Robostar Robot Controller

N2 Series
Programming Manual

Version: N2-PM-K07
Issued Data: May 03, 2021

Robostar

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) 사용 설명서 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
  (사사동 119-38)
  700, Suin-ro, Sangnok-gu, Ansan-City,
  Gyeonggi-do, Republic of South Korea
  (15523)

- 수원공장
  경기도 수원시 권선구 산업로 155번길 37
  (고색동 997)
  37, Saneop-ro 155beon-gil,
  Gwonseon-gu, Suwon-City,
  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

목차

그림 목차

제 1장 프로그램 디자인

1. 기본 프로그램 구조

로보스타 로봇 언어 (RRL, Robostar Robot Language)는 Robostar에서 만든 전용 언어로써 시스템 내에서 로봇 작업 프로그램을 작성할 때 사용되는 명령어 입니다.

일반적으로 프로그램을 작성할 때 Figure 1-1과 같은 구조로 작성합니다. 이 방식은 필수요소가 아닙니다.

※ **JOB 프로그램은 최대 10,000줄까지 작성 가능합니다.**
※ **JOB 프로그램은 한 줄에 최대 100문자까지 작성 가능 합니다.**

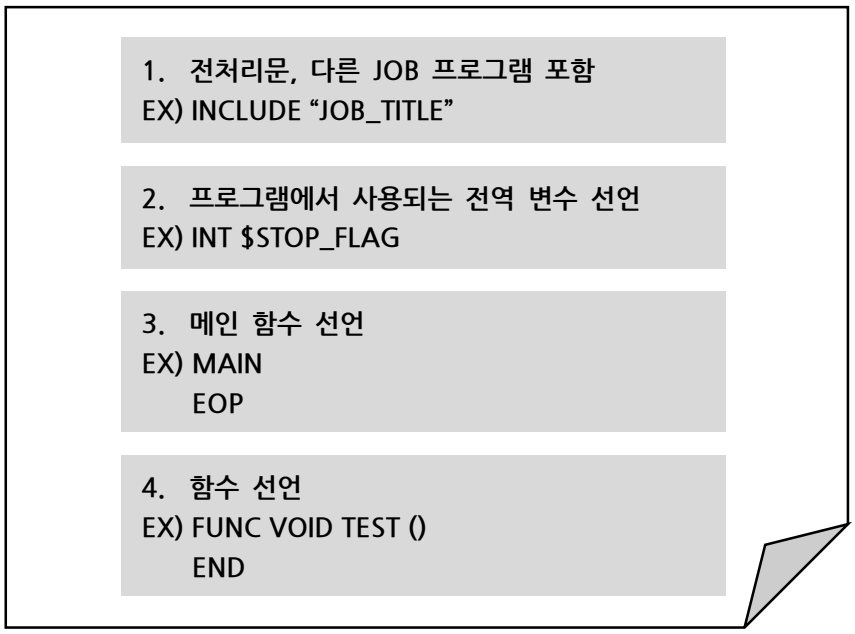


Figure 1-1 기본 프로그램 구조

1.1 메인 함수의 선언

로보스타 로봇 언어의 프로그램은 반드시 하나의 MAIN 함수를 가져야 합니다. Figure 1-2의 MAIN 함수 작성 예시와 같이 MAIN 함수를 선언한 후에, MAIN 함수의 끝에는 반드시 프로그램의 종료를 알리는 EOP 명령어를 작성해 주어야 합니다.

※ 자세한 설명 및 예시는 제 4장 5.1절 MAIN … EOP (프로그램 시작/종료)를 참고하시기 바랍니다.

1	MAIN	1. 메인 함수 시작 선언
2	SVON	2. 서보 ON
3	JMOV GP1	3. 모션 수행 및 시간지연
4	DLAY 2000	
5	JMOV GP2	
6	DLAY 2000	
7	SVOF	7. 서보 OFF
8	EOP	8. 메인 함수 종료

Figure 1-2 메인 함수 작성 예시

1.2 일반 함수의 선언

메인 함수 등 시스템에 미리 선언된 함수 외의 별도의 함수를 선언한 후 호출하여 사용할 수 있습니다.

※ 자세한 설명 및 예시는 제 4장 5.2절 FUNC … END(함수 시작/종료)를 참고하십시오.

Syntax

```
FUNC <변수> <함수명> (매개변수, 매개변수, …)
        ⋮
    (RET)
END
```

1	FUNC INT TEST()	1. 함수 시작 선언, 반환 변수 및 함수명, 매개변수 선언
2	SVON	2. 서보 ON
3	JMOV GP1	3. 모션 수행 및 시간지연
4	DLAY 2000	
5	JMOV GP2	
6	DLAY 2000	
7	SVOF	7. 서보 OFF
8	RET 1	8. 리턴 값 반환
9	END	9. 함수 종료

Figure 1-3 일반 함수 작성 예시

2. 프로그램 호출 및 서브루틴

특정 모션 혹은 기능을 반복 사용하는 경우, 프로그램 내에 그 모션 혹은 기능을 반복하는 루틴을 작성합니다. 그러면 프로그램 실행 시 MAIN 루틴 혹은 다른 루틴에서 해당 루틴을 호출하여 사용할 수 있습니다. 이와 같이 사용되는 루틴을 서브루틴이라 합니다. 만약 이 루틴이 별도의 다른 프로그램에 저장된 후 호출되어 사용되면 이것은 프로그램 호출이라 합니다.

일련의 작업이 서브 루틴이나 다른 프로그램의 단위로 구성되는 경우, 동일한 내용을 반복해서 작성하지 않아도 됩니다. 이는 작성 시간을 줄이고 또한 프로그램의 가독성을 향상시키는 데 효과적입니다.

서브루틴은 반드시 호출 프로그램과 동일한 파일에 포함되어야 합니다. 서브 루틴이 종료되면 서브 루틴을 호출한 프로그램의 다음 행으로 되돌아 갑니다. 서브 루틴은 다른 서브 루틴에서 호출할 수 있습니다. 전역 변수의 사용이 가능하며, 인자를 전달할 수 있습니다.

독립된 개별 파일의 프로그램은 다양한 프로그램에서 호출할 수 있습니다. 프로그램을 호출할 때는 JCALL 문에 프로그램 이름을 지정하여 사용합니다. JCALL문이 프로그램을 호출하면 호출된 프로그램으로 실행 라인이 이동합니다. 호출된 프로그램의 마지막 문이 실행되면 실행 라인이 호출 프로그램의 다음 행으로 되돌아 갑니다. 호출된 프로그램은 다른 프로그램을 호출할 수도 있습니다. 그러나 자신을 호출한 프로그램 또는 자기 자신을 다시 호출할 순 없습니다. 전역 변수의 경우 일반적으로 사용할 수 있으나, 인자 전달은 불가능합니다.

※ 자세한 설명 및 예시는 제 4장 5.10절 JCALL (프로그램 호출) 및 제 4장 5.2절 FUNC … END(함수 시작/종료)를 참고하시기 바랍니다

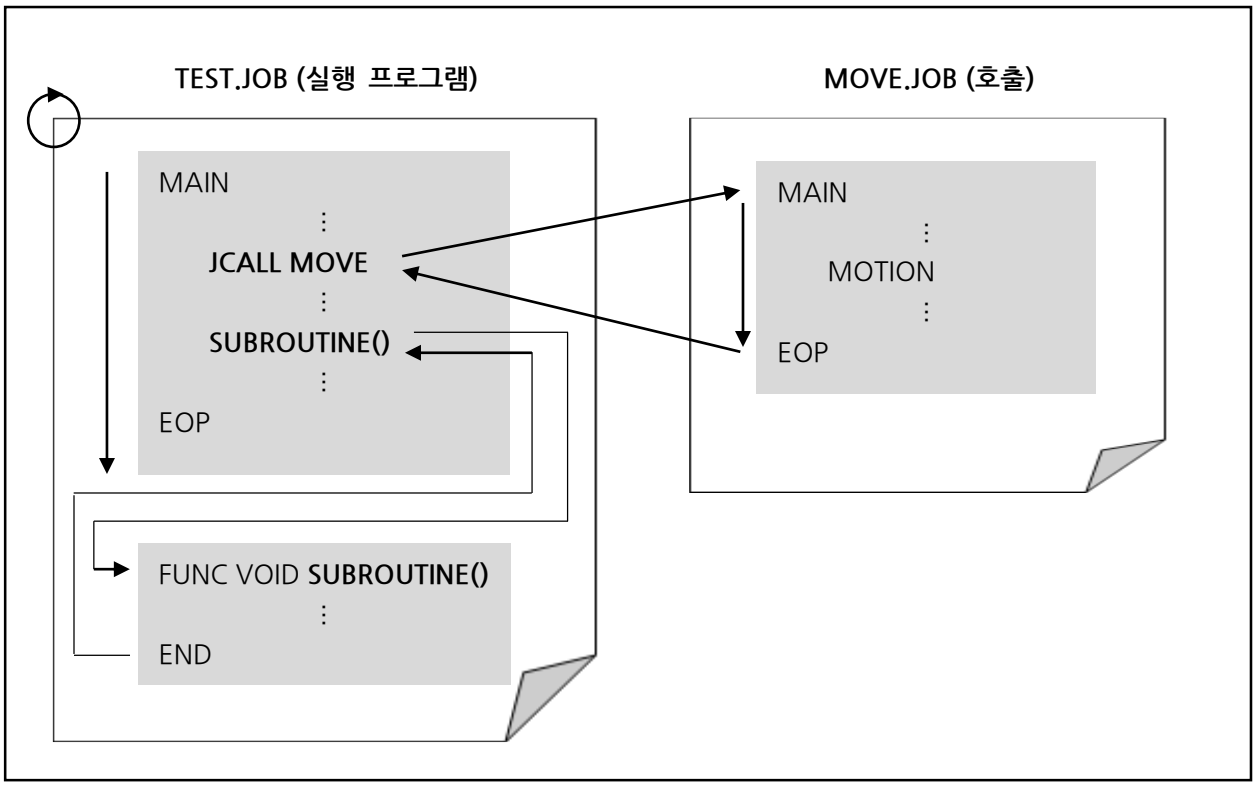


Figure 1-4. 서브 루틴과 프로그램 호출의 차이점

3. Multithreading

로보스타 로봇 언어(RRL, Robostar Robot Language)는 한 프로그램에서 동시에 여러 작업을 처리할 수 있는 멀티 스레드 기능을 제공합니다.

하나의 작업을 하나의 스레드로 처리하는 싱글 스레드 방식으로 두 개의 작업을 처리하게 되는 경우, 한 작업을 마친 후 다음 작업을 시작하게 됩니다. 하지만 멀티 스레드 방식을 사용하게 되면, 여러 개의 스레드를 사용하여 작업을 동시에 처리할 수 있습니다. 로봇이 멀티스레딩 작업에서 제어되는 경우, 개별 JOB 프로그램 또는 함수들을 병행 실행하여 기능을 최적화 할 수 있습니다. 하나의 프로그램 일부분이 중단되거나 긴 작업을 수행하더라도 또 다른 프로그램이 계속 수행되어 사용자에 대한 응답성이 증가됩니다.

※ 서로 다른 스레드에서 동일한 전역변수를 동시에 사용하는 경우 수행 결과를 예측할 수 없게 되거나, 실행 순서가 보장되지 않을 수 있기 때문에 사용 시 주의해야 합니다.
※ JMOV, LMOV 등 모터를 구동하는 명령어는 한 스레드에서 사용해야 합니다.

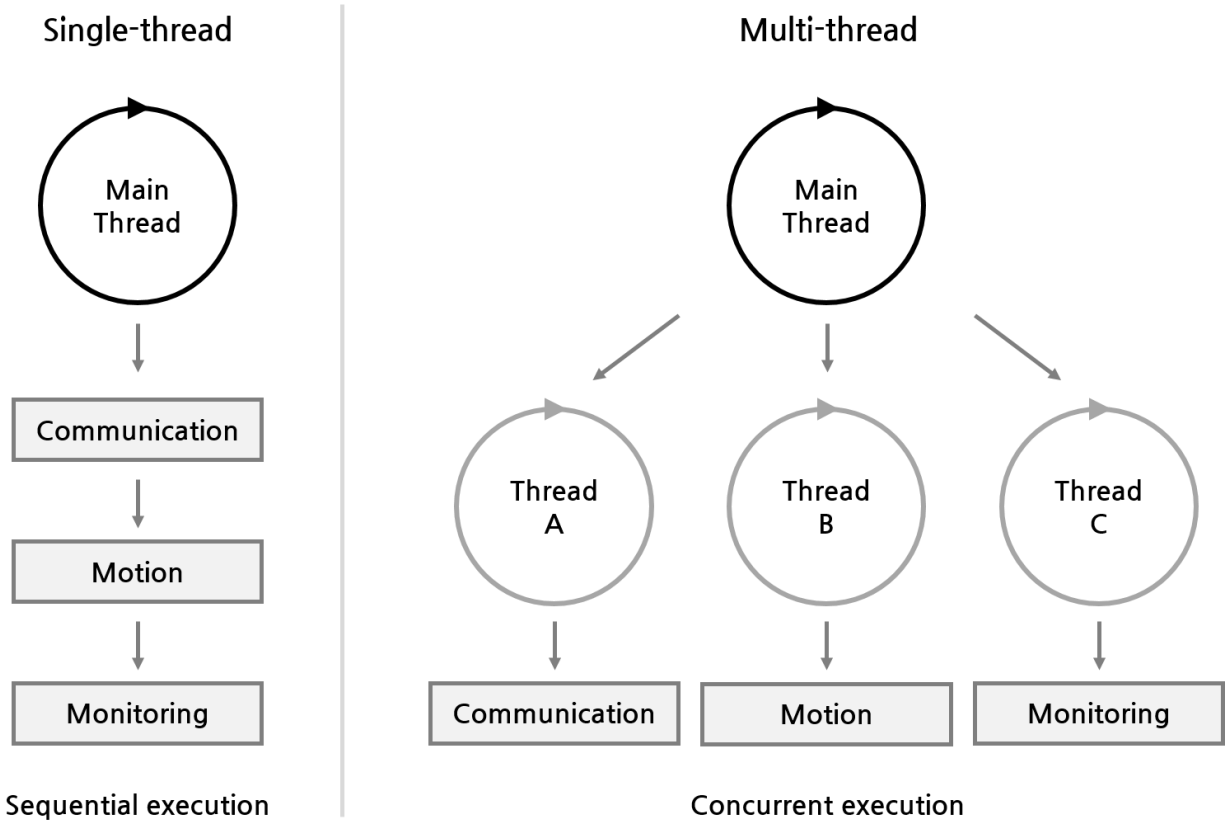


Figure 1-5 싱글 스레드와 멀티 스레드의 차이 예시

System 모드, Robot 모드는 각각 메인 스레드를 제외한 최대 3개의 스레드를 추가적으로 할당하여 최대 4개의 스레드를 동시에 동작할 수 있습니다.

일반적으로 개별 작업을 별도의 프로그램으로 작성하여 메인 프로그램이 특정 조건에 따라 스레드를 할당하고 작업이 완료되면 해제하도록 하여 다양한 작업을 사용할 수 있도록 합니다.

	System 모드	Robot 모드
TH1	SYSTEM THREAD 1(메인 스레드)	ROBOT THREAD 1(메인 스레드)
TH2	SYSTEM THREAD 2	ROBOT THREAD 2
TH3	SYSTEM THREAD 3	ROBOT THREAD 3
TH4	SYSTEM THREAD 4	ROBOT THREAD 4

Figure 1-6 System 및 Robot 모드 스레드 구조

※ 멀티스레딩 기능은 실행 창 모니터링 기능에서 확인할 수 있으며, 멀티스레딩 기능을 사용하기 위한 함수 및 자세한 설명은 제 4장 9절 Multithreading을 참고하시기 바랍니다.

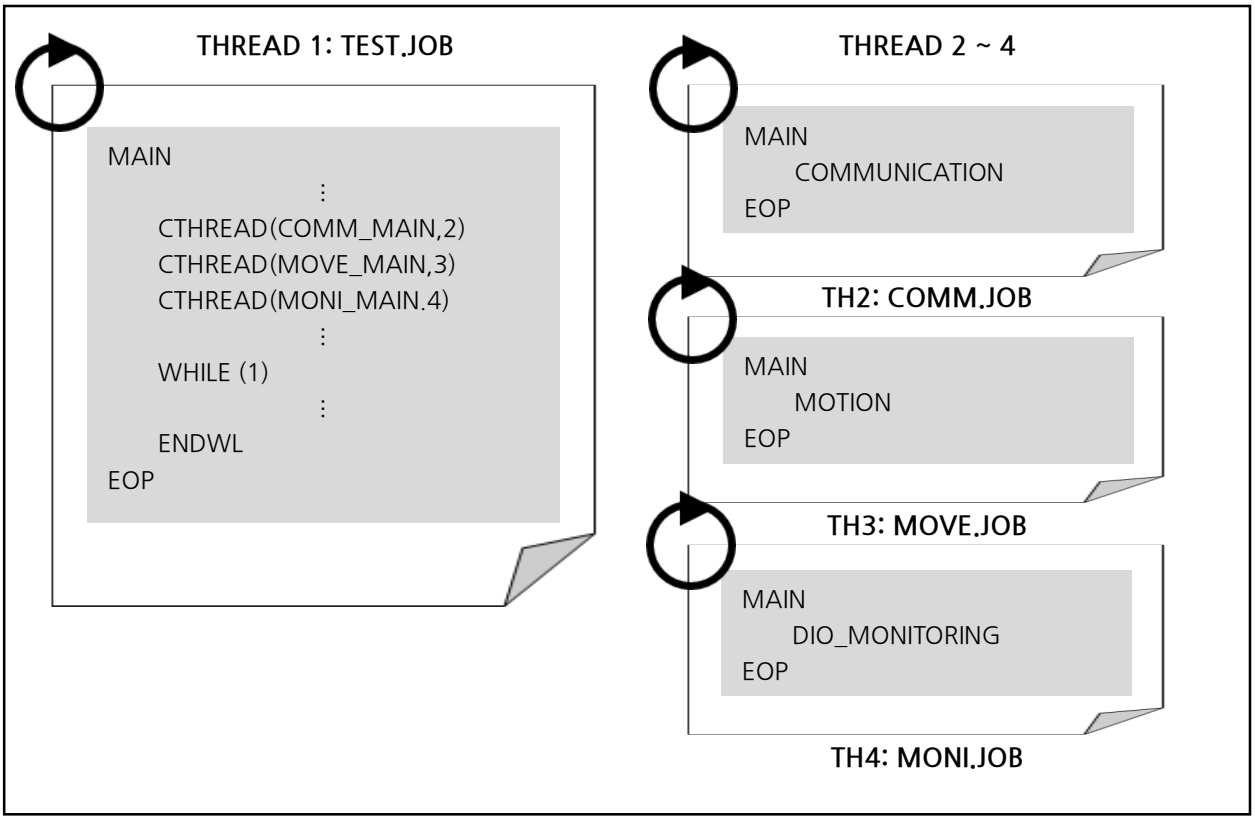


Figure 1-7 멀티스레딩 실행 예시

1) N2 제어기 내 System 모드의 메인 스레드에서는 BGT(Background Task) 파라미터 설정에 따라 BGT프로그램을 구동할 수 있습니다. 구동 가능한 BGT프로그램으로는 디지털 I/O 전용 BGTD.JOB 또는 필드 버스 전용 BGTF.JOB이 있습니다.
2) System 및 Robot 모드는 상호 독립적이기 때문에, System 모드의 메인 JOB 프로그램(BGTD.JOB, BGTF.JOB)은 Robot 모드에서 사용이 불가능합니다.

제 2장 언어 구성 요소 및 문법

1. 언어 구성 요소

로보스타 로봇 언어(RRL, Robostar Robot Language)는 아래와 같은 구성요소로 이루어져 있습니다.

- 식별자(Identifier): 구성 요소를 식별하는 이름
- 변수(Variable): 문자와 숫자로 구성되어 임시로 데이터를 저장하는 공간
- 상수(Constant): 프로그램 실행 중 변경할 수 없는 값
- 연산자(Operator): 특정 수학적 또는 논리적 연산을 수행하도록 지시하는 기호
- 식(Expression): 고유한 값을 얻기 위해 사용된 구성 요소의 조합
- 명령어(Command): 특정 기능을 수행하기 위해 내장된 명령어

2. 이름

명령어, 변수, 함수, 레이블 및 프로그램을 나타내는 이름은 아래의 규약을 따릅니다.

- 이름의 시작은 반드시 영문 대문자 혹은 밑줄로 시작해야 합니다.
- 이름의 시작을 제외하고 영문 대문자, 숫자 및 밑줄을 사용할 수 있습니다.
- 연산자로 사용되는 문자 및 영문 소문자, 영문 외 기타 언어는 사용할 수 없습니다.

3. 변수

변수는 프로그램에서 데이터를 저장할 공간을 만드는 데 사용됩니다. 변수에는 전역 변수 및 지역 변수, 시스템 변수가 있습니다. 시스템 변수는 모든 프로그램에서 사용할 수 있으며, 전역 변수는 모든 프로그램에서 사용할 수 있는 전역 변수와 프로그램에 종속되어 해당 프로그램 내에서만 사용할 수 있는 전역 변수로 구분됩니다. 지역 변수는 변수가 선언된 해당 블록 안에서만 유효합니다.

※ 변수에서 사용할 수 있는 데이터 타입의 종류와 사양은 제 2장 7절 데이터 타입 및 변환을 참고하시기 바랍니다

3.1 전역 변수

전역 변수는 정수형 및 실수형, 조인트 좌표계 위치형 변수가 있습니다. 알파벳 문자 뒤에 정수식이 추가된 형태로 표현됩니다. 전역 변수의 이름은 시스템에 의해 정의되므로, 선언 없이 사용할 수 있습니다. 전역 변수에는 다음 유형을 사용할 수 있습니다.

항목	모든 프로그램에서 공통적으로 사용 가능한 전역 변수
I	Integer type (4 bytes, range: -2,147,483,647 ~ 2,147,438,647) 사용 가능 개수: 1,000개 (0 ~ 999)
	Examples) I0, I[1]
F	Real type (4 bytes, range: 3.402,823E-38 ~ 3.402,823E+38) 사용 가능 개수: 1,000개 (0 ~ 999)
	Examples) F100, F[101]
GP	Global joint position type, 사용 가능 개수: 2,000개 (0 ~ 1,999)
	Examples) GP0, GP[1]

항목	프로그램에 종속된 전역 변수
P	Program joint position type (프로그램 내 값 수정 불가) 사용 가능 개수: 1,000개 (0 ~ 999)
	Examples) P0, P[1]

위의 변수 중 P 변수는 프로그램 내부에서 수정 불가능합니다.

전역 변수가 지정되면, 변수 번호는 식을 사용하여 지정할 수도 있습니다. 이를 간접 참조라고 합니다. 만약 간접 참조가 실행되면 변수 번호는 “[”, “]” 기호로 묶여서 표시됩니다.

〈Examples〉	
I0 = I[2*3]	// I0번 변수에 I6변수를 대입함
F1 = F[I12]	// F1에 I12번째 F변수를 대입함
GP1 = GP[I5+2]	// GP1에 I5번에 2를 더한 값의 GP변수를 대입함
JMOV P[0]	// 현재 위치에서 P0까지 JMOV 이동

※ 자세한 설명 및 사용 방법은 제 4장 2.7절 Program Position P ~ 제 4장 2.10절 Global Position GP를 참조하시기 바랍니다.

3.2 지역 변수

지역 변수로는 정수형 및 실수형, 문자열, 조인트 좌표계 위치형, 베이스 좌표계 위치형, 사용자 좌표계 위치형 변수가 있습니다. 지역 변수는 변수의 선언 후에 사용할 수 있습니다. 지역 변수의 이름은 시스템에 의해 정의되지 않으므로, 반드시 선언해야 합니다. 지역 변수를 참조하는 경우, 참조하기 전에 특정 값을 지역 변수(변수 초기화)에 할당해야 합니다. 초기화 없이 참조되면 실행 시 오류가 발생합니다.

〈Examples〉	
INT AA	// 변수의 선언
AA = 0	// 변수 초기화
INT BB = 4	// 에러 발생(선언과 동시에 초기화 함)

※ 선언과 동시에 변수의 초기화는 불가능합니다.

만약, 아래 표와 같이 "$" 기호를 사용해 변수를 선언한 경우, 전역 변수로 활용이 가능합니다. 즉, 프로그램 내에서 모두 사용할 수 있으며, 변수가 선언된 프로그램을 INCLUDE 명령어를 사용하여 참조 시 다른 프로그램에서도 사용할 수 있습니다.

※ "$"기호를 사용하여 전역 변수를 생성 및 활용하고자 하는 경우 메인 함수 또는 함수 외부에 변수를 선언해야 하며, 변수의 초기화는 함수 내부에서 이루어져야 합니다.

〈Examples〉	
INT $AA	// 정수형 전역 변수 선언
REAL $BB	// 실수형 전역 변수 선언
MAIN	// 메인 함수 선언
$AA = 10	// 정수형 전역 변수 초기화
$BB = 12.4	// 실수형 전역 변수 초기화
INIT()	// 함수 호출
EOP	// 메인 함수 종료
POS $ACUR	// 조인트 좌표계 위치형 전역 변수 선언
FUNC VOID INIT()	// 리턴 값이 없는(VOID) 함수 정의
$ACUR = 〈0,0,0,0,0,0〉	// 조인트 좌표계 위치형 전역 변수 초기화
END	// 함수 종료

※ 자세한 설명 및 사용 방법은 제 4장 2.1절 INT ~ 제 4장 2.6절 UPOS를 참조하시기 바랍니다.

3.3 시스템 변수

시스템 변수는 시스템의 상태를 확인하는 데 사용됩니다. 시스템 변수 이름은 시스템에 의해 보장되므로 변수 선언을 할 필요가 없습니다.

항목	내용
TMR	타이머 카운트 값 지정 혹은 현재 카운트 값을 읽어오는 데 사용 함
MVR	이동거리의 백분율을 확인
HERE	현재 위치를 조인트 값으로 반환 함
HERE_REF	현재 레퍼런스 위치를 조인트 값으로 반환 함

※ 자세한 설명 및 사용 방법은 제 4장 2.11절 TMR ~ 2.14절 HERE_REF를 참고하시기 바랍니다.

4. 함수

함수는 미리 결정된 동작 방식을 이용해 전달 인자의 결과를 얻는데 사용됩니다. 함수 유형에 따라 반환 값이나 전달 인자가 없는 함수도 존재합니다. 사용자가 함수를 정의하여 사용하는 것도 가능합니다. 사용자가 함수를 정의하여 사용하는 경우 전달 인자 및 반환 값은 모든 데이터 타입을 사용할 수 있습니다. 함수 선언 시 반환 값이 없는 경우 VOID를 사용해야 합니다.

