Pallet NO.5의 2번, 3번 작업물 작업않고 통과
PMOV P5 P10
DLAY 10
GOTO A0
EOP
```

3.21 PCNT(팔레트 카운트 값 읽기/쓰기)

기 능 지정된 팔레트 번호의 현재 카운트 값을 읽어 오거나 변경 합니다.

형 식 <정수형 변수> = PCNT<팔레트 번호> : 현재 카운트를 정수형 변수에 적재
PCNT<팔레트 번호>= <정수형 변수> : 현재 카운트를 정수형 변호 값으로 변경

설 명 1) 시스템 변수는 자료형을 선언 하지 않고 사용 합니다.
2) PCNT변수는 해당 팔레트의 현재 카운트 값을 반환 합니다.(읽기용도 사용 시)
3) PCNT 변수를 이용하여 카운트값을 쓰기 용도로 사용할 때 다음 PMOV 명령부터 바로 적용 됩니다.

3.21.1 프로그램 사용 예제(1)

1) 현재 팔레트 카운트 값을 변경

```
MAIN
VEL 200
WHILE 1
JMOV P0
IF IN15==1 THEN ..... 입력 15번이 HIGH일 때 팔레트 0번 카운트 값을
PCNT0=PIN0&0H000F ..... 입력 0~3번 조합된 값으로 설정
ENDIF
PMOV P0 P1
ENDWL
EOP
```

3.21.1 프로그램 사용 예제(2)

1) 현재 팔레트 카운트 값을 읽기

```
MAIN
VEL 200
WHILE 1
JMOV P0
PMOV P0 P1
I0=PCNT0 ..... 현재 팔레트 카운트 값을 정수형 변수 적재
IF I0>=10 THEN
OUT15=1
ENDIF
ENDWL
EOP
```

3.22 WITH, ENDWT (동시처리 명령어)

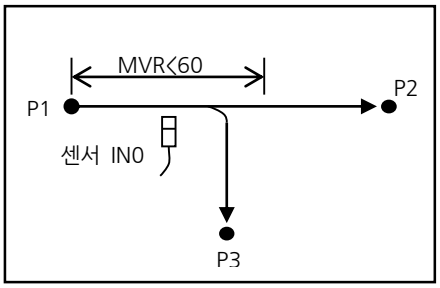
기 능 로봇 동작중 다음 수행문열을 동시 처리 합니다.

형 식
```
WITH
  ...
ENDWT
```

설 명
1) WITH 문 내의 최초 JMOV 명령어를 수행하면서 이후 명령어를 병렬처리합니다.
2) MVR 변수를 사용하여 로봇이동거리를 백단위로 나누어 원하는 지점에서 조건식 등을 처리할 수 있습니다.

3.22.1 프로그램 사용 예제(1)

1) 작업내용
P1에서 P2로 이동중 입력 IN0 신호가 ON되면 이동을 멈추고, P3로 이동합니다.
(단, 센서입력신호는 P1에서 P2까지의 이동거리의 60% 까지만 검사합니다.)



2) 로봇 사용자(USER) I/O

IN		OUT	
0	센서 감지	0	P2 도착완료
1		1	P3 도착완료

3) 프로그램 작성

```
MAIN
VEL 500
JMOV P1
MVR=0 ..... MVR 명령어를 초기화
WITH ..... 
JMOV P2 ..... 포인트 P2로 이동(JMOV만 가능함)
WHILE MVR<60 ..... 이동구간의 60%미만 동안에
IF IN0==1 THEN ..... IN0=1이면 "LABL BB"로 점프
GOTO BB
ENDIF
ENDWL
OUT0=1
```

ENDWT		IF문에 의해 BB로 이동하지 않을 경우 종료 지점
LABL BB		
STOP	.....	로봇이동 동작 정지
ENDWT	.....	WITH문 종료 (반드시 삽입)
JMOV P3		
OUT1=1		
EOP		

CAUTION

▶ WITH문 실행 중 조건식 만족에 의한 점프 명령어로 WITH ~ ENDWT 블록을 빠져나가는 경우, **점프한 최초 스텝에 반드시 ENDWT 명령어를 삽입**해야 합니다.

3.23 OUT, BOUT, POUT, WOUT, DOUT(외부 출력 명령어)

기 능　　지정된 비트단위 또는 포트단위로 지정된 값을 출력합니다.

형 식　　OUT<비트출력포트번호>=<0 또는 1> [펄스유효시간] [→ 또는 ~]
OUT<정수형변수>=<0 또는 1>
BOUT<출력포트번호>=<데이터>
POUT<출력포트번호>=<데이터>
DOUT<출력포트 번호>=<데이터>
<변수>=POUT<출력포트번호>

용 어　　<비트출력포트번호> : 비트출력 포트번호를 설정합니다. (0 ≤ 비트출력번호 ≤ 79)
사용자 출력(User Output) : OUT0 ~ OUT15
확장1 출력(Option Output1) : OUT16 ~ OUT47 → 확장 I/O Card 1장 장착시 해당합니다.
확장2 출력(Option Output2) : OUT48 ~ OUT79 → 확장 I/O Card 2장 장착시 해당합니다.

[펄스유효시간] : 해당 비트출력 포트에 0/1값을 출력(펄스출력)하는 유효시간입니다.

[→ 또는 ~] :
펄스유효시간을 주기로 갖는 주기파형 출력합니다.
펄스출력, 주기파형 출력은 최대 32점(User, Option1, Option2 중 1개)까지 사용가능.
Port를 중복 사용할 수는 없음. (예) User OUT 16점 + Option OUT 16점)
"펄스 출력"과 "주기 파형출력"은 스텝 진행과 동시에 진행 됩니다.
즉, 다음 스텝이 이동명령(MOVE)인 경우 이동하면서 출력비트를 ON 또는 OFF 시킵니다.
"펄스출력"과 "주기파형출력"은 파라미터에서 사용할 Port를 정해야 합니다.
설정방법은 "파라미터 설명서를 참조하십시오.

<출력포트번호> : 출력포트 번호를 설정합니다. (0 ≤ 출력포트번호 ≤ 4)
BOUT 인 경우 : 8bit 묶음(1byte)를 의미합니다.
0일 경우 : OUT0 ~ OUT7 를 지정합니다.　5일 경우 : OUT40 ~ OUT47 을 지정합니다.
1일 경우 : OUT8 ~ OUT15 을 지정합니다.　6일 경우 : OUT48 ~ OUT57 를 지정합니다.
2일 경우 : OUT16 ~ OUT23 을 지정합니다.　7일 경우 : OUT58 ~ OUT63 를 지정합니다.
3일 경우 : OUT24 ~ OUT31 을 지정합니다.　8일 경우 : OUT64 ~ OUT71 를 지정합니다.
4일 경우 : OUT32 ~ OUT39 을 지정합니다.　9일 경우 : OUT72 ~ OUT79 를 지정합니다.

POUT, WOUT 인 경우 : 16bit 묶음(1word)를 의미합니다.
0일 경우 : OUT0 ~ OUT15 를 지정합니다.　3일 경우 : OUT48 ~ OUT63 을 지정합니다.
1일 경우 : OUT16 ~ OUT31 을 지정합니다.　4일 경우 : OUT64 ~ OUT79 를 지정합니다.
2일 경우 : OUT32 ~ OUT47 을 지정합니다.

DOUT 인 경우 : 32bit 묶음(2word)를 의미합니다.
0일 경우 : OUT0 ~ OUT31 를 지정합니다.　3일 경우 : OUT32 ~ OUT63 을 지정합니다.
<데이터> : 해당 출력포트에 출력할 값 (2진수, 16진수가능)을 의미합니다.
예제1) POUT0=10 일 경우 (10(10진수) => 0000 0000 0000 1010 (2진수))
출력포트 1,3번을 ON(1) 시키며, 다른 출력 포트는 OFF (0) 시킵니다.
예제2) POUT0=0H0F0F 일 경우 (0H0F0F (16진수) => 0000 1111 0000 1111 (2진수))
출력포트 0,1,2,3번과 출력포트 8,9,10,11번을 ON(1) 시키며 다른 출력 포트는

OFF (0) 시킵니다.

설 명 1) OUT<비트출력포트번호>=<0 또는 1> [<펄스유효시간>] [→]

가) 지정된 펄스 유효시간만큼 해당비트출력 포트에 0 또는 1값을 출력합니다.

나) 펄스 유효시간이 없으면 계속 유효하며 펄스유효시간이 지나면 이전상태로 복귀합니다.

다) 펄스 유효시간의 단위는 10ms 입니다.

2) OUT<정수형 변수>=<0 또는 1>

"비트출력포트번호"에 정수형 변수를 사용할 수 있습니다.

3) BOUT<정수형 변수>=<데이터>

데이터 값을 해당 출력포트에 출력(8bit)합니다.

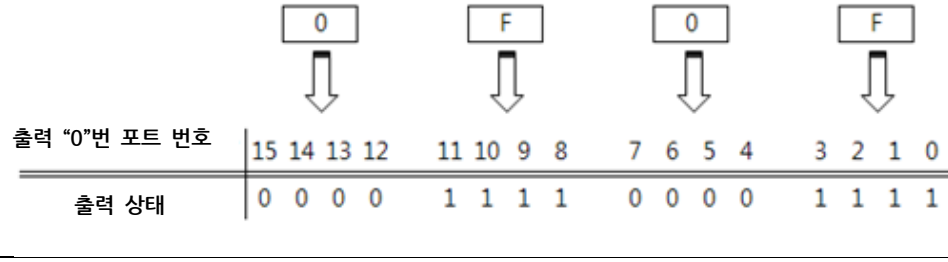
4) POUT<출력포트번호>=<데이터>, WOUT<출력포트번호>=<데이터>

데이터 값을 해당 출력포트에 출력(1word(16bit))합니다.

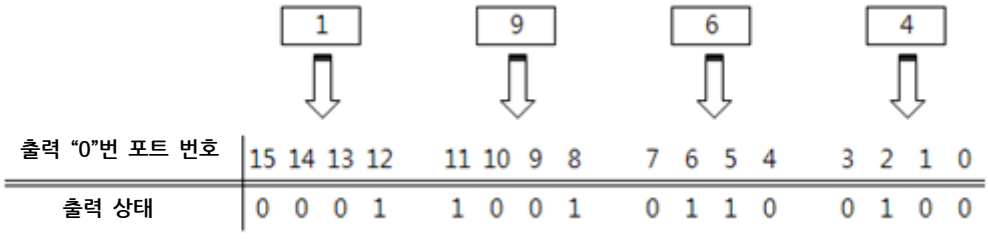
5) DOUT<출력포트번호>=<데이터>

데이터 값을 해당 출력포트에 출력(2word(32bit))합니다.

POUT0=0H0F0F



POUT0=0H1964



CAUTION

- OUT, POUT과 "비트출력포트번호"," 출력포트번호"사이에 공란(Blank)이 들어가지 않도록 주의하시기 바랍니다. 예) OUT_15=1,POUT_0=0H00FF (Syntax Error 발생)
- 출력포트 번호의 데이터값은 우측에서 좌측으로 0,1,2 ~ 15 의 순으로 증가 합니다.
- **내부 접점은 펄스 출력 제어가 되지 않습니다.**
- 확장 I/O 보드는 옵션 구매품 입니다.

3.23.1 프로그램 사용 예제

1) 펄스출력, 데이터출력

```
MAIN
VEL 100
JMOV P10
OUT11=1 100 ..... 출력 포트 11번을 1s 동안 ON(1) 시킴
LMOV P100
POUT0=0H1004 ..... 출력포트 0~15번에 0H1004(16진수)값으로 출력
EOP
```

2) 비트출력포트번호를 변수로 사용

```
MAIN
INT A ..... 정수형 변수 A를 선언
FOR A=0 TO 10 ..... 0 ~ 10까지 반복 수행을 하며
OUT(A)=0 ..... OUT0,OUT1 ~ OUT10 의 출력을 모두 OFF(0)
NEXT
EOP
```

3.24 IN, BIN, PIN, WIN, DIN(외부 입력 명령어)

기 능 지정된 비트단위 또는 포트단위의 입력에서 값을 받습니다.

형 식
IN<입력비트번호>=<0 또는 1>
BIN<입력비트번호>=<0 또는 1>
PIN<입력포트번호>=<정수값 또는 변수>
DIN<입력비트번호>=<0 또는 1>
<변수>=IN<입력비트번호>
<변수>=BIN<입력비트번호>
<변수>=PIN<입력비트번호>
<변수>=DIN<입력포트번호>

용 어
<비트입력포트번호> : 비트입력 포트번호를 설정합니다.(0 ≤ 비트입력번호 ≤ 79)
사용자 입력(User Input) : IN0 ~ IN15
확장1 입력(Option Input1) : IN16 ~ IN47 → 확장 I/O Card 1장 장착시 해당합니다.
확장2 입력(Option Input2) : IN48 ~ IN79 → 확장 I/O Card 2장 장착시 해당합니다.

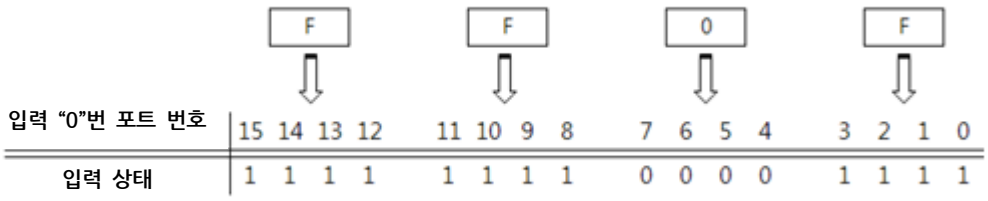
<입력포트번호> : 입력포트 번호를 설정합니다.
BIN 인경우 : 8bit 묶음(1byte)을 의미합니다.
0일 경우 : IN0 ~ IN7 를 지정합니다.
1일 경우 : IN8 ~ IN15 을 지정합니다.
2일 경우 : IN16 ~ IN23 을 지정합니다.
3일 경우 : IN24 ~ IN31 을 지정합니다.
4일 경우 : IN32 ~ IN39 를 지정합니다.
5일 경우 : IN40 ~ IN47 를 지정합니다.
6일 경우 : IN48 ~ IN57 를 지정합니다.
7일 경우 : IN58 ~ IN63 를 지정합니다.
8일 경우 : IN64 ~ IN71 를 지정합니다.
9일 경우 : IN72 ~ IN79 를 지정합니다.

PIN, WIN 인 경우 : 16bit 묶음(1word)을 의미합니다.
0일 경우 : IN0 ~ IN15 를 지정합니다.
1일 경우 : IN 16 ~ IN31 을 지정합니다.
2일 경우 : IN 32 ~ IN 47 을 지정합니다.
3일 경우 : IN 48 ~ IN 63 을 지정합니다.
4일 경우 : IN 64 ~ IN 79 를 지정합니다.

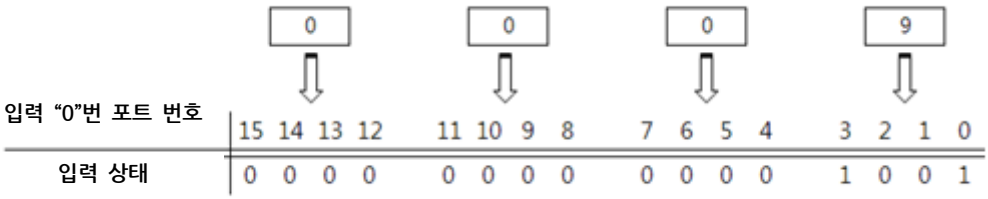
DIN 인 경우 : 32bit 묶음(2word)을 의미합니다.
0일 경우 : IN0 ~ IN31 를 지정합니다.
1일 경우 : IN 32 ~ IN63 을 지정합니다.

설 명
1) IN<비트입력포트번호>
지정된 비트입력포트의 ON/OFF상태(1또는 0)를 IN<비트입력포트번호>변수에 저장합니다.
2) BIN<입력포트번호>, PIN<입력포트번호>, WIN<입력포트번호>, DIN<입력포트번호>
지정된 입력포트의 값을 읽어서 BIN, PIN, WIN, DIN<입력포트번호>변수에 저장합니다.
3) IN<비트입력포트번호>=< 0/1 >
지정된 비트입력포트가 ON/OFF상태(1또는 0)가 될 때까지 대기합니다.
4) BIN<입력포트번호>=<8비트값>,PIN, WIN<입력포트번호>=<16비트값>,
DIN<입력포트번호>=<32비트값>
지정된 입력포트의 값이 지정된 값이 될 때까지 대기합니다.

PIN0=0HFF0F

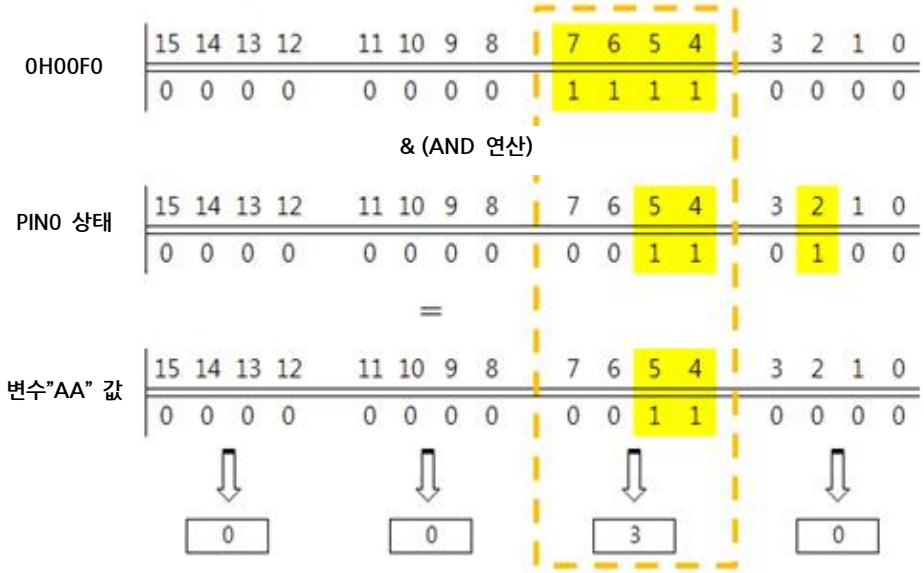


PIN0=0H0009



입력 "0"번 포트의 입력중 "IN4~IN7" 사이의 신호만 체크하려면?

AA=0H00F0&PIN0



CAUTION

- ▶ IN,PIN과 "비트입력포트번호"," 입력포트번호"사이에 공란(Blank)이 들어가지 않도록 주의하시기 바랍니다.
  예) AA=IN_15, AA=PIN_0 (Syntax Error 발생)
- ▶ 입력포트 번호의 데이터값은 우측에서 좌측으로 0,1,2 ~ 15 의 순으로 증가 합니다.
- ▶ 확장 I/O 보드는 옵션 구매품 입니다.

3.24.1 프로그램 사용 예제

1) 입력 대기, Port입력을 정수변수처리 이동선택

```
MAIN
INT D1                  .....   정수형 변수 D1 선언
VEL 100
WHILE 1
JMOV P0
IN1=1                   .....   비트 입력 포트 1번이 ON(1) 될때까지 대기
IF IN0==1 THEN          .....   비트 입력 포트 0번이 ON(1) 일경우
D1=PIN0                 .....   입력포트 0~15 까지의 상태를 D1에 저장
ELSE
PIN0=0HFFFF             .....   입력포트 0~15까지 모두 ON(1) 될때까지 대기
ENDIF
IF D1==0HFF0F THEN      .....   D1의 상태 (즉 PIN0의 상태)를 0HFF0F와 비교
JMOV P1
ENDIF
ENDWL
EOP
```

2) BCD 유닛을 이용한 MODEL 비교(BCD 입력은 IN0 ~ IN7까지 입력을 사용 한 예)

```
MAIN
INT M1,M2,MD            .....   정수형 변수 M1,M2,MD를 선언
WHILE 1
M1=PIN0 & 0H000F        .....   M1에 PIN0와 0H000F를 비교 하여 저장
M2=PIN0&0H00F0          .....   M2에 PIN0와 0H00F0를 비교 하여 저장
M2=(M2>>4)*10           .....   M2의 값을 1의 자리로 쉬프트 후 10을 곱함
MD=M1+M2                .....   연산된 M1과 M2의 값을 더함
IF MD==1 THEN
JMOV P0
ELSE
IF MD==11 THEN
JMOV P1
ENDIF
ENDIF
ENDWL
EOP
```

CAUTION

- ▶ 포트 입력의 경우 16진수 연산을 수행하기 때문에 BCD 유닛등으로 10을 입력 시, 제어기에서는 16으로 인식 합니다.
- ▶ 이 경우 상기 예제와 같이 쉬프트 연산 후, 10진수 변환을 거쳐야 정확한 계산이 수행됩니다.

3.25 CIN,CBIN,CWIN,CDIN,CFIN(필드 버스용 입력 명령어)

기 능 필드 버스 카드에서 지정된 비트단위 또는 포트 단위의 입력 값을 받습니다.

형 식 1. Bit 영역 명령어
<정수형 변수>=CIN<입력비트번호>
<정수형 변수>=CBIN<입력포트번호>
2. Word 영역 명령어
<정수형 변수>=CWIN<입력포트번호>
<정수형 변수>=CDIN<입력포트번호>
<실수형 변수>=CFIN<입력포트번호>

용 어 1. Bit 영역: 1Bit 단위로 읽고 쓰기를 할수 있는 영역 입니다.
<입력비트번호>: 입력 비트번호를 설정합니다(0≤입력비트번호≤127)
1) User 영역 : CIN0 ~ CIN127
<입력포트번호>: 입력포트 번호를 설정합니다.(0≤입력포트번호≤15)
1) User 영역 : CBIN0~CBIN15
2) CBIN<포트번호>: Bit 영역 값을 Byte(8bit)단위(정수)로 받는다.

- 포트번호 0: CIN0 ~ CIN7 - 포트번호 1: CIN8 ~ CIN15
- 포트번호 2: CIN16 ~ CIN23 - 포트번호 3: CIN24 ~ CIN31
- 포트번호 4: CIN32 ~ CIN39 - 포트번호 5: CIN40 ~ CIN47
- 포트번호 6: CIN48 ~ CIN55 - 포트번호 7: CIN56 ~ CIN63
- 포트번호 8: CIN64 ~ CIN71 - 포트번호 9: CIN72 ~ CIN79
- 포트번호 10: CIN80 ~ CIN87 - 포트번호 11: CIN88 ~ CIN95
- 포트번호 12: CIN96 ~ CIN103 - 포트번호 13: CIN104 ~ CIN111
- 포트번호 14: CIN112 ~ CIN119 - 포트번호 15: CIN120 ~ CIN127

2. Word 영역: Word(16Bit)단위로 읽고 쓰기를 할수 있는 영역 입니다.
<입력포트번호>: 입력포트 번호를 설정합니다.(0≤입력포트번호≤15).
1) User 영역: CBIN0 ~ CBIN15
2) CDIN<포트번호>: Byte 영역 값을 Double Word(32Bit)단위(정수)로 받는다.

- 포트번호 0: CBIN0 ~ CBIN3
- 포트번호 1: CBIN4 ~ CBIN7
- 포트번호 2: CBIN8 ~ CBIN11
- 포트번호 3: CBIN12 ~ CBIN15

3) CFIN<포트번호>: Byte 영역 값을 Float(32Bit)단위(실수)로 받는다.

- 포트번호 0: CWIN0 ~ CWIN1 - 포트번호 1: CWIN2 ~ CWIN3
- 포트번호 2: CWIN4 ~ CWIN5 - 포트번호 3: CWIN6 ~ CWIN7
- 포트번호 4: CWIN8 ~ CWIN9 - 포트번호 5: CWIN10 ~ CWIN11
- 포트번호 6: CWIN12 ~ CWIN13 - 포트번호 7: CWIN14 ~ CWIN15

설 명

1) CIN<입력비트번호>
지정된 비트 입력포트의 ON/OFF상태(1또는0)를 IN<비트입력포트번호>변수에 저장 합니다.

2) CIN<정수형 변수>
"입력비트번호"에 정수형 변수를 사용할수 있습니다

3) CBIN<입력포트번호>
지정된 입력포트의 값(8Bit)을 읽어서 CBIN<입력포트번호>변수에 저장합니다.

4) CWIN<입력포트번호>
지정된 입력포트의 값(Word)을 읽어서 CWIN<입력포트번호>변수에 저장합니다.
(최소입력 값 : -32768, 최대입력 값 : 32767)

5) CDIN<입력포트번호>
지정된 입력포트의 값(Double Word)을 읽어서 CDIN<입력포트번호>변수에 저장합니다.

6) CFIN<입력포트번호>
지정된 입력포트의 값(Float)을 읽어서 CFIN<입력포트번호>변수에 저장합니다.
(최소입력 값 : -999999.9 최대입력 값 :999999.9)

7) 필드버스입력명령어 메모리의 공유영역

	CIN(단위 : Bit)			CBIN(단위 : Byte)	CDIN(단위 : Double Word)
입력포트	0	~	7	0	0
	8	~	15	1	
	16	~	23	2	
	24	~	31	3	
	32	~	39	4	1
	40	~	47	5	
	48	~	55	6	
	56	~	63	7	
	64	~	71	8	2
	72	~	79	9	
	80	~	87	10	
	88	~	95	11	
	96	~	103	12	3
	104	~	111	13	
	112	~	119	14	
	120	~	127	15	

	CWIN(단위 : Word)			CFIN(단위 : Float)
입력포트	0	~	1	0
	2	~	3	1
	4	~	5	2
	6	~	7	3
	8	~	9	4
	10	~	11	5
	12	~	13	6
	14	~	15	7

3.25.1 프로그램 사용 예제