※ 자세한 설명 및 사용 방법은 제 4장 5.2절 FUNC … END (함수 시작/종료)를 참고하시기 바랍니다.

5. 분기점

분기점은 프로그램 내 특정 위치를 문장으로 나타냅니다. 분기 위치의 이름을 지정하고 명령어를 사용하여 해당 위치로 이동할 수 있습니다.

※ 자세한 설명 및 사용 방법은 제 4장 5.7절 LABL … GOTO (분기 명령어)를 참고하시기 바랍니다

6. 프로그램

프로그램 내에서 다른 프로그램의 이름을 호출해 사용할 수 있습니다.

<Examples>	
MAIN	// 메인 함수 선언
JCALL MOTION	// MOTION.JOB 호출하여 실행
EOP	// 메인 함수 종료

7. 데이터 타입 및 변환

7.1 데이터 타입

사용자가 선언하여 사용할 수 있는 데이터 타입은 아래 표와 같습니다.

변수형		크기 및 범위	
		설명	사용 가능 개수
숫자	INT	4 bytes, -2,147,483,648 ~ 2,147,438,647	
		Integer type	메모리 범위 내
	REAL	4 bytes, 3.402,823E-38 ~ 3.402,823E+38	
		Real type	메모리 범위 내
문자열	DEFSTR	100자 이내	
		Integer type	1,000개
위치형	POS	48 bytes (J1, J2, J3, J4, J5, J6, E1, E2)	
		Joint position type	메모리 범위 내
	XPOS	48 bytes (X, Y, Z, RX, RY, RZ, E1, E2, ARM, T)	
		Base position type	메모리 범위 내
	UPOS	48 bytes (Xu, Yu, Zu, RX, RY, RZ, E1, E2, ARM, T, U)	
		User position type	메모리 범위 내

사용자가 지역변수를 사용하는 경우 선언할 수 있는 개수는 고정된 메모리 사이즈에 의해 제한됩니다. 아래 표와 같이 지역 변수로 사용하는 경우 1Mbytes 숫자, 위치형 변수를 선언할 수 있고 1,000개의 문자열 변수를 사용할 수 있습니다. “$” 기호를 사용하여 전역변수로 사용하는 경우 동일한 사이즈의 별도의 메모리 영역을 사용합니다. 따라서 최대 2Mbyes의 숫자, 위치형 변수를 선언할 수 있고 2,000개의 문자열 변수를 사용할 수 있습니다.

항목	지역 변수	전역 변수	합계
숫자, 위치형 변수	1Mbytes	1Mbytes	2Mbytes
문자열 변수	1,000개 (100자 이하)	1,000개 (100자 이하)	2,000개

1) 메모리를 초과하여 사용하는 경우 알람이 발생합니다.

위치형 타입의 경우 변수 명에 기호 “.”를 사용해 각 개별 항목에 접근이 가능합니다. 접근할 수 있는 위치 데이터의 개수는 축 개수와 자유도(Degree of freedom, 이하 DOF) 값에 따라 달라질 수 있습니다. 상수를 사용하여 초기화 할 때와는 다르게 Table 2-1의 모든 항목에 접근하여 읽기 또는 쓰기 기능을 수행 할 수 있습니다.

분류	예시
J1축 (또는 X축)	<변수명>.1
J2축 (또는 Y축)	<변수명>.2
J3축 (또는 Z축)	<변수명>.3
J4축 (또는 RX축)	<변수명>.4
J5축 (또는 RY축)	<변수명>.5
J6축 (또는 RZ축)	<변수명>.6
J7축 (E1축)	<변수명>.7
J8축 (E2축)	<변수명>.8
로봇의 자세	<변수명>.ARM
Tool 번호	<변수명>.T
User 번호	<변수명>.U
변수 초기화 완료 유무	<변수명>.USED

Table 2-1 위치형타입 변수의 세부항목

6축 수직다관절 로봇을 사용하는 경우 1 ~ 6축은 로봇의 각 조인트 별 축을 의미합니다. 7 ~ 8축은 로봇의 부가 축 파라미터 설정에 따라 부가 축 설정 시 사용됩니다.

로봇의 자세는 Table 2-2와 같이 입력할 수 있습니다. 0 값을 입력하는 경우 현재 로봇의 JOINT 값을 기준으로 자세를 설정합니다. 각 자세에 대한 보다 자세한 설명은 N2 조작 및 운용 설명서(N2-OM-K□□)를 참고하시기 바랍니다.

Value	Shoulder	Elbow	Wrist
000	NOT DEFINED	NOT DEFINED	NOT DEFINED
001	NOT DEFINED	NOT DEFINED	FLIP
002	NOT DEFINED	NOT DEFINED	NON FLIP
010	NOT DEFINED	ABOVE	NOT DEFINED
011	NOT DEFINED	ABOVE	FLIP
012	NOT DEFINED	ABOVE	NON FLIP
020	NOT DEFINED	BELOW	NOT DEFINED
021	NOT DEFINED	BELOW	FLIP
022	NOT DEFINED	BELOW	NON FLIP
100	REAR	NOT DEFINED	NOT DEFINED
101	REAR	NOT DEFINED	FLIP
102	REAR	NOT DEFINED	NON FLIP
110	REAR	ABOVE	NOT DEFINED
111	REAR	ABOVE	FLIP
112	REAR	ABOVE	NON FLIP
120	REAR	BELOW	NOT DEFINED
121	REAR	BELOW	FLIP
122	REAR	BELOW	NON FLIP
200	FRONT	NOT DEFINED	NOT DEFINED
201	FRONT	NOT DEFINED	FLIP
202	FRONT	NOT DEFINED	NON FLIP
210	FRONT	ABOVE	NOT DEFINED
211	FRONT	ABOVE	FLIP
212	FRONT	ABOVE	NON FLIP
220	FRONT	BELOW	NOT DEFINED
221	FRONT	BELOW	FLIP
222	FRONT	BELOW	NON FLIP

Table 2-2 로봇 자세

Tool 및 User 번호는 각각 0 ~ 15까지 총 16개의 입력이 가능합니다.

마지막으로 변수 초기화 유무 (0: 초기화 안됨, 1: 초기화 완료)를 입력할 수 있습니다. 다만, 상수로 변수를 초기화 할 경우 변수 초기화 유무 값은 1로 자동 설정됩니다. 변수 초기화 유무가 0으로 설정되어 있는 위치형 변수를 이동 명령어에서 참조하는 경우 알람이 발생합니다.

※ 자세한 설명 및 사용 방법은 제 4장 2.4절 POS ~ 제 4장 2.7절 Program Position P를 참조하시기 바랍니다.

1) 각 개별 항목에 접근하여 초기화 하는 경우 반드시 USED 항목에 값을 초기화 완료값(1)을 입력하여 초기화가 완료 되었음을 알려주십시오

7.2 데이터 타입 변환

아래의 규칙에 따라 서로 변환될 수 있습니다. 아래의 규칙을 만족시키지 못하는 경우 알람이 발생할 수 있습니다.

※ 데이터 타입 변환 함수는 제 4장 10.44절 FTOS, 제 4장 10.54절 SVAL, 제 4장 10.3절 JTOB ~ 제 4장 10.8절 UTOB를 참고하시기 바랍니다.

1) 정수와 실수 값

- 다른 유형의 변수에 할당된 경우, 숫자 값은 변수 유형을 충족하도록 변환됩니다.
- 실수가 정수로 변환되면 반올림하여 결과가 출력됩니다.

2) 문자와 숫자 값

변환 전 \ 변환 후	문자	정수
문자		ASC
정수	BINS, CHR	

- 예시) 문자 데이터는 ASC 함수를 사용해서 정수 데이터로 변환됩니다.

3) 위치형 타입

변환 전 \ 변환 후	POS	XPOS	UPOS
POS		JTOB	JTOU
XPOS	BTOJ		BTOU
UPOS	UTOJ	UTOB	

- 예시) POS 데이터는 JTOB 함수를 사용해서 XPOS 데이터로 변환됩니다.

8. 상수

상수는 고정 값을 갖는 표현 식 입니다. 아래 표와 같이 분류 됩니다.

분류		설명
정수	Binary	값으로 0 또는 1만 사용해 표현. 앞에 0B를 붙여 작성
		Example) 0B11
	Decimal	0 ~ 9의 숫자를 사용하여 표현
		Example) 100
	Hexadecimal	0 ~ 9, A ~ F 까지 사용해 표현, 앞에 0H를 붙여 작성
		Example) 0HFF
실수		“.” 기호를 이용하여 소수점 이하 자리까지 표현
		Example) 2.3456
문자열		따옴표를 사용하여 문자열을 표시
		Example) “String statement”
위치형	POS	“<” 및 “>” 기호를 사용하여 표시
		Example) <100, 100, 100, 100, 100, 100>
	XPOS	ARM (자세), T(Tool 번호) 까지 표현 입력
		Example) <100, 100, 100, -30, 30, 60, ARM = 0, T = 2>
	UPOS	ARM (자세), T(Tool 번호), U(User 번호) 까지 입력
		Example) <0, 0, 0, -30, 30, 60, ARM = 0, T = 1, U = 3>

9. 식과 연산자

식은 값을 반환하는 데 사용됩니다. 독립적인 값을 갖는 식과 연산자를 통해 연결되어 여러 요소로 구성된 표현 식이 있습니다.

※ 자세한 설명 및 사용 방법은 제 4장 4절 연산자를 참고하시기 바랍니다.

9.1 연산자 우선순위

연산자	설명	우선순위
(,)	괄호	High
~, !	비트 및 논리 NOT	↑
*, /, %	곱하기, 나누기, 나머지	↑
+, -	덧셈, 뺄셈	↑
<<, >>	왼쪽으로 Shift, 오른쪽으로 Shift	↑
<, <=, >, >=	작음, 작거나 같음, 큼, 크거나 같음	↑
==, !=	같음, 같지 않음	↑
&	비트 AND	↑
^	비트 XOR	↑
|	비트 OR	↑
&&	논리 AND	↑
||	논리 OR	↓
=	대입	Low

9.2 대입연산자

"=" 기호를 사용하여 왼쪽 변수에 식의 결과 또는 값을 대입합니다.
선언한 변수형에 따라 형 변환이 될 수 있습니다.

<Examples>

```
INT A, B, C, D
IN0 = 1                    // IN 명령어
A = 100
B = I5
C = F1
D = A*(B+C)
```

9.3 산술연산자

더하기(+) 및 빼기(-), 곱하기(*), 나누기(/), 나눗셈의 나머지(%)를 구합니다. 나눗셈의 나머지를 구하는 경우 부호는 Table 2-3의 규칙과 같이 결정됩니다.

Divisor / Dividend	(+)	0	(-)
(+)	(+)	Error	(+)
0	0	Error	0
(-)	(-)	Error	(+)

Table 2-3 산술 연산자

1) 0으로 나누거나 나머지를 구하는 경우, 프로그램 실행 시 “[1236] Interpreter error” 알람이 발생합니다.
2) 산술 연산자 사용으로 인해 오버 플로우가 발생할 수 있으며 알람은 발생하지 않습니다. 최대 허용치 값을 유의하시기 바랍니다.

9.4 관계연산자

관계연산자는 두 개의 숫자 값을 비교하는 데 사용됩니다. 결과는 논리 값(1: true, 0: false)으로 얻어집니다. 주로 흐름 제어 명령문의 조건식에 사용됩니다.

관계연산자	설명
<	작음
>	큼
<=	작거나 같음
>=	크거나 같음
==	같음
!=	같지 않음

9.5 논리 및 비트 연산자

논리 연산자는 하나 또는 여러 개의 조건식을 연결하는데 사용되는 기호입니다. 비트 연산자는 비트 단위로 논리적 대응을 하는데 사용되는 기호입니다. 각 기호 별 의미는 아래 표와 같습니다.

	논리	비트
Negation	!	~
Product	&&	&
Addition	||	|
Exclusive addition		^

9.6 배열

로보스타 로봇 언어(RRL, Robostar Robot Language)에서는 1차원 배열의 사용이 가능합니다. 배열이란 같은 타입의 변수들로 이루어진 집합을 의미합니다. 배열을 구성하는 각각의 값을 배열 요소(Element)라고 하며, 배열에서의 위치를 가리키는 숫자는 인덱스(Index)라고 합니다. 인덱스는 0부터 시작하며, 0을 포함한 양의 정수만 가질 수 있습니다.

〈Syntax〉
〈변수 타입〉 〈배열 이름〉'['〈배열 길이〉']'
〈Examples〉
INT AA[10] AA[0] = 10

1) C언어의 포인터 사용은 불가능합니다.
2) 배열의 선언과 동시에 초기화는 불가능합니다.
3) 전체 메모리 사이즈(1MB)를 초과하지 않는 범위 안에서 사용하십시오.

9.7 문자열 운용

두 개의 문자열을 덧셈 기호 “+”를 사용하여 더할 수 있습니다. 덧셈 기호 “+”는 상수 또는 배열 모두 사용 가능합니다.

〈Syntax〉
〈문자열 변수〉 = 〈문자열〉 + 〈문자열〉 〈문자열 변수〉 = 〈문자열 변수〉 + 〈문자열〉 〈문자열 변수〉 = 〈문자열 변수〉 + 〈문자열 변수〉
〈Examples〉
DEFSTR STR1, STR2, STR3 STR1 = “TEST “ + “STATEMENT ” STR2 = STR1 + “COMPLETE” STR3 = STR1 + STR2

1) 더한 문자열이 100자를 초과하지 않도록 주의하십시오.

9.8 위치형 운용

위치형 데이터들은 아래와 같은 작업이 가능합니다.

<Syntax>

<위치형 변수> = <위치형 상수 또는 변수> + <위치형 상수 또는 변수>
<위치형 변수> = <위치형 상수 또는 변수> - <위치형 상수 또는 변수>
<위치형 변수> = <위치형 상수 또는 변수> * <숫자 변수 또는 상수>
<위치형 변수> = <위치형 상수 또는 변수> / <숫자 변수 또는 상수>

<Examples>

```
POS RES_P, AP, BP
REAL A, B

A = 10.1
B = -7.2
AP = <50, 50, 50, 50, 50, 50>
BP = <30, 30, 30, 30, 30, 30>

RES_P = <100, 100, 100, 100, 100, 100> + <100, 100, 100, 100, 100, 100>
PRINT(0,RES_P)
// <200.000,200.000,200.000,200.000,200.000,200.000,0.000,0.000>

RES_P = AP + <100, 100, 100, 100, 100, 100>
PRINT(1,RES_P)
// <150.000,150.000,150.000,150.000,150.000,150.000,0.000,0.000>

RES_P = <100, 100, 100, 100, 100, 100> + BP
PRINT(2,RES_P)
// <130.000,130.000,130.000,130.000,130.000,130.000,0.000,0.000>

RES_P = AP + BP
PRINT(3,RES_P)
// <80.000,80.000,80.000,80.000,80.000,80.000,0.000,0.000>

RES_P = <100, 100, 100, 100, 100, 100> * A
PRINT(4,RES_P)
// <1010.000,1010.000,1010.000,1010.000,1010.000,1010.000,0.000,0.000>

RES_P = <100, 100, 100, 100, 100, 100> / B
PRINT(5,RES_P)
// <-13.889,-13.889,-13.889,-13.889,-13.889,-13.889,-0.000,-0.000>

RES_P = AP * A
PRINT(6,RES_P)
```

```
// <505.000,505.000,505.000,505.000,505.000,505.000,0.000,0.000>

RES_P = BP / B
PRINT(7,RES_P)
// <-4.167,-4.167,-4.167,-4.167,-4.167,-4.167,-0.000,-0.000>
```

1) 서로 다른 위치형 변수끼리는 사용할 수 없습니다. 반드시 동일한 타입의 위치형 변수를 사용해 주십시오.
2) 다른 위치형 변수로 변환하고자 하는 경우에는 제 4장 10.3절 JTOB ~ 제 4장 10.8절 UTOB 절의 변환 함수를 사용하십시오.

제 3장 명령어 리스트

1. 변수 및 상수

1.1 일반 변수

구분		설 명 형 식	사용 예	Ref.
숫자	INT	정수형 변수를 선언 (4 bytes) INT <변수명>	INT AA AA = 0	2.1
	REAL	실수형 변수를 선언 (4 bytes) REAL <변수명>	REAL BB BB = 0.0	2.2
DEFSTR		문자열 변수를 선언 DEFSTR <변수명>	DEFSTR SS SS = "STRING"	2.3
POS		조인트 좌표계 위치형 변수를 선언 POS <변수명>	POS AP AP = <10,10,10,10,10,10>	2.4
XPOS		베이스 좌표계 위치형 변수를 선언 XPOS <변수명>	XPOS BP BP = <0,0,0,0,0,0,ARM = 0,T = 0>	2.5
UPOS		사용자 좌표계 위치형 변수를 선언 UPOS <변수명>	UPOS CP CP = <0,0,0,0,0,0,ARM = 0,T = 0,U = 0>	2.6
P		미리 정의된 조인트 좌표계 위치형 변수로써 프로그램 파일에 저장되며, 해당 프로그램 내에서 전역 변수로 사용됨 **※ JOB 파일에 저장되는 프로그램 조인트 위치형 변수 P는 프로그램 내에서 직접 편집이 불가능함** P<변호> or P[번호]	JMOV P0 JMOV P[1]	2.7
I		미리 정의된 전역 정수 변수 I<번호> or I[번호]	I0 = 100 I[1] = 50	2.8
F		미리 정의된 전역 실수 변수 F<번호> or F[번호]	F0 = 3.14 F[1] = 10.54	2.9
GP		미리 정의된 전역 조인트 좌표계 위치형 변수 GP<변호> or GP[번호]	GP0 = <0,0,0,0,0,0> GP[1] = <10,10,10,10,10,10>	2.10

1) 변수명은 숫자로 시작할 수 없습니다.
2) 변수명은 대문자 알파벳과 숫자로 구성됩니다.
3) 위치형 변수들은 변수명 뒤에 '.' 사용 후 축 번호 또는 옵션을 지정하여 개별 접근할 수 있습니다.
4) 변수 사용에 관한 자세한 설명은 제 2장 3절 변수 및 제 2장 7절 데이터 타입 및 변환, 제 4장 2절 변수 절의 상세 페이지를 참고하시길 바랍니다.
5) 변수명 앞에 $기호를 붙이는 경우 전역 변수로 사용이 가능합니다.

1.2 시스템 변수

구분	설 명 형 식	사용 예	Ref.
TMR(0) TMR(1)	타이머 함수, 정수 값을 입력하는 순간부터 값이 할당되어 10ms 간격마다 값이 1씩 증가 <번호>: 타이머 번호를 선택 (0 또는 1) TMR[(]<번호>[)] = <값> <정수형 변수> = TMR[(]<번호>[)]	INT AA TMR0 = 50 TMR(1) = 0 AA = TMR(0)	2.11

MVR	이동거리의 백분율을 확인
	<로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3)
	<로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용
	<실수형 변수> = MVR[(<로봇 번호>)]
	MVR
		VALUE1 = MVR
		VALUE2 = MVR(1)
		IF MVR <50 THEN
		ENDIF
			2.12

HERE	현재 위치를 조인트 값으로 반환 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <위치형 변수> = HERE[(<로봇 번호>)]	POS AP AP = HERE AP = HERE1 AP = HERE(1)	2.13
HERE_REF	현재 레퍼런스 위치를 조인트 값으로 반환 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <위치형 변수> = HERE_REF[(<로봇 번호>)]	POS AP AP = HERE_REF AP = HERE_REF1 AP = HERE_REF(1)	2.14

1.3 상수

구분	설 명 형 식	사용 예	Ref.
숫자 상수	정수, 실수, 2진 정수, 16진 정수 표시 [0H/0B]<숫자> 숫자 앞에 0H 또는 0B가 없으면 10진수	INT AA AA = 0H12 // 16진수 AA = 0B11 // 2진수	-
문자열 상수	문자열을 표시, 최대 길이 100자 이하 "<문자열>"	DEFSTR STR STR = "STRING CONSTANT"	-
위치형 상수	위치형 상수의 개별 값을 표시 POS: 축(위치값) XPOS: 축(위치값), ARM(자세), T(Tool 번호), UPOS: 축(위치값), ARM(자세), T(Tool 번호), U(User 번호) ※<자세> 할당 시 XPOS위치형 상수로 정의 ※<User 번호> 할당 시 UPOS위치형 상수로 정의 <(위치값)[,<자세>][,<Tool 번호>][,<User 번호>]>	POS AP AP = <1, 2, 3, 4, 5, 6> XPOS XP XP = <0,0,0,0,0,0,ARM = 0,T = 1> UPOS UP UP = <1,2,3,4,5,6,ARM = 0,T = 1, U = 1>	-

2. 전처리기 및 주석

구분	설 명 형 식	사용 예	Ref.
INCLUDE	특정 프로그램 파일을 현재 위치에 첨부하여 하나의 프로그램처럼 사용할 수 있게 함 INCLUDE “<JOB 파일명>”	INCLUDE “TEST”	3.1
/*, */	여러 줄 주석 /* <내용> */	/* JMOV GP0 DLAY 1000 */	3.2
//	한 줄 주석 // <내용>	// JMOV GP0 DLAY 1000	

3. 연산자

구분	설 명 형 식	사용 예	Ref.
=	대입연산자 <변수> = <변수> or <식>	B = A	4.1
+, -, *, /, %	산술 연산자 (가, 감, 승, 제, modulus) <변수> = <A항> <산술연산자> <B항>	C = A+B	4.2
>, <, >=, <=, !=, ==	관계연산자 (초과, 미만, 이상, 이하, 상이, 상동) [<숫자 변수> =]<A항> <관계연산자> <B항>	IF A > B THEN ENDIF	4.3

&&, ||, !	논리연산자
	(AND, OR, Negation)
	[<숫자 변수> =] <A항> <논리연산자> <B항>
		C = !A
		IF (A == 1) && (B == 0) THEN
		ENDIF
			4.4

&, | , ~,
^, <<, >>
	비트연산자
	(BAND, BOR, Complement,
	BXOR, left shift, right shift)
	[<숫자 변수> =]<A항> <비트연산자> <B항>
		C = A << 0H01
		IF A & B THEN
		ENDIF
			4.5

( )	괄호 (<식>)	D = (A+B)*C	4.6

4. 프로그램 제어

구분	설 명 형 식	사용 예	Ref.
MAIN … EOP	주 프로그램 시작 및 종료 MAIN … EOP	MAIN JMOV P1 JMOV P2 EOP	5.1
FUNC … END	함수의 선언 및 종료, 반환 유형(VOID 포함) 및 매개변수 선언 가능. 특정 값 반환 FUNC <반환타입> <함수명>(<매개변수>) [RET <반환 값>] END	FUNC VOID TEST() VEL 200 JMOV P0 END	5.2
RET	함수 종료 및 값을 반환 RET <반환 값>	FUNC INT TEST() RET 1 END	5.3
FOR … TO … [BY …] NEXT	지정된 횟수만큼 실행하는 루프를 생성함. BY <증분량>은 생략 시 '1'로 정의. FOR <초기화문> TO <종료 조건> [BY <증분량>] … NEXT	… FOR AA = 1 TO 5 [BY 2] BB = BB+AA NEXT …	5.4
WHILE … ENDWL	조건 반복 수행 WHILE <조건식> … ENDWL	… WHILE IN0 == 1 JMOV P1 ENDWL …	5.5

IF … THEN … [ELSE IF … THEN] [ELSE …] ENDIF	조건 판단 수행
	IF <조건식> THEN
	…
	[ELSEIF <조건식> THEN]
	…
	[ELSE]
	…
	ENDIF
		…
		IF IN3 == 1 THEN
		GOTO A0
		ELSE
		JMOV P1
		ENDIF
		LABL A0
		…
			5.6

LABL … GOTO	분기점의 위치를 지정하고, 분기점으로 이동 LABL <분기점 명> … GOTO <분기점 명>	… LABL A1 JMOV P1 JMOV P2 GOTO A1 …	5.7

CONTINUE	반복문의 처음으로 복귀
	WHILE <조건식>
	…
	CONTINUE
	ENDWL
		WHILE 1
		IF IN(1) == 1 THEN
		CONTINUE
		ENDIF
		JMOV P0
		ENDWL
			5.8

구분	설 명 형 식	사용 예	Ref.