```
MAIN
INT D1                  .....   정수형 변수 D1 선언
VEL 100
WHILE 1
JMOV P0
IF CIN50==1 THEN        .....   비트 입력 포트 51번이 ON(1) 일경우
D1=CBIN5                .....   입력포트 40~47 까지의 상태를 D1에 저장
ELSE
D1=CBIN6                .....   입력포트 48~45까지의 상태를 D1에 저장
ENDIF
IF D1==0H0F THEN        .....   D1의 상태를 0H0F와 비교
JMOV P1
ELSE
JMOV P2
ENDIF
ENDWL
EOP
```

CAUTION

- ▶ CIN,CBIN,CWIN,CDIN,CFIN과 "비트입력포트번호"," 입력포트번호"사이에 공란(Blank)이 들어가지 않도록 주의하시기 바랍니다.
  예) AA=CIN_15, AA=CBIN_0 (Syntax Error 발생)
- ▶ 입력포트 번호의 데이터값은 우측에서 좌측으로 0,1,2 ~ 15 의 순으로 증가 합니다.
- ▶ 필드버스용 명령어는 PARA→PUB→HWCONF→COMM→FieldBus→MAP 설정값이 "USER_DEF"일 경우만 적용 됩니다.
- ▶ CFIN 사용 시 소수점 3째 자리까지 입력받을 수 있습니다.
- ▶ CWIN(정수), CFIN(실수)은 표현 형태가 다르기 때문에 주의하시기 바랍니다.
  예) 상위에서 CFIN0에 1.1입력시 CWIN0, CWIN1에는 -13107, 16283이 입력이 됩니다.

3.26 COUT,CBOUT,CWOUT,CDOUT,CFOUT(필드 버스용 출력 명령어)

기 능 필드 버스 카드로 비트단위 또는 포트단위로 지정된 값을 출력합니다.

형 식 1. Bit 영역 명령어
COUT<출력비트번호>=<0또는1>
CBOUT<출력포트번호>=<데이터>
2. Word 영역 명령어
CWOUT<출력포트번호>=<데이터>
CDOUT<출력포트번호>=<데이터>
CFOUT<출력포트번호>=<데이터>

용 어 1. Bit 영역: 1Bit 단위로 읽고 쓰기를 할수 있는 영역 입니다.
<출력비트번호>: 출력 비트번호를 설정합니다(0≤출력비트번호≤127)
1) User 영역 : COUT0 ~ COUT127
<출력포트번호>: 출력포트 번호를 설정합니다.(0≤출력포트번호≤15)
1) User 영역 : CBOUT0~CBOUT15
2) CBOUT<포트번호>: Bit 영역 값을 Byte(8bit)단위(정수)로 출력한다.
- 포트번호 0: COUT0 ~ COUT7 - 포트번호 1: COUT8 ~ COUT15
- 포트번호 2: COUT16 ~ COUT23 - 포트번호 3: COUT24 ~ COUT31
- 포트번호 4: COUT32 ~ COUT39 - 포트번호 5: COUT40 ~ COUT47
- 포트번호 6: COUT48 ~ COUT55 - 포트번호 7: COUT56 ~ COUT63
- 포트번호 8: COUT64 ~ COUT71 - 포트번호 9: COUT72 ~ COUT79
- 포트번호 10: COUT80 ~ COUT87 - 포트번호 11: COUT88 ~ COUT95
- 포트번호 12: COUT96 ~ COUT103 - 포트번호 13: COUT104 ~ COUT111
- 포트번호 14: COUT112 ~ COUT119 - 포트번호 15: COUT120 ~ COUT127

2. Word 영역: 1Word(16Bit)단위로 읽고 쓰기를 할수 있는 영역 입니다.
<출력포트번호>: 출력포트 번호를 설정합니다.(0≤출력포트번호≤15).
1) User 영역: CBOUT0 ~ CBOUT15
2) CDOUT<포트번호>: Byte 영역 값을 Double Word(32Bit)단위(정수)로 출력한다.
- 포트번호 0: CBOUT0 ~ CBOUT7
- 포트번호 1: CBOUT8 ~ CBOUT15
- 포트번호 2: CBOUT16 ~ CBOUT23
- 포트번호 3: CBOUT24 ~ CWOUT31

3) CFOUT<포트번호>: Worad 영역 값을 Float(32Bit)단위(실수)로 출력한다.
- 포트번호 0: CWOUT0 ~ CWOUT1 - 포트번호 1: CWOUT2 ~ CWOUT3
- 포트번호 2: CWOUT3 ~ CWOUT4 - 포트번호 3: CWOUT5 ~ CWOUT6
- 포트번호 4: CWOUT7 ~ CWOUT8 - 포트번호 5: CWOUT9 ~ CWOUT10

포트번호 6: CWOUT11 ~ CWOUT12 - 포트번호 7: CWOUT13 ~ CWOUT14

설 명

1) COUT<출력비트번호>=<0또는1>
ON/OFF상태(1또는0)를 해당 출력비트 번호에 출력합니다.
2) COUT<정수형번수>=<0또는1>
"출력비트번호"에 정수형 변수를 사용할수 있습니다.
3) CBOUT<출력포트번호>=<데이터>
데이터(8Bit) 값을 해당 출력포트에 출력합니다.
4) CWOUT<출력포트번호>=<데이터>
데이터(1Word) 값을 해당 출력포트에 출력합니다.
(최소입력 값 : -32768 최대입력 값 : 32767)
5) CDOUT<출력포트번호>=<데이터>
데이터(2Word) 값을 해당 출력포트에 출력합니다.
6) CFOUT<출력포트번호>=<데이터>
데이터(2Word) 값을 해당 출력포트에 출력합니다.
(최소입력 값 : -999999.9 최대입력 값 :999999.9)

7) 필드버스출력명령어 메모리의 공유영역

	COUT (단위 : Bit)			CBOUT (단위 : Byte)	CDOUT (단위 : Double Word)
출력 포트	0	~	7	0	0
	8	~	15	1	
	16	~	23	2	
	24	~	31	3	
	32	~	39	4	1
	40	~	47	5	
	48	~	55	6	
	56	~	63	7	
	64	~	71	8	2
	72	~	79	9	
	80	~	87	10	
	88	~	95	11	
	96	~	103	12	3
	104	~	111	13	
	112	~	119	14	
	120	~	127	15	

	CWOUT (단위 : Word)			CFOUT (단위 : Float)
출력 포트	0	~	1	0
	2	~	3	1
	4	~	5	2
	6	~	7	3
	8	~	9	4
	10	~	11	5
	12	~	13	6
	14	~	15	7

3.26.1 프로그램 사용 예제

1) Bit 영역 출력

```
MAIN
INT A                   .....   정수형 변수 A를 선언
FOR A=32 TO 42          .....   32 ~ 42까지 반복 수행을 하며
COUT(A)=0               .....   OUT32,OUT33 ~ OUT42 의 출력을 모두 OFF(0)
NEXT
EOP
```

2) Word 영역 출력

```
MAIN
POS CURR
WHILE 1
CURR=HERE               .....   로봇 현재 좌표를 CURR에 저장
CFOUT0=CURR.1           .....   CURR 1번축 값을 필드버스
                                (CWOUT0,CWOUT1)로 출력
CFOUT1=CURR.2           .....   CURR 2번축 값을 필드버스
                                (CWOUT2,CWOUT3)로 출력
ENDWL
EOP
```

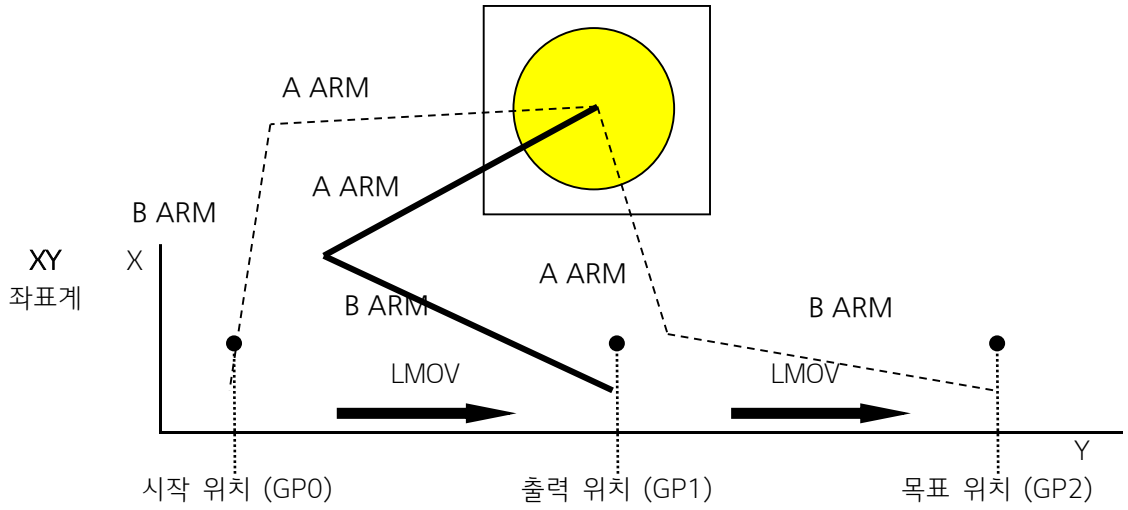
CAUTION

- ▶ COUT,CBOUT,CWOUT,CDOUT,CFOUT과 "출력비트번호"," 출력포트번호"사이에 공란(Blank)이 들어가지 않도록 주의하시기 바랍니다.
  예) COUT_15=1, CBOUT_5=AA (Syntax Error 발생)
- ▶ 출력포트 번호의 데이터값은 우측에서 좌측으로 0,1,2 ~ 15 의 순으로 증가합니다.
- ▶ 필드버스용 명령어는 PARA→PUB→HWCONF→COMM→FieldBus→MAP 설정값이 "USER_DEF"일 경우만 적용됩니다.
- ▶ CFIN 사용 시 소수점 3째 자리까지 입력받을 수 있습니다.
- ▶ CWOUT(정수), CFOUT(실수)은 표현 형태가 다르기 때문에 주의하시기 바랍니다.
  예) 상위에서 CFOUT0에 1.1입력시 CWOUT0, CWOUT에는 -13107, 16283이 입력이 됩니다.

3.27 Moving Out 관련 명령어

기 능 **이동 중 접점 출력**을 합니다.

설 명
1) **이동 중 접점 출력이란 로봇이 구동 중 특정 위치를 지나갈 때 I/O 접점을 통해 출력을 하는 기능**입니다.
2) 사용자는 본 기능 구현을 위해 시작 위치 및 목표 위치를 티칭한 후 이동 궤적 위에 출력 위치를 티칭해야 됩니다.
3) MOINIT, MORNG, MOCLR 3가지 명령어를 통해 출력 기능 설정 및 해제, 출력 위치 및 시간, 출력 접점을 설정할 수 있습니다.
4) 기능 구동 예
   가) 로봇이 시작 위치(GP0)에서 출력 위치(GP2)까지 LMOV 이동 중에 출력 위치(GP1)에서 설정한 I/O 접점으로 출력을 합니다.



CAUTION

- 로봇 티칭 시 출력위치 좌표는 반드시 로봇 이동 궤적 위로 설정해야 합니다. 로봇이 출력위치 좌표를 지나지 않는 경우 출력되지 않습니다.
- 로봇 구동 전 반드시 운용 펌웨어 버전을 확인하시기 바랍니다.
- 펌웨어 03.03.04-C4 RO 170726 또는 03.03.04-CE RO 170726버전부터 사용할 수 있습니다.

3.27.1 MOINIT(Moving Outport Initialize)

기 능 채널 및 포트 정보 설정

형 식 <변수>=MOINIT(<인덱스>, <채널>, <출력포트>, <출력시간>)

용 어 <변수>
정상동작 시 1을 반환하며 에러 발생 시 알람 및 0을 반환합니다.

<인덱스>
이동 중 접점 출력 기능의 파라미터 구분 번호입니다.(범위 : 0 ~ 20)

<채널>
수행 할 로봇 채널입니다.(범위 :1 ~ 3)

<출력포트>
이동 중 출력을 할 접점을 설정합니다.(범위 : 0~79)

<출력시간>
출력 유지 시간을 설정합니다.(범위 : 0~1000, 단위 : 10ms)

설 명
1) 이동 중 접점 출력기능을 사용하기 위한 초기 설정 명령어 입니다.
2) Para_Index는 최대 20개까지 생성할 수 있으며 채널 별 중복 사용이 불가 합니다.
3) Time을 0으로 설정하는 경우 출력위치 ± Offset 범위 안에서 HIGH 출력이 유지됩니다.
4) Time을 0이외로 설정하는 경우 출력위치 ± Offset 범위 안에서 HIGH 트리거 된 시점부터 Time설정 값 *10msec동안 출력이 HIGH를 유지합니다.
5) 펄스출력, 데이터출력

3.27.1.1프로그램 사용 예제

```
MAIN
I0=MOCLR(0)
I1=MOINIT(1,1,0,100)        .....   인덱스 1번 적용, 로봇 1번 채널, 출력접점 0번
    :                               1초간 출력유지
I0=MOCLR(1)
EOP
```

CAUTION

- MOINIT, MORNG, MOCLR 는 Moving Out 기능을 위한 명령어로(03.03.04-C4 RO 170726
- 또는 03.03.04-CE RO 170726) 이상의 펌웨어 버전에서 동작됩니다.
- 같은 번호의 Index 사용을 하는 경우 MOINIT 이전에 MOCLR를 수행하기를 권장합니다.
- Para Index를 채널 별 중복 사용하는 경우 또는 인자에 INT 이 외 다른 데이터 형 값을 입력하는 경우 E1227 INVALID PARAMETER 알람이 발생합니다.
- 표시된 매개변수의 범의를 벗어나는 값을 입력한 경우 E1202 RANGE OVER 알람이 발생합니다.
- 다수의 Index 사용 시, 개별 Out_Port를 설정을 권장하며 동일한 Out_Port 사용시 출력 시간 입력에 주의 바랍니다.

3.27.2 MORNG(Moving Outport Range)

기 능 출력 할 위치 좌표 및 옵셋 범위 설정

형 식 <변수>=MORNG(<인덱스>, <좌표계>, <출력 범위>, <옵셋 범위>)

용 어 <변수>
정상동작 시 1을 반환하며 에러 발생 시 알람 및 0을 반환합니다.

<인덱스>
이동 중 접점 출력 기능의 파라미터 구분 번호입니다.(범위 : 0 ~ 20)

<좌표계>
이동 중 접점을 출력할 로봇 좌표계를 설정합니다.(범위 : 1 ~ 6)
1 : 엔코드 피드백 위치값을 ANGLE 좌표계 변환 4 : 축별 위치 지령 값을 ANGLE 좌표계 변환
2 : 엔코드 피드백 위치값을 BASE 좌표계 변환 5: 축별 위치 지령 값을 BASE 좌표계 변환
3 : 엔코드 피드백 위치값을 TOOL 좌표계 6 : 축별 위치 지령 값을 TOOL 좌표계 변환

<출력범위>
이동 중 접점을 출력할 위치 좌표를 의미하여 POSITION 변수로 입력합니다.

<출력시간>
출력 좌표의 ± 옵셋 값을 의미하며 POSITION 변수로 입력합니다.(ANGLE 좌표계에서 입력)

설 명 1) 출력 할 위치 및 범위를 설정하는 명령어입니다.
2) 좌표계를 통해 비교할 로봇 좌표계를 선택할 수 있습니다.
3) POSITION변수에 대한 설명은 [프로그램 설명서 3.45 POSITION 변수]를 바랍니다.

CAUTION

- ▶ MOINIT, MORNG, MOCLR 는 Moving Out 기능을 위한 명령어로 (03.03.04-C4 RO 170726
- ▶ 또는 03.03.04-CE RO 170726) 이상의 펌웨어 버전에서 동작됩니다.
- ▶ MOINIT에서 초기화 되지 않는 Index 사용 시 E1246 MOINIT Initial Err 알람이 발생합니다.
- ▶ Output_Range는 POS변수 중 XP 변수 사용이 가능하지만 Offset_Range의 경우 XP변수 사용 시 E1227 INVALID PARAMETER 알람이 발생합니다.
- ▶ 표시된 매개변수의 범의를 벗어나는 값을 입력한 경우 E1202 RANGE OVER 알람이 발생합니다.
- ▶ POS변수 사용 시 "P", "GP", "I", "F"등은 변수 명으로 사용하실 수 없습니다.
- ▶ 로봇 티칭 시 기준좌표는 반드시 구동경로 위로 설정해야 합니다. 로봇이 기준좌표를 지나지 않는 경우 출력되지 않습니다.

3.27.3 MOCLR(Moving Outport Clear)

기 능 이동 중 접점 출력기능 해제

형 식 <변수>=MORNG(<인덱스>)

용 어 <변수>
정상동작 시 1을 반환하며 에러 발생 시 알람 및 0을 반환합니다.

<인덱스>
이동 중 접점 출력 기능해제를 위한 파라미터 구분 번호입니다.(범위 : 0 ~ 20)

설 명 1) 0 입력 시 1~20번 인덱스 설정 값들이 해제됩니다.
2) 1~20 입력 시 해당 번호 인덱스 설정 값들이 해제됩니다.
3) 전체 인덱스의 접점 출력기능 해제

CAUTION

- MOINIT, MORNG, MOCLR 는 Moving Out 기능을 위한 명령어로 (03.03.04-C4 RO 170726
- 또는 03.03.04-CE RO 170726) 이상의 펌웨어 버전에서 동작됩니다.
- MOINIT에서 초기화 되지 않는 Index 사용 시 E1246 MOINIT Initial Err 알람이 발생합니다.
- Output_Range는 POS변수 중 XP 변수 사용이 가능하지만 Offset_Range의 경우 XP변수 사용 시 E1227 INVALID PARAMETER 알람이 발생합니다.
- 표시된 매개변수의 범의를 벗어나는 값을 입력한 경우 E1202 RANGE OVER 알람이 발생합니다.
- POS변수 사용 시 "P", "GP", "I", "F"등은 변수 명으로 사용하실 수 없습니다.
- 로봇 티칭 시 기준좌표는 반드시 구동경로 위로 설정해야 합니다. 로봇이 기준좌표를 지나지 않는 경우 출력되지 않습니다.

3.27.3.1프로그램 사용 예제

```
MAIN
INT CUPOSAN, CURPOSXY
INT CURPOSTO, REFPOSAN
INT REFPOSXY, REFPOSTO
INT MTYPE
POS XP0                          .....   이동 중 출력 위치를 입력할 변수 설정
POS OFFSET                       .....   옵셋값을 입력할 변수 설정
CURPOSAN=1                       .....   좌표계 인자 별 정의
CURPOSXY=2
CURPOSTO=3
REFSOSAN=4
REFPOSXY=5
REFPOSTO=6
MTYPE= REFPOSXY                  .....   위치 지령 값을 BASE 좌표계로 변환

XP0=<230.0,0,0.0,-30.0,0.0,0.0,0.0>  .....   위치값 입력
                                             (각 축 좌표 및 ARM Form 설정)
OFFSET=<10.0,0,0.05,10.0,10.0,0.0,0.0>  .....   옵셋 값 입력

I0=MOCLR(0)                      .....   모든 인덱스 초기화
                                         (전체 접점 출력 기능 해제)
I1=MOINIT(1,1,1,0)               .....   채널 및 포트정보 설정
I2=MORNG(1,MTYPE,XP0, OFFSET)    .....   MORNG 명령어 수행 이후 출력
                ⋮
```

CAUTION

- ▶ MOINIT, MORNG, MOCLR 는 Moving Out 기능을 위한 명령어로 (03.03.04-C4 RO 170726 또는 03.03.04-CE RO 170726) 이상의 펌웨어 버전에서 동작됩니다.
- ▶ MOINIT에서 초기화 되지 않는 Index 사용 시 E1246 MOINIT Initial Err 알람이 발생합니다.
- ▶ Output_Range는 POS변수 중 XP 변수 사용이 가능하지만 Offset_Range의 경우 XP변수 사용 시 E1227 INVALID PARAMETER 알람이 발생합니다.
- ▶ 표시된 매개변수의 범의를 벗어나는 값을 입력한 경우 E1202 RANGE OVER 알람이 발생합니다.
- ▶ POS변수 사용 시 "P", "GP", "I", "F"등은 변수 명으로 사용하실 수 없습니다.
- ▶ 로봇 티칭 시 기준좌표는 반드시 구동경로 위로 설정해야 합니다. 로봇이 기준좌표를 지나지 않는 경우 출력되지 않습니다.

3.28 VEL(축 이동 속도 설정 명령어)

기 능 축 이동 속도의 천분율(%)을 설정합니다.

형 식 VEL <속도>

용 어 <속도> : 로봇의 이동속도를 설정합니다. (0 ≤ <속도> ≤ 100.0%)
로봇이동속도 = INIT_V(초기 속도값) * Mv(각 축의 최대속도) * 0.001 * 속도
(0 ≤ <Mv 설정 속도> ≤ 최대 RPM).

설 명
1) 로봇 이동 속도에 대한 명령으로, **최대속도 사용 시 1000으로 설정**합니다.
2) 프로그램에서 속도 설정을 하지 않은 경우 INIT_V 파라미터에 설정 된 속도로 **자동 설정** 됩니다.
3) 속도값을 갖는 **변수(정수형 변수)를 사용**할 수 있습니다.

CAUTION

- 속도값은 100.0% 입니다.
  VEL 1000을 입력 시 100%의 속도로 이동 됩니다. (VEL 100 => 10%)
- 기구부의 허용 최대 RPM을 초과해서 사용시 소음 및 파손의 위험이 있습니다.
  반드시 기구부에 부착된 라벨 확인 후 사용하시기 바랍니다.
- 파라미터 초기화 시, INT_V 설정 값은 30% 이며,
  최대속도 사용 시 설정값을 '100%'로 변경하시기 바랍니다.

3.28.1 프로그램 사용 예제

1) 작업위치에 따라 속도변경

```
MAIN
WHILE 1
VEL 1000 ..... 로봇 이동 속도 (INIT_V) x (Mv) x 0.001 x 1000
JMOV P0
JMOV P1
VEL 200 ..... 로봇 이동 속도 (INIT_V) x (Mv) x 0.001 x 200
LMOV P2
LMOV P3
VEL 1000 ..... 로봇 이동 속도 (INIT_V) x (Mv) x 0.001 x 1000
JMOV P0
ENDWL
EOP
```

3.29 ACC, DEC(가 · 감속 설정 명령어)

기 능 가속/감속 시간의 백분율(%)을 설정

형 식 ACC <가속시간비율>
DEC <감속시간비율>

용 어 <가속시간비율/감속시간비율> : (50 ≤ <가감속시간비율> ≤ 200%)
가 · 감속시간을 프로그램에서 증가시키거나 감소시킵니다.
가 · 감속시간 = At(가감속시간) * 0.01 * 가감속비율(%)
(0.5*최대가감속시간 ≤ <가감속시간> ≤ 2*최대 가 · 감속시간)

설 명
1) 로봇의 가 · 감속시간을 설정합니다.
2) JOB프로그램 내부에서 가 · 감속시간비율을 설정하지 않은 경우에는 100%로 설정 됩니다.
3) 가 · 감속비율값을 갖는 변수(정수형 변수)를 사용할 수 있습니다.
4) 모든 축에 적용됩니다. (SCARA 로봇 : A, B, Z, W축 적용)

> 파라미터 At : A축(0.3초), B축(0.3초), Z축(0.2초), W축(0.5초) 상태에서 "ACC(DEC) 50" 실행하면 파라미터 At : A축(0.15초), B축(0.15초), Z축(0.1초), W축(0.25초)로 적용되어 동작합니다.
> (파라미터 값 자체가 수정되는 것이 아니고 JOB프로그램 내에서만 적용됩니다.)

5) 파라미터 가 · 감속 시간(At)으로 복귀는 "ACC(DEC) 100"을 사용하고 다음스텝부터 적용됩니다. (예제 프로그램을 참조바랍니다.)

CAUTION

- ▶ 파라미터에 설정된 가 · 감속 시간은 기계부 구성에 따라 최적으로 설정된 값입니다.
- ▶ ACC/DEC명령어의 가 · 감속 시간비율이 100보다 작게 설정될 때는 기계부의 소음 및 진동이 발생할 수 있으니 주의하여 사용하십시오.

3.29.1 프로그램 사용 예제

1) 작업위치에 따라 가감속시간 변경(파라미터 에서 At 값이 0.3초로 설정 되어있을 경우의 예)

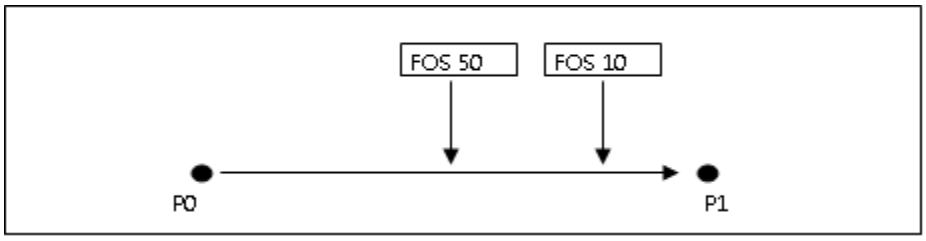
```
MAIN
WHILE 1
VEL 500
ACC 50          .....   가감속 0.15초로 설정
JMOV P1
ACC 200         .....   가감속 0.6초로 설정
LMOV P2
LMOV P3
ACC 100         .....   가감속 0.3초로 설정
JMOV P0
ENDWL
EOP
```

3.30 FOS, PFOS(연속궤적생성 명령어)

기 능 축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적 변경합니다.

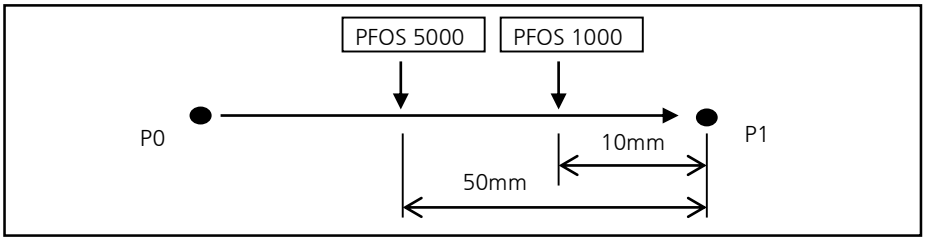
형 식 FOS <거리비율>
PFOS <거리>

용 어 <거리비율> : (0 ≤ <거리비율> < 50%)
로봇이 지정된 위치에 도착하기 전에 다음 위치로 이동시키거나 출력할 때의 시점을 설정합니다.
(거리비율은 두점(Point) 이동 거리에 대한 백분율값으로 설정합니다.)



위 그림은 P0에서 P1으로 이동할 때 각 FOS값에 따라 다음 위치로 이동하거나 출력할 때의 시점을 표시합니다.

<거리> : 로봇이 지정된 위치에 도착하기 전에 다음 위치로 이동시키거나 출력할 때의 거리를 설정합니다. (0 ≤ <거리> ≤ 9999mm) 단위: 0.01mm

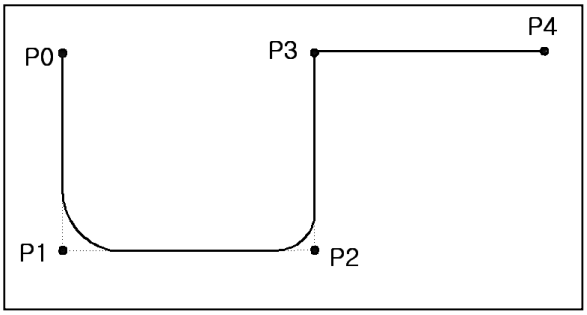


위 그림은 P0에서 P1으로 이동할 때 각 PFOS값에 따라 다음 위치로 이동하거나 출력할 때의 시점을 표시합니다.

설 명
1) FOS 나 PFOS가 설정되면, MOVE 명령은 지정된 위치에 도착하기 전 (설정된 거리비율또는 거리값)에 다음 위치로 이동합니다.
2) OUT 명령인 경우에는 이동중에 출력이 가능합니다.
3) PFOS는 보간 모션(LMOV, CMOV, AMOV)에만 적용됩니다.
4) 작업 프로그램에서 FOS또는 PFOS적용 이전의 상태로 복귀하기위해 "FOS 0" 또는 "PFOS 0"를 사용하고, 명령어는 1스텝 다음 명령어부터 적용됩니다.