BREAK	반복문을 정지하고 탈출
	WHILE <조건식>
	…
	BREAK
	ENDWL
		WHILE 1
		IF IN(1) == 1 THEN
		BREAK
		ENDIF
		JMOV P0
		ENDWL
			5.9

JCALL	JOB 파일을 호출하여 실행 JCALL <JOB 파일명>	JCALL INC	5.10

STOP	로봇 동작 정지
	STOP
		IF IN0 == 1 THEN
		STOP
		ENDIF
			5.11

EXIT	JOB 프로그램 수행 정지
	EXIT
		IF IN0 == 1 THEN
		EXIT
		ENDIF
			5.12

5. 로봇 동작

구분	설 명 형 식	사용 예	Ref.
SVON	서보 전원 ON, <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 SVON[(<로봇 번호>)]	SVON SVON(1)	6.1
SVOF	서보 전원 OFF, <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 SVOF[(<로봇 번호>)]	SVOF SVOF(1)	6.2
JMOV	현재 위치에서 목표점으로 PTP 이동 JMOV <위치형 변수>[, <VEL = 값>, <ACC = 값>, <DEC = 값>, <FOS = 값>, <T = 값>, <U = 값>]	JMOV P1 JMOV P2, VEL = 90, ACC = 20, DEC = 20	6.3
LMOV	현재 위치에서 목표점으로 직선보간 이동 LMOV <위치형 변수>[, <VEL = 값>, <ACC = 값>, <DEC = 값>, <FOS = 값>, <T = 값>, <U = 값>]	LMOV P1 LMOV P2, VEL = 90, ACC = 50, DEC = 50	6.4
CMOV	현재 위치에서 경유점 1, 2를 잇는 원을 그리며 이동 CMOV <경유점1> <경유점2>[, <VEL = 값>, <ACC = 값>, <DEC = 값>, <T = 값>, <U = 값>]	CMOV P1 P2 CMOV P1 P2, VEL = 100	6.5
AMOV	현재 위치에서 경유점 1과 목표점을 잇는 원호를 그리며 이동 AMOV <경유점1> <목표점> [, <VEL = 값>, <ACC = 값>, <DEC = 값>, <T = 값>, <U = 값>]	AMOV P1 P2, VEL = 100	6.6

6. 로봇 동작 조건

구분	설 명 형 식	사용 예	Ref.
WITH … ENDWT	로봇 동작 중 블록 내 수행문 동시 처리 WITH … ENDWT	WITH JMOV P1 WHILE MVR < 80 ENDWL OUT0 = 1 ENDWT	7.1
VEL	축 이동속도의 천분율(%)을 설정 축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001 **※ 초기속도 및 정격속도는 파라미터에서 설정** VEL <천분율 값(%)>	VEL 200 JMOV P1	7.2
ACC	가속 시간의 백분율(%)을 설정 가속시간 = 정격가속시간 X 백분율 값 X 0.01 **※ 정격가속시간은 파라미터에서 설정** ACC <백분율 값(%)>	ACC 50 JMOV P1	7.3
DEC	감속 시간의 백분율(%)을 설정 감속시간 = 정격감속시간 X 백분율 값 X 0.01 **※ 정격감속시간은 파라미터에서 설정** DEC <백분율 값(%)>	DEC 200 JMOV P1	7.4
FOS	축 선단이 목표점에 도달하기 전에 다음 목표점까지의 거리비율(%)을 참조하여 궤적을 변경 함 거리비율은 전체 이동거리의 백분율(%) JMOV, LMOV 에 적용 가능 FOS <거리비율(%)>	FOS 5 JMOV P1 FOS 0 // 해제	7.5
DLAY	지연시간 설정, 지연시간 단위는 1ms DLAY <지연시간>	JMOV P1 DLAY 1000	7.6
OFFS	목표점을 지정된 값만큼 이동 위치형 변수만 입력 가능 (상수 직접 대입 불가) OFFS <편차 값>	OFFS P1 JMOV P2	7.7
OFFSCLR	설정된 OFFS 값을 해지 OFFSCLR	OFFSCLR	7.8
LIMT	각 축의 이동범위를 제한 위치형 변수만 입력 가능 (상수 직접 대입 불가) LIMT <(-)이동범위, (+)이동범위>	LIMT P0 P1	7.9
TRQ	토크 리밋 값을 설정하고, 설정한 토크 리밋 값을 초과한 경우 알람 발생 TRQ <로봇 번호> <축 번호> <토크 리밋 값>	TRQ 1 2 100	7.10
GAIN	게인 그룹을 전환 GAIN <그룹 번호>	GAIN 1 JMOV P0	7.11
IPE	목표 위치 도달까지 허용 시간의 최대 값 설정, 단위 ms	IPE 100	7.12

7. 입출력 제어

7.1 일반 디지털 입출력

구분	설 명 형 식	사용 예	Ref.
IN	비트 입력을 비트 단위로 확인 <비트번호>: 0 ~ 19 (확장 보드 연결 시 최대 83) <대기시간>: 100 ~ 99,999 msec <대기시간> 또는 입력 조건이 만족될 때까지 대기 IN <비트번호> 또는 IN(<비트번호> [, <대기시간>]) = <0/1> <정수형 변수> = IN<비트번호>	INT AA IN10 = 1 AA = IN(0) IN(1,1000) = 1	8.1
OUT	비트 출력을 비트 단위로 출력 또는 확인 ON(= 1), OFF(= 0) <비트번호>: 0 ~ 19 (확장 보드 연결 시 최대 83) <펄스 폭>: 단위 ms, 입력 시 유효시간 이후 이전 상태로 복귀 <펄스 주기>: 단위 ms, 펄스 주기 시간을 입력 **※ <펄스 주기>는 <펄스 폭> 보다 항상 커야 함** OUT[(]<비트번호>[)] =<0/1> OUT[(]<비트번호>[)] =<0/1><펄스 폭> OUT[(]<비트번호>[)] =<0/1><펄스 폭><펄스 주기> <정수형 변수> = OUT<비트번호>	INT AA OUT0 = 1 OUT0 = 1 100 OUT0 = 1 100 200 AA = OUT(0)	8.2
BIN	바이트 입력을 정수 단위로 확인 <바이트번호>: 0 ~ 2 (확장 보드 연결 시 최대 10) <대기시간>: 100 ~ 99,999 msec <대기시간> 또는 입력 조건이 만족될 때까지 대기 BIN<바이트번호> 또는 BIN(<바이트번호> [, <대기시간>]) = <0/1> <정수형 변수> = BIN<바이트번호>	INT AA BIN1 = 124 AA = BIN(0) BIN(0,1000) = 255	8.3
BOUT	바이트 출력을 정수 단위로 출력 또는 확인 <바이트번호>: 0 ~ 2 (확장 보드 연결 시 최대 10) BOUT[(]<바이트번호>[)] = <출력지정 값> <정수형 변수> = BOUT<바이트번호>	INT AA BOUT0 = 0 BOUT(0) = 128 AA = BOUT(1)	8.4
PIN	포트 입력을 정수 단위로 확인 <포트번호>: 0 ~ 1 (확장 보드 연결 시 최대 5) <정수형 변수> = PIN[(]<포트번호>[)]	INT AA, BB AA = PIN(0) BB = PIN1	8.5
POUT	포트 출력을 정수 단위로 출력 또는 확인 <포트번호>: 0 ~ 1 (확장 보드 연결 시 최대 5)	INT AA POUT0 = 0 POUT(0) = 20 AA = POUT(0)	8.6

1) OUT 또는 COUT 명령어를 사용하여 임펄스 신호 또는 펄스 주기 파형을 생성하는 경우 <펄스 폭> 및 <펄스 주기> 값의 입력 단위는 1ms 입니다. 단, 2ms 이하 값은 무시하므로 입력 값은 2의 정수배로 입력 바랍니다.

7.2 필드버스 입출력

구분	설 명 형 식	사용 예	Ref.
CIN	비트 입력을 비트 단위로 확인 <대기시간>: 100 ~ 99,999 msec <대기시간> 또는 입력 조건이 만족될 때까지 대기 CIN <비트번호> 또는 CIN(<비트번호> [, <대기시간>]) = <0/1> <정수형 변수> = CIN<비트번호>	INT AA CIN0 = 1 AA = CIN(1) CIN(1,2000) = 1	8.7
COUT	비트 출력을 비트 단위로 출력 또는 확인 ON(= 1), OFF(= 0) <펄스 폭>: 단위 ms, 입력 시 유효시간 이후 이전 상태로 복귀 <펄스 주기>: 단위 ms, 펄스 주기 시간을 입력 **※ <펄스 주기>는 <펄스 폭> 보다 항상 커야 함** COUT[(]<비트번호>[)] = <0/1> COUT<비트번호> = <0/1><펄스 폭> COUT<비트번호> = <0/1><펄스 폭><펄스 주기> <정수형 변수> = COUT<비트번호>	INT AA, BB COUT0 = 1 COUT0 = 1 100 COUT0 = 1 100 200 AA = COUT0 BB = COUT(1)	8.8
CBIN	바이트 입력을 정수 단위로 확인 <대기시간>: 100 ~ 99,999 msec <대기시간> 또는 입력 조건이 만족될 때까지 대기 CBIN<바이트번호> 또는 CBIN(<바이트번호> [, <대기시간>]) = <0/1> <정수형 변수> = CBIN<바이트번호>	INT AA CBIN1 = 255 AA = CBIN(0) CBIN(1,2000) = 128	8.9
CBOUT	바이트 출력을 정수 단위로 출력 또는 확인 CBOUT[(]<바이트번호>[)] = <출력지정 값> <정수형 변수> = CBOUT<바이트번호>	INT AA CBOUT0 = 0HFF AA = CBOUT(1)	8.10
CWIN	워드 입력을 정수단위로 확인 <정수형 변수> = CWIN[(]<워드번호>[)]	INT AA, BB AA = CWIN0 BB = CWIN(1)	8.13
CWOUT	워드 출력을 정수 단위로 출력 또는 확인 CWOUT[(]<워드번호>[)] = <출력지정 값> <정수형 변수> = CWOUT[(]<워드번호>[)]	INT AA CWOUT0 = 0HFFFF AA = CWOUT(1)	8.14
CDIN	더블 워드 입력을 정수 단위로 확인 <정수형 변수> = CDIN[(]<더블워드번호>[)]	INT AA, BB AA = CDIN0 BB = CDIN(0)	8.11
CDOUT	더블 워드 출력을 정수 단위로 출력 또는 확인 CDOUT[(]<더블워드번호>[)] = <출력지정 값> <정수형 변수> = CDOUT<더블워드번호>	INT AA CDOUT1 = 0HFFFFFFFF AA = CDOUT(0)	8.12
CFIN	더블 워드 입력을 실수 단위 확인 <실수형 변수> = CFIN[(]<더블워드번호>[)]	REAL AA, BB AA = CFIN0 BB = CFIN(0)	8.15
CFOUT	더블 워드 출력을 실수 단위로 출력 또는 확인 CFOUT[(]<더블워드번호>[)] = <출력지정 값> <실수형 변수> = CFOUT<더블워드번호>	REAL AA CFOUT1 = 0HFFFFFFFF AA = CFOUT(0)	8.16

7.3 CNET 입출력

구분	설 명 형 식	사용 예	Ref.
XIN	CNET 또는 MODBUS 디바이스 영역의 비트 입력을 비트 단위로 확인 <정수형 변수> = XIN(<디바이스 영역>, <비트 번호>)	INT DATA DATA = XIN(“M”, 0)	8.17
XOUT	CNET 또는 MODBUS 디바이스 영역의 비트 출력을 비트 단위로 출력 또는 확인 XOUT(<디바이스 영역>, <비트 번호>) = <0/1> <정수형 변수>=XOUT(<디바이스 영역>, <비트 번호>)	INT DATA XOUT(“M”, 0) = 1 DATA = XOUT(“M”, 0)	8.18
XBIN	CNET 또는 MODBUS 디바이스 영역의 바이트 입력을 정수 단위로 확인 <정수형 변수>=XBIN(<디바이스 영역>, <바이트번호>)	INT DATA DATA = XBIN(“D”, 0)	8.19
XBOUT	CNET 또는 MODBUS 디바이스 영역의 바이트 출력을 정수 단위로 출력 또는 확인 XBOUT(<디바이스 영역>,<바이트 번호>)=<출력 값> <정수형 변수> = XBOUT(<디바이스 영역>, <바이트 번호>)	INT DATA XBOUT(“D”, 0) = 0HFF DATA = XBOUT(“D”, 0)	8.20
XWIN	CNET 또는 MODBUS 디바이스 영역의 워드 입력을 정수단위로 확인 <정수형 변수>=XWIN(<디바이스 영역>, <워드 번호>)	INT DATA DATA = XWIN(“M”, 0)	8.21
XWOUT	CNET 또는 MODBUS 디바이스 영역의 워드 출력을 정수 단위로 출력 또는 확인 XWOUT(<디바이스 영역>, <워드 번호>) = <출력 값> <정수형 변수> = XWOUT(<디바이스 영역>, <워드 번호>)	INT DATA XWOUT(“M”, 0) = 0HFFFF DATA = XWOUT(“M”, 0)	8.22
XDIN	CNET 또는 MODBUS 디바이스 영역의 더블 워드 입력을 정수 단위로 확인 <정수형 변수> = XDIN(<디바이스 영역>, <더블 워드 번호>)	INT DATA DATA = XDIN(“D”, 0)	8.23
XDOUT	CNET 또는 MODBUS 디바이스 영역의 더블 워드 출력을 정수 단위로 출력 또는 확인 XDOUT(<디바이스 영역>,<더블 워드 번호>)=<출력 값> <정수형 변수> = XDOUT(<디바이스 영역>, <더블 워드 번호>)	INT DATA XDOUT(“D”, 0) = 0HFFFFFFFF DATA = XDOUT(“D”, 0)	8.24
XFIN	CNET 또는 MODBUS 디바이스 영역의 더블 워드 입력을 실수 단위 확인 <실수형 변수> = XFIN(<디바이스 영역>, <더블 워드 번호>)	REAL DATA DATA = XFIN(“M”, 0)	8.25
XFOUT	CNET 또는 MODBUS 디바이스 영역의 더블 워드 출력을 실수 단위로 출력 또는 확인 XFOUT(<디바이스 영역>, <더블 워드 번호>) = <실수 값>, 또는 <실수형 변수> = XFOUT(<디바이스 영역>, <더블 워드 번호>)	REAL DATA XFOUT(“M”, 0) = 12.45 DATA = XFOUT(“M”, 0)	8.26

8. Multithreading 제어

구분	설 명 형 식	사용 예	Ref.
CTHREAD	thread를 생성 <thread 번호>: 2 ~ 4까지 사용 가능 <thread 알람정지>: 알람이 발생하였을 때, <Thread 정지 기능>: (1: ON(기본값), 0: OFF) [<정수형 변수> =] CTHREAD(<함수명>, <Thread 번호>, [<Thread 알람정지>])	INCLUDE “AA” MAIN INT T_RET T_RET = CTHREAD(AA_MAIN, 2) EOP	9.1
ETHREAD	thread를 종료 <thread 번호>: 2 ~ 4까지 사용 가능 [<정수형 변수> =] ETHREAD(<thread 번호>)	ETHREAD(2)	9.2
TH_STATE	thread 상태 정보 읽기 <실행 모드>: System 모드(1) 또는 Robot 모드(0) 입력 <Thread 번호>: 1 ~ 4까지 사용 가능 - 리턴 값: 0 (TH_ERROR): thread 번호 오류 1 (TH_IDLE): thread 사용 가능 2 (TH_STOP): thread가 정지 3 (TH_RUN): thread가 구동 중 <정수형 변수> = TH_STATE(<실행 모드> ,<thread 번호>)	INT T_STATE T_STATE = TH_STATE(1,1)	9.3
TH_SUSPEND	구동중인 thread 정지 <thread 번호>: 2 ~ 4까지 사용 가능 [<정수형 변수>=]TH_SUSPEND(<thread 번호>)	TH_SUSPEND(2)	9.4
TH_RESUME	정지중인 thread 구동 <thread 번호>: 2 ~ 4까지 사용 가능 [<정수형 변수>=]TH_RESUME(<thread 번호>)	TH_RESUME(2)	9.5

9. 함수

9.1 좌표 변환

구분	설 명 형 식	사용 예	Ref.
TOOL	사용할 툴 번호를 선택 (0 ~ 15) TOOL <번호>	TOOL 3 LMOV P1	10.1
USER	사용할 사용자 좌표계 번호를 선택 (0 ~ 15) USER <번호>	USER 10 LMOV P1	10.2
JTOB	조인트 좌표 위치 변수 → 베이스 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <베이스 좌표 위치 변수> = JTOB([<로봇 번호>,] <조인트 좌표 위치 변수>)	POS AP XPOS XP1, XP2 AP = <0,0,0,0,0,0> AP.T = 2 XP1 = JTOB(AP) XP2 = JTOB(1,AP)	10.3
BTOJ	베이스 좌표 위치 변수 → 조인트 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <조인트 좌표 위치 변수> = BTOJ([<로봇 번호>,] <베이스 좌표 위치 변수>)	POS AP1, AP2 XPOS XP XP = <90,50,90,30,5,45,ARM = 0,T = 1> AP1 = BTOJ(XP) AP2 = BTOJ(1,XP)	10.4
JTOU	조인트 좌표 위치 변수 → 사용자 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <사용자 좌표 위치 변수> = JTOU([<로봇 번호>,] <조인트 좌표 위치 변수>)	POS AP UPOS UP1, UP2 AP = <0,0,0,0,0,0> AP.U = 1 UP1 = JTOU(AP) UP2 = JTOU(1, AP)	10.5
UTOJ	사용자 좌표 위치 변수 → 조인트 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <조인트 좌표 위치 변수> = UTOJ([<로봇 번호>,] <사용자 좌표 위치 변수>)	POS AP1, AP2 UPOS UP UP = <5,5,10,3,6,0,ARM = 0,T = 0,U = 1> AP1 = UTOJ(UP) AP2 = UTOJ(1, UP)	10.6
BTOU	베이스 좌표 위치 변수 → 사용자 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <사용자 좌표 위치 변수> = BTOU([<로봇 번호>,] <베이스 좌표 위치 변수>)	XPOS XP UPOS UP1, UP2 XP = <70,10,20,30,5,45,ARM = 0,T = 1> XP.U = 2 UP1 = BTOU(XP) UP2 = BTOU(1, XP)	10.7
UTOB	사용자 좌표 위치 변수 → 베이스 좌표 위치 변수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <베이스 좌표 위치 변수> = UTOB([<로봇 번호>,] <사용자 좌표 위치 변수>)	XPOS XP1, XP2 UPOS UP UP = <5,5,10,0,0,0,ARM = 0,T = 0,U = 1> XP1 = UTOB(UP) XP2 = UTOB(1, UP)	10.8

구분	설 명 형식	사용 예	Ref.
RELBASE	기준 위치에서 대입된 이동량 만큼 베이스 좌표계 기준으로 상대 이동한 위치 값 계산 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <기준 위치>: 베이스 좌표 위치 변수 <이동량>: 모든 좌표 위치 변수 <베이스 좌표 위치형 변수> = RELBASE([<로봇 번호>,]<기준 위치>,<이동량>)	POS AP, DATA XPOS XP, RP1, RP2 DATA = <100,0,0,0,0,50> AP = HERE AP.T = 0 XP = JTOB(AP) RP1 = RELBASE(XP, DATA) RP2 = RELBASE(1,XP, DATA)	10.9
RELTOOL	기준 위치에서 대입된 이동량 만큼 툴 좌표계 기준으로 상대 이동한 위치 값 계산 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <기준 위치>: 베이스 좌표 위치 변수 <이동량>: 모든 좌표 위치 변수 <베이스 좌표 위치형 변수> = RELTOOL([<로봇 번호>,]<기준 위치>,<이동량>)	POS AP, DATA XPOS XP, RP1, RP2 DATA = <100,0,0,0,0,50> AP = HERE AP.T = 1 XP = JTOB(AP) RP1 = RELTOOL(XP, DATA) RP2 = RELTOOL(1,XP, DATA)	10.10
RELUSER	기준 위치에서 대입된 이동량 만큼 사용자 좌표계 기준으로 상대 이동한 위치 값 계산 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <기준 위치>: 사용자 좌표 위치 변수 <이동량>: 모든 좌표 위치 변수 <사용자 좌표 위치형 변수> = RELUSER([<로봇 번호>,]<기준 위치>,<이동량>)	POS AP, DATA UPOS UP, RP1, RP2 DATA = <100,0,0,0,0,50> AP = HERE AP.T = 0 AP.U = 1 UP = JTOU(AP) RP1 = RELUSER(UP, DATA) RP2 = RELUSER(1,UP, DATA)	10.11
GBASEDISP	두 좌표간 베이스 좌표계 기준의 이동량을 구하는 함수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <위치1>: 모든 위치형 변수 <위치2>: 모든 위치형 변수 **※ 좌표 입력 시 <위치1>과 <위치2>는 같은 타입** <베이스 좌표 위치형 변수> = GBASEDISP([<로봇 번호>,]<위치1>,<위치2>)	POS AP1, AP2 XPOS XP1, XP2, D_X1, D_X2 AP1 = GP0 AP1.T = 0 AP2 = GP1 AP2.T = 0 D_X1 = GBASEDISP(AP1, AP2) D_X2 = GBASEDISP(1,AP1, AP2)	10.12
GTOOLDISP	두 좌표간 툴 좌표계 기준의 이동량을 구하는 함수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용<위치1>: 모든 위치형 변수 <위치2>: 모든 위치형 변수 **※ 좌표 입력 시 <위치1>과 <위치2>는 같은 타입** <베이스 좌표 위치형 변수> = GTOOLDISP([<로봇 번호>,]<위치1>,<위치2>)	POS AP1, AP2 XPOS XP1, XP2, D_X1, D_X2 AP1 = GP0 AP1.T = 1 XP1 = JTOB(AP1) AP2 = GP1 AP2.T = 1 XP2 = JTOB(AP2) D_X1 = GTOOLDISP(XP1, XP2) D_X2 = GTOOLDISP(1,XP1, XP2)	10.13

구분	설 명 형식	사용 예	Ref.
GUSERDISP	두 좌표간 사용자 좌표계 기준의 이동량을 구하는 함수 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <로봇 번호>를 입력하지 않는 경우 JOB 프로그램에 저장된 정보 사용 <위치1>: 모든 위치형 변수 <위치2>: 모든 위치형 변수 **※ 좌표 입력 시 <위치1>과 <위치2>는 같은 타입** <사용자 좌표 위치형 변수> = GUSERDISP([<로봇 번호>,]<위치1>,<위치2>)	POS AP1, AP2 UPOS UP1, UP2, D_U1, D_U2 AP1 = GP0 AP1.T = 1 AP1.U = 1 UP1 = JTOU(AP1) AP2 = GP1 AP2.T = 1 AP2.U = 1 UP2 = JTOU(AP2) D_U1 = GUSERDISP(UP1, UP2) D_U2 = GUSERDISP(1,UP1, UP2)	10.14
GET_TOOL_FRAME	지정한 툴의 오프셋 값을 실수 단위로 출력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <툴 번호>: 번호를 지정해 툴 선택 <문자열 변수> = GET_TOOL_FRAME(<로봇 번호>, <툴 번호>)	DEFSTR STR_TOOL_OFFSET INT INT_ROBOT_CH, INT_TOOL_NUM STR_TOOL_OFFSET = GET_TOOL_TRAME(INT_ROBOT_CH,INT_TOOL_NUM)	10.15
GET_USER_FRAME	지정한 사용자 좌표계의 오프셋 값을 실수 단위로 출력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <사용자 좌표계 번호>: 번호를 지정해 사용자 좌표계 선택 <문자열 변수> = GET_USER_FRAME(<로봇 번호>, <사용자 좌표계 번호>)	DEFSTR STR_USER_OFFSET INT INT_ROBOT_CH, INT_USER_NUM STR_USER_OFFSET = GET_USER_FRAME(INT_ROBOT_CH,INT_USER_NUM)	10.16
SET_TOOL_FRAME	지정한 툴의 오프셋 값을 실수 단위 입력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <툴 번호>: 번호를 지정해 툴 선택 <문자열 변수>: 툴의 오프셋 입력 (<정수형 변수> =) SET_TOOL_FRAME(<로봇 번호>, <툴 번호>, <문자열 변수>)	INT RESULT INT INT_ROBOT_CH,INT_TOOL_NUM DEFSTR STR_OFFS RESULT = SET_TOOL_FRAME(INT_ROBOT_CH,INT_TOOL_NUM,STR_OFFS)	10.17
SET_USER_FRAME	지정한 사용자 좌표계의 오프셋 값을 실수 단위 입력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <사용자 좌표계 번호>: 번호를 지정해 사용자 좌표계 선택 <문자열 변수>: 사용자 좌표계의 오프셋 입력 (<정수형 변수> =) SET_USER_FRAME(<로봇 번호>, <사용자 좌표계 번호>, <문자열 변수>)	DEFSTR STR_USER_OFFSET INT INT_ROBOT_CH, INT_USER_NUM STR_USER_OFFSET = GET_USER_FRAME(INT_ROBOT_CH,INT_USER_NUM)	10.18
READ_TOOL	지정한 로봇에 사용 중인 TOOL 번호 출력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <정수형 변수> = READ_TOOL(<로봇 번호>)	INT TOOL_NUM TOOL 3 TOOL_NUM = READ_TOOL(1)	10.19
READ_USER	지정한 로봇에 사용 중인 USER 번호 출력 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <정수형 변수> = READ_USER(<로봇 번호>)	INT USER_NUM USER 2 USER_NUM = READ_USER(1)	10.20

9.2 수학

구분	설 명 형 식	사용 예	Ref.
ABS	절대값을 취함 <실수형 변수> = ABS(<입력 값>)	REAL AA AA = ABS(-1.5)	10.21
DEG	라디안 각을 각도 값으로 변환 <실수형 변수> = DEG(<입력 값>)	REAL AA AA = DEG(3.14)	10.22
RAD	각도 값을 라디안 각으로 변환 <실수형 변수> = RAD(<입력 값>)	REAL AA AA = RAD(180)	10.23
POW	멱함수 <실수형 변수> = POW (<밑수 값>, <지수 값>)	REAL AA AA = POW(2,4)	10.24
RND	소수부를 반올림하여 실수값을 정수 값으로 변환 <실수형 변수> = RND(<입력 값>)	REAL AA AA = RND(14.8)	10.25
EXP	자연상수의 지수함수 e^x <실수형 변수> = EXP(<입력 값>)	REAL AA AA = EXP(3)	10.26
LN	자연로그 $\log_e X$ <실수형 변수> = LN(<입력 값>)	REAL AA AA = LN(15)	10.27
LOG	상용로그 $\log_{10} X$ <실수형 변수> = LOG(<입력 값>)	REAL AA AA = LOG(100)	10.28
SQRT	평방제곱근을 구함 <실수형 변수> = SQRT(<입력 값>)	REAL AA AA = SQRT(16)	10.29
SIN	사인함수 <실수형 변수> = SIN(<입력 값>)	REAL AA AA = SIN(RAD(30))	10.30
ASIN	아크 사인함수 <실수형 변수> = ASIN(<입력 값>)	REAL AA AA = ASIN(0.5)	10.31
COS	코사인 함수 <실수형 변수> = COS(<입력 값>)	REAL AA AA = COS(0)	10.32
ACOS	아크 코사인함수 <실수형 변수> = ACOS(<입력 값>)	REAL AA AA = ACOS(0.5)	10.33
TAN	탄젠트함수 <실수형 변수> = TAN(<입력 값>)	REAL AA AA = TAN(RAD(45))	10.34
ATAN	아크 탄젠트함수 <실수형 변수> = ATAN(<입력 값>)	REAL AA AA = ATAN(-1.0)	10.35
ATAN2	제2 아크 탄젠트함수 <실수형 변수> = ATAN2(<Y 비율>,<X 비율>)	REAL AA AA = ATAN2(1,-1)	10.36
MIN	최솟값 함수 <실수형 변수> = MIN(<입력 값1>, <입력 값2>)	REAL AA AA = MIN(1,2)	10.37
MAX	최댓값 함수 <실수형 변수> = MAX(<입력 값1>, <입력 값2>)	REAL AA AA = MAX(1.1,2.2)	10.38
RAND	난수 생성 함수 <숫자 변수> = RAND()	REAL AA AA = RAND()	10.39

9.3 문자열

구분	설 명 형 식	사용 예	Ref.
ASC	문자열 첫 문자를 캐릭터 코드로 반환 <정수형 변수> = ASC(<문자열>)	INT VAL VAL = ASC(“STATEMENT”)	10.40
BINS	정수를 이진수 문자열로 변환 <문자열 변수> = BINS(<정수 값>)	DEFSTR STR STR = BINS(16)	10.41
CHR	정수를 문자로 변환 ASCII Code인 문자가 문자열 변수에 저장됨. <문자열 변수> = CHR(<정수 값>)	DEFSTR STR STR = CHR(65)	10.42
FLUSH	통신 입력, 출력 버퍼 클리어 <포트번호> 0 : 시리얼 통신 포트 10 ~ 12 : TCP/IP Server 포트 ~ 65,535: TCP/IP Client 포트 <클리어 버퍼 선택> : 1: 입력 버퍼 클리어 2: 출력 버퍼 클리어 3: 입력,출력 버퍼 클리어 FLUSH <포트번호> <클리어 버퍼 선택>	FLUSH 0 1	10.43
FTOS	정수나 실수를 문자열로 변환 <문자열 변수> = FTOS(<입력 값>)	DEFSTR STR STR = FTOS(10.5)	10.44
HTOS	정수를 16진수의 문자열로 변환 <문자열 변수> = HTOS(<입력 값>)	DEFSTR STR STR = HTOS(100)	10.45
SLEFT	문자열의 왼쪽부터 숫자만큼 문자를 추출해 문자열 변수에 저장 <문자열 변수> = SLEFT(<문자열>, <추출 개수>)	DEFSTR STR STR = SLEFT(“TEST”,2)	10.46
SLEN	문자열 길이를 반환 <정수형 변수> = SLEN(<문자열>)	INT LEN LEN = SLEN(“STATEMENT”)	10.47
SMID	문자열 지정위치부터 자릿수 만큼 문자열 추출 <문자열 변수> = SMID(<문자열>, <지정위치>, <추출문자수>)	DEFSTR STR STR = SMID(“STATEMENT”, 2, 3)	10.48
SPOS	문자열1 내에서 문자열2가 들어가 있는 위치반환 <정수형 변수> = SPOS(<문자열1>, <문자열2>)	INT VAL VAL = SPOS(“TEST LINE”, “LINE”)	10.49
SRIGHT	문자열의 오른쪽부터 숫자만큼 문자를 추출해 문자열 변수에 저장 <문자열 변수> = SRIGHT(<문자열>, <추출 개수>)	DEFSTR STR STR = SRIGHT(“TEST”,2)	10.50
STOK	구분자로 문자열을 추출 후에 지정 된 순번의 문자를 반환합니다. <변수>=STOK(<문자열>,<구분문자>,<번호>)	STRTMP[IVAL] = STOK(STR,TOK,IVAL+1)	10.51
STRIN	구분자까지 문자열 수신 <포트번호>: RS-232(0), Ethernet(10, 11, 12) <대기시간>: 단위 1ms <문자열 변수> = STRIN(<포트번호>, <대기시간>)	DEFSTR STR STR = STRIN(0,1000)	10.52
STROUT	문자열 전송, 전송하지 못한 문자 개수 반환 <포트 번호>: RS-232(0), Ethernet(10, 11, 12) <정수형 변수> = STROUT(<포트번호>, <문자열>)	INT VAL VAL = STROUT(0,“TEST”)	10.53
SVAL	문자열을 숫자로 변환 <숫자 변수> = SVAL(<문자열>)	INT VAL VAL = SVAL(“1234”)	10.54

10. 시스템

10.1 오류

구분	설 명 형 식	사용 예	Ref.