```
MAIN
FOS 10
LMOV P0
LMOV P1
FOS 0
LMOV P2
LMOV P3
LMOV P4
EOP
```

FOS 0 명령어 적용은
LMOV P3 부터 됩니다.



CAUTION

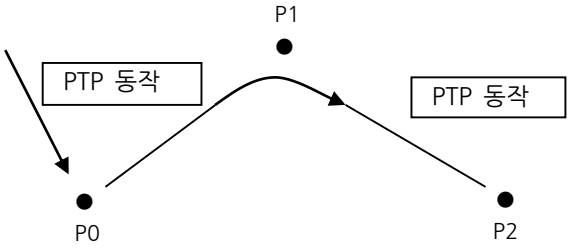
- ▶ 연속된 보간이동위한 포인트 티칭시 두 점 사이의 거리는 최소 5mm이상일 것
  (단 100mm/s 이동 속도 인 경우)
- ▶ 이동중인 궤적의 FOS 비율이 다음 위치의 전체길이 보다 짧을경우, "Too much FOS" 알람이 발생 됩니다.
- ▶ 이동 궤적이 PTP인 경우 궤적 변경이 되며, 보간 동작일 경우 궤적 삽입(원,호 또는 직선)이 됩니다.
- ▶ FOS,PFOS를 사용함으로 연속된 보간동작이 이루어 집니다.
- ▶ 원호(AMOV) 또는 원(CMOV) 동작 시 잘못 된 궤적 변경으로 인해 Auto RUN중 "Unreachable Point", "Inverse Error" 가 발생할 수 있습니다.
- ▶ 특히, 반경이 작은 AMOV또는 CMOV 명령어 사용 시 주의하시기 바랍니다.

3.30.1 프로그램 사용 예제

1) PTP 동작 (JMOV)

```
MAIN
VEL 100
JMOV P0
FOS 5
JMOV P1
FOS 0
JMOV P2
EOP
```

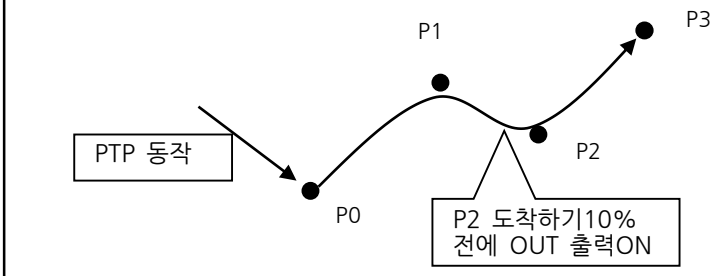
FOS 5 (포인트 P1 도착전 5%)



2) 로봇 이동 중 출력신호(OUT, POUT) On, Off

```
MAIN
VEL 100
JMOV P0
FOS 10
LMOV P1
LMOV P2
OUT0=1
FOS 0
JMOV P3
EOP
```

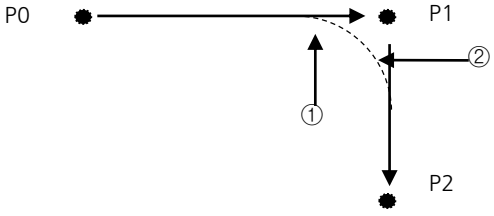
FOS 10



3) 로봇 이동 중 다음스텝 명령어 처리

```
MAIN
WHILE 1
VEL 100
JMOV P0
FOS 10
IF IN0==1 THEN
JMOV P1
ELSE
IN1=1
JMOV P2
ENDIF
FOS 0
ENDWL
EOP
```

FOS 10



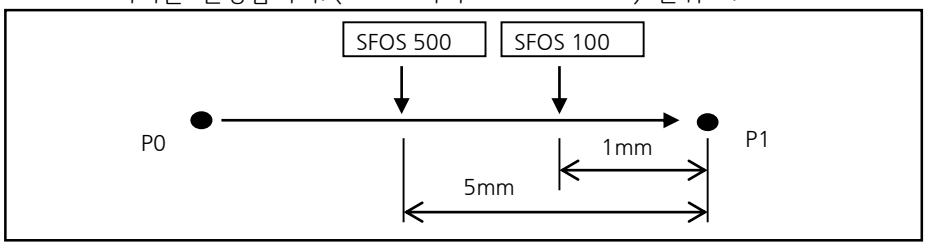
①위치에서
P1도착하기 10%전에 IF 명령어 처리
즉, 입력 "IN0"에 신호가 들어오면 ②(점선)와 같은
궤적을 그리며 이동

3.31 SFOS(연속궤적 및 등속 생성 명령어)

기 능 SFOS 명령어의 경우 축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적을 변경하는 것은 FOS 명령과 동일 하지만, 궤적 변경 시점부터 설정 값을 기준으로 등속을 유지하는 특성이 있습니다.

형 식 SFOS <거리><속도 레벨><원호 각도>

용 어 <거리> : 로봇이 지정된 위치에 도착하기 전에 다음 위치로 이동시키거나 출력할 때의 거리를 설정합니다. (0 ≤ <거리> ≤ 9999mm) 단위: 0.01mm



위 그림은 P0에서 P1으로 이동할 때 각 SFOS값에 따라 다음 위치로 이동하거나 출력할 때의 시점을 표시합니다.

<속도 레벨>: SFOS 명령어가 적용되는 이동 구간에서의 속도를 설정 합니다.
설정 값이 높을 수록 속도는 빨리 집니다.(0≤ <속도 레벨> ≤ 10)
(설정 값 0인 경우: 레벨 5와 동일합니다.)

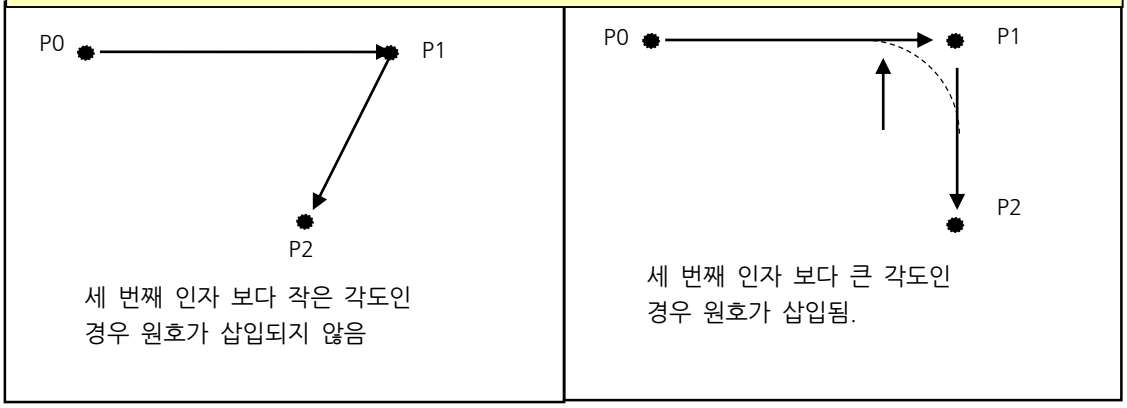
<원호 각도>: 연결되는 두 궤적과 궤적 사이의 원호 삽입 각도를 입력 합니다.
입력된 각도 값보다 작은 경우 원호가 삽입되지 않고, 큰 경우 첫번재 인자 값으로 원호가 삽입 됩니다.(설정 값 0도인 경우: 1도 값과 동일)
설정 값 180도인 경우: 궤적 사이에 원호가 삽입되지 않고, Too much fos 알람이 발생되지 않고 무시됩니다.
(180도 설정 사용시 궤적 변형 및 로봇 이동 중 진동이 발생 할 수 있습니다.)

설 명 1) SFOS가 설정되면, MOVE 명령은 지정된 위치에 도착하기 전 (거리값)에 다음 위치로 이동합니다.
2) MOVE 명령어 사이 DLAY 명령어 사용이 불가능 합니다.
3) 연속된 보간이동에서는 가감속이 없이 연속동작(등속이동)을 할 수 있습니다.
4) SFOS는 보간 모션(LMOV, CMOV, AMOV)에만 적용됩니다.
5) 작업 프로그램에서 SFOS 적용 이전의 상태로 복귀하기위해 해제 명령인 "SFOS 0 0 0" 사용하고, 다음 명령어부터 적용됩니다.

```
MAIN
SFOS 10 0 0
LMOV P0
LMOV P1
SFOS 0 0 0
LMOV P2
LMOV P3
LMOV P4
EOP
```

SFOS 0 명령어 적용은
LMOV P3 부터 됩니다.

SFOS 10 0 80



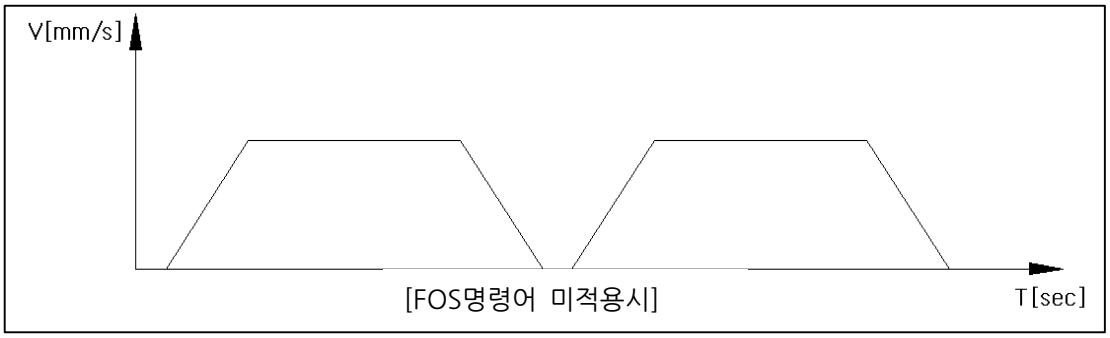
CAUTION

- ▶ SFOS를 사용함으로 연속된 보간동작이 이루어 집니다.
- ▶ 원호(AMOV) 또는 원(CMOV) 동작할 때 잘못된 Point Teaching으로 Auto RUN중 "Unreachable Point","Inverse Error" 가 발생할 수 있습니다.
- ▶ 특히, 크기가 작은 AMOV또는 CMOV 명령어 사용 시 주의하시기 바랍니다.

3.31.1 연속 모션 속도 프로파일 패턴

1) FOS 0인 경우

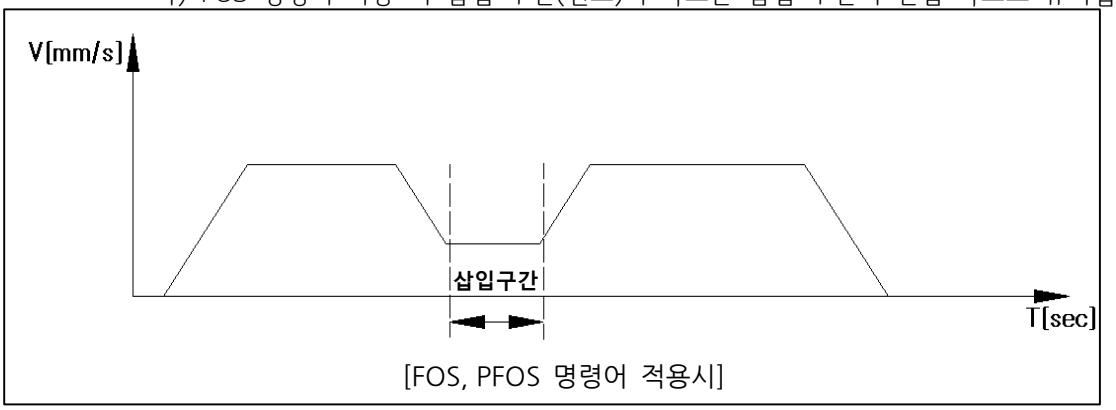
가) FOS 관련 명령어가 없는 경우 궤적 연결 부분에서 속도가 0으로 됩니다.



[FOS명령어 미적용시]

2) FOS 10 & PFOS 10

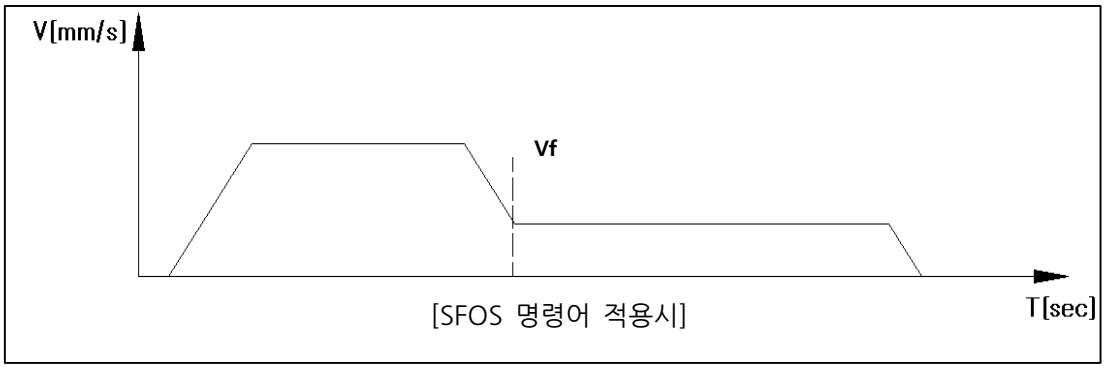
가) FOS 명령어 사용 시 삽입 구간(원호)의 속도는 삽입 구간의 진입 속도로 유지됩니다.



[FOS, PFOS 명령어 적용시]

3) SFOS 10 0 0

가) SFOS 설정 값과 속도 레벨로 다음 궤적의 최고 속도(Vf)가 결정 되어, 이동 속도가 유지 됩니다.



[SFOS 명령어 적용시]

CAUTION

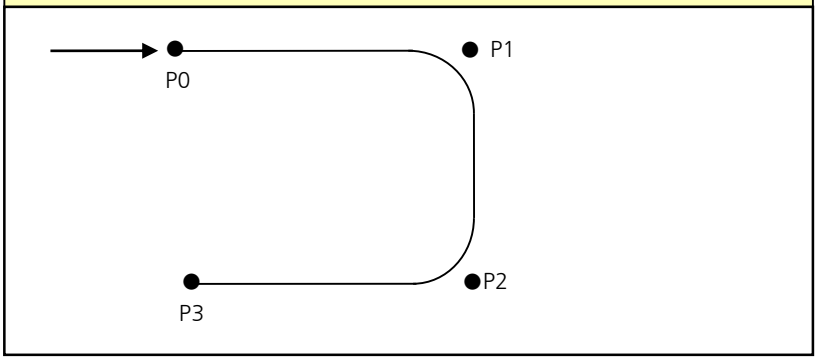
▶ 연결 궤적의 길이가 너무 작아 vf을 유지할 수 없는 경우, 등속이 유지 되지 않으며, 이럴 경우 속도 레벨을 작게 하거나, SFOS 량을 구간내의 최소 길이로 설정 합니다.

3.31.2 프로그램 사용 예제

1) 직선 보간 모션(LMOV)

```
MAIN
VEL 100
LMOV P0
SFOS 150 0 70
LMOV P1
LMOV P2
SFOS 0 0 0
LMOV P3
EOP
```

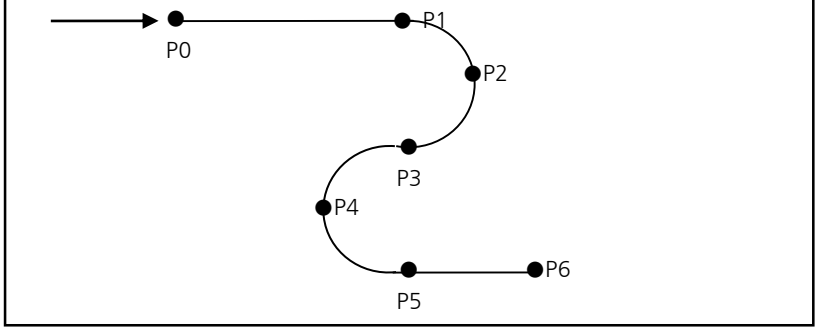
SFOS 150 0 70
P1, P2 도착 1.5mm전에 다음 위치로 이동



2) 원호 보간 모션(AMOV)

```
MAIN
VEL 100
JMOV P0
LMOV P1
SFOS 150 0 180
AMOV P2 P3
AMOV P4 P5
SFOS 0 0 0
LMOV P6
EOP
```

SFOS 150 0 180(0)
P1, P2 도착 1.5mm전에 다음 위치로 이동



3.32 SVON, SVOF (서보 ON/OFF 명령어)

기 능 서보 ON, 서보 OFF를 합니다.

형 식
SVON : 모든 축 서보 ON
SVON [<지정축>] : 지정 축 서보 ON
SVOF : 모든 축 서보 OFF
SVOF [<지정축>] : 지정 축 서보 OFF

설 명
1) SVOF
로봇 동작중 서보를 OFF 시켜 로봇 Body를 Free(손으로 로봇을 이동시킬 수 있는 상태) 상태로 전환합니다.(현재 구동중인 모든 축을 서보 OFF 시킵니다.)

2) SVON
로봇 Body를 서보 ON 시킵니다.
SVOF 후 SVON시키지 않은 상태에서 로봇동작(RUN)은 Dry RUN(로봇 Body는 움직이지 않고 JOB 프로그램이 1Step씩 실행) 상태입니다.
현재 서보 OFF된 모든 축을 서보 ON 시킵니다.

3) SVON(SVOF) <지정축> : 개별 축을 서보 ON/OFF 시킵니다.
SCARA 로봇 지정축 → A축 : 1, B축 : 2, Z축 : 3, W축 : 4
직각 로봇 지정축 → X축 : 1, Y축 : 2, Z축 : 3, W축 : 4, E1 : 5, E2 : 6

3.32.1 프로그램 사용 예제

1) PTP 동작 (JMOV)

```
MAIN
VEL 100
JMOV P0
IF IN1==1 THEN
SVOF 4 ..... 4번째 축을 서보 OFF 시킴
ENDIF
JMOV P1 ..... 4번째 축 서보 OFF 상태에서 포인트 P1으로 이동
IF IN1==0 THEN
SVON 4 ..... 4번째 축을 서보 ON 시킴
ENDIF
JMOV P2 ..... 모든 축이 서보 ON 상태로 포인트 P2로 이동
EOP
```

2) PTP 동작 (JMOV)

```
MAIN
VEL 500
JMOV P0
MVR=0
WITH
JMOV P1
WHILE MVR<100
IF IN0==1 THEN
STOP                ..........      P1으로 이동중 IN0=1이면
SVOF                                로봇 정지 후 서보 OFF
GOTO L1
ELSE
EXIT
ENDIF
ENDWL

LABL L1
ENDWT
IN0=1                               입력 IN0 가 "1"이 된 후
SVON                ..........      서보 ON
JMOV P2
EOP
```

CAUTION

▶ 서브루틴에서 SVON, SVOF를 사용하면 "Syntax Error"가 발생됩니다.

```
MAIN
VEL 100
IN0=1
JMOV P0
CALL A0
EOP

SUBR A0
SVOF
OUT1=1
IN1=1
SVON
RET
```

3.33 DLAY(지연시간 명령어)

기 능　　지연시간을 설정합니다.

형 식　　DLAY <지연시간>

용 어　　<지연시간> : 지연하고자 하는 시간을 설정합니다. (1=10ms)
(1 ≤ <지연시간> ≤ 999)

설 명　　1) 설정시간을 숫자로 설정합니다.
2) 설정시간은 0.01초~9.99초까지 가능합니다.
3) 시간 값을 가진 변수(정수형 변수)를 사용할 수 있습니다.

3.33.1 프로그램 사용 예제

1) PTP 동작 (JMOV)

```
MAIN
VEL 100
JMOV P10
DLAY 10          .....    100ms(0.1초) 동안 P10 위치에서 대기
JMOV P100
EOP
```

3.34 OFFS(옵셋 지정 명령어)

기 능 티칭 된 목표 지점을 지정 된 값만큼 임시 이동 시킵니다.

형 식 OFFS P<번호>
OFFS GP<번호>
OFFS <위치형변수>

용 어 <번호> : 티칭한 위치 좌표 번호를 설정합니다.
P : JOB 별로 개별 사용 되는 LOCAL POINT (0 ≤ 번호 ≤ 1999)
GP : 공통으로 사용 되는 GLOBAL POINT (0 ≤ 번호 ≤ 1023)

<위치형 변수> : POS형으로 선언된 변수이름을 의미합니다.

설 명 1) JOB 프로그램의 **임의의 POINT에 Offset값을 각 직각좌표로 저장**.
OFFSET 명령어가 수행된 후, **이후 스텝의 이동명령(MOV)시 옵셋값 만큼 증감**하여 로봇이 이동합니다.
2) 실행되고 있는 **JOB 프로그램 내에서만** 유효합니다.
MAIN 프로그램의 OFFS값은 **Job Call(JCALL)된 프로그램에서 적용되지 않습니다.**

옵셋이 적용된 위치로 이동 된다.

3.34.1 프로그램 사용 예제

1) PTP 및 CP 동작(JMOV,LMOV)<SCARA 로봇>

P11 <10.23, 22.37, 7.96, 120.00, 0, 0>
P12 <30.23, 22.37, 20.96, 200.00, 0, 0>
P100 <100, 100, 0, 0, 0, 0>

```
MAIN
VEL 100
JMOV P10        .....   포인트 P10 으로 PTP 이동
OFFS P100       .....   P100에 저장된 값 으로 옵셋 적용
LMOV P11        .....   P11 + P100 의 위치로 CP이동
                        <110.23,122.37,7.96,120.00, 0, 0>
LMOV P12        .....   P12 + P100 의 위치로 CP이동
                        <130.23, 122.37, 20.96, 200.00, 0, 0>
EOP
```

CAUTION

- JOB프로그램에 저장된 POINT가 파라미터의 "RANG"이내 에 위치하여 POINTTeaching할 때, "Range OVER" Error가 발생하지 않으나 JOB 프로그램이 실행될 때 OFFS에 의해 "Range OVER" Error가 발생할 수 있습니다.
- Point Teaching 할때에는 ANGLE 좌표로 입력하시기 바랍니다.
- OFFS 을 해제 하려면, <0,0,0,0,0,0> 의 값을 할당 해야 합니다.

3.35 USEROFFS(사용자 좌표계에서 옵셋된 위치 반환 명령어)

기 능 사용자 좌표계 방향으로 옵셋 된 축 좌표계 기준위치를 반환

형 식 <반환 위치>=USEROFFS(<사용자 좌표계 번호>, <기준위치>, <옵셋 이동량>)

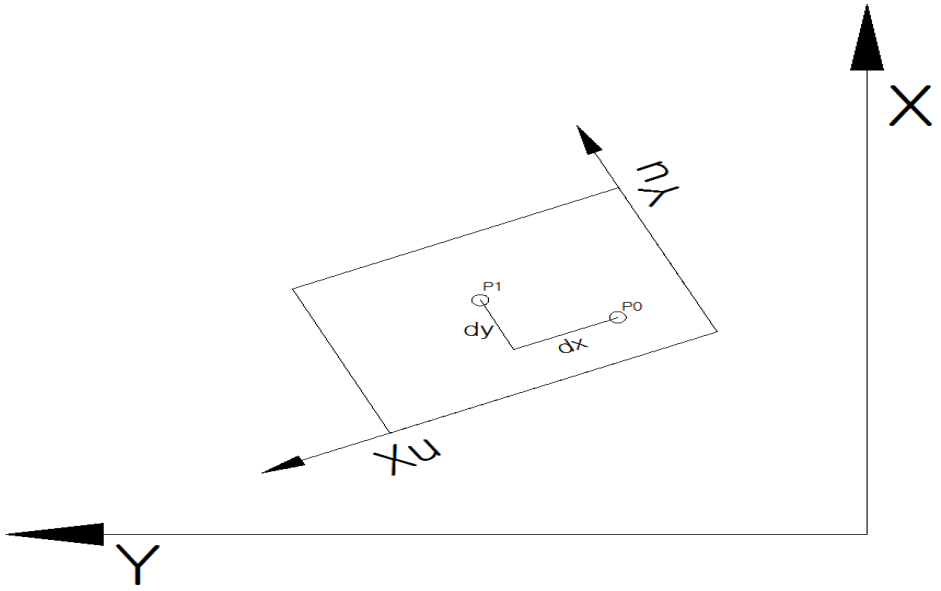
용 어 <사용자 좌표계 번호> : 파라미터에서 "USER"에서 설정한 사용자 좌표계
자료타입 : "상수", "I 변수", "변수"

<기준 위치>: 축 및 직교 좌표계 기준의 원점 위치
자료 타입: "P<번호>", "GP<번호>", "위치형 변수"

<옵셋 이동량>: 이동을 위한 옵셋 량
자료 타입: "P<번호>", "GP<번호>", "위치형 변수"

<반환 위치>: 옵셋량을 적용 후 축 좌표계 또는 직교 좌표계 기준의 위치 반환 값
자료 타입: " P<번호>", "GP<번호>", "위치형 변수"

설 명 1) 축 및 직교 좌표의 위치에서 사용자 좌표계 방향으로 옵셋 이동량 만큼 이동된 위치를 반환 합니다.



X,Y: 직교 좌표계 축
Xu, Yu: 사용자 좌표계 축
dx, dy: 옵셋 이동량
P0: 기준 위치

3.35.1 프로그램 사용 예제

```
MAIN
VEL 300
POS AP, OFFS          .....   반환 값이 옵셋량 지정을 위한 위치 변수 선언
OFFS=< 30,30,0,0,0,0>  .....   옵셋량 < 30,30,0,0,0,0> 지정
JMOV P0
AP = USEROFFS(1, P0, OFFS)  .....   사용자 좌표계 기준으로 옵셋된 위치 반환
JMOV AP
EOP
```

CAUTION

- ▶ 사용자 좌표계가 정의 되어 있지 않는 번호 선택 시 알람이 발생합니다.
- ▶ USEROFFS의 기준 위치 및 옵셋 이동량에 따라 "Range Over Error"알림이 발생합니다.

3.36 SETUSER(사용자 좌표계 생성 명령어)

기 능 기존에 티칭한 포인트 변수 3개를 사용하여 사용자 좌표계 생성

형 식 <정수형 변수> = SETUSER(<사용자 좌표계 번호>,<UP0>,<UP1>,<UP2>)

용 어 <정수형 변수>: 실행 완료 시, 0이 저장되며 에러 발생 시, 해당 에러 번호 저장됨

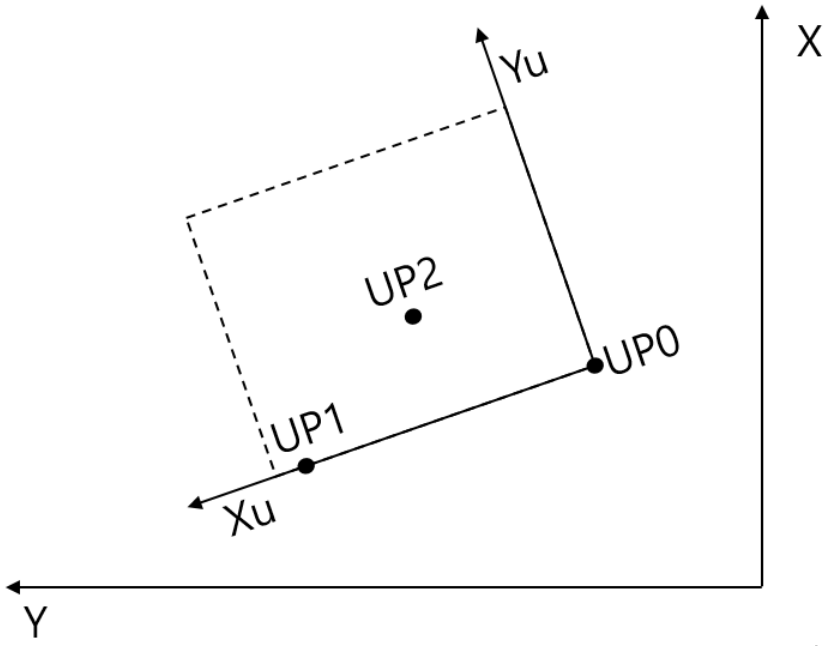
<사용자 좌표계 번호> : 생성 혹은 편집할 사용자(USER) 좌표계 번호
자료타입 : "상수", "I 변수", "변수"

<UP0> : 사용자 좌표계의 원점 좌표
자료타입 : "POS변수" 및 "POSITION변수"

<UP1> : 사용자 좌표계의 X축 지정 좌표
자료타입 : "POS변수" 및 "POSITION변수"

<UP2> : 사용자 좌표계의 X축과 Y축의 +방향으로 이루어진 평면 상의 좌표
자료타입 : "POS변수" 및 "POSITION변수"

설 명 1) 티칭한 포인트 변수 3개를 사용하여 원하는 번호의 사용자 좌표계를 설정합니다.
2) 설정된 사용자 좌표계는 해당 JOB 및 다른 JOB, 파라미터의 USER 항목에서도 확인 및 사용 가능합니다.
※ UP0 ~ 2에 대한 자세한 설명은 N1-OM-K□□를 참고하시기 바랍니다.



X,Y: 직교 좌표계 축
Xu, Yu: 사용자 좌표계 축
UP0: 사용자 좌표계 원점
UP1: 사용자 좌표계 X 축 위치
UP2: 사용자 좌표계 XY 평면 위치

3.36.1 프로그램 사용 예제

```
MAIN
POS XP(3)                                  ..... 사용자 좌표계 생성을 위한 위치형 변수 선언
XP(0) = <300,200,0,0,0,0,0>                .....
XP(1) = <308,66.205,0,0,0,0,0>             ..... 사용자 좌표계 생성을 위한 원점, X축,XY평면 위치 입력
XP(2) = <303.66,213.66,0,0,0,0,0>          .....
I0 = SETUSER(2, XP(0), XP(1), XP(2))       ..... 입력한 세 점을 사용하여 사용자 좌표계 2번 생성

I1 = SETUSER(5, GP7, GP1, GP4)             ..... 미리 티칭된 GP 세 점을 사용하여 사용자 좌표계
                                                 5번 생성
EOP
```

CAUTION

- ▶ 명령어에 사용되는 UP0, UP1, UP2.포인트 변수 간 거리가 최소 10mm 미만인 경우, "RANGE OVER"알람이 발생하며, 상세 메시지에 알람이 발생한 JOB 파일 이름과 "CHK_DIST"메세지가 출력됩니다.
- ▶ 해당 명령어 사용 시, 이미 티칭된 사용자 좌표계도 변경할 수 있습니다
- ▶ 해당 명령어는 로봇이 고속으로 동작해야 하는 명령어 구간에서는 사용을 권장하지 않습니다

3.37 CALLUSER(사용자 좌표계 오프셋 확인 명령어)

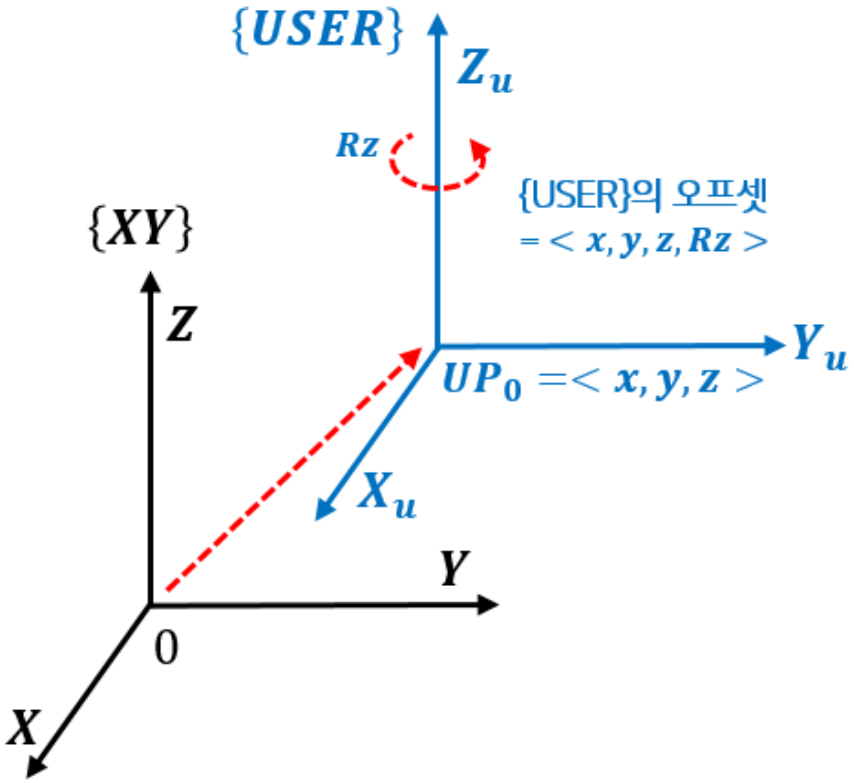
기 능 설정된 사용자 좌표계의 오프셋 값 확인

형 식 <좌표계 오프셋> = CALLUSER(<사용자 좌표계 번호>)

용 어 <좌표계 오프셋> : 명령어에 입력한 사용자 좌표계의 오프셋 값 저장
자료타입 : "AP변수", "GP 포인트 변수"

<사용자 좌표계 번호> : 파라미터의 "USER"에서 설정한 사용자 좌표계 번호
자료타입 : "상수", "I 변수", "변수"

설 명 1) 미리 설정되 있는 사용자 좌표계의 오프셋 값을 확인합니다

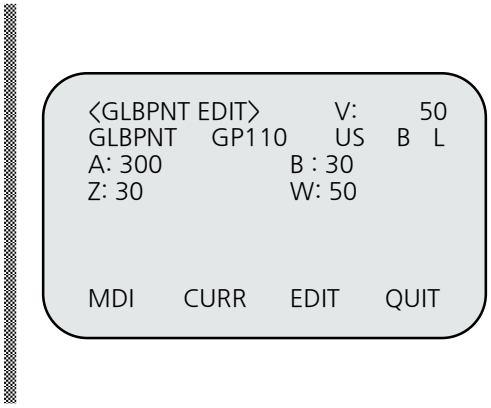


{XY}: 직교 좌표계
{USER}: 사용자 좌표계
x, y, z: 사용자 좌표계 기준점의 x,y,z 좌표
Rz: {XY}의 Z 축 기준으로 {USER}가 회전한 각도

3.37.1 프로그램 사용 예제

```
MAIN
POS AP                  .....   사용자 좌표계 오프셋값 저장을 위한 위치 변수 선언
AP = CALLUSER(1)        .....   사용자 좌표계 1번의 오프셋 값 저장
GP110 = CALLUSER(5)     .....   사용자 좌표계 5번의 오프셋 값 저장
EOP
```

3.37.2 SETUSER를 사용한 사용자좌표계 오프셋 저장 예시



SETUSER를 사용하여 글로벌 포인트 변수에 오프셋을 저장한 경우, 변수 내 각 축 값은 다음과 같습니다.

변수 내 성분	저장되는 값
A축 값	오프셋 X좌표 값[mm]
B축 값	오프셋 Y좌표 값[mm]
Z축 값	오프셋 Z좌표 값[mm]
W축 값	직교좌표 Z축 기준 회전 오프셋(Rz) 값[Degree]

CAUTION

- ▶ 사용자 좌표계가 정의 되어 있지 않는 번호 선택 시 알람이 발생합니다.
- ▶ 읽어온 오프셋 값은 로봇의 XY 좌표계 기준 사용자 좌표계의 오프셋 값입니다.
  (Rz의 경우, 양수값으로만 출력됩니다 : -15º → 345º)

3.38 ADJUSER(사용자 좌표계 오프셋 조정 명령어)

기 능 직교 좌표계 기준 사용자 좌표계의 위치 및 회전량(Rz) 조정

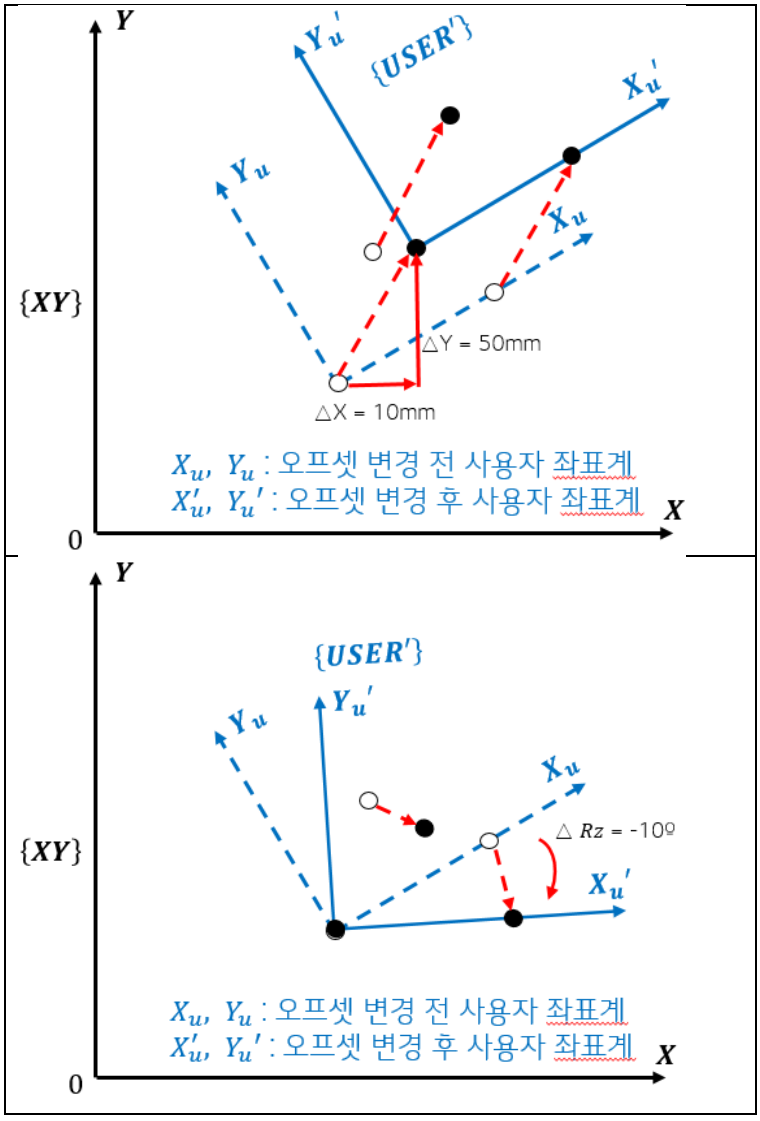
형 식 <정수형 변수> = ADJUSER(<사용자 좌표계 번호>,<오프셋 좌표값>)

용 어 <정수형 변수>: 실행 완료 시, 0이 저장되며 에러 발생 시, 해당 에러 번호 저장됨

<사용자 좌표계 번호> : 파라미터에서 "USER"에서 설정한 사용자 좌표계
자료타입 : "상수", "I 변수", "변수"

<오프셋 좌표값>: 직교 좌표계 기준 사용자 좌표계의 이동을 위한 증분량
자료타입 : "AP변수", "GP 포인트 변수"

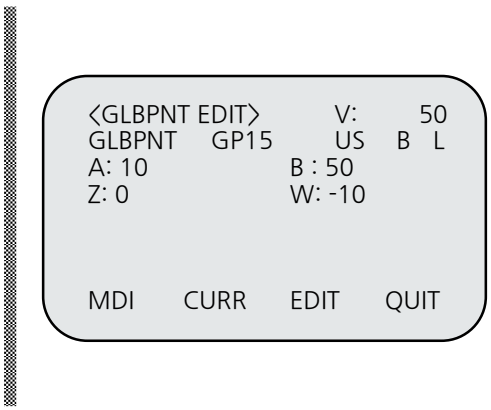
설 명 1) 미리 설정되 있는 사용자 좌표계의 오프셋 값을 증분량 만큼 조정합니다



3.38.1 프로그램 사용 예제

```
MAIN
POS AP                          .....   사용자 좌표계 증분량 지정을 위한 위치 변수 선언
AP = <0,0,0,0,0,0>              .....   위치 변수 초기화
AP . 1 = F0                     .....   직교 좌표계 기준 X 좌표방향 옵셋량 입력
AP . 2 = F1                     .....   직교 좌표계 기준 Y 좌표방향 옵셋량 입력
AP . 3 = F2                     .....   직교 좌표계 기준 Z 좌표방향 옵셋량 입력
I11 = ADJUSER(4, AP)            .....   AP변수를 사용한 사용자 좌표계 4번의 오프셋 변경
I17 = ADJUSER(2, GP15)          .....   글로벌 포인트 15번 변수를 사용한 사용자 좌표계
                                        2번의 오프셋 변경
EOP
```

3.38.2 ADJUSER를 사용하기 위한 사용자 좌표계 증분값 입력 예시



ADJUSER와 글로벌 포인트 변수를 사용하여 오프셋을 변경하려 할 경우, 변수 내 각 축 값들의 의미는 다음과 같습니다.

변수 내 성분	저장되는 값
A축 값	X좌표 증분량(△X)[mm]
B축 값	Y좌표 증분량(△Y)[mm]
Z축 값	Z좌표 증분량(△Z)[mm]
W축 값	직교좌표 Z축 기준 회전 증분량(△Rz)[Degree]

CAUTION

- ▶ 사용자 좌표계가 정의 되어 있지 않는 번호 선택 시 알람이 발생합니다.
- ▶ ADJUSER의 옵셋 이동량에 따라 "Range Over Error"알림이 발생합니다.
- ▶ 해당 명령어 실행 후, 오프셋이 변경된 좌표계는 파라미터 내 USER항목내 저장된 티칭 포인트들도 변경됩니다.

3.39 LIMT(축 제한 명령어)

기 능　　각 축의 이동범위를 제한합니다.

형 식　　LIMT P<번호1> P(번호2)
LIMT <위치형변수> (위치형변수)

용 어　　<번호1> : 마이너스(-) 리밋값을 JOB 프로그램의 POINT에 설정합니다.
(0 ≤ <번호> ≤ 1999)
(번호2) : 플러스(+) 리밋값을 JOB 프로그램의 POINT에 설정합니다.
(0 ≤ <번호> ≤ 1999)

설 명　　1) JOB 프로그램내에서의 **소프트웨어 리밋을 설정**합니다.
2) 실행되고 있는 **JOB 프로그램내에서만** 유효합니다.
MAIN 프로그램의 LIMT값은 **Job Call(JCALL)된 프로그램에서도 적용**됩니다.

3.39.1 프로그램 사용 예제

1) PTP 동작(JMOV, SCARA 로봇)

```
MAIN
VEL 100
JMOV P0          .....
LIMIT P100 P101  .....
JMOV P1          .....
EOP              .....
```

JOB 프로그램내에서의 소프트웨어 리밋은 아래와 같이 적용됨.
A: -100도 ~ +100도
B: -100도 ~ +100도
Z: 0 mm ~ 100 mm
W:-300도 ~ +300도

수평다관절 로봇에서 리밋을 P100, P101에 설정할 경우

P100 위치: (A:-100.00 B:-100.00 Z:000.00 W:-300.00 E1:-100.00 E2:-100.00)
P101 위치: (A:100.00 B:100.00 Z:100.00 W:300.00 E1:100.00 E2:100.00)

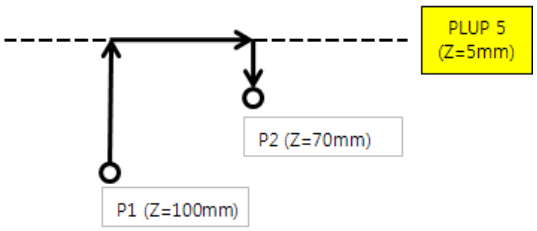
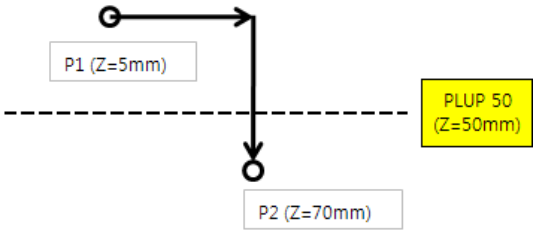
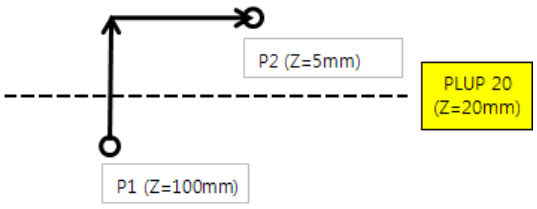
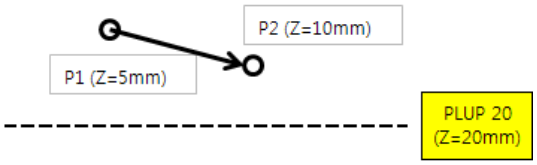
3.40 PLUP(Pull up 동작 설정 명령어)

기 능 Pull up 동작을 위한 Z축 값을 설정합니다.

형 식 PLUP <Z축값>

용 어 <Z축값> : Z축 풀업 값을 설정 합니다.(단위 : mm)
(0 ≤ Z축값 ≤ Z축 스트로크(STROKE))

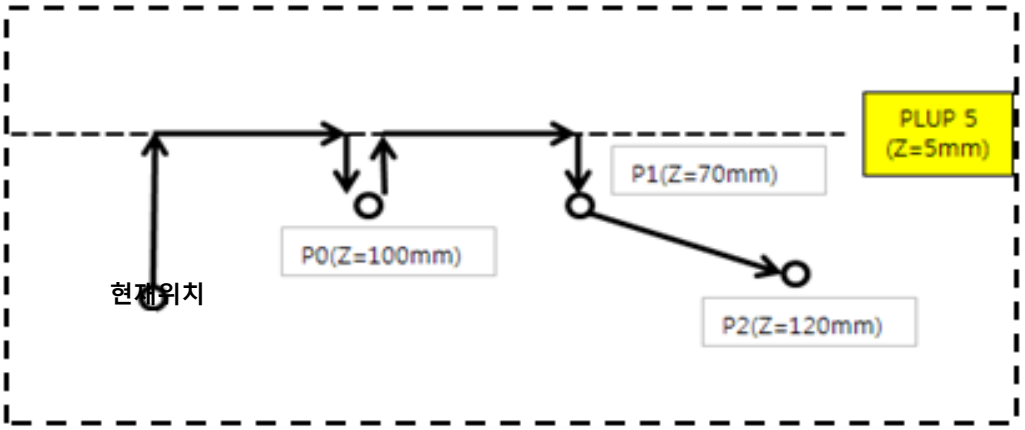
설 명
1) Z축이 있는 로봇에만 적용. Z축이 2개 이상 있을 경우 첫번째 Z축만 적용 됩니다.
2) PTP이동(JMOV, PMOV, IMOV)에만 적용 되며, 보간이동에는 적용 되지 않습니다.
3) Z축 풀업값을 가진 변수 (정수형 변수)를 사용 할 수 있습니다.
4) Z축 풀업값 적용 해제는 "PLUP 0"를 사용 하고, 다음 스텝부터 적용 됩니다.
5) Z축 풀업 값에 따른 로봇의 동작은 아래와 같습니다.

구분	로봇의 동작	해설
1. P1, P2 의 값이 모두 풀업 값 보다 **큰 경우**		1. P1에서 Z축을 풀업 값 까지 상승합니다. 2. Z축을 제외한 나머지 축을 P2의 풀업 값 위치로 이동합니다. 3. Z축을 P2위치까지 하강합니다.
2. P1의 Z축 값은 풀업 값 보다 **작고,** P2의 Z축 값은 풀업 값 보다 **큰 경우**		1. Z축을 제외한 나머지 축을 P2의 풀업값 위치로 이동합니다. 2. Z축을 P2위치까지 하강합니다.
3. P1의 Z축 값은 풀업 값 보다 **크고,** P2의 Z축 값은 풀업 값 보다 **작은 경우**		1. P1에서 Z축을 P2의 Z축 위치까지 상승합니다. 2. P2 위치로 이동합니다.
4. P1, P2 의 값이 모두 풀업 값 보다 **작은 경우**		1. PLUP 명령과 상관 없이 P2로 이동합니다.

3.40.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
PLUP 5          .....   Z축 풀업 값을 5mm 로 설정
JMOV P0
JMOV P1
PLUP 0          .....   Z축 풀업 값을 해제
JMOV P2
EOP
```



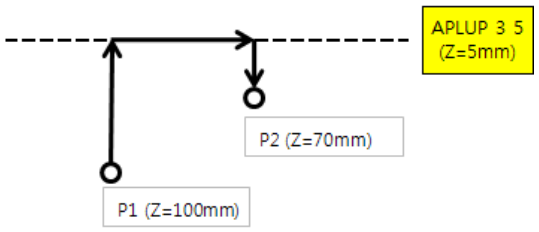
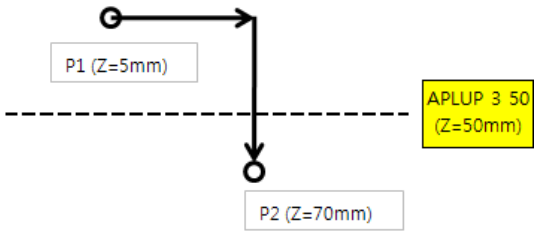
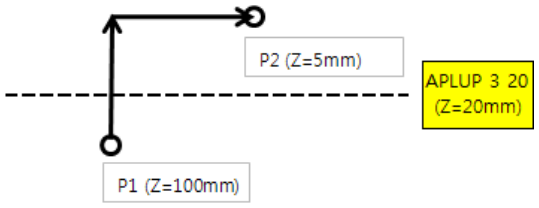
3.41 APLUP(Pull up 동작 설정 명령어)

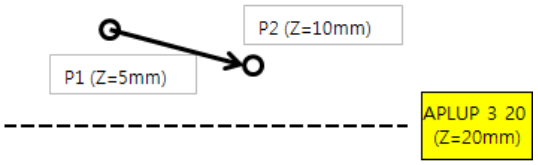
기 능 Z축이 여러 개 있는 경우, Pull up 동작을 하기 위한 Z축을 선정하고 해당 Z축의 Pull up 값을 설정합니다.