SETERR	사용자가 임의의 알람을 발생시킴
	SETERR(<사용자 알람 번호> [, <사용자 알람 메시지>])
		IF IN(0) == 1 THEN
		SETERR(1, "User Err 1")
		ENDIF
			11.1

RESET	알람 해제 RESET[(<로봇 번호>)]	RESET RESET(1)	11.2
RERROR	가장 마지막 알람 코드반환 <로봇 번호>: 번호를 입력하여 로봇 지정(0 ~ 3) 0 입력 시 모든 로봇 <정수형 변수> = RERROR(<로봇 번호>)	INT ECODE ECODE = RERROR(0) ECODE = RERROR(1)	11.3
RERRCNT	현재 발생한 알람 개수 반환 <로봇 번호>: 번호를 입력하여 로봇 지정(0 ~ 3) 0 입력 시 모든 로봇 <정수형 변수> = RERRCNT(<로봇 번호>)	INT ECOUNT ECOUNT = RERRCNT(0) ECOUNT = RERRCNT(1)	11.4
RERRCODE	인덱스 번호에 해당하는 알람 코드 반환 <로봇 번호>: 번호를 입력하여 로봇 지정(0 ~ 3) 0 입력 시 모든 로봇 <인덱스 번호>는 알람 발생 순서를 의미함 <정수형 변수> = RERRCODE(<로봇 번호>, <인덱스 번호>)	INT ECODE ECODE = RERRCODE(0,1) ECODE = RERRCODE(1,1)	11.5
RERRTEXT	인덱스 번호에 해당하는 알람 내용 반환 <로봇 번호>: 번호를 입력하여 로봇 지정(0 ~ 3) 0 입력 시 모든 로봇 <인덱스 번호>는 알람 발생 순서를 의미함 <문자열 변수> = RERRTEXT(<로봇 번호>, <인덱스 번호>)	DEFSTR ETEXT ETEXT = RERRTEXT(0,1) ETEXT = RERRTEXT(1,2)	11.6
READ_DERR	드라이버 알람 코드 읽기 <로봇 번호>: 번호를 입력하여 로봇 지정(1 ~ 3) <축 번호>: 번호를 입력하여 축 지정(DOF) <정수형 변수> = READ_DERR(<로봇 번호>, <축 번호>)	INT VALUE VALUE = READ_DERR(1,1)	11.7

10.2 상태 정보

구분	설 명 형 식	사용 예	Ref.
RSTATE	로봇의 상태를 확인. - <상태 인덱스>의 의미 0: ALL, 상태 정보 전체를 반환 1: Alarm 상태 ➢ 0: Normal ➢ 1: Alarm 2: MC Ready 상태 ➢ 0: MC OFF ➢ 1: MC ON 3: Origin 완료 상태 ➢ 0: 원점 미수행 ➢ 1: 원점 수행 완료 4: In Position 상태 ➢ 0: 목표 위치 미 도달 ➢ 1: 목표 위치 도달 완료 5: Reserved 6: Servo State (ON/OFF) 상태 ➢ 0: Servo OFF ➢ 1: Servo ON 7 :Reserved 8: T/P Mode와 상위 설비 Mode 일치 상태 ➢ 0: T/P와 상위 모드 일치 ➢ 1: T/P와 상위 모드 불일치 9: System Mode 상태 ➢ 0: Robot ➢ 1: System 10: T/P Mode 상태 ➢ 0: Manual ➢ 1: Auto 11: Emergency 상태 ➢ 0: Normal ➢ 1: TP / Front / System EMG 12: Robot 활성화 상태 ➢ 0: Robot Disable ➢ 1: Robot Enable 13: Warning 상태 ➢ 0: Normal ➢ 1: FAN/Encoder 배터리 경고 발생 14: Encoder Battery 경고 상태 ➢ 0: Normal ➢ 1: Encoder 배터리 경고 발생 15: 로봇 모션 수행 여부 ➢ 0: 모션 정지 ➢ 1: 모션 이동 중 16: FAN 동작 여부 확인 ➢ 0: Normal ➢ 1: FAN 경고 발생 <정수형 변수> = RSTATE(<로봇 번호>, <상태 인덱스>)	INT R_ST R_ST = RSTATE(1,0)	11.8

구분	설 명 형 식	사용 예	Ref.
SWLIMIT	각 축의 최소 또는 최대 이동범위를 확인. <최대/최소>: 0:이동범위 최솟값, 1:이동범위 최댓값 <실수형 변수> =SWLIMIT(<로봇 번호>,<축 번호>, <최대/최소>)	REAL L_LIMIT, R_LIMIT L_LIMIT = SWLIMIT(1,1,0) R_LIMIT = SWLIMIT(1,1,1)	11.9
READ_RPM	현재 RPM 값을 확인. <실수형 변수> =READ_RPM(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_RPM(1,1)	11.10
READ_LOAD	현재 부하율 값 확인. <실수형 변수> =READ_LOAD(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_LOAD(1,2)	11.11
READ_TRQMAX	최대 부하율 값 확인. <실수형 변수> =READ_TRQMAX(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_TRQMAX(1,3)	11.12
READ_VEL	현재 속도 값 확인. <실수형 변수>=READ_VEL(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_VEL(1,4)	11.13
READ_TRQ	현재 토크 값 확인 <실수형 변수>=READ_TRQ(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_TRQ(1,1)	11.14
READ_MTEMP	현재 모터 엔코더 온도 값 확인 <실수형 변수> =READ_MTEMP(<로봇 번호>,<축 번호>)	REAL VALUE VALUE = READ_MTEMP(1,1)	11.15
READ_CTEMP	현재 제어기 내부 온도 값 확인 <실수형 변수>=READ_CTEMP()	REAL VALUE VALUE = READ_CTEMP()	11.16

10.3 로그 및 데이터 저장

구분	설 명 형 식	사용 예	Ref.
PRINT	모니터링 화면에서 확인 가능한 로그 출력 <인덱스> 범위: 0 ~ 999 PRINT(<인덱스>, <로그 내용1> [{, <로그 내용2>, … <로그 내용N>}])	PRINT(0,"PRINT LOG")	11.17
WLOG	시간 정보를 포함하여 사용자 로그를 순서대로 파일로 저장 WLOG(<로그 내용1> [{, <로그 내용2>, … <로그 내용N>}])	WLOG("WLOG TEST")	11.18
CLOG	**※ 현재 지원하지 않습니다.**		11.19

구분	설 명 형 식	사용 예	Ref.
WGPNT	전역 포인트 변수를 파일로 저장 <동기화>: 파일이 디스크에 동기화 될 때까지 대기 여부를 선택(0: 대기 안 함, 1: 대기) - 리턴 값: 0(정상), 1(쓰기 오류) [<정수형 변수> =] WGPNT(<동기화>)	WGPNT(1) INT VAL VAL = WGPNT(1)	11.20
WGINT	전역 정수 변수를 파일로 저장 <동기화>: 파일이 디스크에 동기화 될 때까지 대기 여부를 선택(0: 대기 안 함, 1: 대기) - 리턴 값: 0(정상), 1(쓰기 오류) [<정수형 변수> =] WGINT(<동기화>)	WGINT(1) INT VAL VAL = WGINT (1)	11.21
WGFLT	전역 실수 변수를 파일로 저장 <동기화>: 파일이 디스크에 동기화 될 때까지 대기 여부를 선택(0: 대기 안 함, 1: 대기) - 리턴 값: 0(정상), 1(쓰기 오류) [<정수형 변수> =] WGFLT(<동기화>)	WGFLT(1) INT VAL VAL = WGFLT (1)	11.22
TACT_START	공정 시간 측정을 위한 측정 시작 위치 설정 TACT_START()	INT VAL TACT_START() DELAY 1000 VAL = TACT_STOP()	11.23
TACT_STOP	공정 시간 측정을 위한 측정 종료 위치 설정 <정수형 변수> = TACT_STOP()		11.24
INIT_POS	모든 위치형 변수를 초기화 INIT_POS(<번호>)	INIT_POS(P5) INIT_POS(GP3)	11.25
GET_TWTIME	출하 이후 제어기 총 누적 통전시간 확인 <변수> = GET_TWTIME()	DEFSTR TW_TIME TW_TIME = GET_TWTIME()	11.26
GET_TSVTIME	출하 이후 모터 총 누적 통전 시간을 확인 <변수> = GET_TSVTIME(<로봇 번호>,<축 번호>)	DEFSTR TSV_TIME TSV_TIME = GET_ TSVTIME(1, 1)	11.27
GET_TMVTIME	출하 이후 모터 총 누적 이동 시간을 확인 <변수> = GET_TMVTIME(<로봇 번호>,<축 번호>)	DEFSTR TMV_TIME TMV_TIME = GET_ TMVTIME(1, 1)	11.28
GET_TMVDIST	출하 이후 모터 총 누적 이동거리를 확인 <변수> = GET_TMVDIST(<로봇 번호>,<축 번호>)	DEFSTR TMV_DIST TMV_DIST = GET_ TMVDIST(1, AXIS)	11.29
GET_WTIME	제어기 누적 통전시간을 확인 <변수> = GET_WTIME()	DEFSTR W_TIME W_TIME = GET_WTIME()	11.30
GET_SVTIME	모터 누적 통전 시간을 확인 <변수> = GET_SVTIME(<로봇 번호>,<축 번호>)	DEFSTR SV_TIME SV_TIME = GET_ SVTIME(1, AXIS)	11.31
GET_MVTIME	모터 누적 이동 시간을 확인 <변수> = GET_MVTIME(<로봇 번호>,<축 번호>)	DEFSTR MV_TIME MV_TIME = GET_ MVTIME(1, AXIS)	11.32
GET_MVDIST	모터 누적 이동거리를 확인 <변수> = GET_MVDIST(<로봇 번호>,<축 번호>)	REAL MV_DIST MV_DIST = GET_ MVDIST(1, AXIS)	11.33
INIT_WTIME	제어기 누적 통전 시간을 초기화 <변수> = INIT_WTIME()	DEFSTR W_TIME W_TIME = INIT_WTIME()	11.34
INIT_SVTIME	모터 누적 통전 시간을 초기화 <변수> = INIT_SVTIME(<로봇 번호>,<축 번호>)	DEFSTR SV_TIME SV_TIME = INIT_ SVTIME(1, AXIS)	11.35
INIT_MVTIME	모터 누적 이동 시간을 초기화 <변수> = INIT_SVTIME(<로봇 번호>,<축 번호>)	DEFSTR MV_TIME MV_TIME = INIT_ MVTIME(1, AXIS)	11.36
INIT_MVDIST	모터 누적 이동거리를 초기화 <변수> = INIT_MVDIST(<로봇 번호>,<축 번호>)	REAL MV_DIST MV_DIST = INIT_ MVDIST(1, AXIS)	11.37

11. System 모드 전용 명령어

구분	설 명 형 식	사용 예	Ref.
PSEL	지정한 Thread 번호에 JOB 프로그램 매칭 <thread 번호>: 1 ~ 4까지 사용 가능 [<정수형 변수> =] PSEL(<Thread 번호>, "<JOB 파일명>")	INT R_VAL R_VAL = PSEL(1,"TEST")	12.1
PSTART	지정한 Thread 번호의 JOB 프로그램 실행 <thread 번호>: 1 ~ 4까지 사용 가능 [<정수형 변수> =] PSTART(<Thread 번호>)	PSTART(1)	12.2
PSTOP	지정한 Thread 번호의 JOB 프로그램 정지 <thread 번호>: 1 ~ 4까지 사용 가능 [<정수형 변수> =] PSTOP(<Thread 번호>)	PSTOP(1)	12.3
PSTATE	지정한 Thread 번호의 상태 확인 <thread 번호>: 1 ~ 4까지 사용 가능 <정수형 변수> = PSTATE(<Thread 번호>)	INT STATE STATE = PSTATE(1)	12.4
PRESTART	지정한 Thread 번호의 JOB 프로그램 재 실행 <thread 번호>: 1 ~ 4까지 사용 가능 [<정수형 변수> =] PRESTART(<Thread 번호>)	PRESTART(1)	12.5
PEXIT	지정한 Thread 번호의 JOB 프로그램 해제 <thread 번호>: 1 ~ 4까지 사용 가능 [<정수형 변수> =] PEXIT(<Thread 번호>)	PEXIT(1)	12.6

1) 현재 System 모드 전용 명령어의 Thread 번호는 1만 지원합니다.
추후 1~4 까지 지원 할 예정입니다.

12. TCP IP Client 전용 명령어

구분	설 명 형식	사용 예	Ref.
NET_OPEN	제어기의 TCP/IP Client 소켓 생성 및 Server 로 접속 요청 <IP 번호>: Server의 IP 번호 입력 변수 <Port 번호>: Server의 Port 번호 입력 변수 <Wait 시간>: 통신 접속 요청 시간 변수(msec) <정수형 변수> =NET_OPEN(<IP번호>,<Port번호>,<Wait 시간>)	INT FD_NUM DEFSTR IP_NUM INT PORT_NUM INT WAIT_TIME IP_NUM = "192.168.1.196" PORT_NUM = 10000 WAIT_TIME = 1000 FD_NUM = NET_OPEN(IP_NUM, PORT_NUM, WAIT_TIME)	13.1
NET_CLOSE	제어기의 연결된 TCP/IP 통신을 종료하고 Client 소켓을 제거 <FD>: Socket File Descriptor NET_CLOSE(<FD>)	INT FD_NUM …… RET_VEL = NET_CLOSE(FD_NUM)	13.2
NET_STATUS	제어기의 TCP/IP 통신상태를 확인 <FD>: Socket File Descriptor	INT FD_NUM …… RET_VEL = NET_STATUS(FD_NUM)	13.3

	<정수형 변수> = NET_STATUS(<FD>)		
NET_KEEPALIVE	제어기의 TCP/IP Client Socket의 세션체크 옵션 설정 <FD>: Socket File Descriptor <시작주기>: 최초 세션 체크 주기 입력 변수 <반복횟수>: 세션 체크 무 응답 시 반복 횟수 <반복주기>: 세션 체크 반복 주기	INT FD_NUM …… INT RET_VEL INT KEEP_IDLE INT KEEP_CNT INT KEEP_INTERVAL FD_NUM = 10000 KEEP_IDEL = 10 KEEP_CNT = 20 KEEP_INTERVAL = 3 RET_VEL = NET_KEEPALIVE(FD_NUM, KEEP_IDLE, KEEP_CNT, KEEP_INTERVAL)	13.4
	<정수형 변수> = NET_KEEPALIVE(<FD>, <시작주기>, <반복횟수>, <반복주기>)		
NET_FD	TCP/IP Server 에 접속한 Client의 Socket File Descriptor를 반환	INT FD_NUM INT INDEX INDEX = 10 FD_NUM = NET_FD(<INDEX>)	13.5
	<정수형 변수> = NET_FD(<Server Index>)		
NET_STRIN	구분자까지 문자열 수신 <파일 기술자 번호>: 소켓 정보 접근을 위한 파일 기술자(File descriptor) 번호 <대기시간>: 단위 1ms	DEFSTR STR INT FD_NUM …… STR = NET_STRIN(FD_NUM,1000)	13.6
	<문자열 변수> = NET_STRIN(<파일 기술자 번호>, <대기시간>)		
NET_STROUT	문자열 전송, 전송하지 못한 문자 개수 반환 구분자까지 문자열 수신 <파일 기술자 번호>: 소켓 정보 접근을 위한 파일 <문자열 변수>: 출력할 문자열	INT VAL DEFSTR STR INT FD_NUM …… VAL = NET_STROUT(FD_NUM, STR)	13.7
	<정수형 변수> = NET_STRIN(<포트번호>, <대기시간>)		

13. 영역 검출 전용 명령어

구분	설 명 형식	사용 예	Ref.
SET_INRANGE_ENB	In Range 기능을 활성화 또는 비 활성화 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <활성화 Flag>: (0: Disable, 1: Enable) SET_INRANGE_ENB(<로봇 번호>, <In Range 번호>, <활성화 Flag>)	SET_INRANGE_ENB(1, 1, 0) SET_INRANGE_ENB(1, 1, 1)	14.1
SET_INRANGE_POS	두 좌표값을 입력해 In Range 영역을 설정 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <영역 최소 좌표>: Base 또는 User 좌표값 <영역 최대 좌표>: Base 또는 User 좌표값 SET_INRANGE_POS(<로봇 번호>, <In Range 번호>, <영역 최소 좌표>, <영역 최대 좌표>)	XPOS XMIN, XMAX XMIN = JTOB(GP0) XMAX = JTOB(GP1) SET_INRANGE_POS(1, 1, XMIN, XMAX) UPOS UMIN, UMAX UMIN = JTOU(GP0) UMAX = JTOU(GP1) SET_INRANGE_POS(1, 2, UMIN, UMAX)	14.2
SET_INRANGE_ALARM	In Range 검출 시 알람 활성화 도는 비활성화 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <활성화 Flag>: (0: Disable, 1: Enable) SET_INRANGE_ALARM(<로봇 번호>, <In Range 번호>, <활성화 Flag>)	SET_INRANGE_ALARM(1, 1, 0) SET_INRANGE_ALARM (1, 1, 1)	14.3
SET_INRANGE_OUT	In Range 검출 시 출력 접점 번호 설정 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <접점 번호>: (0 ~ 접점 최대 번호) SET_INRANGE_OUT(<로봇 번호>, <In Range 번호>, <접점 번호>)	SET_INRANGE_OUT(1, 1, 0) SET_INRANGE_OUT(1, 1, 19)	14.4
SET_INRANGE _MEASURE	In Range 영역 데이터 타입 설정 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <데이터 타입>: (0: Feedback, 1: Reference) SET_INRANGE_MEASURE(<로봇 번호>, <In Range 번호>, <데이터 타입>)	SET_INRANGE_MEASURE(1, 1, 0) SET_INRANGE_MEASURE(1, 1, 1)	14.5
GET_INRANGE_MIN	In Range 최소 영역 좌표값 반환 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <좌표계>: (0: BASE, 1: USER) <위치형 변수> = GET_INRANGE_MIN(<로봇 번호>, <In Range 번호>, <좌표계>)	XPOS XMIN, XMAX XMIN = GET_INRANGE_MIN(1, 1, 0) XMAX = GET_INRANGE_MAX(1, 1, 0) UPOS UMIN, UMAX UMIN = GET_INRANGE_MIN(1, 1, 1) UMAX = GET_INRANGE_MAX(1, 1, 1)	14.6
GET_INRANGE_MAX	In Range 최대 영역 좌표값 반환 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <좌표계>: (0: BASE, 1: USER) <위치형 변수> = GET_INRANGE_MAX(<로봇 번호>, <In Range 번호>, <좌표계>)		14.7

GET_INRANGE_ALARM	In Range 검출 시 알람 설정 값 반환 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <정수형 변수> = GET_INRANGE_ALARM(<로봇 번호>, <In Range 번호>)	INT AA AA = GET_INRANGE_ALARM(1, 1)	14.8
GET_INRANGE_OUT	In Range 검출 시 출력 접점 번호 반환 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <정수형 변수> = GET_INRANGE_OUT(<로봇 번호>, <In Range 번호>)	INT AA AA = GET_INRANGE_OUT(1, 1)	14.9
GET_INRANGE_VECTOR	In Range 중심점으로 부터의 벡터량 반환 <로봇 번호>: 번호를 지정해 로봇 선택(1 ~ 3) <In Range 번호>: In Range 번호 선택(1 ~ 16) <정수형 변수> = GET_INRANGE_VECTOR(<로봇 번호>, <In Range 번호>)	XPOS VEC VEC = GET_INRANGE_VECTOR(1, 1)	14.10

제 4장 명령어 해설

1. 명령어 해설 형식

아래 그림과 같은 형식으로 명령어의 의미와 사용법에 관해 자세히 설명합니다.

(1)<절 번호> (2)<명령어 이름>

(3) Usage	
	명령어 용도에 관해 설명합니다.
(4) Syntax	
	명령어의 올바른 사용 방법을 설명합니다. 명령어에 전달 인자가 포함된 경우 “<” 및 “>”로 표현합니다. 복수의 전달 인자가 있는 경우 “,” 가 사용됩니다. 무시할 수 있는 요소는 대괄호 “[“ 및 “]”로 묶습니다. 배열을 표시하는 경우는 ‘[‘ 및 ‘]’로 표시합니다. 요소 집합에 하나의 요소만 필요할 경우 집합은 대괄호 “{” 및 “}” 안에 기술되며 각 전달 인자는 구분 기호”|”로 구분됩니다. 반드시 전달 인자를 필요하거나 함수 등의 특수한 용도의 경우 명령어 뒤 소괄호 “(“ 및 “)” 안에 전달 인자를 기술합니다.
(5) Description	
	명령어의 특징 및 기능에 관해 자세히 설명합니다.
(6) Return value	
	명령어를 사용하였을 때 그 결과로서 돌려받게 되는 값이 존재하는 경우 해당 반환 값의 형태 및 용도에 대해 설명합니다.
(7) Arguments	
	포함되는 전달 인자의 형태 및 용도에 대해 설명합니다.
(9) Examples	
	명령어를 사용하는 방법에 대한 예시를 보여줍니다.
(10) Related terms	
	다른 관련 명령이나 설명을 보여줍니다.

Figure 4-1 명령어 해설 형식

2. 변수

2.1 INT

Usage

프로그램 내부에서 사용할 수 있는 정수형 변수를 선언합니다.

Syntax

INT [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

4바이트 크기의 정수형 변수를 선언합니다.
선언 후 반드시 초기화 바랍니다. 단, 선언과 동시에 초기화는 불가능합니다.
Range: -2,147,483,647 ~ 2,147,438,647

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
INT $AA                          // 전역 변수 선언

MAIN                             // 메인 함수 선언
    INT D_1, D_2, D_3[2]         // 지역 변수 선언
    $AA = 15                     // 전역 변수 초기화
    D_1 = 0                      // D_1에 상수 값 대입
    D_2 = I100                   // D_2에 전역 변수 I100값 대입
    D_3[0] = $AA                 // D_3에 전역 변수 $AA값 대입
EOP                              // 메인 함수 종료
```

Related terms

I, REAL, DEFSTR

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.2 REAL

Usage

프로그램 내부에서 사용할 수 있는 실수형 변수를 선언합니다.

Syntax

REAL [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

4바이트 크기의 실수형 변수를 선언합니다.
선언 후 반드시 초기화 바랍니다. 단, 선언과 동시에 초기화는 불가능합니다.
Range: 3.402,823E-38 ~ 3.402,823E+38

※ 소수점 3번째자리까지 입력되며, 소수점 4번째 자리부터 반올림됩니다.

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
REAL $AA                        // 전역 변수 선언

MAIN                            // 메인 함수 선언
    REAL D_1, D_2, D_3[2]       // 지역 변수 선언
    $AA = 15.15                 // 전역 변수 초기화
    D_1 = 0.58                  // D_1에 상수 값 대입
    D_2 = F100                  // D_2에 전역 변수 F100값 대입
    D_3[0] = $AA                // D_3에 전역 변수 $AA값 대입
EOP                             // 메인 함수 종료
```

Related terms

F, INT, DEFSTR

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.3 DEFSTR

Usage

프로그램 내부에서 사용할 수 있는 문자열 변수를 선언합니다.

Syntax

DEFSTR [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

문자열 변수를 선언합니다.
선언 후 반드시 초기화 해 주어야 합니다. 단, 선언과 동시에 초기화는 불가능 합니다.
최대 글자수는 100자를 넘지 않아야 합니다.

※ 100자를 초과하여 입력하는 경우 이후 문자는 변수에 입력되지 않습니다.