형 식 APLUP <축 번호> <Z축값>

용 어 <축 번호> : Pull up할 Z축 번호를 설정 합니다.
(ex: RSA-40의 경우 XYZW순(Z축이 세번째 축)이므로 정수 3 입력)
<Z축값> : Z축 풀업 값을 설정 합니다.(단위 : mm)
(0 ≤ Z축값 ≤ Z축 스트로크(STROKE))

설 명
6) Z축이 있는 로봇에만 적용. Pull up을 실행할 Z축 번호를 설정한 후 적용됩니다.
7) PTP이동(JMOV, PMOV, IMOV)에만 적용 되며, 보간이동에는 적용 되지 않습니다.
8) Z축 풀업값을 가진 변수 (정수형 변수)를 사용 할 수 있습니다.
9) Z축 풀업값 적용 해제는 "APLUP <축 번호> 0"를 사용 하고, 다음 스텝부터 적용 됩니다.
10) Z축 풀업 값에 따른 로봇의 동작은 아래와 같습니다.

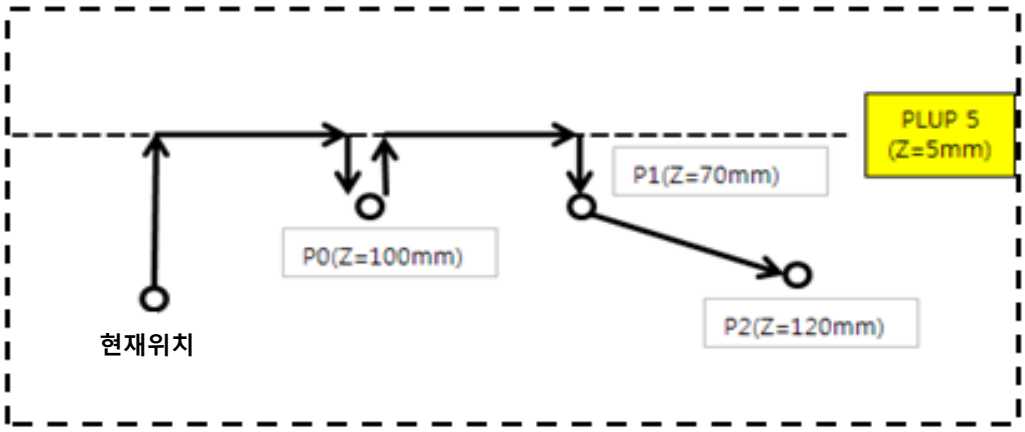
구분	로봇의 동작	해설
1. P1, P2 의 값이 모두 풀업 값 보다 **큰 경우**		4. P1에서 Z축을 풀업 값 까지 상승합니다. 5. Z축을 제외한 나머지 축을 P2의 풀업 값 위치로 이동합니다. 6. Z축을 P2위치까지 하강합니다.
2. P1의 Z축 값은 풀업 값 보다 **작고,** P2의 Z축 값은 풀업 값 보다 **큰 경우**		3. Z축을 제외한 나머지 축을 P2의 풀업값 위치로 이동합니다. 4. Z축을 P2위치까지 하강합니다.
3. P1의 Z축 값은 풀업 값 보다 **크고,** P2의 Z축 값은 풀업 값 보다 **작은 경우**		3. P1에서 Z축을 P2의 Z축 위치까지 상승합니다. 4. P2 위치로 이동합니다.

4. P1, P2 의 값이 모두 풀업 값 보다 **작은 경우**		2. APLUP 명령과 상관 없이 P2로 이동합니다.

3.41.1 프로그램 사용 예제

2) 포인트 티칭한 위치좌표로 이동 (직교로봇 중 XYZZZ로봇의 경우)

```
MAIN
VEL 100
APLUP 3 5          ..........   첫번째 Z축의 풀업 값을 5mm 로 설정
JMOV P0
JMOV P1
APLUP 3 0          ..........   첫번째 Z축의 풀업 값을 해제
JMOV P2
EOP
```



3.42 JTOU(사용자 좌표계 위치 반환 명령어)

기 능　축 및 직교 좌표의 위치에서 사용자 좌표의 위치로 변환

형 식　<반환 위치>=JTOU(<사용자 좌표계 번호>, <시작 위치>)
예) AP = JTOU(1, P0)

용 어　<사용자 좌표계 번호>: 파라미터에서 "USER" 에서 설정한 사용자 좌표계
자료 타입: "상수", "I 변수", "변수"

<기준 위치>: 축 및 직교 좌표계 기준의 위치
자료 타입: "P<번호>", "GP<번호>", "위치형 변수"

<반환 위치>: 사용자 좌표계 기준의 위치
자료 타입: "P<번호>", "GP<번호>", "위치형 변수"

설 명　1) 축 및 직교 좌표의 위치를 선댁된 사용자 좌표의 위치 데이터로 반환합니다.

3.42.1 프로그램 사용 예제

```
MAIN
VEL 300
POS UP,AP          .....    위치형 변수 선언
JMOV P0
UP = JTOU (1, P0)  .....    축 좌표계 위치를 사용자 좌표계 위치로 변환
UP.1 = UP.1+30     .....    사용자 좌표계 기준으로 옵셋
UP.2 = UP.2+50     .....    사용자 좌표계 기준으로 옵셋
EOP
```

3.43 UTOJ(사용자 좌표계 위치에서 축 좌표계 위치 반환)

기 능 **사용자 좌표계 기준의 위치를 축, 직교 좌표계 기준의 위치로 반환합니다.**

형 식 <반환 위치>=UTOJ(사용자 좌표계 번호, 시작 위치)
예) AP = UTOJ(1, P0)

용 어 <사용자 좌표계 번호>: 파라미터에서 "USER" 에서 설정한 사용자 좌표계
자료 타입: "상수", "I 변수", "변수"

<기준 위치>: 축 및 직교 좌표계 기준의 위치
자료 타입: " P<번호>", "GP<번호>", "위치형 변수"

<반환 위치>: 축, 직교 좌표계 기준의 위치
자료 타입: " P<번호>", "GP<번호>", "위치형 변수"

설 명 1) 선택된 사용자 좌표계 기준의 위치 데이터를 축, 직교 좌표계 기준의 위치데이터로 반환합니다.

3.43.1 프로그램 사용 예제

```
MAIN
VEL 300
POS UP,AP ..... 위치형 변수 선언
JMOV P0
UP = JTOU(1, P0) ..... 축 좌표계 위치를 사용자 좌표계 위치로 변환
UP.1= UP.1+30 ..... 사용자 좌표계 기준으로 옵셋
UP.2=UP.2+50 ..... 사용자 좌표계 기준으로 옵셋
AP=UTOJ(1,UP) ..... 사용자 좌표계 위치를 축 좌표계 위치로 변환
JMOV AP
EOP
```

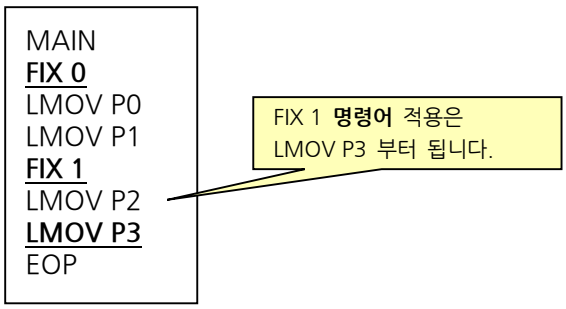
3.44 FIX(보간 동작시 W축 동작 선택 명령어)

기 능 보간 동작에서 W축 동작 선택

형 식 FIX <형식>

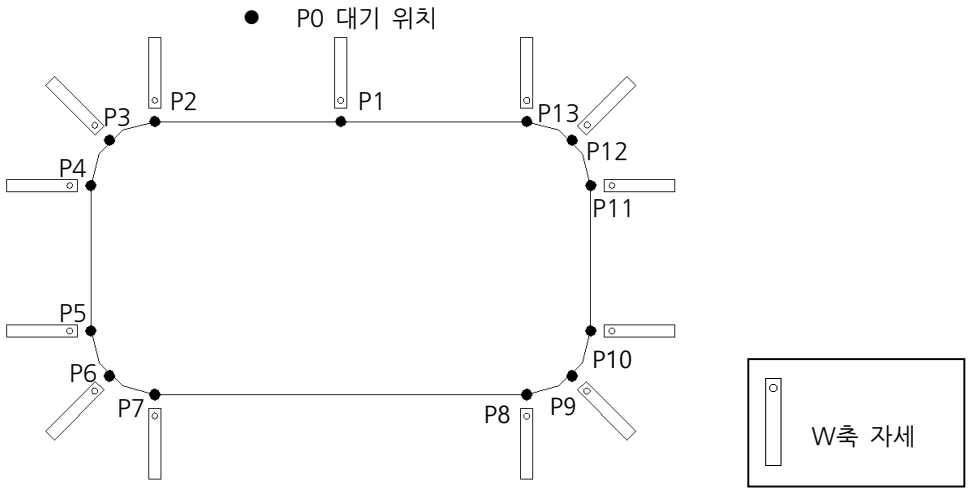
용 어 <형식>
FIX 1 : 로봇의 핸드 자세가 고정 (W축 자세 고정)
FIX 0 : 로봇의 핸드 자세가 회전 (W축 자세 회전)

설 명
1) 로봇이 보간 동작을 할 때 로봇 핸드(W축)의 자세를 고정 또는 회전 합니다.
2) PTP 동작에서는 FIX 설정과 관계 없이 교시점(티칭 된 POINT)으로 이동 합니다.
3) "FIX 0"으로 설정 되면 로봇의 회전축(W축)은 교시점의 W축 좌표값으로 보간 이동 합니다.
4) "FIX 1"로 설정 되면 로봇의 회전축(W축)은 시작점의 자세를 유지하면서 보간 이동 합니다.
5) "FIX 1"로 설정 되면 도착축의 회전축(W축)좌표값은 최초 교시점과 다른 값을 가질 수 있습니다.
6) 작업 프로그램내에서 FIX명령이 사용 되지 않으면 "FIX 1"로 설정 됩니다.
단 ROSEP Robot의 경우는 프로그램내에서 FIX명령이 사용 되지 않으면 "FIX 0"로 설정 됩니다.
7) 작업 프로그램에서 FIX 적용 이전의 상태로 복귀하기 위해 "FIX 1"을 사용 하고 명령어는 2 스텝 다음 명령어부터 적용 됩니다.



3.44.1 프로그램 사용 예제

1) W축 자세를 회전 시키면서 보간 동작 작업




```
MAIN
VEL 100
JMOV P0
JMOV P1
FIX 0 ..... W축 자세 회전
LMOV P2
AMOV P3 P4
LMOV P5
AMOV P6 P7
LMOV P8
AMOV P9 P10
LMOV P11
AMOV P12 P13
FIX 1 ..... W축 자세 고정
LMOV P1
EOP
```

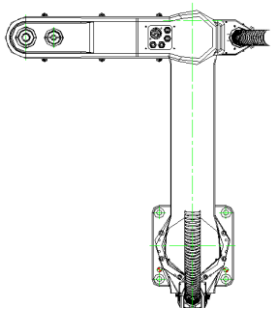
3.45 FORM (로봇 ARM 형상 지정 명령어)

기 능 로봇 동작시 로봇 ARM 형태로 설정합니다.

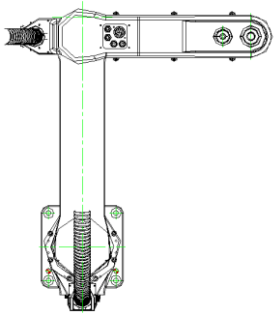
형 식 FORM <형태>

용 어 <형태> : 수평 다관절 로봇 (SCARA) 에서 이동 할 때 ARM (A,B 축) 의 형상을 설정합니다. (LEFT, RIGHT, NO)

설 명 1) **수평 다관절 로봇(SCARA) 에만 적용**
2) 로봇이 이동 할 때 ARM의 형상을 설정 합니다.
3) 형태에는 3가지 종류(LEFT, RIGHT, NO)가 있습니다.



[FORM RIGHT]



[FORM LEFT]

3.45.1 프로그램 사용 예제

1) 포인트 티칭한 위치좌표로 이동

```
MAIN
VEL 100
JMOV P10
FORM LEFT ..... ARM 형상을 LEFT로 설정
JMOV P11        P11 위치로 이동
FORM RIGHT ..... ARM 형상을 RIGHT로 설정
JMOV P11        P11 위치로 이동
EOP
```

CAUTION

- ▶ JMOV 명령어에서만 로봇 자세가 변경됩니다.
- ▶ 수평 다관절 로봇은 하나의 좌표 값에 대해 두가지의 자세가 있습니다. (LEFT FORM, RIGHT FORM)
- ▶ LEFT FORM으로 포인트 저장 하였어도 프로그램에서 "FORM RIGHT"가 지정 되면 저장된 POINT의 자세를 무시 하고 RIGHT FORM 으로 이동 합니다.
- ▶ FORM 명령어는 JMOV에서만 적용됩니다.

3.46 TRQ(충돌 감지 명령어)

기 능 충돌 감지시 토크 리밋 알람이 발생합니다.

형 식 TRQ <지정축> <토크리밋>

용 어 <지정축> : 1 → A(X)축, 2 → B(Y)축, 3 → Z축, 4 → W축, 5 → E1축, 6 → E2축
<토크리밋> : 범위 → 1~300[%] (300%는 모터 정격토크의 3배를 의미합니다.)

설 명 1) 로봇 동작중 지정된 각 축의 토크리밋을 설정하여 그 이상의 토크가 파라미터 TOL로 설정한 시간이상으로 동작시 Torque Limit Alarm이 발생합니다.
(N1-OM 1.2.4.9장을 참조바랍니다.)
2) 토크리밋값을 240 이상으로 설정시 토크값은 제한되지만 알람발생은 되지 않습니다.

3.46.1 프로그램 사용 예제

```
MAIN
VEL 500
FOS 10
JMOV P0
WHILE 1
LMOV P1
TRQ 1 50 ..... A(X)축의 토크 리밋을 50%로 설정
TRQ 2 50 ..... B(Y)축의 토크 리밋을 50%로 설정
LMOV P2
LMOV P1
LMOV P3
ENDWL
EOP
```

P2, P1, P3로 이동중 파라미터 TOL로 설정한 시간 이상 외부 충격이 인가될 경우 알람 발생

3.47 TQL(출력 토크값 제한 명령어)

기 능　　각 축의 최대 출력 토크값 제한.

형 식　　TQL <지정축> <토크리밋>

용 어　　<지정축> : 1 → A(X)축, 2 → B(Y)축, 3 → Z축, 4 → W축, 5 → E1축, 6 → E2축
<토크리밋> : 범위 → 1~300[%] (300%는 모터 정격토크의 3배를 의미합니다.)

설 명　　1) 통상적으로 순간 토크는 정격의 약 3배 정도 허용하고 있으나, 이로 인해 모터 또는 기계 강도에 문제가 생길 우려가 있을 경우에 한해서만 **최대 토크를 제한**하기 위함.
2) 토크가 부족하여 일정시간 동안 위치에 도달 하지 못할 경우 "In Position Error"등의 알람이 발생 됩니다.

3.47.1 프로그램 사용 예제

```
MAIN
VEL 500
FOS 10
JMOV P0
WHILE 1
LMOV P1
TQL 1 50 ..... A(X)축의 최대 출력 토크를 정격의 50%로 제한
TQL 2 50 .....
LMOV P2
LMOV P1
LMOV P3
ENDWL
EOP
```

P2, P1, P3 로 이동시 TQL로 설정된 최대 출력토크 이하의 토크로 이동함.
토크가 부족하여 이동 하지 못할 경우 알람 발생

3.48 INPOS (목표점 도달 지정 명령어)

기 능 로봇동작시 목표점 도달정도를 표시합니다.

형 식 INPOS <목표점 도달정도지정>

용 어 <목표점 도달정도지정> : 로봇이 목표점에 도달하였는지 확인하는 허용값을 설합니다.
(단위 : 펄스량)

설 명
1) INPOS 명령어를 사용하면 로봇이 이동 명령을 수행한 후 모든 축이 목표점에 허용 정도 이내로 도달하지 않으면 다음명령을 수행하지 않습니다.
2) 로봇 축의 현재 위치값과 목표 교시점과의 차이가 INPOS 명령에서 설정한 펄스(Pulse)량 이내가 되면 도착점에 도달한 것으로 인정하여 다음 명령을 수행합니다.
3) INPOS 명령어 적용해제는 "INPOS 0"를 사용하고, 다음 스텝 명령어부터 적용됩니다.

3.48.1 프로그램 사용 예제

1) INPOS 사용 이동

```
MAIN
VEL 100
JMOV P10
INPOS 100 ..... 도착 완료 허용 값을 100 PULSE로 설정
JMOV P1         P1,P2위치로 이동한 후 각축의 현재값과 티칭된
JMOV P2         값이 100PULSE 이내가 될때까지 대기
INPOS 0   ..... INPOS 명령 적용 해제
JMOV P3
EOP
```

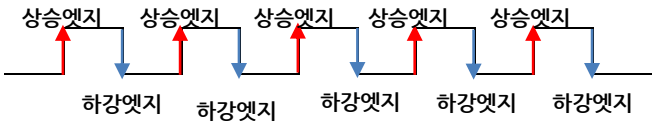
CAUTION

- INPOS 명령을 사용할 경우 현재위치를 정확히 찾기 위해 DELAY가 발생 되어 로봇 이동 Cycle Time이 증가 될 수 있습니다.
- FOS 동작이나 WITH 명령 수행중에는 INPOS 명령이 적용 되지 않습니다.

3.49 MAPPING 명령어

기 능 제어기에 연결된 센서신호를 이용하여 MAPPING동작을 합니다..

용 어 MAPPING이란?
센서에 검출된 데이터(**상승엣지** 시점의 위치 및 물체개수, **하강엣지** 시점의 위치 및 물체 개수)를 N1제어기로 보내는 동작을 의미합니다.
예) 카세트 내에 있는 5개의 웨어퍼를 검출할 때 센서 신호 파형



3.49.1 MINIT(MAPPING 초기화 명령어)

기 능 MAPPING 기능 초기화 명령어입니다.

형 식 MINIT <센서 타입>

용 어 <센서 타입>: MAPPING 기능에 사용되는 센서 타입을 설정 합니다.
MINIT 1: NC 센서인 경우
MINIT 2: NO 센서인 경우

설 명 1) MAPPING관련 제어기 내부 변수를 초기화 합니다.
2) MAPPING 동작시 사용되는 센서 타입을 설정 합니다

CAUTION

- MAPPING 기능은 MINIT, MSTART, MREAD 3개의 명령어가 조합되어 하나의 기능을 수행합니다.
- 정확한 MAPPING 동작을 위해서는 MAPPING 초기화(MINIT), MAPPING 시작(MSTART), MAPPING 데이터 읽기(MREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.49.2 MSTART(MAPPING 검출 명령어)

기 능 MAPPING 기능 시작 명령어입니다.

형 식 MSTART <센서신호 핀> <축 지정> <GP 번호>

용 어 <센서신호 핀> : MAPPING 동작에 사용되는 센서신호 핀 번호를 할당합니다.

<축 지정> : MAPPING 동작 수행 축을 지정합니다.

<GP 번호>: MAPPING 검출 데이터값을 저장할 글로벌 포인트(GP) 시작 번호를 지정합니다,

설 명
1) MAPPING 수행시 사용되는 센서신호 핀 번호와 지정 축 그리고 MAPPING 관련 데이터를 저장할 글로벌 포인트 시작 지점을 설정합니다.
2) MAPPING 동작 중 감지된 상승 엣지와 하강 엣지 수를 글로벌 포인트 시작위치에 저장합니다.
3) MAPPING 검출 가능한 최대 개수는 50개 입니다.

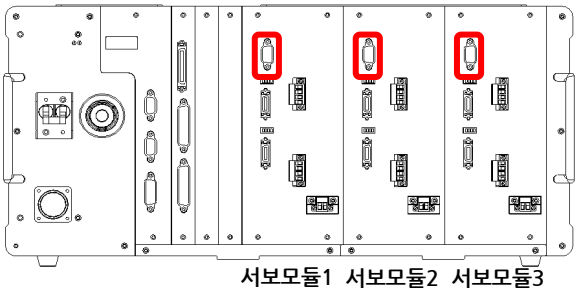
CAUTION

- ▶ MAPPING 기능을 사용하기 위해서는 센서 입력 핀 번호와 축 지정 번호가 동일한 Servo Module에 해당되어야 합니다. 그렇지 않은 경우, MAPPING 기능이 수행되지 않습니다.
- ▶ MAPPING 기능은 MINIT, MSTART, MREAD 3개의 명령어가 조합되어 하나의 기능을 수행합니다.
- ▶ 정확한 MAPPING 동작을 위해서는 MAPPING 초기화(MINIT), MAPPING 시작(MSTART), MAPPING 데이터 읽기(MREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.49.2.1MAPPING 센서 입력 포트 관련

1) MAPPING 기능을 정확하게 사용하기 위해서는 센서신호 번호와 해당되는 서보모듈 축을 사용해야합니다.

2) 각각의 서보 모듈은 **2개의 센서신호 핀**을 갖고 있으며, 아래 표 "센서신호"와 같이 할당 됩니다.



서보모듈1 서보모듈2 서보모듈3

	서보 모듈 1	서보 모듈 2	서보 모듈 3
센서 신호	1	3	5
	2	4	6

서보 모듈 외부 IO	
핀 번호	센서 연결 시 기능
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	P24V
9	센서신호(1, 3, 5)
10	센서신호(2, 4, 6)
11	G24V
12	-
13	-
14	-
15	-

예) 스카라 로봇 Z축 MAPPING 기능 사용 시 MSTART 명령어 설정 방법
일반적으로 스카라 로봇 Z축의 경우 서보 모듈 2에 연결되고, MAPPING 데이터를 GP100부터 저장, 센서는 센서신호3에 연결되는 경우 설정 방법은 아래와 같습니다.

MSTART 3 3 GP100
센서신호 Z축 GPNT INDEX

3.49.3 MREAD(MAPPING 위치 데이터 갱신 명령어)

기 능 MAPPING 데이터 갱신 명령어입니다.

형 식 MREAD <제한 시간>

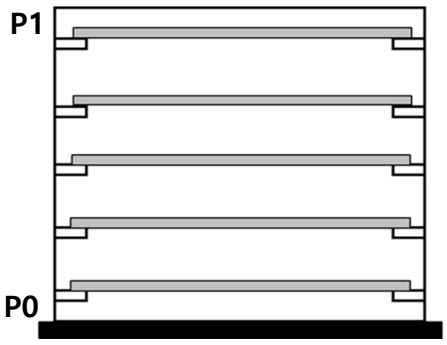
용 어 <제한 시간> : MAPPING 데이터를 읽어오는 제한 시간입니다.(단위 : ms)

설 명 1) MAPPING 센서에서 검출된 위치 데이터를 MSTART에서 설정한 글로벌 포인트 시작번호 다음부터 차례로 저장합니다.
2) 최소 시간 설정은 예상 **검출 포인트 개수 * 20ms** 이상으로 설정 해야 합니다.

3.49.3.1 프로그램 사용 예제

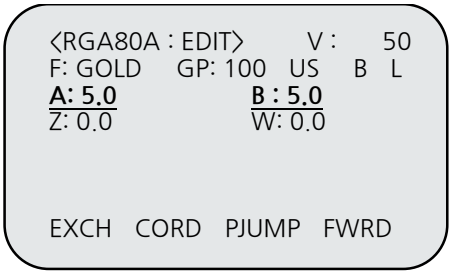
1) 카세트의 글라스 두께 측정

```
MAIN            .....   Program 시작
JMOV P0         .....   대기 위치 P0 이동
MINIT 1         .....   MAPPING 초기화
MSTART 3 3 GP100 .....  센서신호 3번과 Z축 선택 및 MAPPING 데이터는
                        GP100부터 매칭
JMOV P1         .....   P1 위치로 이동
MREAD 500       .....   MAPPING 데이터 GPNT에 저장
EOP                     Program 종료
```



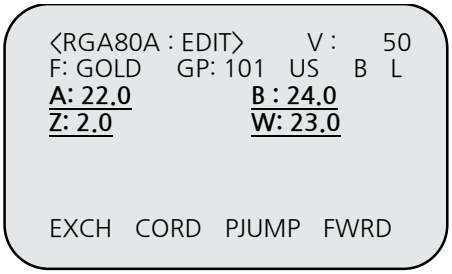
2) MAPPING 데이터 저장 방법 설명(카세트에 글라스가 5개 있는 경우)

상승 하강 엣지 개수 화면



A축은 상승 엣지 수 B축은 하강 엣지 수를 의미합니다.

상승 하강 엣지시 위치



A축 : 상승 엣지시 위치값
B축 : 하강 엣지시 위치값
Z축 : 두 위치 편차
W축 : 두 위치의 평균 위치 값

가) GP100의 첫번째 축에 센서의 상승 엣지 개수 5가 입력되고, 두번째 축에 센서의 하강 엣지 개수 5가 입력됩니다.

나) 그 다음 MAPPING을 수행하고 나서 MREAD 명령을 만나면 GP101의 첫번째 축에 첫번째 상승 엣지가 발생한 위치가 저장되고, GP101의 두번째 축에 첫번째 하강 엣지가 발생한 위치가 저장됩니다.

다) GP101의 세번째 축에는 하강 엣지 발생 위치와 상승 엣지 발생 위치 편차가 저장되고, GP101의 네번째 축에는 하강 엣지 발생 위치와 상승 엣지 발생 위치 합의 평균이 저장됩니다.

CAUTION

- ▶ MAPPING 기능을 사용하기 위해서는 센서신호 핀 번호와 축 지정 번호가 동일한 Servo Module에 해당되어야 합니다. 그렇지 않은 경우, MAPPING 기능이 수행되지 않습니다.
- ▶ MAPPING 기능은 MINIT, MSTART, MREAD 3개의 명령어가 조합되어 하나의 기능을 수행합니다.
- ▶ 정확한 MAPPING 동작을 위해서는 MAPPING 초기화(MINIT), MAPPING 시작(MSTART), MAPPING 데이터 읽기(MREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.50 LATCH 명령어

기 능 제어기에 연결된 센서신호를 이용하여 LATCH동작을 합니다..

용 어 LATCH란?
센서에 검출된 데이터(위치 및 물체 검출개수)를 N1제어기로 보내는 동작을 의미합니다.

3.50.1 NINIT(LATCH 초기화 명령어)

기 능 LATCH 기능 초기화 명령어입니다.

형 식 NINIT <센서 타입>

용 어 <센서 타입>: LATCH 동작에 사용되는 센서 타입을 설정 합니다.
NINIT 1: NC 센서인 경우
NINIT 2: NO 센서인 경우

설 명 1) LATCH 관련 제어기 내부 변수를 초기화 합니다.
2) LATCH 동작시 사용되는 센서 타입을 설정 합니다.

CAUTION

- LATCH 기능은 NINIT, NSTART, NREAD 3개의 명령어가 조합되어 하나의 기능을 수행 합니다.
- 정확한 LATCH 동작을 위해서는 LATCH 초기화(NINIT), LATCH 시작(NSTART), LATCH 데이터 읽기(NREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.50.2 NSTART(LATCH 검출 명령어)

기 능 LATCH 기능 검출 명령어입니다.

형 식 NSTART <센서신호 핀> <축 지정> <GP 번호>

용 어 <센서신호 핀>: LATCH 동작에 사용되는 센서신호 핀 번호 할당합니다.

<축 지정> : LATCH 동작 수행 축을 지정합니다.

<GP 번호> : LATCH 검출 데이터값을 저장할 글로벌 포인트(GP) 시작 번호 지정합니다.

설 명 1) LATCH 수행시 사용되는 센서신호 핀 번호와 지정 축 그리고 LATCH 관련 데이터를 기록 할 글로벌 포인트지점을 설정합니다.

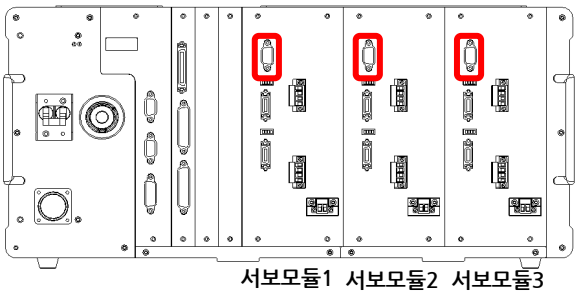
2) LATCH 동작 중 감지된 상승엣지 개수와 입력한 축의 위치를 지정한 글로벌 포인트에 저장합니다.

⚠ CAUTION

- ▶ LATCH 기능은 NINIT, NSTART, NREAD 3개의 명령어가 조합되어 하나의 기능을 수행 합니다.
- ▶ 정확한 LATCH 동작을 위해서는 LATCH 초기화(NINIT), LATCH 시작(NSTART), LATCH 데이터 읽기(NREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.50.2.1LATCH 센서 입력 포트 관련

1) LATCH 기능을 정확하게 사용하기 위해서는 센서신호 번호와 해당되는 서보모듈 축을 사용해야 합니다.
2) 각각의 서보 모듈은 **2개의 센서신호 핀**을 갖고 있으며, 아래 표의 "센서신호"와 같이 할당 됩니다.



서보모듈1 서보모듈2 서보모듈3

	서보 모듈 1	서보 모듈 2	서보 모듈 3
센서 신호	1	3	5
	2	4	6

서보 모듈 외부 IO	
핀 번호	센서 연결 시 기능
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	P24V
9	센서신호(1, 3, 5)
10	센서신호(2, 4, 6)
11	G24V
12	-
13	-
14	-
15	-

예) 트랜스퍼 로봇 R축 LATCH 기능 사용 시 NSTART 명령어 설정 방법
일반적으로 트랜스퍼 로봇 R축의 경우 서보 모듈 2에 연결되고, LATCH 데이터를 GP100부터 저장, 센서는 센서신호3 에 연결되는 경우 설정 방법은 아래와 같습니다.

NSTART 3 3 GP100
센서신호 Z축 GPNT INDEX

3.50.3 NREAD(LATCH 위치 데이터 갱신 명령어)

기 능 LATCH 데이터 갱신 명령어입니다.

형 식 NREAD <제한 시간>

용 어 <제한 시간> : LATCH 데이터를 읽어오는 제한 시간입니다.(단위 : ms)

설 명 1) LATCH 센서에서 검출된 위치 데이터를 NSTART에서 설정한 글로벌 포인트 시작번호 다음부터 차례로 저장합니다.
2) 최소 시간 설정은 예상 **검출 포인트 개수 * 20ms** 이상으로 설정 해야 합니다.

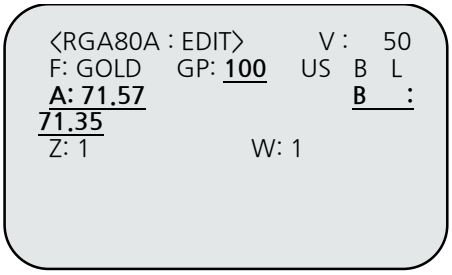
3.50.4 프로그램 사용 예제

1) 입력센서의 LATCH된 위치 값 측정

```
MAIN            .....   Program 시작
JMOV P0         .....   대기 위치 P0 이동
NINIT 1         .....   LATCH 초기화
DALY 5          .....   50ms 지연시간
NSTART 3 3 GP100 .....  LATCH 데이터(센서신호3, Z축)를 GP100변수에 저장
NSTART 4 4 GP100 .....  LATCH 데이터(센서신호4, W축)를 GP100변수에 저장
JMOV P1         .....   P1 위치로 이동
NREAD 500       .....   LATCH 데이터 GPNT에 저장
EOP                     Program 종료
```

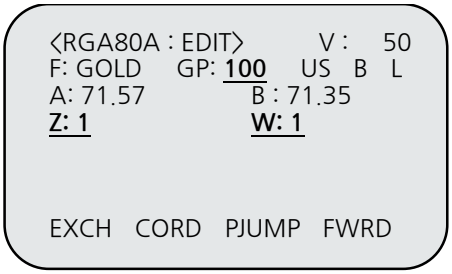
2) LATCH 데이터 저장 방법 설명(센서 2개 사용 및 GP변수를 100으로 설정한 경우)

위치데이터 저장화면



A축 : 센서신호3 LATCH 시의 위치 값
B축 : 센서신호4 LATCH 시의 위치 값

물체 검출 개수 데이터 화면



Z축 : 센서신호3 LATCH 시 검출 개수
W축 : 센서신호4 LATCH 시 검출 개수

가) 위의 예에서 NREAD시 GP100의 A축에 센서신호3의 LATCH된 위치가 입력되고, B축에 센서신호4의 LATCH된 위치가 입력됩니다.

나) GP100의 Z축에 센서신호3의 물체 검출 개수가 저장되고, GP100의 W축에 센서신호4의 물체 검출 개수가 저장됩니다.

다) GP100에 센서신호3, 센서신호4에 대한 위치 및 물체 검출 개수가 저장됩니다.

CAUTION

- LATCH 기능은 NINIT, NSTART, NREAD 3개의 명령어가 조합되어 하나의 기능을 수행 합니다.
- 정확한 LATCH 동작을 위해서는 LATCH 초기화(NINIT), LATCH 시작(NSTART), LATCH 데이터 읽기(NREAD) 순서로 진행해야 합니다. 그렇지 않은 경우 "Latch Sequence Err" 알람이 발생 합니다.

3.51 변수

3.51.1 변수의 종류

1) 변수는 일반 변수, POSITON 변수, 시스템 변수로 나뉘어 집니다.

		LOCAL	GLOBAL
일반 변수	정수형	INT	I(번호)
	실수형	REAL	F(번호)
POSITION 변수		POS, P(번호)	GP(번호)
시스템 변수		CNT,TMR,MVR,HERE	

2) LOCAL 변수 : 현재 구동중인 프로그램 내에서만 사용 가능 한 변수입니다.
   가) INT : 정수형 변수로 산술연산, 입/출력 DATA 임시저장 등에 사용합니다.
   나) REAL : 실수형 변수로 산술 연산, POINT DATA 연산 등에 사용합니다.
   다) POS : 위치형 변수로 단순 변수와 배열 변수로 구분합니다.
   라) CNT : 입력 포트를 통해 들어오는 펄스 입력을 업카운트(UP COUNT) 하여 저장합니다.
   마) TMR : 파라미터 설정값에 따라 초기 화면 값에서 1씩 증가합니다.
   바) MVR : 두 POINT 사이를 이동 하는 경우,전체 거리에 대한 현재 위치의 백분율 입니다.
   사) HERE : 로봇 현재 위치 값을 갖고 있는 변수입니다.
   아) P(번호) : 사용자가 티칭한 위치 값을 가지고 있는 변수입니다.

3) GLOBAL 변수 : 전체 프로그램 및 채널에서 공통으로 사용 가능 한 변수입니다.
   가) I : 정수형 변수로 INT로 선언된 LOCAL 변수와 동일한 기능입니다.
   나) F : 실수형 변수로 REAL로 선언된 LOCAL 변수와 동일한 기능입나다.
   다) GP : 사용자가 티칭한 위치 값을 가지고 있는 변수입니다.

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Global INT	0 ≤ 번호 ≤ 499	0 ≤ 번호 ≤ 1999
Global FLOAT	0 ≤ 번호 ≤ 499	(0 ≤ 번호 ≤ 1999
Local Point	0 ≤ 번호 ≤ 1999	0 ≤ 번호 ≤ 1999
Global Point	0 ≤ 번호 ≤ 1023	0 ≤ 번호 ≤ 14999

3.51.2 사용 방법

1) 사용위치
   가) 프로그램에서 사용하는 일반변수는 반드시 프로그램 시작부분인 "MAIN" 다음에 선언
   나) 부 프로그램(Subroutine)이나 프로그램의 시작부분 이외에서는 선언할 수가 없습니다.

2) 구성
   가) 일반변수의 변수명은 영문자와 숫자로 구성된 8자리의 문자 스트링입니다.
   나) 변수의 첫자는 반드시 영문자이거나 '_'입니다.(숫자 사용 불가)
   (단,"P", "GP", "I", "F" 등은 POINT변수 및 GLOBAL변수와 중복되므로 **사용할 수 없습니다**.)

3) 초기화
   가) 변수는 초기화 후 사용해야 합니다.
   나) CNT, TMR 변수는 변수사용 이전 STEP에서 초기화를 수행 해야 합니다.
   다) MVR 변수는 MOVE 명령어가 시작되면 0으로 초기화 됩니다.

3.52 정수형(INT, I, II), 실수형 (REAL, F) 변수

3.52.1 LOCAL 변수

기 능 프로그램 내부에서 사용 가능한 정수형, 실수형 변수의 선언합니다.

형 식 INT <변수명>,<변수명>,---
REAL <변수명>,<변수명>,---

용 어 <변수명> :
영문자와 숫자로 구성된 8자리의 문자 스트링입니다.
변수의 첫자는 반드시 영문자 이거나 '_'입니다.

설 명 1) **MAIN 블록의 시작 위치**에서 선언해야 하고, 부 프로그램내에서는 선언할수 없습니다.
2) INT로 선언된 변수는 정수형 변수입니다.(정수값 범위 : $-2.14 \times 10^{9} \sim 2.14 \times 10^{9}$)
3) REAL로 선언된 변수는 실수형 변수입니다.(실수값 범위 : $\pm 8.4 \times 10^{-37} \sim \pm 3.4 \times 10^{38}$)

3.52.2 프로그램 사용 예제

1) 정수형, 실수형 변수 사용

```
MAIN
INT J, BDOUT ..... 정수형 변수 J 와 BDOUT를 선언
REAL FF1 ..... 실수형 변수 FF1을 선언
WHILE 1
BDOUT=0H000F
J=1 ..... 변수 초기화
FF1=10.5
FOR J=0 TO 10 ..... AP1 위치 좌표로 이동
JMOV P1 ..... AP2 위치 좌표로 이동
JMOV P2
POUT0=BDOUT ..... 정수형 변수 BDOUT을 사용 하여 출력
NEXT
FF1=FF1+100.0 실수형 변수 산술 연산
ENDWL
EOP
```

CAUTION

- "P", "GP", "I", "F" 등은 POINT 변수 및 GLOBAL 변수와 중복되므로 사용할 수 없습니다.
  ("Syntax Error" , "Duplicated Symbol" 등 ALARM 발생)
- 실수형 변수와 정수형 변수의 연산을 혼합 사용시 정수형으로 변경 됩니다.
  9.5 + 10 = 19 (실수와 정수 연산시 오차 발생)
  9.5 + 10.0=19.5 (실수 + 실수 연산으로 변경 사용)

3.52.3 정수형 GLOBAL 변수 I, II

기 능 INT로 선언된 정수형 변수와 동일한 기능을 가지며 전체 프로그램에서 공통으로 사용 가능한 변수 입니다.

형 식 I<정수> : 사용 가능한 정수의 범위는 0~499 까지 입니다.
II<정수> : 사용 가능한 정수의 범위는 0~499 까지 입니다.
→ I와 <정수>, II와 <정수> 사이에는 공란(BLANK)이 있으면 안됩니다.
→ I와 (정수), I와 (정수변수) 사이에는 공란(BLANK)이 있으면 안됩니다.

설 명
1) LOCAL 변수와 달리 **프로그램에서 선언하지 않고** 사용합니다.
2) II<정수>는 **I<정수>를 중복**해서 사용한 것과 동일합니다..
예) I35 가 20 이라면 II35 는 I20 과 동일합니다.
3) 프로그램에서 직접 I<정수> 나 II<정수>, I(정수), I(변수)의 값을 변경할 수 있습니다.
4) VEL, ACC, DEC, FOS, PFOS, PLUP, DLAY, INPOS 명령에 사용할 수 있습니다.
예) VEL I20 : I20 에 저장된 값으로 속도(Velocity)를 설정합니다.
5) IF - (ELSE) - ENDIF 블록과 WHILE-ENDWL 블록에서 사용할 수 있습니다.
예) IF I7 > A0 THEN 예) WHILE II49 ≤ 100
※ FOR-NEXT 블록에서는 사용할 수 없습니다.
6) I<정수>형은 포인트 전역변수(Global Point Variable)에 사용할 수 있습니다.
예) I9 가 3 이라면 PI9 는 P3 과 동일합니다.
※ PII<정수>형태는 사용할 수 없습니다.

3.52.4 프로그램 사용 예제

```
MAIN
INT L,M,H          ..... 정수형 변수 L,M,H를 선언
L=I0               ..... L=300 할당
M=I1               ..... M=600 할당
H=I2               ..... H=1000 할당
VEL H              ..... 속도를 H(1000) 으로 지정
JMOV P0
WHILE I10<I11      ..... I10<I11 을 만족하는 동안
FOS I4             ..... FOS I4 => FOS 10
LMOV PI25          ..... P(I25) 로 CP 이동
VEL L              ..... 속도를 L(300) 으로 지정
AMOV PI26 PI27     ..... P(I26) 과 P(I27)을 경유하는 원호 보간
LMOV PI28          ..... P(I28) 로 CP 이동
I25=I25+4
I26=I26+4          ..... 1회 이동 후 I25,26,27의 값을 4씩 증가
I27=I27+4
I10=I10+1          ..... 카운터 I10을 1증가
ENDWL
EOP
```

I 변수 값
I0 = 300
I1 = 600
I2 = 1000
I4 = 10
I10 = 1
I11 = 4
I25 = 1
I26 = 2
I27 = 3
I28 = 4

3.52.5 실수형 GLOBAL 변수 F

기 능 REAL로 선언된 실수형 변수와 동일한 기능을 가지며 전체 프로그램에서 공통으로 사용 가능한 변수 입니다.

형 식 F<정수> : 사용 가능한 정수의 범위는 0~499 까지 입니다.
→ F와 <정수> 사이에는 공란(BLANK)이 있으면 안됩니다.

설 명 1) LOCAL 변수와 달리 **프로그램에서 선언하지 않고** 사용합니다.
2) 프로그램에서 직접 F<정수>의 값을 변경할 수 있습니다.
3) IF - (ELSE) - ENDIF 블록과 WHILE-ENDWL 블록에서 사용할 수 있습니다.
예) IF F7 > A0 THEN 예) WHILE F49 ≤ 100
※ FOR-NEXT 블록에서는 사용할 수 없습니다.

3.52.6 프로그램 사용 예제

```
MAIN
POS XP
XP=P1
FOR I0=0 TO 10
XP.1=F0 ..... 위치형 변수 XP1 의 X축(1번째축)에 F0의 값을 대입
LMOV XP ..... XP의 위치로 CP 이동
F0=F0+10.0 ..... F0 값을 10.0씩 증가
NEXT
EOP
```

CAUTION

▶ 실수형 변수와 정수형 변수의 연산을 혼합 사용시 정수형으로 변경 됩니다.
9.5 + 10 = 19 (실수와 정수 연산시 오차 발생)
9.5 + 10.0=19.5 (실수 + 실수 연산으로 변경 사용)

3.53 POSITION 변수

3.53.1 POS 변수

기 능 위치형 변수 선언 (단순변수 및 배열변수)

형 식 단순변수 → POS <변수명>, <변수명>, ...
배열변수 → POS <변수명>(크기), {<변수명>(크기)}, ...

용 어 <변수명> :
변수명은 **영문자와 숫자로 구성**된 8자리의 문자 스트링 및 배열의 크기입니다.
변수의 **첫자는 반드시 영문자** 입니다.("P", "GP" ,"I", "F" 문자 단독사용 제외)

(크기) : 배열변수의 크기는 괄호 안의 숫자(정수)로 나타냅니다.
배열 변수의 선언의 괄호 안의 숫자는 **방의 개수**를 나타냅니다.
예) POS AA(3) 선언시 – 사용 가능한 변수는 AA(0), AA(1), AA(2) 3개.

설 명
1) **MAIN 블록의 시작**에서 선언하여야 하고, 부 프로그램내에서는 선언할 수 없습니다.
2) POS로 선언된 변수값은 각축의 각도, 리드값 또는 직교 좌표값입니다.
   -. **X로 시작**되는 변수값(예: XA, XA1, ...) → 직교 좌표값 변수
   -. **X이외의 문자**로 시작되는 변수 → 각축의 각도, 리드값 변수, 각축의 증분량변수
3) POS로 선언된 변수의 각 **성분값**은 **<변수명>.<번호>**로 사용할 수 있습니다.
   예) POS XP1 → XP1.1 XP1.2 XP1.3 XP1.4...
4) 각도값 POS 변수는 **최대 6가지 성분**을 가지며,
   변수명(단순변수, 배열변수)=<1축성분, 2축성분, 3축성분, 4축성분, 5축성분, 6축성분>
   형식으로 초기화할 수 있습니다.
5) 직교 좌표값 POS 변수는 **최대 7가지 성분**을 가지며,
   변수명(단순변수, 배열변수)=
   <X축성분, Y축성분, Z축성분, 4축성분, 5축성분, 6축성분, Arm Form값>
   형식으로 초기화할 수 있습니다. → 기계부가 SCARA 로봇일 경우만 적용됩니다.
   (Arm Form 값 : 0 → LEFT form, 1 → RIGHT form, 2 → NO form)
6) 각도값 POS변수와 직교좌표값 POS변수는 서로 연산 및 지정이 가능하며, 이 경우 저장되는 값은 형식에 따라 변환됩니다.

CAUTION

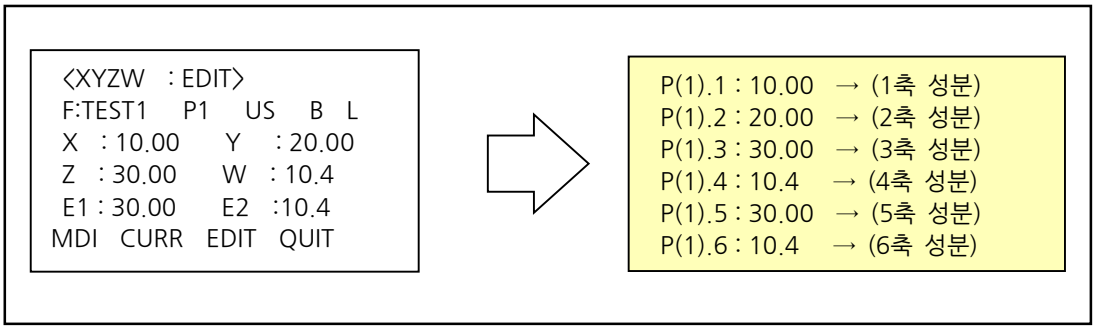
- "P", "GP", "I", "F" 등은 POINT 변수 및 GLOBAL 변수와 중복되므로 사용할 수 없습니다. ("Syntax Error" , "Duplicated Symbol" 등 ALARM 발생)
- 성분값에 ARM FORM이 없는 경우 각도값으로 판단되어 XP변수에 담을 때 XY좌표계로 변환됩니다.
- 5, 6번축은 5축 이상 제어 사용 시 확장성을 고려한 사용 하지 않는 축이며, 4축이하 제어시 성분 값은 '0.0'을 입력하셔야 합니다.

3.53.2 POINT 변수

기 능 Job Editor에서 사용자가 Teaching한 모든 Point를 사용합니다.

형 식 LOCAL변수 → P(번호) (0 ≤ 번호 ≤ 1999)
GLOBAL변수 → GP(번호) (0 ≤ 번호 ≤ 1023)

설 명 1) POS변수와 달리 초기에 변수 선언을 하지 않습니다.
2) POINT 변수의 각 **성분값**은 **<포인트번호>.<번호>**로 사용할 수 있습니다.



3.53.3 프로그램 사용 예제

1) POINT변수, POSITION변수를 활용한 감속 포인트 적용

```
MAIN
POS MM,XA,XB                ..... 위치형 변수 MM, XA, XB를 선언
MM=<10.1,10.2,10.3,10.4,0,0> ..... MM 위치 변수 초기화
XA=<400.0,50.0,10.0,0.0,0.0,0.0,1> ..... XA 위치 변수 초기화
XB=XA                       ..... XB 변수 값을 XA 값와 동일하게 초기화
XB.3=XA.3-40.0              ..... XB의 Z(3번째축)값을 40mm 감산
VEL 1000
JMOV P0                     ..... P0 위치로 PTP 이동
LMOV XB                     ..... XB 위치 좌표로 CP 이동 (감속 포인트)
VEL 300
LMOV XA                     ..... XA 위치 좌표로 CP 이동
IMOV2 MM                    ..... 위치형 변수 MM값 만큼 증분 이동
VEL 1000
JMOV P0
EOP
```

CAUTION

▶ 변수명에 따라 성분값이 변환되어 저장됩니다.
XP=<각도값> : 입력 성분값이 직각좌표계로 변환되어 저장됩니다.
XP=<직각좌표값>: 입력 성분값인 직각 좌표계로 저장됩니다.
AP=<각도값>: 입력 성분값인 각도값으로 저장됩니다.
AP=<직각좌표값>: 입력 성분값이 각도값으로 변환되어 저장됩니다.