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
DEFSTR $STR                     // 전역 변수 선언

MAIN                            // 메인 함수 선언
    DEFSTR D_1, D_2[2]          // 지역 변수 선언
    $STR = "STRING"             // 전역 변수 초기화
    D_1 = "CHECK"               // D_1에 상수 값 대입
    D_2[0] = $STR               // D_2에 전역 변수 $STR값 대입
EOP                             // 메인 함수 종료
```

Related terms

INT, REAL

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.4 POS

Usage

프로그램 내부에서 사용할 수 있는 조인트 좌표계 위치형 변수를 선언합니다.

Syntax

POS [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

조인트 좌표계 위치형 변수를 선언합니다. 최대 DOF까지 위치 값을 저장할 수 있습니다. 선언 후 반드시 초기화 바랍니다. 단, 선언과 동시에 초기화는 불가능합니다. 초기화는 위치형 상수 또는 개별 요소 접근 방식으로 할 수 있습니다.

※ 개별 요소 접근 방식으로 초기화 시, 반드시 USED 요소를 1로 초기화해야 합니다.

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
POS $AP                                  // 전역 변수 선언

MAIN                                     // 메인 함수 선언
    POS D_1, D_2[2]                      // 지역 변수 선언
    $AP = <100,100,100,100,100,100>      // 전역 변수 초기화, 6축
    D_1.1 = 100.0                        // 지역 변수 개별 요소 접근 방식
    D_1.2 = 120.1
    D_1.3 = 130.2
    D_1.4 = 140.3
    D_1.5 = 150.4
    D_1.6 = 160.5
    D_1.USED = 1                         // D_1 초기화 완료
    D_2[0] = $AP                         // D_2에 전역 변수 $AP값 대입
EOP                                      // 메인 함수 종료
```

Related terms

XPOS, UPOS, P, GP

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.5 XPOS

Usage

프로그램 내부에서 사용할 수 있는 베이스 좌표계 위치형 변수를 선언합니다.

Syntax

XPOS [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

베이스 좌표계 위치형 변수를 선언합니다. 최대 DOF까지 위치 값을 저장할 수 있습니다. 선언 후 반드시 초기화 바랍니다. 단, 선언과 동시에 초기화는 불가능합니다. 초기화는 위치형 상수 또는 개별 요소 접근 방식으로 할 수 있습니다.

※ 개별 요소 접근 방식으로 초기화 시, 반드시 USED 요소를 1로 초기화해야 합니다.

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
XPOS $XP                                          // 전역 변수 선언
MAIN                                              // 메인 함수 선언
    XPOS D_1, D_2[2]                              // 지역 변수 선언
    $XP = <100,100,100,100,100,100,ARM = 0,T = 1> // 전역 변수 초기화, 6축
    D_1.1 = 100.0                                 // 지역 변수 개별 요소 접근 방식
    D_1.2 = 120.1
    D_1.3 = 130.2
    D_1.4 = 140.3
    D_1.5 = 150.4
    D_1.6 = 160.5
    D_1.ARM = 111                                 // 로봇 자세 대입
    D_1.T = 2                                     // 로봇 Tool 번호 대입
    D_1.USED = 1                                  // D_1 초기화 완료
    D_2[0] = $XP                                  // D_2에 전역 변수 $AP값 대입
EOP                                               // 메인 함수 종료
```

Related terms

POS, UPOS, P, GP

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.6 UPOS

Usage

프로그램 내부에서 사용할 수 있는 사용자 좌표계 위치형 변수를 선언합니다.

Syntax

UPOS [$]<변수 이름>['[' <배열 개수> ']'] [, <변수 이름>['[' <배열 개수> ']'], …]

Description

사용자 좌표계 위치형 변수를 선언합니다. 최대 DOF까지 위치 값을 저장할 수 있습니다. 선언 후 반드시 초기화 바랍니다. 단, 선언과 동시에 초기화는 불가능합니다. 초기화는 위치형 상수 또는 개별 요소 접근 방식으로 할 수 있습니다.

※ 개별 요소 접근 방식으로 초기화 시, 반드시 USED 요소를 1로 초기화해야 합니다.

1) "$" 기호를 변수 이름 앞에 붙여 전역 변수로 사용할 수 있습니다. 전역 변수로 사용할 경우 함수 밖에 선언해야 합니다.
2) <변수 이름>은 영문자와 숫자를 사용할 수 있으며, 변수의 첫 글자는 반드시 영문자 이거나 "_" 기호만 사용이 가능합니다.
3) "," 기호를 사용해 두 개 이상의 변수를 선언할 수 있습니다.
4) '['<배열 개수>']'를 지정해 여러 개의 데이터를 하나로 묶어 사용할 수 있습니다.

Examples

```
UPOS $UP                                                    // 전역 변수 선언
MAIN                                                        // 메인 함수 선언
    UPOS D_1, D_2[2]                                        // 지역 변수 선언
    $UP = <100,100,100,100,100,100, ARM = 0, U = 0>         // 전역 변수 초기화, 6축
    D_1.1 = 100.0                                           // 지역 변수 개별 요소 접근 방식
    D_1.2 = 120.1
    D_1.3 = 130.2
    D_1.4 = 140.3
    D_1.5 = 150.4
    D_1.6 = 160.5
    D_1.ARM = 111                                           // 로봇 자세 대입
    D_1.T = 2                                               // 로봇 Tool 번호 대입
    D_1.U = 3                                               // 로봇 User 번호 대입
    D_1.USED = 1                                            // D_1 초기화 완료
    D_2[0] = $UP                                            // D_2에 전역 변수 $AP값 대입
EOP                                                         // 메인 함수 종료
```

Related terms

POS, XPOS, P, GP

1) 변수 이름 선언 시 "P", "GP", "I", "F"로 시작하는 문자는 시스템에 미리 선언된 전역 변수들과 중복되므로 사용할 수 없습니다.
2) 변수의 선언과 동시에 초기화는 불가능하므로 주의하십시오.

2.7 Program Position P

Usage

시스템에서 미리 선언한 조인트 좌표계 위치형 변수로 파일에 위치 값이 저장되며 프로그램에 종속되는 특징을 갖습니다. 해당 프로그램 내에서 전역변수로 사용됩니다.

Syntax

P<번호> 또는 P'['<번호>']'

Description

POS로 선언된 조인트 좌표계 위치형 변수와 마찬가지로 최대 DOF까지의 위치 값을 저장할 수 있습니다.
GP와 달리 JOB 프로그램 파일마다 각각 다른 값을 저장할 수 있으며, 해당 프로그램 내에서만 사용이 가능한 전역 변수 입니다.
지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
프로그램 실행 시 편집은 불가능하며, 개별 요소 접근으로도 값을 읽어 오는 것만 가능합니다.

1) <번호>는 배열처럼 입력 가능하며, 숫자를 바로 작성하여 사용할 수 있습니다. 문자 P와 <번호> 간에는 공란이 있으면 안됩니다.
2) <번호>의 범위는 0 ~ 999까지로 총 1,000개 사용할 수 있습니다.

Examples

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = GP0.1          // AA변수에 GP0.1 값 대입
    BB = GP[0].2        // BB 변수에 GP[0].2 값 대입
    SVON                // 서보 ON
    JMOV P0             // P0위치까지 JMOV로 이동
EOP                     // 메인 함수 종료
```

Related terms

POS, XPOS, UPOS, GP

1) 프로그램 실행 중 값을 변경할 수 없습니다. 티칭 모드에서만 변수의 데이터 편집이 가능합니다.

2.8 Global Integer I

Usage

시스템에서 미리 선언한 정수형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다.

Syntax

I<번호> 또는 I'['<번호>']'

Description

INT로 선언된 정수형 변수와 마찬가지로 4바이트 크기의 정수 값을 저장할 수 있습니다.
Range: -2,147,483,647 ~ 2,147,438,647

지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
프로그램 실행 중 값을 변경할 수 있습니다. 값을 변경하고 WGINT 함수를 사용해 변경한 값을 파일에 저장할 수 있습니다.

1) <번호>는 배열처럼 입력 가능하며, 숫자를 바로 작성하여 사용할 수 있습니다. 문자 I와 <번호> 간에는 공란이 있으면 안됩니다.
2) <번호>의 범위는 0 ~ 999까지로 총 천 개까지 사용할 수 있습니다.

Examples

```
MAIN                    // 메인 함수 선언
    I[0] = 100          // 100을 0번째 I 변수에 대입
    I[999] = -1         // -1을 999번째 I 변수에 대입
EOP                     // 메인 함수 종료
```

Related terms

INT, F, WGINT

1) 프로그램 실행 중 변경한 값을 저장하기 위해서 반드시 WGINT 함수를 사용해야 합니다.
2) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
3) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

2.9 Global Real F

Usage

시스템에서 미리 선언한 실수형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다.

Syntax

F<번호> 또는 F'['<번호>']'

Description

REAL로 선언된 실수형 변수와 마찬가지로 4바이트 크기의 실수 값을 저장할 수 있습니다.
Range: 3.402,823E-38 ~ 3.402,823E+38

※ 소수점 6번째자리까지 입력되며, 소수점 7번째 자리부터 반올림됩니다.

지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
프로그램 실행 중 값을 변경할 수 있습니다. 값을 변경하고 WGFLT 함수를 사용해 변경한 값을 파일에 저장할 수 있습니다.

1) <번호>는 배열처럼 입력 가능하며, 숫자를 바로 작성하여 사용할 수 있습니다. 문자 F와 <번호> 간에는 공란이 있으면 안됩니다.
2) <번호>의 범위는 0 ~ 999까지로 총 천 개까지 사용할 수 있습니다.

Examples

```
MAIN                    // 메인 함수 선언
    F[0] = 100          // 상수 값을 0번째 F 변수에 대입
    F[999] = -1         // 상수 값을 999번째 F 변수에 대입
EOP                     // 메인 함수 종료
```

Related terms

REAL, I, WGFLT

1) 프로그램 실행 중 변경한 값을 저장하기 위해서 반드시 WGFLT 함수를 사용해야 합니다.
2) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
3) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

2.10 Global Position GP

Usage

시스템에서 미리 선언한 조인트 좌표계 위치형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다.

Syntax

GP<번호> 또는 GP'['<번호>']'

Description

POS로 선언된 조인트 좌표계 위치형 변수와 마찬가지로 최대 DOF까지의 위치 값을 저장할 수 있습니다.
지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
프로그램 실행 중 값을 변경할 수 있습니다. 값을 변경하고 WGPNT 함수를 사용해 변경한 값을 파일에 저장할 수 있습니다.
위치형 상수를 통해 값을 대입할 수 있고, 개별 요소 접근으로도 값을 변경할 수 있습니다.
※ 개별 요소 접근 방식으로 초기화 시, 반드시 USED 요소를 1로 초기화해야 합니다.

1) <번호>는 배열처럼 입력 가능하며, 숫자를 바로 작성하여 사용할 수 있습니다. 문자 GP와 <번호> 간에는 공란이 있으면 안됩니다.
2) <번호>의 범위는 0 ~ 1,999까지 총 2,000개 사용할 수 있습니다.

Examples

```
MAIN                                        // 메인 함수 선언
    GP[20] = <100,100,100,100,100,100>      // 상수 값을 20번째 GP 변수에 대입
    GP[20].1 = 200                          // GP 20번째 개별 요소 접근
    GP[20].2 = 50
    GP[20].3 = GP[20].1+GP[20].2
    GP[20].USED = 1
EOP                                         // 메인 함수 종료
```

Related terms

POS, XPOS, UPOS, P, WGPNT

1) 프로그램 실행 중 변경한 값을 저장하기 위해서 반드시 WGPNT 함수를 사용해야 합니다.
2) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
3) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

2.11 TMR

Usage

시스템에서 미리 선언한 타이머 변수로, 카운트 값을 지정하거나 카운트 값 확인을 위한 변수입니다.

Syntax

TMR[(]<번호>[)] = <값>
<변수> = TMR[(]<번호>[)]

Description

지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
정수 값을 입력하는 순간부터 값이 할당되며, 10ms 간격마다 값이 1씩 증가합니다.
4.PARA → PUB → 1.HW_CONF(2/3) → 3.TMR 항목에서 설정한 값을 만나면 초기화 됩니다.

※ 자세한 설명은 N2 조작 및 운용 설명서(N2-OM-K□□)를 참고하시기 바랍니다.

1) <번호>는 괄호 사이에 숫자를 바로 작성하여 사용합니다.
2) 입력 값은 정수형 변수 및 상수만 가능합니다.

Return value

<변수> Data type: <정수형 변수>
<정수형>의 범위는 0 ~ 60,000까지입니다.

Arguments

<번호> Data type: <정수형 변수/상수>
<번호>의 범위는 0 ~ 1까지입니다.

Examples

```
MAIN                        // 메인 함수 선언
    INT AA, BB              // 정수형 변수 선언

    AA = 100                // AA 변수에 초기값 대입
    TMR0 = 50               // TMR 0번에 카운트 값 50 입력
    TMR1 = AA               // TMR 1번에 변수 AA 값 입력
    TMR(0) = 50             // TMR 0번에 카운트 값 50 입력
    TMR(1) = AA             // TMR 1번에 변수 AA 값 입력
    DLAY 100                // 시간지연
    BB = TMR(0)             // BB 변수에 TMR 0번 카운트 값 저장
EOP                         // 메인 함수 종료
```

2.12 MVR

Usage

시스템에서 미리 선언한 변수로, 전체 이동거리에 대한 이동거리비율을 반환합니다.

Syntax

<변수> = MVR[(<로봇 번호>)]
MVR[(<로봇 번호>)] = <값>

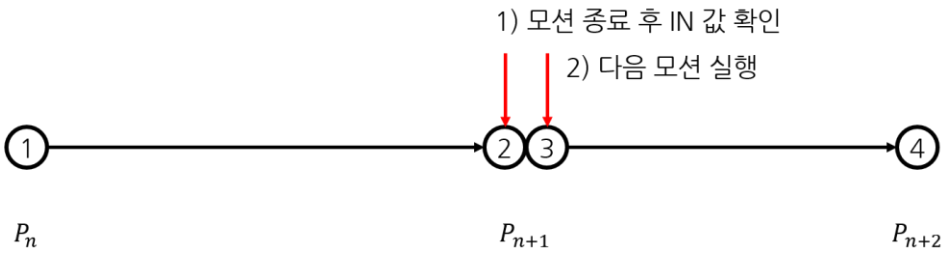
Description

지역 변수와 달리 프로그램에서 선언하지 않고 사용할 수 있습니다.
전체 이동거리에 대한 백분율 값을 반환합니다.

- MVR = (현재 로봇 이동거리 / 로봇 전체 이동거리) * 100

로봇 동작 중 조건분기 및 I/O 병렬 처리에 자주 사용됩니다. 이 경우 WITH … ENDWT문과 병행하여 사용됩니다.
아래 그림과 같이 위치(P_n, P_{n+1})의 이동 중에 다음 위치(P_{n+1}, P_{n+2}) 이동 실행 시 필요한 조건을 미리 처리함으로써 전체 프로그램 실행 속도를 단축하는 등의 장점을 가질 수 있습니다.

1) 일반 사용 예시



2) MVR 사용 예시

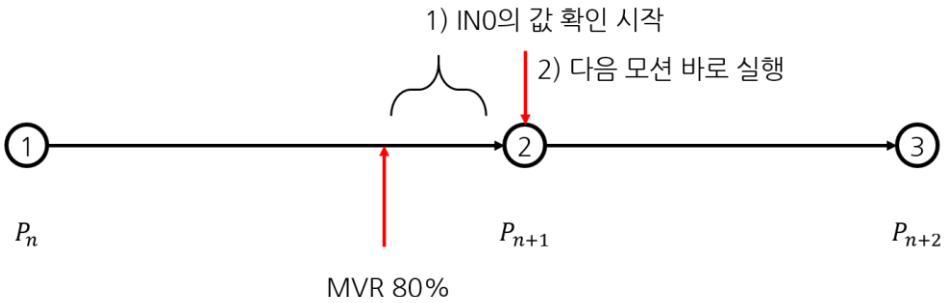


Figure 4-2 일반 사용 및 MVR 사용 비교 예시

위의 그림은 위치(P_n, P_{n+1})의 이동 중에 다음 위치(P_{n+1}, P_{n+2}) 이동 실행 조건인 IN0의 값을 미리 확인해 전체 프로그램의 실행 속도를 단축하는 예시를 도식화 한 것입니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇의 이동거리비율을 가져올 수 있습니다.

Return value

<변수>	Data type: <정수형 변수> 또는 <실수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수> <값>의 범위는 0 ~ 100까지입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    REAL VALUE                          // 실수형 변수 선언
    VALUE = MVR                         // MVR 값 대입
    VALUE = MVR(1)                      // 로봇 1번의 MVR 값 대입
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    MVR = 0                             // MVR 값 0으로 설정
    WITH                                // WITH문 시작
        JMOV GP0                        // 현재 위치에서 GP0로 JMOV
        WHILE MVR < 100                 // MVR 값이 100보다 작은 경우 반복 실행
            IF (MVR >80)&&(IN0 == 1) THEN   // 이동거리가 80%초과, IN0가 1인 경우
                OUT0 = 1                // OUT0의 신호를 1로 변경
                GOTO BB                 // 분기점 BB로 이동
            ENDIF                       // 조건문 종료
        ENDWL                           // 조건 반복문 종료
    ENDWT                               // WITH 문 종료
    LABL BB                             // 분기점 BB 선언
    JMOV GP1                            // 현재 위치에서 GP1로 JMOV
EOP                                     // 메인 함수 종료
```

Related terms

HERE, HERE_REF, WITH … ENDWT, WHILE … ENDWL, IF …(ELSE IF …ELSE…) ENDIF

1) 이동거리가 짧고 고속으로 운전하는 경우 MVR 값이 빠르게 변하여 확인이 불가능할 수 있습니다. MVR 값 설정에 유의 바랍니다. 또한 이 경우 로봇의 속도를 낮추어 사용하시기 바랍니다.

2.13 HERE

Usage

시스템에서 미리 선언한 변수로 현재 위치를 조인트 각도 값으로 반환합니다.

Syntax

<변수> = HERE
<변수> = HERE[(]<로봇 번호>[)]

Description

로봇의 현재 위치를 조인트 각도 값으로 반환합니다. 다른 위치형 변수에 값을 저장하려고 하는 경우 알람이 발생합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇의 위치 값도 가져올 수 있습니다.

Return value

<변수>	Data type: <조인트 좌표계 위치형>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    POS CUR_POS         // 조인트 좌표계 위치형 변수 선언
    CUR_POS = HERE      // 현재 조인트 각도 값 대입
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    POS R1_POS          // 조인트 좌표계 위치형 변수 선언
    R1_POS = HERE1      // 1번째 로봇의 현재 조인트 각도 값 대입
    R1_POS = HERE(1)    // 1번째 로봇의 현재 조인트 각도 값 대입
EOP                     // 메인 함수 종료
```

Related terms

MVR, POS, P, GP, HERE_REF

2.14 HERE_REF

Usage

시스템에서 미리 선언한 변수로 현재 레퍼런스 위치를 조인트 각도 값으로 반환합니다.

Syntax

<변수> = HERE_REF
<변수> = HERE_REF [([)<로봇 번호>([)]

Description

로봇의 현재 레퍼런스 위치를 조인트 각도 값으로 반환합니다. 다른 위치형 변수에 값을 저장하려고 하는 경우 알람이 발생합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇의 위치 값도 가져올 수 있습니다.

Return value

<변수>	Data type: <조인트 좌표계 위치형>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    POS CUR_POS             // 조인트 좌표계 위치형 변수 선언
    CUR_POS = HERE_REF      // 현재 레퍼런스 조인트 각도 값 대입
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    POS R1_POS              // 조인트 좌표계 위치형 변수 선언
    R1_POS = HERE_REF1      // 1번째 로봇의 현재 레퍼런스 조인트 각도 값 대입
    R1_POS = HERE_REF(1)    // 1번째 로봇의 현재 레퍼런스 조인트 각도 값 대입
EOP                         // 메인 함수 종료
```

Related terms

MVR, POS, P, GP, HERE

3. 전처리기 및 주석

3.1 INCLUDE

Usage

특정 프로그램 파일을 현재 위치에 첨부하여 사용할 수 있습니다.

Syntax

INCLUDE “<JOB 파일명>”

Description

INCLUDE 명령어는 메인 함수를 작성하기 전에 사용되어야 합니다.
첨부한 JOB 파일 내부의 함수 및 전역 변수의 사용이 가능합니다.
첨부한 파일의 JOB 메인 프로그램을 실행하고자 하는 경우 JOB 파일 뒤에 “_MAIN()”를 추가해야 합니다.

1) <JOB 파일명>에는 파일 마침표 “.” 와 확장자 “JOB”을 제외한 파일의 이름만 작성해 주십시오.
2) 첨부된 파일 내부의 함수 및 전역변수는 선언한 이름과 형식을 그대로 사용할 수 있습니다

Examples

Example 1

```
INCLUDE “INC”                    // 파일 INC.JOB 을 첨부함

MAIN                             // 메인 함수 선언
    …
EOP                              // 메인 함수 종료
```

Example 2

```
INCLUDE “INC”                    // 파일 INC.JOB 을 첨부함

MAIN                             // 메인 함수 선언
    INC_MAIN()                   // INC 파일의 메인 함수 실행
EOP                              // 메인 함수 종료
```

Example 3

```
INCLUDE “INC”                    // 파일 INC.JOB 을 첨부함

MAIN                             // 메인 함수 선언
    SUB_FUNC(5)                  // INC 파일의 SUB_FUNC() 함수 실행
EOP                              // 메인 함수 종료
```

Related terms

JCALL

3.2 주석 (Comment)

Usage

프로그램 실행에 어떠한 효과도 갖지 않도록 처리합니다.

Syntax

// [<주석>]
/* [<주석>] */

Description

일반적으로 주석은 보통 기술이나 참조, 설명문을 작성하는 용도로 사용됩니다.
실행 시 아무 영향을 주지 않습니다.

1) "//" 기호를 사용하는 경우 해당 기호가 표시된 위치를 기준으로 우측 방향 한 줄을 주석으로 처리합니다.
2) "/*" 및 "*/" 기호를 사용하는 경우 해당 기호 사이의 모든 문자열을 주석으로 처리합니다. 여러 줄을 주석으로 처리할 경우 사용됩니다.

Arguments

<주석>	모든 문자열

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // 현재 위치에서 GP0까지 JMOV
    //DLAY 1000         // 한 줄 주석처리 됨
    JMOV GP1            // 현재 위치에서 GP1까지 JMOV
    DLAY 1000           // 1초 지연
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    /*                  // 여러 줄 주석 시작
    JMOV GP0
    DLAY 1000
    */                  // 여러 줄 주석 종료
    JMOV GP1            // 현재 위치에서 GP1까지 JMOV
    DLAY 1000           // 1초 지연
EOP                     // 메인 함수 종료
```

4. 연산자

4.1 대입연산자

Usage

변수에 새 값을 할당하는데 사용됩니다. 이 연산자는 상수에서 식에 이르기까지 모두 사용할 수 있습니다.

Syntax

〈변수〉 = 〈모든 상수, 변수 또는 식〉

Description

저장하는 변수형에 맞게 데이터를 변환하여 저장합니다.
형 변환이 불가능한 경우 알람이 발생합니다.

Return value

〈변수〉 Data type: 〈모든 변수〉

Arguments

Data type: 값 또는 식의 결과 값이 반환 값과 동일한 형태의 값

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA, BB, CC      // 정수형 변수 선언
    AA = 1              // 변수에 상수 값 대입
    BB = AA             // 변수에 변수 대입
    CC = (AA+BB)*5      // 변수에 식 대입
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    DEFSTR STR1, STR2   // 문자열 변수 선언
    STR1 = “CONSTANT”   // 변수에 상수 값 대입
    STR2 = STR1+” ASSIGN”   // 변수에 식 대입
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    POS AP, BP, CP, DP  // 위치형 변수 선언
    AP = P0             // 변수에 프로그램 변수 대입
    BP = GP0            // 변수에 전역 변수 대입
    CP = 〈0,0,0,0,0,0〉  // 변수에 상수 대입
    DP = AP+BP+CP       // 변수에 식 대입
EOP                     // 메인 함수 종료
```

4.2 산술연산자

Usage

산술 연산을 수행하는 연산자입니다.

Syntax

<변수> = <A항><산술연산자><B항>

Description

산술 연산의 유형을 정의하는데 사용합니다. 아래 표의 규칙에 따릅니다.

기호	의미	사용 가능한 데이터 타입	
		A항	B항
+	더하기	모든 데이터 타입	
-	빼기	문자열을 뺀 모든 타입	
*	곱하기	문자열을 뺀 모든 타입	정수 또는 실수
/	나누기	문자열을 뺀 모든 타입	정수 또는 실수
%	나머지	정수(실수를 입력해도 정수로 변환 됨)	

나눗셈의 나머지(%)를 구하는 경우 부호의 결정은 제2장 9.3절 산술연산자를 참고하십시오.

Return value

<변수> Data type: <모든 변수>

Arguments

Data type: 값 또는 식의 결과 값이 반환 값과 동일한 형태의 값

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR                          // 정수 덧셈
    I2 = I0 + I1                        // 위치형 덧셈
    GP2 = P0 + P1                       // 문자열 덧셈
    STR = “ADD” + “STRING”              // 메인 함수 종료
EOP
```

Example 2

```
MAIN                                    // 메인 함수 선언
    F2 = F0 - F1                        // 실수 뺄셈
    GP2 = GP0 - GP1                     // 위치형 뺄셈
    GP3 = GP0 - 1                       // 위치형 개별 데이터 모두에 빼기 1
EOP                                     // 메인 함수 종료
```

Example 3

```
MAIN                                    // 메인 함수 선언
    I2 = I0 * I1                        // 정수 곱셈
    GP2 = P0 * 1.5                      // 위치형 개별 데이터 모두에 곱하기 1.5
EOP                                     // 메인 함수 종료
```

Example 4

```
MAIN                                    // 메인 함수 선언
    F2 = F0 / F1                        // 실수 나눗셈
    GP2 = <100,100,100,100,100,100> / 2 // 위치형 개별 데이터 모두에 나누기 2
EOP                                     // 메인 함수 종료
```

Example 5

```
MAIN                                    // 메인 함수 선언
    I2 = I0 % I1                        // 정수 나머지 구하기
EOP                                     // 메인 함수 종료
```

4.3 관계연산자

Usage

조건식에 사용되는 피연산자들을 비교하여 지정된 관계의 유효성을 확인하는 연산자입니다.

Syntax

[<변수> =]<A항><관계연산자><B항>

Description

주로 정수형과 실수형의 자료를 대상으로 하며, IF 문 또는 WHILE 문의 조건식에 사용됩니다. 아래 표는 각 기호의 의미를 설명합니다.

기호	의미
<	작음
>	큼
<=	작거나 같음
>=	크거나 같음
==	같음
!=	같지 않음

Return value

<변수> Data type: <정수형 변수>
관계 연산의 결과는 논리 값 (true: 1, false: 0)이 반환됩니다.

Arguments

Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    IF I0 > I1 THEN         // I0의 값이 I1 보다 클 때,
        OUT0 = 1            // OUT0의 신호를 1로 출력
    ENDIF                   // 조건문 종료
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    INT VALUE               // 정수형 변수 선언
    VALUE = F0 > F1         // F0와 F1값 비교 후 F0가 클 경우 1 반환
EOP                           F1이 클 경우 0 반환
                            // 메인 함수 종료
```

4.4 논리연산자

Usage

하나 혹은 두 개의 피연산자의 논리적 관계를 확인하기 위해 사용되는 연산자입니다.

Syntax

[<변수> =]<A항><논리연산자><B항>
[<변수> =]<논리연산자><A항>

Description

논리연산자는 일반적인 산술 변환을 수행하지 않습니다. 대신 논리 값(true: 1, false: 0)에 해당하는 개념으로 각 피연산자를 평가합니다.

기호	의미
&&	논리 AND 연산자는 두 피연산자가 모두 0이 아닌 값이 있는 경우 값 1을 반환합니다. 피연산자 중 하나가 0이면 결과는 0입니다.
||	논리 OR 연산자는 두 피연산자가 모두 0 값이 있는 경우 값 0을 반환합니다. 피연산자 중 하나라도 0이 아닌 값이 있는 경우 결과는 1입니다.
!	논리 부정(논리 NOT) 연산자는 해당 피연산자가 true(0이 아닌 값) 이면 0을 생성하고, false(0)이면 1을 생성합니다.

Return value

<변수> Data type: <정수형 변수>
논리 연산의 결과는 논리 값 (true: 1, false: 0)이 반환됩니다.

Arguments

Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

```
MAIN                                // 메인 함수 선언
    IF I0 < I1 && I2 > I3 THEN      // I0의 값이 I1보다 작고 I2의 값이 I3보다 큰 경우
        OUT(0) = 1                  // OUT0의 신호를 1로 출력
    ENDIF                           // 조건문 종료

    IF IN0 == 1 || IN1 == 1 THEN    // IN0 값이 1이거나 IN1 값이 1인 경우
        JMOV GP0                    // 현재 위치에서 GP0까지 JMOV
    ENDIF                           // 조건문 종료

    INT VALUE                       // 정수형 변수 선언
    VALUE = 0                       // 값 초기화
    IF !VALUE THEN                  // VALUE 변수의 값이 1(true)이 아니라면
       OUT(0) = 1                   // OUT0의 신호를 1로 출력
    ENDIF                           // 조건문 종료
EOP                                 // 메인 함수 종료
```

4.5 비트연산자

Usage

한 개 혹은 두 개의 피 연산자에 대해 비트 단위로 적용되는 연산입니다.

Syntax

[<변수> =]<A항><비트연산자><B항>
[<변수> =]<비트연산자><A항>

Description

비트연산자는 논리연산자와 비슷하지만 비트 단위로 연산을 수행한다는 점이 가장 큰 차이점 입니다. 또한 비트 단위로 전체 비트를 왼쪽이나 오른쪽으로 이동시킬 때도 사용됩니다.

기호	의미
&	비트 단위의 AND 연산자는 양쪽 비트가 모두 1이면 해당 결과 비트는 1로 설정됩니다. 그렇지 않으면 해당 결과 비트는 0으로 설정됩니다.
^	비트 단위의 XOR 연산자는 한 비트가 0이고 다른 비트가 1인 경우 해당 비트는 1로 설정됩니다. 그렇지 않으면 해당 결과 비트는 0으로 설정됩니다.
|	비트 단위의 OR 연산자는 어느 한쪽 비트가 1이면 해당 결과 비트는 1로 설정됩니다. 그렇지 않으면 해당 결과 비트는 0으로 설정됩니다.
<<	두 번째 피연산자가 지정하는 위치 수만큼 첫 번째 피연산자를 왼쪽으로 이동합니다.
>>	두 번째 피연산자가 지정하는 위치 수만큼 첫 번째 피연산자를 오른쪽으로 이동합니다.
~	비트 NOT 연산자는 피연산자의 비트 보수를 만듭니다. 비트가 1이면 0으로, 0이면 1로 반전시킵니다.

Return value

<변수> Data type: <정수형 변수>
관계 연산의 결과는 논리 값 (true: 1, false: 0)이 반환됩니다.

Arguments

Data type: <정수형 변수/상수>
피 연산자는 모두 정수 값이어야 합니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    IF I0 & I1 THEN     // I0와 I1의 비트 AND 연산이 참인 경우
        OUT0 = 1        // OUT0의 신호를 1로 출력
    ENDIF               // 조건문 종료
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT VALUE           // 정수형 변수 선언
    VALUE = I0 ^ I1     // I0와 I1의 비트 XOR 연산을 저장
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    IF I0 | I1 THEN     // I0와 I1의 비트 OR 연산이 참인 경우
        JMOV GP0        // 현재 위치에서 GP0까지 JMOV
    ENDIF               // 조건문 종료
EOP                     // 메인 함수 종료
```

Example 4

```
MAIN                    // 메인 함수 선언
    INT A, B            // 정수형 변수 선언
    A = 0B1011          // 변수 초기화
    B = A<<2            // 왼쪽으로 2만큼 쉬프트
EOP                     // 메인 함수 종료
```

Example 5

```
MAIN                    // 메인 함수 선언
    INT A, B            // 정수형 변수 선언
    A = 0B1011          // 변수 초기화
    B = A>>1            // 오른쪽으로 1만큼 쉬프트
EOP                     // 메인 함수 종료
```

Example 6

```
MAIN                    // 메인 함수 선언
    INT A, B            // 정수형 변수 선언
    A = 0B1011          // 변수 초기화
    B = ~A              // A 변수 값 반전하여 저장
EOP                     // 메인 함수 종료
```

4.6 괄호

Usage

식 또는 하위 식 주변에 한 쌍의 괄호를 표시함으로써 일반 연산자 우선순위를 무시하고 더 높은 우선 순위의 표현 식보다 먼저 평가될 수 있도록 하는 연산자 입니다.

Syntax

(<식>)

Description

괄호 안에 있는 것들을 그룹화 합니다.

Examples

```
MAIN                    // 메인 함수 선언
    INT A, B, C, RES    // 정수형 변수 선언
    A = 1               // 변수 초기화
    B = 2
    C = 3

    RES = (A+B)*C       // 우선순위가 높은 곱하기(*)연산보다 더하기(+)연산을 먼저 처리
EOP                     // 메인 함수 종료
```

5. 프로그램 제어

5.1 MAIN … EOP (프로그램 시작/종료)

Usage

메인 프로그램 시작 및 종료를 나타냅니다.

Syntax

```
MAIN
    …
EOP
```

Description

모든 프로그램은 메인 함수를 한 개만 작성할 수 있으며, 반드시 하나의 메인 함수를 가져야 실행할 수 있습니다.

1) MAIN 과 EOP는 반드시 같이 사용되어야 합니다.
2) EOP(End of Program)는 메인 프로그램 블록의 끝을 의미하며 JOB 프로그램 실행 시 이 명령을 만나면 JOB 프로그램이 종료됩니다. 단, JCALL에 의해 호출된 JOB은 EOP 명령어를 만나면 호출된 JOB의 수행을 종료하고, 호출한 JOB으로 복귀합니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP1            // 현재 위치에서 GP1로 JMOV
    JMOV GP2            // 현재 위치에서 GP2로 JMOV
EOP                     // 메인 함수 종료
```

Related terms

FUNC … END

5.2 FUNC … END (함수 시작/종료)

Usage

함수의 시작 및 종료를 나타냅니다.

Syntax

```
FUNC <반환 유형> <함수명> ([<매개변수>])
    …
    [RET <변수>]
END
```

Description

자주 반복해서 사용하거나 프로그램 코드 크기가 큰 경우 별도의 함수로 분할하여 작성할 수 있습니다.
프로그램 내 어느 곳에서든 여러 개의 함수를 선언하여 사용할 수 있습니다.
메인 프로그램에서 함수명과 매개변수를 입력하면 해당 함수를 실행할 수 있으며, 재귀호출(Recursive Call)도 가능합니다. 단, 함수의 호출 깊이는 100을 초과할 수 없습니다.

1) FUNC과 END 는 반드시 같이 사용되어야 합니다.
2) END는 함수 블록의 끝을 의미하며 함수 블록 실행 시 이 명령을 만나면 함수 실행이 종료됩니다.
3) <반환 유형>은 VOID(반환 없음)를 포함한 모든 데이터 타입(INT 및 REAL, DEFSTR, POS, XPOS, UPOS)의 사용이 가능합니다. 단, 시스템에서 미리 선언된 전역변수(P, GP, I, F) 및 시스템 변수(TMR 등)를 사용할 순 없습니다.
4) <함수명>은 영문자와 숫자로 구성할 수 있으며, 미리 선언된 전역 변수들과 동일하게 사용할 수는 없습니다.
5) <매개변수>는 생략이 가능하지만, 필요에 의해 함수 내부의 연산에 사용될 각종 변수들을 매개변수로 받아 사용할 수 있습니다. <매개 변수>는 모든 데이터 타입(INT 및 REAL, DEFSTR, POS, XPOS, UPOS)의 사용 가능합니다. 단, 시스템에서 미리 선언된 전역변수(P, GP, I, F) 및 시스템 변수(TMR 등)를 사용할 수 없습니다.

Arguments

<매개변수>	Data type: <모든 데이터 타입, 없는 경우 생략>

Examples

Example 1

- **반환 유형(VOID) 및 매개 변수 없는 타입**

```
FUNC VOID TEST()                    // 반환 값 및 매개변수가 없는 함수 선언
    SVON                            // 서보 ON
    JMOV GP0                        // 모션 실행
    JMOV GP1
END                                 // 함수 종료

MAIN                                // 메인 함수 선언
    TEST()                          // 함수 호출
EOP                                 // 메인 함수 종료
```

Example 2

- **반환 유형은 없고(VOID), 매개 변수(조인트 좌표계 위치형)는 있는 타입**

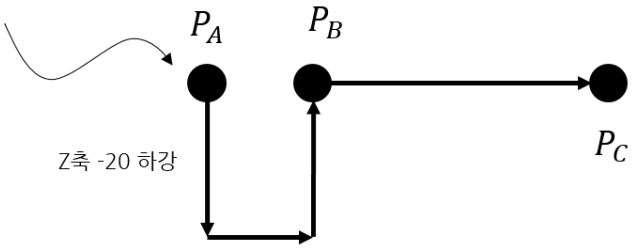


Figure 4-3 함수 동작 예시

```
MAIN                                // 메인 함수 선언
    MOVE(GP[0], GP[1])              // A, B 위치형 변수를 인자로 함수 호출
    LMOV GP[2]                      // GP[2](C위치)까지 LMOV
EOP                                 // 메인 함수 종료

FUNC VOID MOVE(POS A, POS B)        // 반환 값이 없고, 위치형 매개변수를 갖는 함수 선언
    XPOS TEMP                       // 베이스 좌표계 위치형 변수 선언
    SVON                            // 서보 온
    JMOV A                          // A 위치까지 JMOV
    TEMP = JTOB(A)                  // A 위치를 베이스 좌표계 위치값으로 변환하여 저장
    TEMP.3 = TEMP.3 - 20            // A 위치에서 Z축으로 20 하강한 위치 값 저장
    LMOV TEMP                       // TEMP 위치까지 LMOV
    TEMP = JTOB(B)                  // B 위치를 베이스 좌표계 위치 값으로 변환하여 저장
    TEMP.3 = TEMP.3 - 20            // B 위치에서 Z축으로 20 하강한 위치 값 저장
    LMOV TEMP                       // TEMP 위치까지 LMOV
    LMOV B                          // B 위치까지 LMOV
END                                 // 함수 종료
```

Example 3

- **반환 유형은 있고(실수), 매개 변수는 없는 타입**

```
MAIN                                    // 메인 함수 선언
    REAL VALUE                          // 실수형 변수 선언
    VALUE = TEST()                      // 함수 호출 및 VALUE 변수에 반환 값 저장
    PRINT(0, VALUE)                     // 모니터링 로그 0번 인덱스에 출력
EOP                                     // 메인 함수 종료

FUNC REAL TEST()                        // 실수 값을 반환하고 매개변수는 없는 함수 선언
    REAL RES                            // 실수형 변수 선언
    RES = F[0]+F[1]                     // 실수 값 덧셈
    RET RES                             // 결과 값 반환
END                                     // 함수 종료
```

Example 4

- **반환 유형(문자열) 및 매개 변수(정수, 문자열)가 모두 있는 타입**

```
MAIN                                    // 메인 함수 선언
    INT R_VAL                           // 정수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언
    WHILE 1                             // 반복문 시작
        STR = TEST("IN(0)", IN(0))      // 함수 호출 및 STR 변수에 반환 값 저장
        PRINT(0, STR)                   // 모니터링 로그 0번 인덱스에 출력
        R_VAL = STROUT(0,STR)           // RS-232 포트로 문자열 전송
        DLAY 100                        // 0.1초 지연
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료

FUNC DEFSTR TEST(DEFSTR A, INT B)       // 문자열을 반환하고 문자열 및 정수형 매개변수를 갖
    DEFSTR DATA, RES                      는 함수 선언
    DATA = HTOS(B)                      // 문자열 변수 선언
    RES = A + ":" + DATA                // 정수 값을 16진수 문자열로 변환
    RET RES                             // 각 문자열을 더해 변수에 저장
END                                     // 문자열 반환
                                        // 함수 종료
```

Example 5

- **반환 유형 및 매개 변수가 모두 있는 타입 (베이스 좌표계 위치형 예시)**

```
MAIN                                    // 메인 함수 선언
    XPOS A_P, B_P, X_T                  // 베이스 좌표계 위치형 변수 선언
    A_P = JTOB(GP[0])                   // 베이스 좌표계 위치 값 획득
    B_P = JTOB(GP[1])
    X_T = MD_X(A_P, B_P)                // 함수 호출 및 X_T 변수에 반환 값 저장
    SVON                                // 서보 ON
    LMOV X_T                            // 반환 위치로 LMOV
EOP

FUNC XPOS MD_X(XPOS A, XPOS B)          // 베이스 좌표계 위치형의 매개변수와 반환 값을 갖는
    XPOS RES                              함수 선언
    RES = (A+B)/2                       // 베이스 좌표계 위치형 변수 선언
    RET RES                             // 두 베이스 좌표 값을 더하고 반으로 나눈 결과 저장
END                                     // 베이스 좌표계 위치 값 반환
                                        // 함수 종료
```

Example 6

● **재귀 함수 예시**

```
INT $COUNT                          // 전역 변수 선언
MAIN                                // 메인 함수 선언
    $COUNT = 0                      // 변수 초기화
    RECALL()                        // 함수 호출
EOP                                 // 메인 함수 종료

FUNC VOID RECALL()                  // 함수 선언
    $COUNT = $COUNT+1               // 전역 변수 $COUNT값에 1 더함
    PRINT(0,"COUNT",$COUNT)         // 모니터링 로그 출력
    IF $COUNT < 50 THEN             // 전역 변수 $COUNT값이 50 미만인 경우
        RECALL()                    // 재귀 호출
    ENDIF                           // 조건문 종료
END                                 // 함수 종료
```

Related terms

MAIN ··· EOP, RET

5.3 RET

Usage

함수의 실행을 마치고 값을 반환하며 함수를 호출한 곳으로 복귀합니다.

Syntax

RET <반환 값>

Description

메인 함수에서는 사용될 수 없습니다.

<반환 값>은 함수의 반환 값으로 함수 선언에 따라 결정됩니다.
함수 선언 시 <반환유형>이 VOID인 경우 RET 명령어를 사용할 수 없습니다.
<반환 값>은 VOID 제외한 모든 데이터 타입(INT 및 REAL, DEFSTR, POS, XPOS, UPOS)의 사용 가능합니다. 단, 시스템에서 미리 선언된 전역변수(P, GP, I, F) 및 시스템 변수(TMR 등)를 사용할 수 없습니다.

Arguments

Data type: 함수 선언에 따라 결정됨, VOID의 경우 생략

Examples

```
FUNC INT TEST(INT A, INT B)    // 정수형 값을 반환하고 매개변수를 갖는 TEST 함수 선언
    INT VALUE                  // 정수형 변수 선언
    VALUE = A*B
    RET VALUE                  // 계산 값 반환
END                            // 함수 종료
```

Related terms

FUNC … END

5.4 FOR … TO … [BY] … NEXT (반복 수행문)

Usage

조건이 false(0)가 될 때까지 블록을 반복 수행합니다.

Syntax

FOR <초기화 문> TO <종료 조건> [BY <증분량>]
…
NEXT

Description

주로 지정된 횟수만큼 실행해야 하는 루프를 생성합니다. 초기 값이 설정된 후, 종료 조건을 만족할 때까지 FOR 블록을 반복 수행 합니다.

1) FOR … TO … NEXT는 반드시 같이 사용되어야 합니다.
2) <초기화 문>은 루프 인덱스를 초기화 하는데 사용됩니다. 한 번만 실행됩니다. 식 또는 변수를 포함할 수 있습니다.
3) <종료 조건>은 루프 종료 기준을 확인하는데 사용됩니다.
4) <증분량>을 지정할 수 있습니다. 생략 시 루프 인덱스가 1씩 증가합니다. 반복자의 증가, 감소 두 가지 다 사용 가능합니다

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT AA, SUM             // 정수형 변수 선언
    SUM = 0                 // 변수 초기화
    FOR AA = 1 TO 10 BY 2   // 1부터 10까지 2씩 증가
        SUM = SUM+AA        // SUM 변수와 AA 변수를 더해 SUM 변수에 저장
        JMOV GP[SUM]        // SUM 값에 해당하는 변수의 위치로 JMOV
    NEXT                    // FOR문 종료
    PRINT(0, SUM)           // 모니터링 로그 0번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL AA, SUM            // 실수형 변수 선언
    SUM = 0.0               // 변수 초기화
    FOR AA = 0.0 TO 1 BY 0.1 // 0 부터 1까지 0.1씩 증가
      SUM = SUM+AA          // SUM 변수와 AA 변수를 더해 SUM 변수에 저장
    NEXT                    // FOR문 종료
    PRINT(0,SUM)            // 모니터링 로그 0번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Related terms

WHILE… ENDWL

5.5 WHILE … ENDWL (조건 반복 수행문)

Usage

조건식이 참(true)인 동안만 블록을 반복 수행합니다.

Syntax

```
WHILE <조건식>
    …
ENDWL
```

Description

WHILE 문은 조건이 만족(조건식의 결과가 true, 즉, 0이 아닌 값)이 되는 동안 블록을 반복 수행합니다. 조건식의 결과가 항상 참인 경우에는 무한 반복 수행을 하게 됩니다.

1) WHILE … ENDWL은 반드시 같이 사용되어야 합니다.
2) <조건식>의 연산 결과가 true(0 이외의 값)이면 블록 내 문장들이 실행되고, false(0)이면 ENDWL 다음의 문장들이 실행됩니다.
3) <조건식> 입력 시 연산자 우선순위에 주의하십시오.

Examples

Example 1	WHILE 1 ENDWL	// WHILE ~ ENDWL 블록을 무한 실행
Example 2	WHILE IN0 == 1 ENDWL	// IN0가 1로 입력되는 동안 블록 반복 실행
Example 3	WHILE IN1 == 1 && IN0 == 1 ENDWL	// IN0와 IN1이 1로 입력되는 동안 블록 반복 실행
Example 4	WHILE MVR > 10 ENDWL	// MVR 변수 값이 10 보다 큰 동안만 블록을 반복 실행

Related terms

FOR … TO … [BY] … NEXT

5.6 IF …THEN … ENDIF (조건 분기 명령어)

Usage

조건식의 연산 결과에 따라 블록을 실행합니다.

Syntax

```
IF <조건식> THEN
    …
[ELSE IF <조건식> THEN]
    …
[ELSE]
    …
ENDIF
```

Description

조건식의 연산 결과에 따라 블록의 실행 여부를 결정할 수 있습니다.

1) IF … THEN … ENDIF는 반드시 같이 사용되어야 합니다.
2) ELSEIF, ELSE는 사용자가 선택적으로 사용할 수 있습니다.
3) 조건식의 연산 결과가 true(0 이외의 값)이면 THEN 이하의 문장들이 실행되고, false(0)이면 ELSE 다음의 문장들이 실행됩니다.
4) IF 문 내에서 다시 IF 문을 사용할 수 있습니다. 이 때, IF와 ENDIF가 논리적으로 쌍을 이룰 수 있도록 주의하십시오.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    IF IN0 == 1 THEN    // IN0가 1이면
        JMOV GP0        // 현재 위치에서 GP0까지 JMOV
    ENDIF               // IF문 종료
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    INT BB                  // 정수형 변수 선언
    IF I0 == 1 THEN         // I0의 값이 1이면
        STOP                // 로봇 정지
    ELSEIF I0 == 2 THEN     // I0의 값이 2이면
        EXIT                // 프로그램 종료
    ELSE                    // 조건식이 하나도 만족되지 않는 경우
        BB = IN0            // IN0의 값을 정수형 변수에 저장
        IF BB == 1 THEN     // BB 변수가 1이면
            DLAY 10         // 10ms 대기
            JMOV GP0        // 현재 위치에서 GP0까지 JMOV
        ENDIF               // IF문 종료
    ENDIF                   // IF문 종료
EOP                         // 메인 함수 종료
```

5.7 LABL … GOTO (분기 명령어)

Usage

분기점 위치를 지정하고, 해당 분기 위치로 이동하는 명령어 입니다.

Syntax

```
LABL <레이블 명>
    …
GOTO <레이블 명>
```

Description

GOTO 명령은 조건없이 해당 분기점으로 분기하며 동일 블록 내에서 가능합니다.
예를 들면, MAIN 블록 내에서 FUNC 블록으로, FUNC 블록 내에서 다른 FUNC 블록으로 분기할 수 없습니다.
IF 및 FOR, WHILE 블록을 사용할 때, 내부에서 외부로의 분기는 가능하지만, 반대의 경우는 불가능합니다.

1) <레이블 명>은 영문자와 숫자로 구성할 수 있으며, 선언 시 첫 글자를 숫자로 시작할 수 없습니다. 또한, 미리 선언된 전역 변수들과 동일하게 사용할 수 없습니다. 또, 한 파일 안에 동일한 레이블 명을 중복해서 선언할 수 없습니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    VEL 500             // 속도 변경
LABL REPEAT             // 분기점 선언
    JMOV GP0            // GP0까지 JMOV
    JMOV GP1            // GP1까지 JMOV
    GOTO REPEAT         // 분기점으로 이동
    SVOF                // 서보 OFF
EOP                     // 메인 함수 종료
```

Exampl2

```
FUNC VOID TEST()        // 함수 선언
    SVON                // 서보 ON
    OUT0 = 1            // OUT0에 1을 출력
LABL REPEAT             // 분기점 선언
    JMOV GP0            // GP0까지 JMOV
    JMOV GP1            // GP1까지 JMOV
    OUT0 = 0            // OUT0에 0을 출력
    GOTO REPEAT         // 분기점으로 이동
END                     // 함수 종료
```

5.8 CONTINUE

Usage

반복문 루프 내에서 뒤의 구문을 수행하지 않고 바로 반복문 후처리 구문 및 조건식으로 분기할 때 사용합니다.

Syntax

```
WHILE <조건식>
    IF <조건식> THEN
        CONTINUE
    ENDIF
ENDWL
```

Description

CONTINUE문을 만나면 반복문의 맨 첫 줄로 이동하게 됩니다.

Examples

```
MAIN                                    // 메인 함수 선언
    WHILE 1                             // 무한 반복
        SVON                            // 서보 ON
        JMOV GP1                        // GP1까지 JMOV
        IF IN(1) == 1 THEN              // IN1의 값이 1인 경우
            CONTINUE                    // 첫 줄로 이동(→ WHILE 1)
        ENDIF                           // 조건식 종료
        JMOV GP0                        // GP0까지 JMOV
    ENDWL                               // WHILE문 종료
EOP                                     // 메인 함수 종료
```

Related terms

FOR … TO … [BY] … NEXT, WHILE … ENDWL, BREAK

5.9 BREAK

Usage

반복문 루프 내에서 어떤 특정 조건이 발생했을 때 탈출을 위해 사용합니다.

Syntax

```
WHILE <조건식>
    IF <조건식> THEN
        BREAK
    ENDIF
ENDWL
```

Description

BREAK문을 만나면 반복문을 탈출합니다.

Examples

```
MAIN                                // 메인 함수 선언
    WHILE 1                         // 무한 반복
        SVON                        // 서보 ON
        JMOV GP1                    // GP1까지 JMOV
        IF IN(1) == 1 THEN          // IN1의 값이 1인 경우
            BREAK                   // 반복문 탈출
        ENDIF                       // 조건식 종료
        JMOV GP0                    // GP0까지 JMOV
    ENDWL                           // WHILE문 종료
EOP                                 // 메인 함수 종료
```

Related terms

FOR … TO … [BY] … NEXT, WHILE … ENDWL, CONTINUE

5.10 JCALL (프로그램 호출)

Usage

JOB 파일을 호출해 즉시 실행합니다.

Syntax

JCALL <JOB 파일명>

Description

호출된 JOB 프로그램이 EOP 명령어를 만나면 호출한 프로그램의 JCALL 명령어 다음으로 복귀합니다.
INCLUDE 명령어와 마찬가지로 호출되는 파일 내부의 전역 변수 및 함수의 사용이 가능합니다.
JCALL은 재귀호출이 되지 않도록 프로그램을 작성해야 합니다.

1) <JOB 파일명>에는 파일 마침표 "." 와 확장자 "JOB"을 제외한 파일의 이름만 작성해 주십시오.

Examples

```
MAIN                    // 메인 함수 선언
    JCALL TEST          // TEST.JOB 프로그램 호출 및 즉시 실행
EOP                     // 메인 함수 종료
```

Related terms

INCLUDE

5.11 STOP (로봇 정지)

Usage

로봇의 동작을 정지합니다.

Syntax

STOP

Description

이동 중인 로봇을 정지시키고, 다음 STEP의 명령어를 처리합니다. 즉, 수행 중인 모션을 정지한 후 다음 STEP을 계속 실행합니다.

Examples

```
MAIN                                // 메인 함수 선언
    SVON                            // 서보 ON
    WITH                            // WITH 문 실행
        JMOV GP0                    // GP0까지 JMOV
        WHILE MVR < 90              // MVR 값이 90 미만인 동안만 반복 실행
            IF IN0 == 1 THEN        // IN0가 1인 경우
                GOTO AA             // 분기점 AA로 이동
            ENDIF                   // IF문 종료
        ENDWL                       // WHILE 문 종료
LABL AA                             // 분기점 AA 선언
        STOP                        // 로봇 정지
    ENDWT                           // WITH문 종료
EOP                                 // 메인 함수 종료
```

Related terms

EXIT

5.12 EXIT (JOB 프로그램 정지)

Usage

JOB 프로그램 실행을 정지합니다.

Syntax

EXIT

Description

EXIT 명령어는 JOB 프로그램 실행을 정지합니다. 이때, 제어기는 "EXIT Instruction" 알람을 발생합니다.

Examples

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    WITH                                // WITH 문 실행
        JMOV GP0                        // GP0까지 JMOV
        WHILE MVR < 90                  // MVR 값이 90 미만인 동안만 반복 실행
            IF IN0 == 1 THEN            // IN0가 1인 경우
                GOTO AA                 // 분기점 AA로 이동
            ENDIF                       // IF문 종료
        ENDWL                           // WHILE 문 종료
LABL AA                                 // 분기점 AA 선언
        EXIT                            // 프로그램 정지 및 EXIT 알람 발생
    ENDWT                               // WITH문 종료
EOP                                     // 메인 함수 종료
```

Related terms

STOP

6. 로봇 동작

6.1 SVON (서보 ON)

Usage

서보 모터 전원을 켭니다.