3.54 시스템 변수(CNT, TMR, MVR, HERE)

3.54.1 CNT, TMR 변수

기 능 CNT : 카운터값을 저장합니다.
TMR : 타이머값을 저장합니다.

형 식 CNT<펄스입력 BIT 번호>=<초기값>
TMR0=<초기값>
TMR1=<초기값>

용 어 <펄스입력 BIT 번호> : 입력 받고자 하는 INPUT BIT 번호입니다.
<초기값> : 카운터(CNT), 타이머(TMR)가 시작되는 정수값입니다.

설 명 1) 시스템 변수는 **자료형을 선언하지 않고** 사용합니다.
2) CNT 변수는 **펄스 입력 BIT가 결정되는 순간 "0"이 저장**되며
이 후 **매 펄스입력(10m 이상의 펄스)을** 카운트합니다. (0 ≤ 입력범위 ≤ 65,535)
3) TMR변수는 **정수값을 입력하는 순간**부터 값이 할당되어 시스템 파라미터에서
정의된 시간간격으로 1씩 증가합니다.
설정 값 적용은 **파라미터 모드의 TMR을 참조** 하시기 바랍니다.

3.54.2 프로그램 사용 예제

1) CNT, TMR 변수 사용

```
MAIN
VEL 100
CNT0=0          .....   카운터 초기화
TMR1=-100       .....   타이머 초기화
WHILE CNT0<20   .....   입력 BIT "0"번에 펄스가 20개 입력될때까지
                        LOOP 실행
JMOV P0
IF(TMR1>0) THEN .....   타이머가 1초 이상 이면
GOTO TEST               (-100 ~> 0 : 1ms가 100개 발생)
ENDIF
DLAY 100
JMOV P1
ENDWL
LABL TEST
EOP
```

CAUTION

▶ CNT와 펄스입력 포트번호 사이와 TMR와 "0" 또는 "1" 사이에 공란(BLANK)이 들어가지 않도록 주의 하시기 바랍니다. 예) CNT_0=0 (X), TMR_1=-100 (X)

3.54.3 MVR 변수

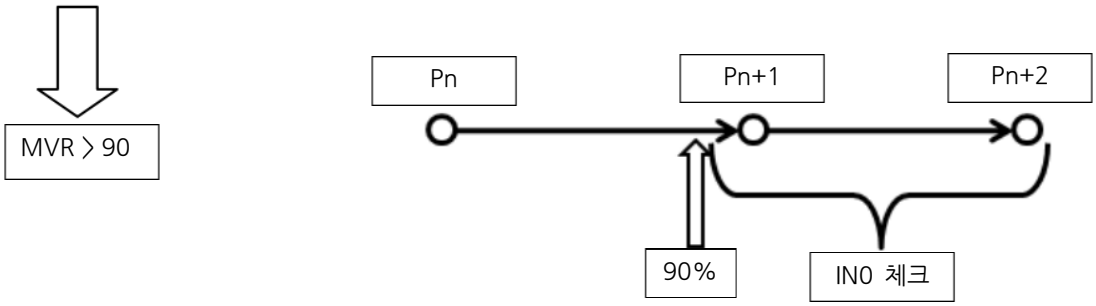
기 능 전체 이동거리에 대한 이동거리 비율을 설정합니다.

형 식 MVR <전체 이동거리에 대한 이동거리의 백분율값>

설 명
1) 두 Point(Pn, Pn+1)사이를 이동하는 경우 전체 이동거리에 대한 백분율 값입니다.
(즉, MVR = (현재 로봇 이동거리/로봇 전체이동거리)*100)
2) 로봇 동작중 조건분기 및 I/O병렬 처리에 사용합니다.
3) 이동구간을 100단위로 나누어 반드시 WITH문과 병행하여 사용합니다.
4) 두 Point (Pn, Pn+1)의 이동거리가 짧고 고속으로 운전할 때는 MVR값 설정에 유의 하시기 바랍니다.

CAUTION

▶ Pn Point에서 Pn+1 Point로 이동 중 90% 이후에 입력 "IN1"를 확인하는 경우



1. 로봇이 Pn+1위치의 90%정도 도달하였을 때는 다음 이동 POINT(Pn+2)로 이동을 위한 연산을 시작합니다.
2. 위의 그림과 같이 로봇이 90%위치에 있어도 **현재 #1위치를 연산중**입니다.
3. 이동거리가 짧고 고속으로 운전할 때에는 Pn출발 후 90% 이후에는 입력 Bit'0'의 상태를 확인할 수 없는 경우도 발생합니다. (이 경우 로봇 이동 속도를 낮추어 사용 하시기 바랍니다.)

▶ 로봇 이동 중 신호가 입력 시 현재좌표 READ(HERE 명령어) 기능 사용

1. 로봇 이동 중 신호가 입력 되었을 때 정지 후 현재좌표 READ기능은 로봇이동 속도에 따라 현재좌표 READ한 값에 편차가 발생할 수 있습니다. (이 경우 로봇의 이동 속도를 낮추어 사용 하시기 바랍니다.)

3.54.4 프로그램 사용 예제

1) 로봇 이동 중 입력Bit 상태에 따라 이동 궤적 수정.

```
MAIN
VEL 100
FOS 10
JMOV P100
MVR=0                                  이동거리 비율 초기화
WITH
JMOV P0
WHILE MVR<100                  ....... P0포인트를 100% 이동중
IF MVR>10 THEN                 ....... 이동거리가 10% 위치에서
OUT1=1 50                      ....... 출력을 0.5초 동안 ON(1)으로 유지
ENDIF
IN1=1
GOTO L1
ENDWL
LABL L1
STOP
ENDWT
JMOV P1
EOP
```

2) P1 포인트 출발하여 80%이후에 입력 "IN0"의 신호입력에 따라 P2 포인트로 이동

```
MAIN
VEL 100
WHILE IN1==1
PLUP 10
JMOV P0
MVR=0                                  이동거리 비율 초기화
WITH
JMOV P1
WHILE MVR<100                  ....... P1포인트를 100% 이동중
IF (MVR>80)&&(IN0==1)          ....... 이동거리가 80% 이고, IN0가 ON(1)된 경우
THEN
PLUP 0
JMOV P22
GOTO BB
ENDIF
ENDWL
ENDWT
IN0=1
PLUP 0
JMOV P2
LABL BB
ENDWT
OUT1=1 50
ENDWL
EOP
```

3.54.5 HERE 변수

기 능 현재 위치 값을 저장하는 변수입니다.

형 식 HERE
HERE<정수>

용 어 <정수> : 채널 번호 (1 또는 2)
채널간 로봇의 현재 위치 값을 읽어올 경우 사용 합니다.

설 명 1) 로봇의 현재 위치 값을 각도값 또는 리드값으로 저장
2) 다른 채널에 있는 로봇의 현재 위치를 저장.

3.54.6 프로그램 사용 예제

1) HERE 변수를 활용하여 현재 위치를 READ

```
MAIN
VEL 100
POS CURR
JMOV P0
MVR=0
WITH
JMOV P1
WHILE MVR<100
IF MVR>50 THEN                    P1 포인트로 이동 중 50% 지점에서
CURR=HERE            .....        현재 위치값을 위치형 변수 CURR에 저장
GOTO LLL
ENDIF
ENDWL
LABL LLL
ENDWT
JMOV P2
JMOV CURR
EOP
```

2) HERE<n>변수를 활용하여 다른 채널의 로봇 현재위치를 READ

```
MAIN
VEL 100
POS AP
JMOV P0
WHILE AP.1<10.0
AP=HERE2             .....        2번 채널 로봇의 현재위치를 저장
DLAY 10
ENDWL
OUT0=1
JMOV P1
EOP
```

3.55 RSTATE (로봇 상태 확인 명령어)

기 능 로봇의 상태를 읽어오는 명령어입니다.

형 식 <정수형변수>=RSTATE(<채널>, <STATE INDEX>)

설 명

INDEX	STATE NAME	내용
0	ALL	상태 정보 전체를 반환
1	ALARM	알람 상태 반환
2	READY	구동 준비 상태 반환
3	ORIGIN	원점 완료 상태 반환 (ABS Type 일경우 항상 ON)
4	INPOSITION	목표위치 도달 완료
5	RUN(JOB 구동중)	JOB RUNNING 상태 반환
6	SERVO ON	SERVO MOTOR ON/OFF 상태 반환
7	STOP(JOB 정지중)	JOB STOP 상태 반환
8	RUN MODE	MANUAL RUN 구동상태 반환
9	SYSTEM MODE	SYSTEM 모드 구동상태 반환
10	TEACH MODE	MANUAL RUN MODE 진입 상태 반환
11	EMG	비상 정지 발생시 ON
12	PSEL	JOB LOAD 완료.
13	EECH	EECH 명령어 상태 반환
14	Reserved	예약된 영역 입니다.
15	MOVING(로봇 구동중)	해당 채널의 Robot 이동 중 상태 반환
16	TIMEOUT	String command Time out 상태 반환
17	1 AXIS MOVING FLAG	해당 채널의 첫번째 축 이동 중 상태 반환
18	2 AXIS MOVING FLAG	해당 채널의 두번째 축 이동 중 상태 반환
19	3 AXIS MOVING FLAG	해당 채널의 세번째 축 이동 중 상태 반환
20	4 AXIS MOVING FLAG	해당 채널의 네번째 축 이동 중 상태 반환
21	5 AXIS MOVING FLAG	해당 채널의 다섯번째 축 이동 중 상태 반환
22	6 AXIS MOVING FLAG	해당 채널의 여섯번째 축 이동 중 상태 반환
23	Reserved	예약된 영역 입니다.
.....	.....	
31	Reserved	
32	1 AXIS SENSOR ALIGN	해당 채널의 첫번째 축 센서 ALIGN 상태 반환
33	2 AXIS SENSOR ALIGN	해당 채널의 두번째 축 센서 ALIGN 상태 반환
34	3 AXIS SENSOR ALIGN	해당 채널의 세번째 축 센서 ALIGN 상태 반환
35	4 AXIS SENSOR ALIGN	해당 채널의 네번째 축 센서 ALIGN 상태 반환
36	5 AXIS SENSOR ALIGN	해당 채널의 다섯번째 축 센서 ALIGN 상태 반환
37	6 AXIS SENSOR ALIGN	해당 채널의 여섯번째 축 센서 ALIGN 상태 반환

3.55.1 프로그램 사용 예제

1) CH1의 상태를 모두 확인 하여 I/O로 개별 출력

```
MAIN
INT TEMP
TEMP=RSTATE(1,0)          .....   CH1의 상태 전체 확인
OUT10=(TEMP>>5)&0H01      .....   SVON 상태 출력
OUT9=(TEMP>>4)&0H01       .....   JOB RUN 상태 출력
OUT8=TEMP&0H01            .....   Alarm 상태 출력
EOP
```

2) CH2의 상태를 개별로 읽어 I/O로 출력

```
MAIN
OUT10= RSTATE(2,6)        .....   CH2 SVON 상태 출력
OUT9= RSTATE(2,5)         .....   CH2 JOB RUN 상태 출력
OUT8= RSTATE(2,1)         .....   CH2 Alarm 상태 출력
EOP
```

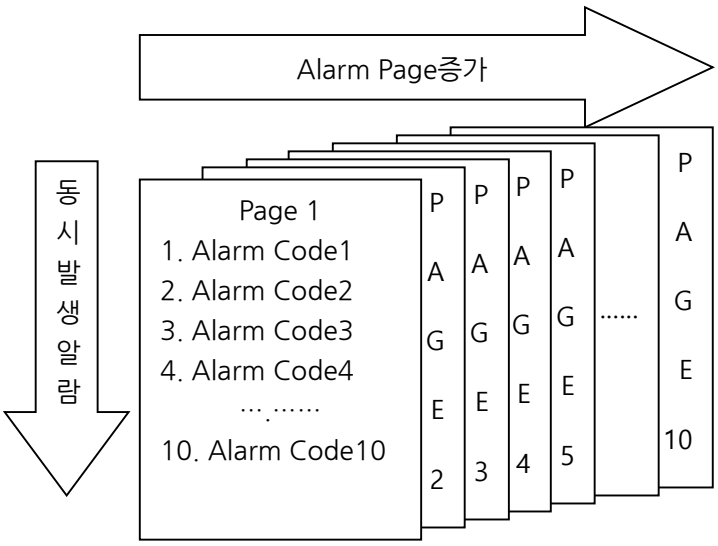
3.56 RERROR(알람 코드 확인 명령어)

기 능 로봇의 알람 코드를 읽어오는 명령어입니다.

형 식 <정수형변수>=RERROR(<PAGE NO>, <INDEX>)

용 어 <PAGE NO> : 알람 이력 번호
Alarm History의 이력 번호이며, 하나의 Page에 10개의 Index를 저장합니다.
(Page번호가 1번이면 가장 최근에 발생한 이력 이며, 10번이 가장 오래된 이력 번호 입니다.)

<INDEX> : 동시 발생된 알람 테이블 번호
동시에 발생될수 있는 Alarm Code는 최대 10개까지 입니다.
(Alarm code가 없을 경우에는 0을 반환 합니다.)



3.56.1 프로그램 사용 예제

1) Alarm Code Read

```
MAIN
INT ERR,ERR1
WHILE 1
ERR=RERROR(1,1)        ..........   Page1 의 첫번째 Alarm code 확인
IF ERR != 0 THEN       ..........   Alarm Code가 있으면
ERR1=RERROR(1,2)       ..........   Page1 의 두번째 Alarm code 확인
ENDIF
ENDWL
EOP
```

3.57 GFTOGP(GF를 GP에 저장하는 명령어)

기 능 GF(Global Float)의 값을 GP(Global Point) 변수에 저장하는 명령어입니다.

형 식 GFTOGP <Global Float index> <Global Point index>

용 어 <Global Float Index> : 0~499
<Global Point Index> : 0~1023

	RO(일반 로봇 소프트웨어)	TR(반도체 이송용 로봇 소프트웨어)
Global Float	500EA(0~499)	2000EA(0~1999)
Global Point	1024EA(0~1023)	15000EA(0~14999)

설 명 1) 저장된 GF 변수의 시작 번호와 저장할 GP 번호를 지정하면 시작 번호로부터 6개의 GF Data가 지정한 GP에 저장됩니다.

3.57.1 프로그램 사용 예제

1) 로봇 현재 위치를 GP(Global Point)변수에 저장

```
MAIN
POS CURR
CURR=HERE ..... 로봇의 현재 위치 좌표 Read
F10=CURR.1 ..... 로봇 첫번째축 좌표 F10번에 저장
F11=CURR.2 ..... 로봇 두번째축 좌표 F11번에 저장
F12=CURR.3 ..... 로봇 세번째축 좌표 F12번에 저장
F13=CURR.4 ..... 로봇 네번째축 좌표 F13번에 저장
F14=CURR.5 ..... 로봇 다섯번째축 좌표 F14번에 저장
F15=CURR.6 ..... 로봇 여섯번째축 좌표 F15번에 저장
GFTOGP F10 GP50 ..... F10~F15값을 GP50번에 저장
EOP
```

3.58 REMCMD(SYSTEM COMMAND)

기 능 System Command 명령어입니다.

형 식 REMCMD <채널> <Command>

용 어 <채널>: Command(명령)를 수행할 채널입니다.
<Command>:

Command	명칭	설명
1	Servo ON	해당 채널의 로봇을 Servo ON 시킨다.
2	Servo OFF	해당 채널의 로봇을 Servo OFF 시킨다.
3	PSEL	선댁된 JOB 프로그램 LOAD 시킨다.
4	START	선댁된 JOB을 구동시킨다.
5	STOP	구동중인 JOB을 정지시킨다.
6	RESET	알람을 클리어 시킨다.
7	EMG STOP	로봇을 비상정지 시킨다.

3.58.1 프로그램 사용 예제

1) REMCMD를 사용 한 CH1의 로봇 JOB 수행

```
MAIN
INT CHECK
REMCMD 1 3 ..... CH1에 JOB 파일을 LOAD 합니다.
CHECK=RSTATE(1,12) ..... 선택된 JOB LOAD 완료 정보 확인
IF CHECK==1 THEN
REMCMD 1 1 ..... CH1의 로봇을 Servo ON 시킴
CHECK=RSTATE(1,6) ..... CH1의 SERVO ON/OFF 상태 정보 확인
IF CHECK==1 THEN
REMCMD 1 4 ..... CH1의 JOB을 RUN 시킨다.
ENDIF
ENDIF
EOP
```

3.59 상수

기 능 16진수 및 2진수로 표시합니다.

형 식 0H<숫자>
0B<숫자>

용 어 <숫자> : 숫자 앞에 0H 또는 0B가 없으면 10진수입니다.
0H : 16진수 입니다.
0B : 2진수 입니다.

설 명 1) 0H<숫자>는 16진수이며, 숫자는 8자리(16비트)까지 가능합니다.
2) 0B<숫자>는 2진수이며, 숫자는 16자리(16비트)까지 가능합니다.
3) 0H나 0B가 없으면, 10진 정수 또는 실수입니다.

3.59.1 프로그램 사용 예제

```
MAIN
INT AA
VEL 100
WHILE 1
AA=PIN0 & 0H00FF                入력포트0번의 값을 16진수 00FF 와 비교
IF AA==1 THEN
POUT0=0B00110100                출력포트0번에 2진수 00110100 출력
JCALL MD1
ENDIF
IF AA==1 THEN
POUT0=0B01110100                출력포트0번에 2진수 01110100 출력
JCALL MD2
ENDIF
ENDWL
EOP
```

3.60 연산자

3.60.1 배정 연산자

명령어	기 능
=	오른쪽 수식을 평가하여 그 결과를 왼쪽변수의 변수형에 맞게 변환하여 **배정**합니다. (예: IN0=1)

왼쪽 \ 오른쪽	정수형	실수형	위치 형	카운터형	타이머형
정수형	정 수 형	정 수 형	불 가	정 수 형	정 수 형
실수형	실 수 형	실 수 형	불 가	실 수 형	실 수 형
위치형	불 가	불 가	위 치 형	불 가	불 가
카운터형	정 수 형	정 수 형	불 가	정 수 형	정 수 형
타이머형	정 수 형	정 수 형	불 가	정 수 형	정 수 형

3.60.2 산술 연산자

명령어	기 능
*/,+,-,%	서로 다른 자료형에 대한 각 연산자의 연산결과는 다음표와 같습니다. 표에서 '불가'는 **연산이 이루어질수 없는 수식**의 경우를 말합니다.

1) **+, -** 연산자

	정수형	실수형	위치형	카운터형	타이머형
정수형	정 수 형	실 수 형	불 가	정 수 형	정 수 형
실수형	실 수 형	실 수 형	불 가	실 수 형	실 수 형
위치형	불 가	불 가	위 치 형	불 가	불 가
카운터형	정 수 형	정 수 형	불 가	정 수 형	정 수 형
타이머형	정 수 형	정 수 형	불 가	정 수 형	정 수 형

2) ***** 연산자

	정수형	실수형	위치형	카운터형	타이머형
정수형	정 수 형	실 수 형	위 치 형	정 수 형	정 수 형
실수형	실 수 형	실 수 형	위 치 형	실 수 형	실 수 형
위치형	위 치 형	위 치 형	불 가	위 치 형	위 치 형
카운터형	정 수 형	실 수 형	위 치 형	정 수 형	정 수 형
타이머형	정 수 형	실 수 형	위 치 형	정 수 형	정 수 형

3) **/** 연산자

	정수형	실수형	위치형	카운터형	타이머형
정수형	정 수 형	실 수 형	불 가	정 수 형	정 수 형
실수형	실 수 형	실 수 형	불 가	실 수 형	실 수 형
위치형	위 치 형	위 치 형	불 가	위 치 형	위 치 형
카운터형	정 수 형	실 수 형	불 가	정 수 형	정 수 형
타이머형	정 수 형	실 수 형	불 가	정 수 형	정 수 형

4) **%** 연산자

정수간 연산만이 가능하며 나눗셈 연산 수행 시 **몫은 버리고 나머지를 결과 값**으로 갖습니다.
예) AA=10%3일 경우 **AA의 값은** 1

3.60.3 관계 연산자

명령어	기 능
>, <, ≤, ≥, ==, !=	주로 정수형과 실수형 자료를 대상으로 하며 IF 문과 WHILE 문의 조건식에 사용됩니다. 관계 연산의 결과는 논리값 (true,false 또는 0/1)입니다

3.60.4 논리 연산자

명령어	기 능
&&, ||, !	논리연산자는 논리값만을 대상으로 합니다. 즉 관계연산의 결과를 피연산자로 사용합니다. 논리연산의 결과는 논리값 입니다.

3.60.5 비트 연산자

명령어	기 능
&, | <<, >> ~	논리연산자 이동연산자 1 의 보수

3.60.6 주석문

명령어	기 능	사 용 예
//, /* */	주석문은 프로그램의 이해을 돕기 위한 역할을 합니다. 주석문은 컴파일러에 의해 여백으로 간주되기 때문에 프로그램 실행에 영향을 주지 않습니다.	// 한줄 주석 /* 문자열 주석 */
@	특수 문자열은 사용자가 작성한 내용을 로봇 운영중에 확인 할 수 있습니다. 주석문과 동일하게 프로그램 실행에 영향을 주지 않습니다. 일반 및 CE 용: Ver03.02.14 이상에서 지원됩니다. KCs 용: Ver03.03.04 이상에서 지원됩니다.	@Aing Test @Date:2013.12.24

```
@Aing Test     //특수 문자열
@Date:2013.12.24
@JOB Ver1.0a
//-----------------------------------------
/*                주석문
PULL_UP 명령어를 이용한
2포인트 AING JOB
*/
//-----------------------------------------
//          ROBOT POROGRAM
MAIN

VEL 1000 //속도 100%
PLUP 10  //PULL_UP 높이 10

WHILE 1
    JMOV P0
    JMOV P1
ENDWL

EOP
//-----------------------------------------
```

CAUTION

- ▶ 특수 문자열은 프로그램내의 최상단에 있어야 합니다.
- ▶ 특수 문자열은 최대 4 줄, 15 자 이내로 적성해야 티칭 펜던트에 정확히 출력됩니다.

3.61 내장 함수

함 수	기 능	사용 예
EXP(X)	지수 e^x	EXP(3) 은 20.085
LOG(X)	상용로그 $\log_{10}X$	LOG(100) 은 2
LN(X)	자연로그 $\log_eX$	LN(15)는 2.708
SQRT(X)	제곱근 $\sqrt{}$	SQRT(16)은 4
POW(x,y)	x^y	POW(2,10)은 1024
ABS(K)	정수 K 의 절대값	ABS(-12)는 12
RND(X)	실수 X 를 반올림하여 정수값	RND(97.62)는 98
SIN(X)	sine, x 단위 : radian	SIN(RAD(30))는 0.5
COS(X)	cosine, x 단위 : radian	COS(0)는 1
TAN(X)	tangent, x 단위 : radian	TAN(RAD(45))는 1.0
ASIN(X)	arcsine, - π/2~π/2 의 값 반환	DEG(ASIN(0.5))는 30
ACOS(X)	arccosine, 0 ~ π의 반환	DEG(ACOS(0.5))는 60
ATAN(X)	arctangent, -π/2 ~π/2 의 값 반환	DEG(ATAN(1.0))은 45
ATAN2(Y,X)	제 2 arctangent, -π~π 의 값 반환	DEG(ATAN2(-1,-1))은 -135
DEG(X)	radian 을 angle 값으로 변환	DEG(3.1416)은 180.0
RAD(X)	angle 을 radian 값으로 변환	RAD(180.0)는 3.1416

3.62 ASC

기 능　　문자열의 첫문자를 아스키 코드로 반환합니다.

형 식　　<정수형 변수> = ASC(<문자열>)

용 어　　문자열 변수란?
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<문자열>
문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인 문자들의 집합입니다. 예) "ABCDEF"

아스키 코드란?
문자에 해당하는 숫자(정수)를 말합니다.
예) 문자'A'는 정수(65)이고 문자 'B'는 정수(66)입니다.
아스키 코드표를 참조 하십시오.

설 명　　1) 입력된 문자열의 첫문자를 반환합니다.

3.62.1 프로그램 사용 예제

```
MAIN
DEFSTR AA
INT BB
AA = "XYZW"  ..... 문자열 변수 AA에 "XYZW"대입
BB = ASC(AA) ..... 문자열 변수 AA에서 첫번째 문자 "X"를 캐릭터코드(88)로 반환
EOP
```

3.63 CHR

기 능 아스키 코드 범위의 정수를 문자로 반환합니다.

형 식 <문자열변수> = CHR(<정수>)

용 어 <문자열변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<정수>
아스키 코드값 내의 정수(0~127)입니다.

설 명 1) 입력된 정수에 해당하는 아스키 코드를 반환합니다.