Syntax

SVON[(<로봇 번호>)]

Description

JOB 프로그램에 할당된 로봇 전축의 서보 모터 전원을 켭니다.
서보 모터의 전원이 켜지지 않은 상태에서 이동 명령을 사용하면 알람이 발생합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇의 서보 모터 전원을 켤 수 있습니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    SVOF                // 서보 OFF
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언
    STATE = RSTATE(1,6) // 로봇 1번 서보 상태 저장
    IF STATE == 0 THEN  // 서보가 OFF인 경우
        SVON(1)         // 로봇 1번의 서보 ON
    ENDIF               // 조건문 종료
EOP                     // 메인 함수 종료
```

Related terms

SVOF

6.2 SVOF (서보 OFF)

Usage

서보 모터 전원을 끕니다.

Syntax

SVOF[(<로봇 번호>)]

Description

JOB 프로그램에 할당된 로봇 전축의 서보 모터 전원을 끕니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇의 서보 모터 전원을 끌 수 있습니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    SVOF                // 서보 OFF
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                            // 메인 함수 선언
    INT STATE                   // 정수형 변수 선언
    STATE = RSTATE(1,6)         // 로봇 1번 서보 상태 저장
    IF STATE == 1 THEN          // 서보가 ON인 경우
        SVOF(1)                 // 로봇 1번의 서보 OFF
    ENDIF                       // 조건문 종료
EOP                             // 메인 함수 종료
```

Related terms

SVON

6.3 JMOV (PTP 이동)

Usage

현재 위치에서 목표점으로 이동합니다.

Syntax

JMOV <목표위치> [, FOS = <값>, VEL = <값>, ACC = <값>, DEC = <값>, T = <값>, U = <값>]

Description

점에서 점으로 이동하는 PTP(Point to Point) 이동을 하게 됩니다.
이동하는 경로는 로봇의 자세에 의존하게 되며, 그 경로는 직선동작에 한정되지 않습니다.

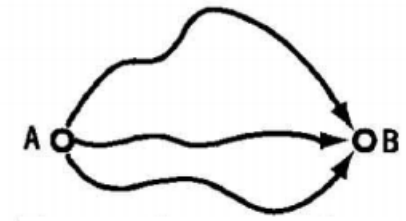


Figure 4-4 PTP 이동 예시

명령어 사용 시 FOS, VEL, ACC, DEC, T, U의 부가항목을 추가로 입력할 수 있습니다. FOS, VEL과 같은 키워드를 지정하고 값을 대입하면 모션 명령어에 변화를 줄 수 있습니다. 각각의 키워드는 순서와 관계없이 사용할 수 있으며, 일부만 사용해도 됩니다.

1) <목표위치>는 모든 위치형 변수의 사용이 가능합니다. JMOV 명령어를 사용하게 되는 경우 입력되는 위치형 변수를 조인트 좌표계 위치형 변수로 변환하여 이동하게 됩니다.
2) FOS(Factor of smooth path)는 정지할 필요가 없는 중간 경유점을 정지하지 않고 통과하여 연속적으로 다음 목표 위치로 이동시키는 기능을 의미합니다. 다음 목표점까지의 거리비율(%)을 참조하여 궤적을 변경합니다. 거리비율은 전체 이동거리의 백분율(%)입니다.
3) VEL은 축 이동속도를 의미하며, 천분율(%)로 나타냅니다. 아래 수식과 같이 이동속도를 계산합니다. 정격속도 및 초기속도는 MOTION 파라미터에서 설정됩니다.
   - 축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001
4) ACC는 축 가속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 가속시간을 계산합니다. 정격가속시간은 MOTION 파라미터에서 설정됩니다.
   - 가속시간 = 정격가속시간 X 백분율 값 X 0.01
5) DEC는 축 감속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 감속시간을 계산합니다. 정격감속시간은 MOTION 파라미터에서 설정됩니다.
   - 감속시간 = 정격감속시간 X 백분율 값 X 0.01
6) T는 툴 번호를, U는 사용자 좌표계 번호를 의미합니다. 모션 수행 시 사용할 툴을 선택하거나 좌표계를 선택하는데 사용됩니다.

Arguments

〈목표위치〉	Data type: 〈모든 위치형 변수〉
FOS	Factor of smooth path Data type: 〈정수형 변수/상수〉 FOS의 범위는 0 ~ 50%까지 입니다.
VEL	Velocity Data type: 〈정수형 변수/상수〉 VEL의 범위는 1 ~ 1,000%까지 입니다.
ACC	Acceleration Data type: 〈정수형 변수/상수〉 ACC의 범위는 1 ~ 400%까지 입니다.
DEC	Deceleration Data type: 〈정수형 변수/상수〉 DEC의 범위는 1 ~ 400%까지 입니다.
T	Tool number Data type: 〈정수형 변수/상수〉 T의 범위는 0 ~ 15입니다. TOOL 명령어와 같은 역할을 합니다.
U	User number Data type: 〈정수형 변수/상수〉 U의 범위는 0 ~ 15입니다. USER 명령어와 같은 역할을 합니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS AP                              // 위치형 변수 선언
    AP = 〈400.0, 50.0, 10.4, 10.0, 0, 0〉  // 변수 초기화
    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지 JMOV
    JMOV AP                             // AP까지 JMOV
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    XPOS XP                             // 위치형 변수 선언
    XP = 〈250,250,350,-30,30,60,ARM = 0,T = 2〉  // 변수 초기화
    SVON                                // 서보 ON
    JMOV XP                             // XP까지 JMOV
EOP                                     // 메인 함수 종료
```

Example 3

```
MAIN                                                    // 메인 함수 선언
    SVON                                                // 서보 ON
    JMOV GP0, FOS = 30, VEL = 80, ACC = 50, DEC = 50    // 조건에 맞추어 GP0로 JMOV
    JMOV GP1, DEC = 150, VEL = 200, ACC = 150           // 조건에 맞추어 GP1로 JMOV
EOP                                                     // 메인 함수 종료
```

Related terms

LMOV, FOS, VEL, ACC, DEC, TOOL, USER

6.4 LMOV (직선 보간 이동)

Usage

현재 위치에서 목표점으로 직선 보간 이동합니다.

Syntax

LMOV <목표위치> [, FOS = <값>, VEL = <값>, ACC = <값>, DEC = <값>, T = <값>, U = <값>]

Description

시작 위치에서 목표 위치에 도달하는 경로가 직선이 되도록 보간합니다.

Figure 4-5 직선보간 이동 예시

명령어 사용 시 FOS, VEL, ACC, DEC, T, U의 부가항목을 추가로 입력할 수 있습니다. FOS, VEL과 같은 키워드를 지정하고 값을 대입하면 모션 명령어에 변화를 줄 수 있습니다. 각각의 키워드는 순서와 관계없이 사용할 수 있으며, 일부만 사용해도 됩니다.

1) <목표위치>는 모든 위치형 변수의 사용이 가능합니다. LMOV 명령어를 사용하게 되는 경우 입력되는 위치형 변수를 베이스 좌표계 위치형 변수로 변환하여 이동하게 됩니다.
2) FOS(Factor of smooth path)는 정지할 필요가 없는 중간 경유점을 정지하지 않고 통과하여 연속적으로 다음 목표 위치로 이동시키는 기능을 의미합니다. 다음 목표점까지의 거리비율(%)을 참조하여 궤적을 변경합니다. 거리비율은 전체 이동거리의 백분율(%)입니다.
3) VEL은 축 이동속도를 의미하며, 천분율(%)로 나타냅니다. 아래 수식과 같이 이동속도를 계산합니다. 정격속도 및 초기속도는 MOTION 파라미터에서 설정됩니다.
   - 축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001
4) ACC는 축 가속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 가속시간을 계산합니다. 정격가속시간은 MOTION 파라미터에서 설정됩니다.
   - 가속시간 = 정격가속시간 X 백분율 값 X 0.01
5) DEC는 축 감속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 감속시간을 계산합니다. 정격감속시간은 MOTION 파라미터에서 설정됩니다.
   - 감속시간 = 정격감속시간 X 백분율 값 X 0.01
6) T는 툴 번호를, U는 사용자 좌표계 번호를 의미합니다. 모션 수행 시 사용할 툴을 선택하거나 좌표계를 선택하는데 사용됩니다.

Arguments

<목표위치>	Data type: <모든 위치형 변수>
FOS	Factor of smooth path

Data type: <정수형 변수/상수>
FOS의 범위는 0 ~ 50%까지 입니다.

VEL — Velocity
Data type: <정수형 변수/상수>
VEL의 범위는 1 ~ 1,000%까지 입니다.

ACC — Acceleration
Data type: <정수형 변수/상수>
ACC의 범위는 1 ~ 400%까지 입니다.

DEC — Deceleration
Data type: <정수형 변수/상수>
DEC의 범위는 1 ~ 400%까지 입니다.

T — Tool number
Data type: <정수형 변수/상수>
T의 범위는 0 ~ 15입니다. TOOL 명령어와 같은 역할을 합니다.

U — User number
Data type: <정수형 변수/상수>
U의 범위는 0 ~ 15입니다. USER 명령어와 같은 역할을 합니다.

Examples

Example 1

```
MAIN                                        // 메인 함수 선언
    XPOS XP                                 // 위치형 변수 선언
    XP = <250,250,350,-30,30,60,ARM = 0,T = 2>  // 변수 초기화
    SVON                                    // 서보 ON
    LMOV XP                                 // XP까지 LMOV
EOP                                         // 메인 함수 종료
```

Example 2

```
MAIN                                        // 메인 함수 선언
    POS AP                                  // 위치형 변수 선언
    AP = <30.0, -60.0, 10.4, 10.0, 0, 0>    // 변수 초기화
    SVON                                    // 서보 ON
    LMOV AP, T = 1                          // T1번을 사용해 AP로 LMOV
EOP                                         // 메인 함수 종료
```

Example 3

```
MAIN                                        // 메인 함수 선언
    SVON                                    // 서보 ON
    LMOV GP0, FOS = 30, VEL = 50, ACC = 80, DEC = 40,   // 조건 참조해, GP0로 LMOV
    U = 1
    LMOV GP1, DEC = 150, VEL = 200, ACC = 150, T = 1    // 조건 참조해, GP1로 LMOV
EOP                                         // 메인 함수 종료
```

Related terms

JMOV, FOS, VEL, ACC, DEC, TOOL, USER

1) 로봇의 위치와 자세에 따라 동작할 수 없는 경우가 존재합니다. 이때, “[1219] Range Over error”, 또는 “[1415] ik position error” 등의 알람이 발생할 수 있습니다. 로봇 동작 조건 명령어를 함께 사용하시는 경우 주의하시기 바랍니다.
2) 작업 포인트 티칭에 유의하여 주십시오.

6.5 CMOV (원형 보간 이동)

Usage

현재 위치에서 경유점 1, 2를 잇는 원을 그리며 이동합니다.

Syntax

CMOV <경유점1> <경유점2> [, VEL = <값>, ACC = <값>, DEC = <값>, T = <값>, U = <값>]

Description

동작 목표위치에 도달하는 경로를 원을 그리며 보간합니다. 원형 보간 동작을 할 때 총 3개의 위치 좌표(시작, 경유점1, 경유점2)가 필요합니다.
CMOV를 사용할 때에는 Figure 4-6과 같이 JMOV를 사용하여 이동 후 CMOV를 사용할 것을 권장합니다.

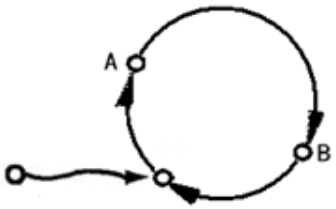


Figure 4-6 원형 보간 예시

명령어 사용 시 VEL, ACC, DEC, T, U의 부가항목을 추가로 입력할 수 있습니다.
VEL과 같은 키워드를 지정하고 값을 대입하면 모션 명령어에 변화를 줄 수 있습니다.
각각의 키워드는 순서와 관계없이 사용할 수 있으며, 일부만 사용해도 됩니다.

1) <경유점1>과 <경유점2>는 모든 위치형 변수의 사용이 가능합니다. CMOV 명령어를 사용하게 되는 경우 입력되는 위치형 변수를 베이스 좌표계 위치형 변수로 변환하여 이동하게 됩니다.
2) VEL은 축 이동속도를 의미하며, 천분율(%)로 나타냅니다. 아래 수식과 같이 이동속도를 계산합니다. 정격속도 및 초기속도는 MOTION 파라미터에서 설정됩니다.
   - 축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001
3) ACC는 축 가속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 가속시간을 계산합니다. 정격가속시간은 MOTION 파라미터에서 설정됩니다.
   - 가속시간 = 정격가속시간 X 백분율 값 X 0.01
4) DEC는 축 감속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 감속시간을 계산합니다. 정격감속시간은 MOTION 파라미터에서 설정됩니다.
   - 감속시간 = 정격감속시간 X 백분율 값 X 0.01
5) T는 툴 번호를, U는 사용자 좌표계 번호를 의미합니다. 모션 수행 시 사용할 툴을 선택하거나 좌표계를 선택하는데 사용됩니다.

Arguments

<경유점1> <경유점2>	Data type: <모든 위치형 변수>
VEL	Velocity Data type: <정수형 변수/상수> VEL의 범위는 1 ~ 1,000%까지 입니다.
ACC	Acceleration Data type: <정수형 변수/상수> ACC의 범위는 1 ~ 400%까지 입니다.
DEC	Deceleration Data type: <정수형 변수/상수> DEC의 범위는 1 ~ 400%까지 입니다.
T	Tool number Data type: <정수형 변수/상수> T의 범위는 0 ~ 15입니다. TOOL 명령어와 같은 역할을 합니다.
U	User number Data type: <정수형 변수/상수> U의 범위는 0 ~ 15입니다. USER 명령어와 같은 역할을 합니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    JMOV GP1                            // GP1까지 JMOV
    CMOV GP2 GP3                        // GP2, GP3을 경유하는 원형 보간
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지JMOV
    CMOV GP1 GP2, DEC = 50, VEL = 100, ACC = 50, T = 1
                                        // GP1, GP2를 경유하는 원형 보간
EOP                                     // 메인 함수 종료
```

Related terms

AMOV, VEL, ACC, DEC, TOOL, USER

1) 로봇의 위치와 자세에 따라 동작할 수 없는 경우가 존재합니다. 이때, “[1219] Range Over error”, 또는 “[1415] ik position error” 등의 알람이 발생할 수 있습니다. 로봇 동작 조건 명령어를 함께 사용하시는 경우 주의하시기 바랍니다.
2) 작업 포인트 티칭에 유의하여 주십시오.

6.6 AMOV (원호 보간 이동)

Usage

현재 위치에서 경유점 1, 2를 원호를 그리며 이동합니다.

Syntax

AMOV <경유점1> <경유점2> [, VEL = <값>, ACC = <값>, DEC = <값>, T = <값>, U = <값>]

Description

동작 목표위치에 도달하는 경로를 원호를 그리며 보간합니다. 원호 보간 동작을 할 때 총 3개의 위치 좌표(시작, 경유점1, 경유점2)가 필요합니다.
AMOV를 사용할 때에는 Figure 4-7과 같이 JMOV를 사용하여 이동한 후 AMOV 를 사용할 것을 권장합니다.

Figure 4-7 원호 보간 예시

명령어 사용 시 VEL, ACC, DEC, T, U의 부가항목을 추가로 입력할 수 있습니다.
VEL과 같은 키워드를 지정하고 값을 대입하면 모션 명령어에 변화를 줄 수 있습니다.
각각의 키워드는 순서와 관계없이 사용할 수 있으며, 일부만 사용해도 됩니다.

1) <경유점1>과 <경유점2>는 모든 위치형 변수의 사용이 가능합니다. AMOV 명령어를 사용하게 되는 경우 입력되는 위치형 변수를 베이스 좌표계 위치형 변수로 변환하여 이동합니다.
2) VEL은 축 이동속도를 의미하며, 천분율(%)로 나타냅니다. 아래 수식과 같이 이동속도를 계산합니다. 정격속도 및 초기속도는 MOTION 파라미터에서 설정됩니다.
   - 축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001
3) ACC는 축 가속시간을 의미하며, 백분율(%)로 나타냅니다.
   아래 수식과 같이 가속시간을 계산합니다. 정격가속시간은 MOTION 파라미터에서 설정됩니다.
   - 가속시간 = 정격가속시간 X 백분율 값 X 0.01
4) DEC는 축 감속시간을 의미하며, 백분율(%)로 나타냅니다. 아래 수식과 같이 감속시간을 계산합니다. 정격감속시간은 MOTION 파라미터에서 설정됩니다.
   - 감속시간 = 정격감속시간 X 백분율 값 X 0.01
5) T는 툴 번호를, U는 사용자 좌표계 번호를 의미합니다. 모션 수행 시 사용할 툴을 선택하거나 좌표계를 선택하는데 사용됩니다.

Arguments

<경유점1> <경유점2>	Data type: <모든 위치형 변수>
VEL	Velocity Data type: <정수형 변수/상수> VEL의 범위는 1 ~ 1,000%까지 입니다.
ACC	Acceleration Data type: <정수형 변수/상수> ACC의 범위는 1 ~ 400%까지 입니다.
DEC	Deceleration Data type: <정수형 변수/상수> DEC의 범위는 1 ~ 400%까지 입니다.
T	Tool number Data type: <정수형 변수/상수> T의 범위는 0 ~ 15입니다. TOOL 명령어와 같은 역할을 합니다.
U	User number Data type: <정수형 변수/상수> U의 범위는 0 ~ 15입니다. USER 명령어와 같은 역할을 합니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    JMOV GP1                            // GP1까지 JMOV
    AMOV GP2 GP3                        // GP2를 경유하며 GP3까지 원호보간
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지JMOV
    AMOV GP1 GP2, DEC = 50, VEL = 150, ACC =  // GP1을 경유하며 GP2까지 원호보간
    50, T = 1
EOP                                     // 메인 함수 종료
```

Related terms

CMOV, VEL, ACC, DEC, TOOL, USER

1) 로봇의 위치와 자세에 따라 동작할 수 없는 경우가 존재합니다. 이때, “[1219] Range Over error”, 또는 “[1415] ik position error” 등의 알람이 발생할 수 있습니다. 로봇 동작 조건 명령어를 함께 사용하시는 경우 주의하시기 바랍니다.
2) 작업 포인트 티칭에 유의하여 주십시오.

7. 로봇 동작 조건

7.1 WITH … ENDWT (동시처리 명령어)

Usage

로봇 동작과 함께 블록 내 수행문을 동시처리합니다.

Syntax

```
WITH
    …
ENDWT
```

Description

WITH문 블록 내에 최초 이동 명령어를 수행하면서 반복문 및 조건식 등을 사용하여 이동이 완료되기 전 미리 필요한 작업을 병렬로 진행할 수 있습니다.

Examples

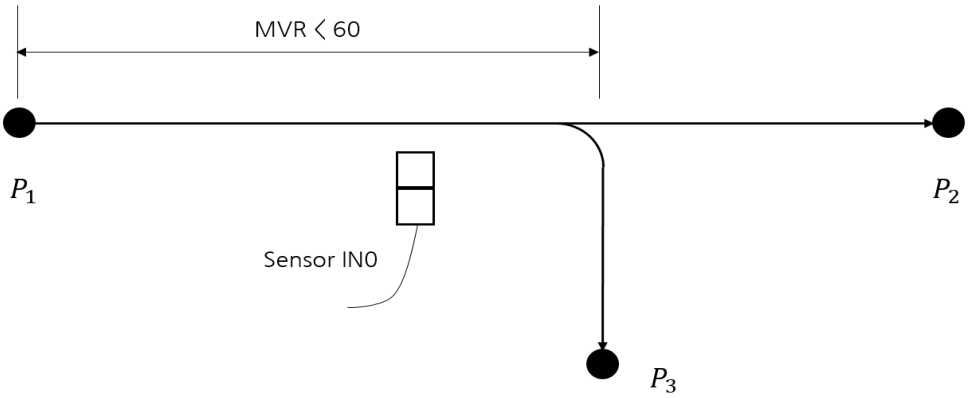


Figure 4-8 WITH문 사용 예시

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    JMOV GP1                            // GP1까지 JMOV
    WITH                                // WITH문 사용, 동시 처리
        JMOV GP2                        // GP2 까지 JMOV
        WHILE MVR < 60                  // MVR이 60 미만인 경우 반복 수행
            IF IN0 == 1 THEN            // IN0가 1인 경우
                STOP                    // 로봇 정지
                GOTO AA                 // 분기점 AA로 이동
            ENDIF                       // IF문 종료
        ENDWL                           // WHILE 문 종료
    ENDWT                               // WITH문 종료

LABL AA                                 // 분기점 AA 선언
    JMOV GP3                            // GP3까지 JMOV
EOP                                     // 메인 함수 종료
```

Related terms

MVR

1) WITH문 실행 중 분기 명령어로 WITH … ENDWT 문을 탈출 시, 로봇을 정지하고자 하는 경우 분기 명령어 이전에 로봇을 멈추어 주십시오.

7.2 VEL (축 이동속도 설정)

Usage

축 이동속도의 천분율(%)을 설정합니다.

Syntax

VEL <속도 비율>

Description

로봇 이동속도에 대한 명령으로, 전체 프로그램 이동 명령어의 속도를 변경합니다.
정격속도 및 초기속도는 MOTION 파라미터에서 설정됩니다.

축 이동속도 = 초기속도 X 정격속도 X 천분율 값 X 0.001

이동 명령어에서 속도를 변경하는 경우 명령어에 삽입된 속도 값을 우선합니다.

1) <속도 비율>은 축 이동속도의 비율을 의미하며, 천분율(%) 단위로 설정할 수 있습니다.

Arguments

<속도 비율>	Data type: <정수형 변수/상수> <속도 값>의 범위는 1 ~ 1,000%까지 입니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    VEL 1000            // 이동 속도: 초기속도 X 정격속도 X 0.001 X 1,000
    JMOV GP0            // GP0까지 JMOV
    VEL 200             // 이동 속도: 초기속도 X 정격속도 X 0.001 X 200
    JMOV GP1            // GP1까지 JMOV
EOP                     // 메인 함수 종료
```

Related terms

JMOV, LMOV, CMOV, AMOV, FOS, ACC, DEC

1) 기구 허용 최대 RPM을 초과하여 사용하는 경우 소음 및 파손의 위험이 있습니다. 반드시 기구부에 부착된 라벨을 확인한 후 사용하시기 바랍니다.
2) JOB 프로그램 내에서만 적용되고 파라미터의 값의 변경은 발생하지 않습니다.

7.3 ACC (축 가속시간 설정)

Usage

가속 시간의 백분율(%)을 설정합니다.

Syntax

ACC <가속시간 비율>

Description

로봇 가속시간에 대한 명령으로, 전체 프로그램 이동 명령어의 가속시간을 변경합니다. 정격가속시간은 MOTION 파라미터에서 설정되며, 백분율 값을 조정하여 가속시간을 변경합니다.

가속시간 = 정격가속시간 X 백분율 값 X 0.01

이동 명령어에서 가속시간을 변경하는 경우 명령어에 삽입된 가속시간을 우선합니다.

1) <가속시간 비율>은 축 가속시간의 비율을 의미하며, 백분율(%) 단위로 설정할 수 있습니다

Arguments

<가속시간 비율>	Data type: <정수형 변수/상수> <가속시간 비율>의 범위는 1 ~ 400%까지 입니다.

Examples

- 파라미터에서 정격가속시간이 0.4초로 설정되어 있는 경우의 예

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    ACC 200             // 가속시간: 정격가속시간(0.4s) X 0.01 X 200 = 0.8s
    JMOV GP0            // GP0까지 JMOV
    ACC 50              // 가속시간: 정격가속시간(0.4s) X 0.01 X 50 = 0.2s
    JMOV GP1            // GP1까지 JMOV
EOP                     // 메인 함수 종료
```

Related terms

JMOV, LMOV, CMOV, AMOV, FOS, VEL, DEC

1) 가속시간 비율이 100%미만으로 설정되는 경우 기계부의 소음 및 진동이 발생할 수 있으니 주의하여 사용하십시오.
2) JOB 프로그램 내에서만 적용되고 파라미터의 값의 변경은 발생하지 않습니다.

7.4 DEC (축 감속시간 설정)

Usage

감속 시간의 백분율(%)을 설정합니다.

Syntax

DEC <감속시간 비율>

Description

로봇 감속시간에 대한 명령으로, 전체 프로그램 이동 명령어의 감속시간을 변경합니다. 정격감속시간은 MOTION 파라미터에서 설정되며, 백분율 값을 조정하여 감속시간을 변경합니다.

감속시간 = 정격감속시간 X 백분율 값 X 0.01

이동 명령어에서 감속시간을 변경하는 경우 명령어에 삽입된 감속시간을 우선합니다.

1) <감속시간 비율>은 축 감속시간의 비율을 의미하며, 백분율(%) 단위로 설정할 수 있습니다

Arguments

<감속시간 비율>	Data type: <정수형 변수/상수> <감속시간 비율>의 범위는 1 ~ 400%까지 입니다.