3.63.1 프로그램 사용 예제

```
MAIN
DEFSTR  AA
AA = CHR(65)  ..... 문자열 변수 AA에 아스키코드 65에 해당하는
EOP           "A"대입
```

3.64 FLUSH

기 능 입력, 출력 버퍼를 클리어합니다.

형 식 FLUSH <PORT> <정수>

용 어 <PORT>: COM Port를 선택(1~2)
1) 일반 및 CE제어기인 경우
1: HOST(RS-232) Port 선택
2: EXT_COM(RS-485) Por를 선택

2) KCs 제어기인 경우
1:COM1(RS-232) Port 선택
2:COM2(RS-232) Port 선택

<정수>: 클리어 버퍼 선택(1~3)
1: 입력 버퍼 클리어
2: 출력 버퍼 클리어
3: 입력, 출력 버퍼 클리어

설 명 1) 시리얼 통신 송수신에 사용하는 입출력 버퍼를 클리어 합니다.

3.64.1 프로그램 사용 예제

```
MAIN
DEFSTR AA
AA = STRIN(1,1000)  ..... Serial 통신으로 입력을 받아 문자열 AA에 저장
FLUSH 1 1           ..... RS-232 Port 입력 버퍼 클리어
EOP
```

3.65 FTOS

기 능　　정수나 실수 데이터를 문자열로 변환합니다.

형 식　　<문자열변수> = FTOS(<정수> or <실수>)

용 어　　<문자열변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<정수> or <실수>
문자열로 변환할 정수나 실수입니다.

설 명　　1) 정수나 실수 수치 데이터를 문자열로 변환합니다.
예) FTOS(1234) → "1234"

3.65.1 프로그램 사용 예제

```
MAIN
DEFSTR  AA,BB
AA = FTOS(1234)        .....    1234를 문자열 "1234"로 변환 후 문자열 변수 AA에 대입
BB = FTOS(-123.456)    .....    -123.456를 문자열 "-123.456"로 변환 후
                                문자열 변수 BB에 대입
EOP
```

3.66 HTOS

기 능 정수를 16진수의 문자열로 변환합니다.

형 식 <문자열변수>= HTOS(<정수>)

용 어 <문자열변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<정수>
16진수 문자열로 변환할 정수입니다.

설 명 1) 입력된 정수를 16진수 문자열로 변환합니다.

3.66.1 프로그램 사용 예제

```
MAIN
DEFSTR  AA
AA = HTOS(10)  ..............  정수 10을 16진수 문자열("A")로 변환하여 문자열 AA에 대입
EOP
```

3.67 SLEFT

기 능　　입력된 문자열의 좌측 문자열 추출합니다.

형 식　　<문자열 변수> = SLEFT(<문자열>, <정수>)

용 어　　<문자열변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<문자열>
문자열 변수나 문자열 상수를 의미합니다.
문자열 상수는 " "로 둘러싸인 문자들의 집합입니다.

<정수>
추출할 문자의 수입니다.

설 명　　1) 입력된 문자열을 왼쪽에서부터 입력된 정수만큼 추출합니다.

3.67.1 프로그램 사용 예제

```
MAIN
DEFSTR   AA,BB
AA = "XYZW"          .....    문자열 변수 AA에 "XYZW" 대입
BB = SLEFT(AA,2)     .....    문자열 변수 AA의 왼쪽부터 2문자("XY")을 추출
EOP
```

3.68 SLEN

기 능 입력된 문자열의 길이를 반환합니다.

형 식 <정수형변수>=SLEN(<문자열>)

용 어 <문자열>
문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인문자들의 집합입니다.
예) "ABCDEF"

설 명 1) 입력된 문자열의 길이를 반환합니다.

3.68.1 프로그램 사용 예제

```
MAIN
DEFSTR  AA
INT LL
AA = "XYZW"          .....    문자열 변수 AA에 "XYZW"대입
LL = SLEN(AA)        .....    문자열 변수 AA의 길이(4)를 정수형 변수 LL에 대입
EOP
```

3.69 SMID

기 능　　문자열 지정위치부터 자릿수 만큼 문자열 추출합니다.

형 식　　<문자열 변수> = SMID(<문자열>, <지정위치>, <정수>)

용 어　　<문자열변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<문자열>
문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인문자들의 집합입니다.

<지정위치>
문자열 추출할 시작 위치, 문자열의 시작은 0입니다.

<정수>
지정위치부터 읽어올 문자 개수입니다.

설 명　　1) 지정위치에서부터 사용자 입력 정수만큼 문자를 추출합니다.

3.69.1 프로그램 사용 예제

```
MAIN
DEFSTR   AA, BB
AA = "ABCDEFG"          .....    문자열 변수 AA에 "ABCDEFG"대입
BB = SMID(AA,2,3)       .....    문자열 변수 AA의 2번째위치부터 문자 3개("CDE")반환
EOP
```

3.70 SPOS

기 능　　문자열1에서 문자열2가 매칭되는 시작위치 반환합니다.

형 식　　정수형변수 = SPOS(<문자열1>, <문자열2>)

용 어　　<문자열1><문자열2>
문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인 문자들의 집합입니다. 예) "ABCDEF"

설 명　　1) 문자열1에서 문자열2와 매칭되는 시작위치를 반환합니다.

3.70.1 프로그램 사용 예제

```
MAIN
DEFSTR   AA, BB
INT PP
AA = "XYZW"          .....    문자열 변수 AA에 "XYZW" 대입
BB = "Z"             .....    문자열 변수 BB에 "Z" 대입
PP = SPOS(AA, BB)    .....    문자열 변수 AA에서 BB와 일치하는 위치 반환
EOP
```

3.71 SRIGHT

기 능　　입력된 문자열의 우측 문자열을 추출합니다.

형 식　　<문자열 변수> = SRIGHT(<문자열>, <정수>)

용 어　　<문자열 변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<문자열>
문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인 문자들의 집합입니다.

<정수>
추출할 문자의 수입니다.

설 명　　1) 입력된 문자열을 오른쪽에서부터 입력된 정수만큼 추출합니다.

3.71.1 프로그램 사용 예제

```
MAIN
DEFSTR   AA, BB
AA = "ABCDEF"          .....    문자열 변수 AA에 "ABCDEF" 대입
BB = SRIGHT(AA,2)      .....    문자열 변수 AA의 오른쪽 두문자를("EF")반환
EOP
```

3.72 STRIN

기 능 시리얼통신을 통하여 입력 된 문자열을 읽어옵니다.

형 식 <문자열 변수> = STRIN(<PORT>, <정수>)

용 어 <문자열 변수>
DEFSTR을 사용하여 선언한 변수를 의미하며, 문자열을 다루기 위해 사용합니다.

<PORT>: Serial Port(1~2), Ethernet TCP/IP(10~12)
1) 일반 및 CE제어기인 경우
1: HOST(RS-232) Port 선택
2: EXT_COM(RS-485) Por를 선택

2) KCs 제어기
1:COM1(RS-232) Port 선택
2:COM2(RS-232) Port 선택

3) Ethernet Port 지원 제어기에 해당함.
10: TCP/IP String Port 1 선택
11: TCP/IP String Port 2 선택
12: TCP/IP String Port 3 선택

<정수>
타임아웃 시간(단위: ms)의미하며 입력한 타임아웃 시간만큼 시리얼 입력을 기다립니다. 타임아웃시간동안 데이터가 들어오지 않으면, 시스템 영역에 타임아웃 상태를 체크하고 다음 스텝명령을 수행합니다

설 명 1) 사용자가 지정한 시간만큼 대기하면서 시리얼 통신을 통한 입력을 읽어옵니다.

3.72.1 프로그램 사용 예제

```
MAIN
DEFSTR   PACKET
INT LEN, OVERTM
PACKET = STRIN(1,1000)        .....   시리얼 통신을 통한 문자열 입력(입력 없을시 1s 대기)
LEN = SLEN(PACKET)            .....   입력받은 문자열 길이 저장
SS1=SYS1
OVERTM=(SS1>12) &0H01         .....   TIMEOUT 유무를 읽어옴
IF OVERTM==0&&LEN>0 THEN      .....   TIMEOUT이 아니고, 입력받은 문자열이 존재한 경우
.....
ENDIF
EOP
```

3.73 STROUT

기 능 시리얼 통신을 통하여 문자열 출력합니다.

형 식 <정수형변수> = STROUT(<PORT>, <문자열>)

용 어 <PORT> : Serial Port(1~2), Ethernet TCP/IP(10~12)

1) 일반 및 CE제어기인 경우

1: HOST(RS-232) Port 선댁합니다.

2: EXT_COM(RS-485) Por를 선댁합니다.

2) KCs 제어기인 경우

1:COM1(RS-232) Port 선댁합니다.

2:COM2(RS-232) Port 선댁합니다.

3) Ethernet Port 지원 제어기에 해당함.

10: TCP/IP String Port 1 선댁

11: TCP/IP String Port 2 선댁

12: TCP/IP String Port 3 선댁

<문자열>

문자열 변수나 문자열 상수를 의미합니다. 문자열 상수는 " "로 둘러싸인 문자들의 집합입니다.

설 명 1) 입력한 문자열을 시리얼 통신을 통해 출력을 내보내며, 전송하지 못한 문자수 반환

예)"ABCDEFGH"의 문자를 전송하는 경우 성공시 반환값은 0

전부 전송하지 못한 경우 반환값은 8

3.73.1 프로그램 사용 예제

```
MAIN
DEFSTR   SENDPK
INT STRRET
SENDPK="ABCDEF"          .....   SENDPK 문자열 변수에 "ABCDEF" 대입
STRRET=STROUT(1,SENDPK)  .....   시리얼 통신포트로 문자열("ABCDEF") 출력
IF STRRET !=0 THEN       .....   전송 실패 확인
.....
ENDIF
EOP
```

3.74 SVAL

기 능　　문자열을 숫자로 변환합니다.

형 식　　<정수형변수 or 실수형변수> = SVAL(<문자열>)

용 어　　<문자열>
문자열 변수나 문자열 상수를 의미합니다.
문자열 상수는 " "로 둘러싸인 문자들의 집합입니다.

설 명　　1) 입력한 문자열을 정수 또는 실수 데이터로 변환합니다.

3.74.1 프로그램 사용 예제

```
MAIN
INT AI
REAL BR
AI=SVAL("1234")  .....  문자열 "1234"를 정수 1234로 변환
BR=SVAL("12.234")  .....  문자열 "12.234"를 실수 12.234로 변환
EOP
```

3.75 VCMD

기 능 Vision point 및 관련 변수를 초기화 합니다.

형 식 VCMD <INDEX> <작업종류>

용 어 <INDEX>: 0~9까지 입력 합니다.
입력된 Index는 Serial Port로 전송합니다.

<작업종류>: 1 또는 2를 입력 합니다.
Serial Port로 전송할 DATA를 선택 합니다.

설 명 1) Serial Port로 전송되는 DATA format은 다음과 같습니다.
가) 작업종류가 1인 경우: STX, 0XFF, 'V' 'Q' Index ETX LRC
나) 작업종류가 2인 경우: STX, 0XFF, 'V' 'R' Index ETX LRC

3.75.1 프로그램 사용 예제

```
MAIN
VCMD 5 2          .....   Serial Port로 STX 0xFF 'V' 'R' '5' ETX LRC를
                          전송합니다.
EOP
```

3.76 VVAL

기 능　　Vision Point로 받은 DATA를 반환 합니다.

형 식　　정수형 변수=VVAL

용 어　　<정수형 변수> : INT로 선언된 변수명입니다.

설 명　　1) Serial CMD로 받은 DATA를 반환합니다.
가) "VA", "VB", "VM", "VN" 프로토콜 명령어를 참조 하시기 바랍니다.
(N1_옵션 로보스타 프로토콜 설명서 참조바랍니다.)

3.76.1 프로그램 사용 예제

```
MAIN
JMOV P1000 ..... 카메라 위치로 이동
CALL H_ON ..... 판정 요청
IF VVAL != 0 THEN ..... 판정 값을 반환 합니다.
CALL ERROR ..... 불량 서브루틴 호출
ELSE
CALL GOOD ..... 양품 서브루틴 호출
ENDIF
EOP
SUBR ERROR
JMOV P100 ..... 불량 위치 이동
RET
SUBR GOOD
JMOV P200 ..... 양품 위치 이동
RET
```

3.77 VDAT

기 능 Vision CMD 데이터 수신을 지정한 시간만큼 대기합니다.

형 식 VDAT 대기시간

용 어 대기시간: 정수형 DATA 입니다.(단위는 10ms 입니다.)

설 명 1) Serial CMD DATA가 들어 올 때까지 대기 합니다.
가) "VA", "VB", "VM", "VN" 프로토콜 명령어를 참조 하시기 바랍니다.
(N1_옵션 로보스타 프로토콜 설명서 참조바랍니다.)

3.77.1 프로그램 사용 예제

```
MAIN
JMOV P1000 ..... 카메라 위치로 이동
CALL H_ON ..... 판정 요청
VDAT 100 ..... 1초 동안 Vision CMD 수신 대기
IF VVAL != 0 THEN
CALL ERROR ..... 불량 서브루틴 호출
ELSE
CALL GOOD ..... 양품 서브루틴 호출
ENDIF
EOP
SUBR H_ON
OUT2=1 ..... 작업물 판정 요청
RET
SUBR ERROR
JMOV P100 ..... 불량 위치 이동
RET
SUBR GOOD
JMOV P200 ..... 양품 위치 이동
RET
```

3.78 VCNT

기 능 전송 받은 Vision Point Data의 개수를 반환 합니다.

형 식 <정수형 변수>=VCNT

용 어 <정수형 변수> : INT로 선언된 변수명입니다.

설 명 1) Serial CMD DATA가 들어 올 때까지 대기 합니다.
가) "VA", "VB", "VM", "VN" 프로토콜 명령어를 참조 하시기 바랍니다.
(N1_옵션 로보스타 프로토콜 설명서 참조바랍니다.)

3.78.1 프로그램 사용 예제

```
MAIN
INT COUNT
JMOV P1000 ..... 카메라 위치로 이동
CALL H_ON ..... 판정 요청
COUNT=VCNT ..... Vision Point Data 반환
IF COUNT>0 THEN
JMOV P2000 ..... Vision Point로 이동
ELSE
JMOV P10 ..... 대기 위치 이동
ENDIF
EOP
SUBR H_ON
OUT1=1 ..... 요청 신호 출력
RET
```

3.79 Safety 관련 명령어

N1 Category4 전용 명령어로 03.03.04-C4로 시작하는 펌웨어 버전에서만 동작됩니다.
(N1-Category4 매뉴얼을 참고바랍니다.)

3.79.1 STEMP(System Temperature)

기 능 제어기 관련 온도값을 읽어 오는 명령어입니다.

형 식 <실수형 변수>= STEMP(<모드>, <인덱스>)

용 어 <모드>
0 : 기능 비활성화을 시킵니다.
1 : 기능 활성화를 시킵니다.

<인덱스>
0 : 제어기 내부 온도 상태를 읽어옵니다.
1 : 예약된 영역입니다.
2 : 예약된 영역입니다.
3 : 예약된 영역입니다.
4 : 예약된 영역입니다.
5 : 예약된 영역입니다.
6 : 예약된 영역입니다.
7 : 예약된 영역입니다.
8 : 예약된 영역입니다.
9 : 예약된 영역입니다.
10 : 예약된 영역입니다.
11 : 예약된 영역입니다.
12 : 예약된 영역입니다.
13 : 예약된 영역입니다.
14 : 예약된 영역입니다.
15 : 예약된 영역입니다.

설 명 1) 제어기 내부 온도 값을 Safety Interface 보드를 통해 읽어옵니다.

3.79.1.1 프로그램 사용예제

```
MAIN
REAL TEMP
TEMP = STEMP(1,0)  ..... 제어기 내부 온도 읽어와 내부 실수형 변수에 저장
EOP
```

CAUTION

- ▶ STEMP 명령어를 통해 읽어오는 제어기 내부 온도값의 경우 Safety Interface B/D 의 온도 센서로 읽어 옵니다.
- ▶ KCs 제어기 타입 및 펌웨어 03.03.04-C4 R0 170726 버전부터 사용할 수 있습니다.

3.79.2 SUIN(Safety User Input)

기 능 Safety 관련 접점 상태를 읽어옵니다.

형 식 <정수형 변수>= SUIN(<모드>, <인덱스>)

용 어 <모드>
1 : Safety 사용자 입력 접점을 읽을 수 있는 모드입니다.
2 : Safety 사용자 출력 접점을 읽을 수 있는 모드입니다.
3 : Safety 접점상태를 읽을 수 있는모드입니다.

<인덱스>
모드가 '1'인경우 :
0 : 전체 Safety User 입력 값을 읽어옵니다.
1 : Safety User 입력1 접점상태를 읽어옵니다.
2 : Safety User 입력2 접점상태를 읽어옵니다.
3 : Safety User 입력3 접점상태를 읽어옵니다.
4 : Safety User 입력4 접점상태를 읽어옵니다.
5 : 예약된 영역입니다.
6 : 예약된 영역입니다.
7 : 예약된 영역입니다.
8 : 예약된 영역입니다.
9 : 예약된 영역입니다.
10 : 예약된 영역입니다.
11 : 예약된 영역입니다.
12 : 예약된 영역입니다.
13 : 예약된 영역입니다.
14 : 예약된 영역입니다.
15 : 예약된 영역입니다.

모드가 '2'인경우 :
0 : 전체 Safety User 출력 값을 읽어옵니다.
1 : Safety User 출력1 접점상태를 읽어옵니다.
2 : Safety User 출력2 접점상태를 읽어옵니다.
3 : Safety User 출력3 접점상태를 읽어옵니다.
4 : Safety User 출력4 접점상태를 읽어옵니다.
5 : Safety User 출력5 접점상태를 읽어옵니다.
6 : Safety User 출력6 접점상태를 읽어옵니다.
7 : 예약된 영역입니다.
8 : 예약된 영역입니다.
9 : 예약된 영역입니다.
10 : 예약된 영역입니다.
11 : 예약된 영역입니다.
12 : 예약된 영역입니다.
13 : 예약된 영역입니다.
14 : 예약된 영역입니다.
15 : 예약된 영역입니다.

모드가 '3'인경우 :
0 : Satey 레지스터 전체 값을 읽어옵니다.
1 : Interlock A 접점상태를 읽어옵니다.
2 : Interlock M 접점상태를 읽어옵니다.
3 : Extern Mode NC 접점상태를 읽어옵니다.
4 : Extern Mode NO 접점상태를 읽어옵니다.
5 : LIGHT CURTAIN 접점상태를 읽어옵니다.
6 : Enable hold 접점상태를 읽어옵니다.
7 : Extern Reset 접점상태를 읽어옵니다.
8 : 예약된 영역입니다.
9 : 예약된 영역입니다.
10 : 예약된 영역입니다.
11 : 예약된 영역입니다.

설 명 1) Safety User 입/출력 및 Safety 접점 상태를 읽어옵니다.

3.79.2.1프로그램 사용 예제

1) Safety User Input 접점 상태값을 개별로 읽어 GINT 에 저장

```
MAIN
I0 = SUIN(1,0)  .....  Safety User Input 1~4 전체 접점 상태 값을
                       정수로 반환합니다.
I1 = SUIN(1,1)  .....  Safety User Input 1 접점 상태를 GINT 1에 저장
I2 = SUIN(1,2)  .....  Safety User Input 2 접점 상태를 GINT 2에 저장
I3 = SUIN(1,3)  .....  Safety User Input 3 접점 상태를 GINT 3에 저장
I4 = SUIN(1,4)  .....  Safety User Input 4 접점 상태를 GINT 4에 저장
EOP
```

2) Safety User Output 접점 상태값을 개별로 읽어 I/O로 출력

```
MAIN
I0 = SUIN(2,0)  .....  Safety User Output 1~6 전체 접점 상태 값을
                       정수로 반환합니다.
OUT1 = SUIN(2,1)  .....  Safety User Output 1 접점 상태 출력
OUT2 = SUIN(2,2)  .....  Safety User Output 2 접점 상태 출력
OUT3 = SUIN(2,3)  .....  Safety User Output 3 접점 상태 출력
OUT4 = SUIN(2,4)  .....  Safety User Output 4 접점 상태 출력
OUT5 = SUIN(2,5)  .....  Safety User Output 5 접점 상태 출력
OUT6 = SUIN(2,6)  .....  Safety User Output 6 접점 상태 출력
EOP
```

3) Safety 접점 상태값을 개별로 읽어 I/O로 출력

```
MAIN
I0 = SUIN(3,0)  .....  Safety 레지스터 전체 값을 정수로 반환합니다.
OUT1 = SUIN(3,1)  .....  Interlock M 상태 출력(1:Interlock M, 0: Normal)
OUT2 = SUIN(3,2)  .....  Interlock A 상태 출력(1: Interlock A, 0: Normal)
OUT3 = SUIN(3,3)  .....  Extern mode NO 상태 출력(1: Manual, 0:Auto)
OUT4 = SUIN(3,4)  .....  Extern mode NC 상태 출력(1: Auto, 0:Manual)
OUT5 = SUIN(3,5)  .....  Light Curtain 상태 출력(1:Error, 0:Normal)
OUT6 = SUIN(3,6)  .....  Enable Hold 상태 출력(1: Normal, 0:Hold Stop)
OUT6 = SUIN(3,7)  .....  Extern Reset 상태 출력(1:Reset Enable, 0:Reset Disable)
EOP
```

CAUTION

- ▶ SUIN 명령어 모드 "1"인 경우 Safety User Input 접점 상태를 반환하며, 외부 접점이 입력된 경우 ON(1)을 반환하며, 외부 접점이 차단된 경우 OFF(0)를 반환 합니다.
- ▶ SUIN 명령어 모드 "2"인 경우 Safety User Output 접점 상태를 반환하며, 점점 출력이 켜진 경우 ON(1)을 반홚며, 출력이 꺼진 경우 OFF(0)를 반환 합ㄴ디ㅏ.
- ▶ SUIN 명령어 모드 "3"인 경우 Safety 접점 상태를 반환하며, 프로그램 예제의 점점 상태 반환 설명을 참고하시기 바랍니다.

  ※ Reserved 상태의 접점은 0으로 반환됩니다.
- ▶ KCs 제어기 타입 및 펌웨어 03.03.04-C4 R0 170726 버전부터 사용가능합니다.

3.79.3 SUOUT(Safety User Output)

기 능 Safety 사용자 출력 포트를 지정된 값으로 출력합니다.

형 식 <정수형 변수>= SUOUT(<인덱스>, <1 또는 0>)

용 어 <인덱스>

0 : 전체 Safety User 출력을 On/Off 시킵니다.
1 : Safety User 출력1을 On/Off 시킵니다.
2 : Safety User 출력2을 On/Off 시킵니다.
3 : Safety User 출력3을 On/Off 시킵니다.
4 : Safety User 출력4을 On/Off 시킵니다.
5 : Safety User 출력5을 On/Off 시킵니다.
6 : Safety User 출력6을 On/Off 시킵니다.
7 : 예약된 영역입니다.
8 : 예약된 영역입니다.
9 : 예약된 영역입니다.
10 : 예약된 영역입니다.
11 : 예약된 영역입니다.
12 : 예약된 영역입니다.
13 : 예약된 영역입니다.
14 : 예약된 영역입니다.
15 : 예약된 영역입니다.
16 : 예약된 영역입니다.

설 명 1) 사용자가 지정한 Safety User 출력포트를 출력할 수 있는 명령어입니다.

3.79.3.1프로그램 사용 예제

1) Safety User Output 접점 상태값을 개별로 읽어 I/O로 출력

```
MAIN
INT RETVAL
RETVAL = SUOUT (0,1)  ..... Safety User Output 1~6 전체 포트를 ON(1) 시킴
RETVAL = SUOUT (0,0)  ..... Safety User Output 1~6 전체 포트를 OFF(0) 시킴
RETVAL = SUOUT (1,1)  ..... Safety User Output 1 출력 포트를 ON(1) 시킴
RETVAL = SUOUT (1,0)  ..... Safety User Output 1 출력 포트를 OFF(0) 시킴
EOP
```

CAUTION

- SUOUT 명령어 개별 출력(INDEX값이 1~6인 경우) 사용 시 해당 파라미터 설정이 "NONE"으로 되어 있어야 하며, 특수 기능으로 설정된 경우 E1245 "User Output Num Err" 가 발생합니다.
- SUOUT 명령어 전체 출력(INDEX값이 0인 경우) 사용 시 SAFETY USER I/O 파라미터 설정값이 "NONE"으로 설정된 포트만 동작됩니다.
- SUOUT 명령어 실행이 정상적인 경우 리턴 되는 반환 값은 정수 1 입니다.
- 파라미터 설정에 대한 자세한 사항은 N1-IM-CA4 매뉴얼을 참조하시기 바랍니다.
- KCs 제어기 타입 및 펌웨어 03.03.04-C4 R0 170726 버전부터 사용가능합니다.

Rev.	수정일자	내용	수정자	S/W Version
V.1	2012.07.30	초판 인쇄		
V.2	2012.12.31	내용 추가 및 수정	Kimjs	
V.3	2014.07.10	오탈자 및 문맥 수정		
V.4	2018.03.02	명령어 추가	Minhj	
V.5	2019.09.18	오탈자 및 문맥 수정	Minhj	
V.6	2020.03.14	오탈자 및 명령어 추가	Jangsh	
V.7	2022.01.18	오탈자 및 명령어 추가	Jangsh	

Robostar

N1 ROBOT CONTROLLER

CONTROLLER MANUAL

FIRST EDITION JULY 2012

ROBOSTAR CO, LTD

ROBOT R&D CENTER