Examples

- 파라미터에서 정격감속시간이 0.6초로 설정되어 있는 경우의 예

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    DEC 200             // 가속시간: 정격가속시간(0.6s) X 0.01 X 200 = 1.2s
    JMOV GP0            // GP0까지 JMOV
    DEC 50              // 가속시간: 정격가속시간(0.6s) X 0.01 X 50 = 0.3s
    JMOV GP1            // GP1까지 JMOV
EOP                     // 메인 함수 종료
```

Related terms

JMOV, LMOV, CMOV, AMOV, FOS, VEL, ACC

1) 감속시간 비율이 100%미만으로 설정되는 경우 기계부의 소음 및 진동이 발생할 수 있으니 주의하여 사용하십시오.
2) JOB 프로그램 내에서만 적용되고 파라미터의 값의 변경은 발생하지 않습니다.

7.5 FOS (Factor of smooth path 설정)

Usage

축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적을 변경합니다.

Syntax

FOS <거리비율>

Description

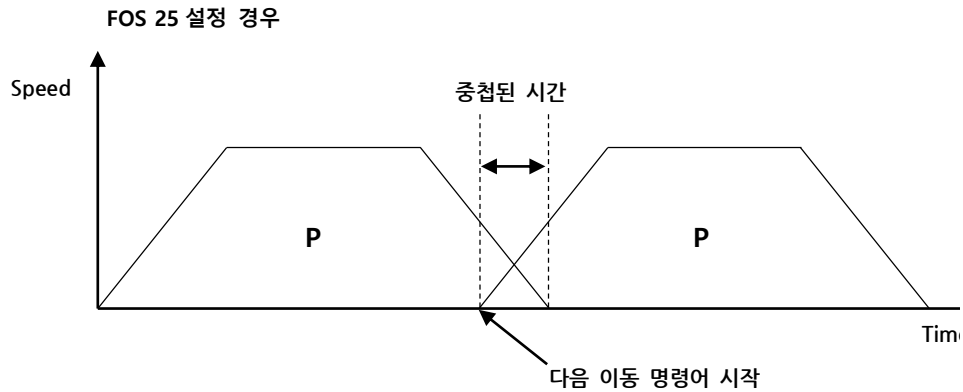
FOS 명령어는 도착 위치에 도달 전 다음 위치로 이동하는 명령어 입니다.
<거리비율>은 진행 중인 이동 모션의 감속시간 내 거리 비율을 설정 하여 다음 이동 명령어를 수행합니다. 좌표계가 같은 이동 명령어는 연속 이동 명령어를 수행 할 수 있지만 다른 경우 먼저 수행한 명령어가 완료된 후 수행 됩니다. 명령어 별 동작 여부 확인은 Table 4-1 참고 바랍니다.
예를 들어 JMOV와 JMOV 또는 LMOV, AMOV, CMOV 간 이동 명령어는 연속 이동 할 수 있습니다.
하지만 JMOV와 LMOV 또는 AMOV 또는 CMOV 명령어 사용 시 JMOV 이동 명령어 수행이 완료 후 다음 이동 명령어를 수행 합니다.
같은 이동 거리에 이동 속도가 다른 경우 연결 구간이 변경되어 이동 궤적이 달라집니다. 속도에 따라 연결 구간은 길이에 비례합니다.

현재 이동명령어 \ 다음 이동 명령어	JMOV	LMOV	AMOV	CMOV
JMOV	연속 이동 가능	연속 이동 불가능	연속 이동 불가능	연속 이동 불가능
LMOV	연속 이동 불가능	연속 이동 가능	연속 이동 가능	연속 이동 가능
AMOV	연속 이동 불가능	연속 이동 가능	연속 이동 가능	연속 이동 가능
CMOV	연속 이동 불가능	연속 이동 가능	연속 이동 가능	연속 이동 가능

Table 4-1 FOS동작 가능 명령어 확인

※ FOS 명령어를 사용시 INPOS 신호는 출력되지 않습니다.

1) <가감속 시간 비율>은 현재 이동 명령어의 감속 시간의 비율(%) 설정 됩니다.
예) FOS 25 설정 시 감속 시간의 50% 시점에 다음 이동 명령어 시작.



예) FOS 50 설정 시 감속 시간의 0% 시점에 다음 이동 명령어 시작

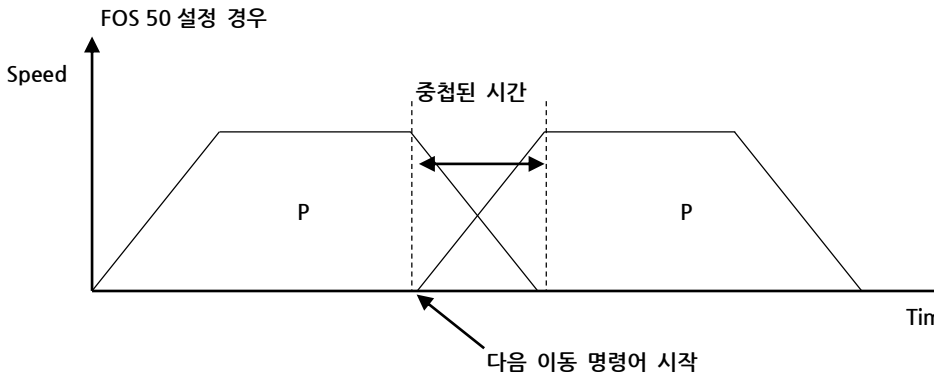


Figure 4-9 FOS 비율

Arguments

〈거리비율〉 Data type: 〈정수형 변수/상수〉
〈감속시간 비율〉의 범위는 0 ~ 50%까지 입니다.

Examples

Example 1

- P1 도착 전 FOS 20%

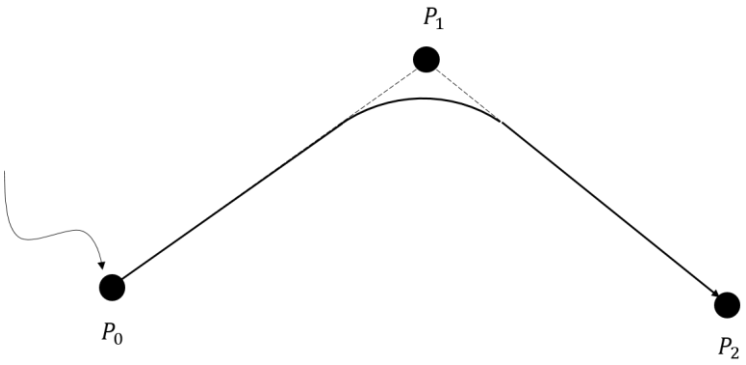


Figure 4-10 FOS 사용 기본 예

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    FOS 20              // FOS 20% 설정
    JMOV GP1            // GP1까지 JMOV
    FOS 0               // FOS 0% 설정
    JMOV GP2            // GP2까지 JMOV
EOP                     // 메인 함수 종료
```

Example 2 • GP1에서 GP2 이동 중 OUT0 신호 출력

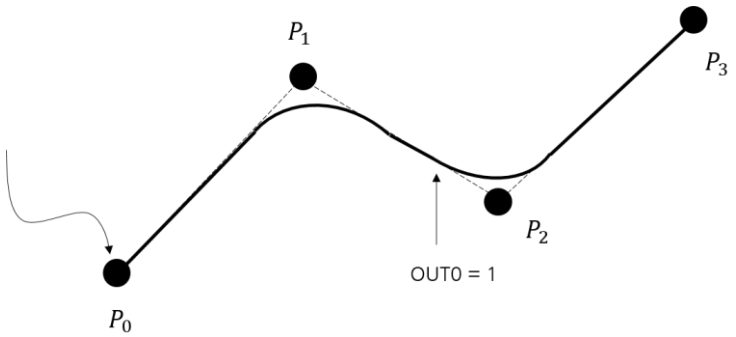


Figure 4-11 이동 중 신호 출력 예

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    FOS 10              // FOS 10% 설정
    LMOV GP1            // GP1까지 LMOV
    LMOV GP2            // GP2까지 LMOV
    OUT0 = 1            // OUT0에 1값 출력
    FOS 0               // FOS 0% 설정
    JMOV GP3            // GP3까지 JMOV
EOP                     // 메인 함수 종료
```

Example 3 • P0에서 P1 이동 중 IN0 신호 확인 후 경로 선택

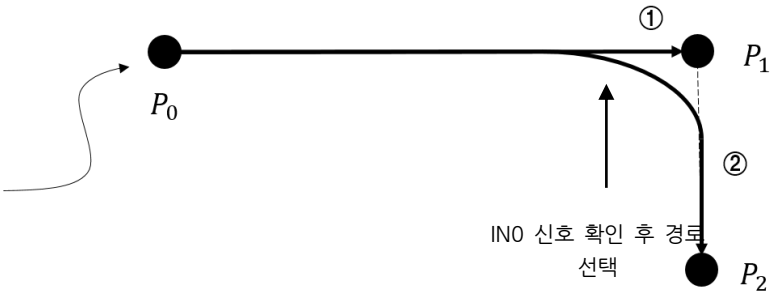


Figure 4-12 이동 중 신호 입력 확인 예

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    FOS 10              // FOS 10% 설정
    JMOV GP1            // GP1까지 JMOV
    IF IN0 == 1 THEN    // IN0가 1인 경우
        JMOV GP2        // GP2까지 JMOV
    ENDIF               // IF문 종료
EOP                     // 메인 함수 종료
```

Related terms

JMOV, LMOV, CMOV, AMOV, VEL, ACC, DEC

1) 이동 명령어 사이에 지연 명령어(DLAY), 계산식 및 시퀀스 연산이 많은 경우 이동 명령어 연결 구간에서 속도가 낮아질 수 있습니다.
2) 현재 이동 명령어와 다음 이동 명령어의 가감속 시간이 다른 경우 연결 구간에서 속도가 낮아질 수 있습니다.
3) FOS 명령어를 사용시 동일 이동 거리에도 이동 속도가 변경되면 연결 구간의 이동 궤적이 변경됩니다.

7.6 DLAY (지연시간 설정)

Usage

지정된 시간 동안 프로그램의 처리를 일시 중단합니다.

Syntax

DLAY <지연시간>

Description

입력한 <지연시간>을 지날 때까지 프로그램의 처리가 멈춥니다.

1) <지연시간>의 단위는 1ms 입니다.

Arguments

<지연시간>	Data type: <정수형 변수/상수>

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    DLAY 1000           // 1초 지연
    JMOV GP1            // GP1까지 JMOV
    DLAY 2000           // 2초 지연
EOP                     // 메인 함수 종료
```

7.7 OFFS (오프셋 지정)

Usage

목표 지점을 지정 된 값만큼 가감해 이동시킵니다.

Syntax

OFFS <편차값>

Description

OFFS 명령어가 수행된 후, 이동 명령어를 사용하면 모두 오프셋 값만큼 증감한 위치로 이동합니다.
Figure 4-13에서 보듯이 A에서 B까지 점선 이동하는 경로가 본래 경로라면, OFFS 명령어를 사용하여 A에서 B까지 실선 이동하는 경로로 변경할 수 있습니다.

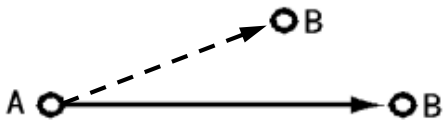


Figure 4-13 OFFS 명령어 예시

OFFS 명령어는 JOB 프로그램 실행 시에만 유효하며, 전체 프로그램에 공통적으로 사용됩니다. 따라서, JCALL, INCLUDE 명령어를 통해 사용한 프로그램에도 적용됩니다.

1) <편차값>은 위치형 상수를 제외한 모든 위치형 변수로 사용이 가능합니다.
2) <편차값>으로 입력되는 위치형 변수의 좌표계를 참조하여 이동 명령어의 최종 목표 위치를 갱신합니다.

Arguments

<편차값>	Data type: <모든 위치형 변수>

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    POS PDATA                 // 위치형 변수 선언
    PDATA = <0,0,0,0,30,30>   // 위치형 변수 초기화
    SVON                      // 서보 ON
    OFFS PDATA                // OFFSET 설정
    JMOV GP0                  // GP0로 JMOV
EOP                           // 메인 함수 종료
```

Example 2

```
MAIN                          // 메인 함수 선언
    OFFS GP1                  // OFFSET 설정
    SVON                      // 서보 ON
    JMOV GP0                  // GP0로 JMOV
    OFFS GP2                  // OFFSET 재 설정
    JMOV GP0                  // GP0로 JMOV
EOP                           // 메인 함수 종료
```

Example 3

```
MAIN                                        // 메인 함수 선언
    XPOS XP                                 // 베이스 좌표계 위치형 변수 선언
    UPOS UP                                 // 사용자 좌표계 위치형 변수 선언
    XP = <10,10,10,10,10,10,ARM = 0,T = 1>  // 베이스 좌표계 위치형 변수 초기화
    UP = <5,10,15,0,0,0,ARM = 0,T = 1,U = 1>// 사용자 좌표계 위치형 변수 초기화
    SVON                                    // 서보 ON
    OFFS XP                                 // 베이스 좌표계 위치형 변수로 OFFSET
    LMOV GP0                                // GP0까지 LMOV
    OFFS UP                                 // 사용자 좌표계 위치형 변수로 OFFSET
    JMOV GP1                                // GP1까지 JMOV
EOP                                         // 메인 함수 종료
```

Related terms

OFFSCLR

1) 티칭 값이 S/W 이동 제한 범위를 벗어나지 않더라도 JOB 프로그램 실행 시, OFFS 명령어를 사용하는 경우 “[1219] Range Over error” 가 발생할 수 있습니다.
2) OFFS을 해제하려면 OFFSCLR 명령어를 사용하십시오.

7.8 OFFSCLR (오프셋 해제)

Usage

설정된 오프셋 값을 해지합니다.

Syntax

OFFSCLR

Description

사용자가 OFFS 명령어를 사용하여 설정한 오프셋 값을 해지하고, 원래 위치 값을 참조하도록 변경합니다.
OFFS 명령어와 마찬가지로 전체 프로그램에 적용됩니다.

Examples

```
MAIN                    // 메인 함수 선언
    OFFS GP1            // OFFSET 설정
    SVON                // 서보 ON
    JMOV GP0            // GP0로 JMOV
    OFFSCLR             // OFFSET 해제
    JMOV GP0            // GP0로 JMOV
EOP                     // 메인 함수 종료
```

Related terms

OFFS

7.9 LIMT (축 이동범위 제한)

Usage

각 축의 이동범위를 제한합니다.

Syntax

LIMT <(-)이동범위> <(+)이동범위>

Description

JOB 프로그램 내에서의 S/W 이동 제한 범위를 설정합니다.
LIMT 명령어는 JOB 프로그램 실행 시에만 유효하며, 전체 프로그램에 공통적으로 사용됩니다. 따라서 JCALL, INCLUDE 명령어를 통해 사용한 프로그램에도 적용됩니다.

1) (-)이동범위는 (-)방향으로 이동할 수 있는 최소 위치를 의미합니다.
2) (+)이동범위는 (+)방향으로 이동할 수 있는 최대 위치를 의미합니다.
3) (-) 및 (+)이동범위를 입력할 때에는 조인트 좌표계 위치형 변수를 사용해야 하며, 위치형 상수는 사용할 수 없습니다.

Arguments

<(-)이동범위>	Data type: <조인트 좌표계 위치형 변수>
<(+)이동범위>	Data type: <조인트 좌표계 위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    LIMT GP0 GP1                        // S/W 이동 제한 범위 변경
    SVON                                // 서보 ON
    JMOV GP2                            // GP2까지 JMOV
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    POS AP, BP                          // 조인트 좌표계 위치형 변수 선언
    AP = <-100,-100,-100,-100,-100,-100>  // 조인트 좌표계 위치형 변수 초기화
    BP = <100,100,100,100,100,100>
    LIMT AP BP                          // S/W 이동 제한 범위 변경
    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지 JMOV
EOP                                     // 메인 함수 종료
```

1) 티칭 값이 S/W 이동 제한 범위를 벗어나지 않더라도 JOB 프로그램 실행 시, LIMT 명령어를 사용하는 경우 “[1219] Range Over error” 가 발생할 수 있습니다.
2) (+)이동범위가 (-)이동범위보다 작은 값이 입력되지 않도록 주의해 주십시오.

7.10 TRQ (충돌 감지 명령어)

Usage

토크 리밋 값을 설정하여 충돌 시 알람을 발생하는 명령어입니다.

Syntax

TRQ <로봇 번호> <축 번호> <토크 리밋 값>

Description

로봇 동작 중 지정된 각 축의 토크리밋을 설정하여 그 이상의 토크가 파라미터 TOL 시간이상 동작 시 “[1434] Over Trq error” 알람을 발생합니다.

1) <로봇 번호>를 입력하여 로봇을 지정합니다.
2) <축 번호>를 입력하여 토크 값을 감시할 축을 지정합니다.
3) <토크 리밋 값>을 입력하여 알람을 발생시킬 제한 값을 설정합니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF 까지입니다.
<토크 리밋 값>	Data type: <정수형 변수/상수> <토크 리밋 값>의 범위는 1 ~ 300까지입니다.

Examples

```
MAIN                    // 메인 함수 선언
    TRQ 1 1 50          // J1 축의 토크 리밋을 50%로 설정
    TRQ 1 2 100         // J2 축의 토크 리밋을 100%로 설정
    LMOV GP0            // 현재 위치에서 GP0까지 LMOV
    LMOV GP1            // 현재 위치에서 GP1까지 LMOV
    TRQ 1 1 300         // J1축의 토크 리밋을 300%로 설정
    TRQ 1 2 300         // J2축의 토크 리밋을 300%로 설정
EOP                     // 메인 함수 종료
```

1) JOB START시 TRQ 값은 300으로 초기화 됩니다.
2) 서보 OFF 상태에서는 토크 값을 모니터링 하지 않습니다.

7.11 GAIN (게인 그룹 전환)

Usage

게인 그룹을 전환합니다.

Syntax

GAIN <그룹 번호>

Description

모션 실행 시 사용할 게인 그룹을 선택합니다.
게인 그룹에는 위치 루프 게인, 속도 루프 게인, 속도 루프 적분 시정수, 토크 명령 필터 시정수가 포함되어 있습니다.

1) <그룹 번호>를 입력하여 게인 그룹을 선택합니다.

Arguments

<그룹 번호>	Data type: <정수형 변수/상수> <그룹 번호>의 범위는 1 ~ 2까지입니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    GAIN 1              // 게인 그룹1 적용
    JMOV GP0            // 게인 그룹1이 적용된 상태로 GP0으로 JMOV
    GAIN 2              // 게인 그룹2 적용
    JMOV GP1            // 게인 그룹2가 적용된 상태로 GP1으로 JMOV
EOP                     // 메인 함수 종료
```

1) 게인 전환 시 대기시간 및 전환시간이 존재하므로 유의하시기 바랍니다.

7.12 IPE (In-position 허용 시간 설정)

Usage

목표 위치 도달(In-Position)까지의 허용 시간 최대 값을 설정합니다.

Syntax

IPE <허용 시간>

Description

목표 위치 값 근처에 도달하였을 때부터 목표 위치로 수렴하기까지 허용되는 시간의 최대 값을 설정합니다.
허용 시간 값 내에서 목표 위치 오차 허용 범위(IPA)내에 수렴되어야 합니다.

1) <허용 시간>의 단위는 1ms 입니다.
2) FOS 동작 시, Inposition Check는 하지 않습니다.

Arguments

<허용 시간>	Data type: <정수형 변수/상수> <허용 시간>의 범위는 0 ~ 5,000(ms)까지입니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    IPE 100             // In-Position 허용 시간 최대 값 설정
    JMOV GP0            // GP0 까지 JMOV
    JMOV GP1            // GP1 까지 JMOV
EOP                     // 메인 함수 종료
```

1) 허용 시간 이내에 목표 위치까지 도달하지 못하는 경우 [1426] Inposition error 알람이 발생하므로 값 설정에 유의하시기 바랍니다.
2) IPE를 0으로 설정하는 경우 JOB 구동 중 In-Position 여부를 확인하지 않습니다.

8. 입출력 제어

8.1 IN (디지털 비트 입력)

Usage

1비트 단위로 지정된 비트번호의 값을 확인 합니다.

Syntax

IN[(]<비트번호>[)] = <0/1>
IN(<비트번호>[, <대기시간>]) = <0/1>
<변수> = IN[(]<비트번호>[)]

Description

IN 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 비트에 0 또는 1 값을 대입하는 경우로 해당 비트의 입력 신호가 대입한 값과 같아질 때까지 대기하거나, 사용자가 지정한 시간까지만 대기합니다.
두 번째 방법은 지정된 비트번호의 신호 값을 읽어 변수에 저장합니다.

1) <비트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 IN과 <번호>간에는 공란이 있으면 안됩니다.
2) <비트번호>의 범위는 0 ~ 19까지이며, 확장 보드 사용 시 최대 83까지 입력 가능합니다.
3) <대기시간>의 설정은 괄호를 사용할 때만 가능하며, 설정 범위는 100 ~ 99,999 msec까지 입니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<0/1> Data type: <정수형 변수/상수>
0(OFF) 또는 1(ON)만 입력할 수 있습니다.

<비트번호> Data type: <정수형 변수/상수>
- 일반 입력: IN0 ~ IN19
- 확장 1 입력: IN20 ~ IN51
- 확장 2 입력: IN52 ~ IN83

<대기시간> Data type: <정수형 변수/상수>
설정 범위는 100 ~ 99,999 msec 입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    IN0 = 1             // IN0번 입력 신호가 1일 때까지 대기
    JMOV GP1            // GP1까지 JMOV
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    WHILE 1             // 반복문 시작
        IN(0,1000) = 1  // IN0번 입력 신호가 1일 때 까지 1초 대기
        IF IN0 == 0 THEN // IN0 신호 값이 0 일 때
            JMOV GP0    // GP0로 JMOV
        ELSE            // 그 외의 경우
            JMOV GP1    // GP1로 JMOV
        ENDIF           // 조건문 종료
    ENDWL               // 반복문 종료
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    INT STATE, NUM      // 정수형 변수 선언

    NUM = 0             // 정수형 변수 초기화
    STATE = IN0         // IN0 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = IN(0)       // IN0 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
    STATE = IN(NUM)     // NUM 변수 값 참조하여 IN 신호 저장
    PRINT(2,STATE)      // 모니터링 로그 2번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Related terms

BIN, PIN

1) 신호 값 대기 시 사용에 주의하십시오.
2) 대기시간 입력 시 설정한 대기시간이 지나면 다음 구문을 실행하므로 프로그램 작성에 주의하십시오.

8.2 OUT (디지털 비트 출력)

Usage

지정된 비트번호에 지정된 값을 출력합니다.

Syntax

OUT[(]<비트번호>[)] = <0/1>
OUT[(]<비트번호>[)] = <0/1> <펄스 폭>
OUT[(]<비트번호>[)] = <0/1> <펄스 폭> <펄스 주기>
<변수> = OUT[(]<비트번호>[)]

Description

OUT 명령어는 4가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 비트번호에 0 또는 1 값을 출력합니다.
두 번째 방법은 지정된 비트번호에 임펄스 신호를 내보냅니다.
세 번째 방법은 지정된 비트 번호에 펄스 주기 파형을 내보냅니다.
마지막으로 지정된 비트번호로 출력한 신호 값을 읽어 변수에 저장합니다.

1) <비트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 OUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <비트번호>의 범위는 0 ~ 19까지이며, 확장 보드 사용 시 최대 83까지 입력 가능합니다.
3) <펄스 폭>을 추가로 입력하는 경우 입력한 시간만큼 임펄스 신호를 내보내고 이전 값으로 돌아갑니다.
4) <펄스 주기>가 입력되면 입력한 시간을 주기로 하는 펄스 주기 파형을 내보냅니다. <펄스 주기>는 <펄스 폭>과 항상 함께 입력합니다.

※ <펄스 주기>는 <펄스 폭>보다 큰 값을 입력합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<0/1> Data type: <정수형 변수/상수>
0(OFF) 또는 1(ON)만 입력할 수 있습니다.

<비트번호> Data type: <정수형 변수/상수>
- 일반 입력: IN0 ~ IN19
- 확장 1 입력: IN20 ~ IN51
- 확장 2 입력: IN52 ~ IN83

<펄스 폭> Data type: <정수형 변수/상수>
단위는 1ms 이나 2ms 이하 값은 무시되므로 2의 배수로 입력해 주십시오.

<펄스 주기> Data type: <정수형 변수/상수>
단위는 1ms 이나 2ms 이하 값은 무시되므로 2의 배수로 입력해 주십시오.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    OUT(0) = 1          // 0번 출력 신호에 1값 출력
    DLAY 1000           // 1초 대기
    OUT0 = 0            // 0번 출력 신호에 0 값 출력
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언

    STATE = OUT1        // OUT1 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = OUT(1)      // OUT1 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
    STATE = OUT(I0)     // 전역 변수 I0번 값 참조하여 OUT 신호 저장
    PRINT(2,STATE)      // 모니터링 로그 2번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    OUT0 = 0            // 0번 출력 신호에 0 값 출력
    OUT(0) = 1 100      // 0번 출력 신호에 0.1초 길이의 임펄스 출력
EOP                     // 메인 함수 종료
```

Example 4

```
MAIN                    // 메인 함수 선언
    OUT1 = 0            // 1번 출력 신호에 0 값 출력
    OUT1 = 1 1000 5000  // 1번 출력 신호에 5초 주기에 1초 펄스 폭을 갖는 파형 출력
EOP                     // 메인 함수 종료
```

Related terms

BOUT, POUT

1) OUT명령어를 사용하여 임펄스 신호 또는 펄스 주기 파형을 생성하는 경우 <펄스 폭> 및 <펄스 주기> 값의 입력 단위는 1ms 입니다. 단, 2ms 이하 값은 무시하므로 입력 값은 2의 배수로 입력해 주십시오.
   - OUT0 = 1 1 → 펄스 출력 없음, 출력 값 계속 유지
   - OUT0 = 1 15 → 14ms 길이의 임펄스 출력
   - OUT0 = 1 10 15 → 14ms주기에 10ms의 펄스 폭을 갖는 파형 출력
   - OUT0 = 1 15 29 → 28ms 주기에 14ms의 펄스 폭을 갖는 파형 출력

8.3 BIN (디지털 바이트 입력)

Usage

바이트 단위로 지정된 바이트영역의 값을 확인 합니다.

Syntax

BIN[(]<바이트번호>[)] = <값>
BIN(<바이트번호>[, <대기시간>]) = <값>
<변수> = BIN[(]<바이트번호>[)]

Description

BIN 명령어는 3가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 바이트번호 영역에 0 ~ 255값을 대입하는 경우로 해당 영역의 입력 신호가 대입한 값과 같아질 때까지 대기합니다.
두 번째 방법은 사용자가 지정한 시간까지만 대기합니다.
마지막으로 지정된 바이트번호 영역의 신호 값을 읽어 변수에 저장합니다.

1) <바이트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 BIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <바이트번호>의 범위는 0 ~ 2까지이며, 확장 보드 사용 시 최대 10까지 입력 가능합니다.
3) <대기시간>의 설정은 괄호를 사용할 때만 가능하며, 설정 범위는 100 ~ 99,999 msec까지 입니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 양의 정수 값(0 ~ 255)이 반환됩니다.
데이터 값은 우측에서 좌측으로 증가됩니다.

비트 / 정수 값	F E D C	B A 9 8	7 6 5 4	3 2 1 0
10	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 0
193	0 0 0 0	0 0 0 0	1 1 0 0	0 0 0 1
0HF4	0 0 0 0	0 0 0 0	1 1 1 1	0 1 0 0
0B11110011	0 0 0 0	0 0 0 0	1 1 1 1	0 0 1 1

Arguments

<값> Data type: <정수형 변수/상수>
8비트 양의 정수 값(0 ~ 255) 만 입력할 수 있습니다.

<바이트번호> Data type: <정수형 변수/상수>
- 일반 입력: BIN0 ~ 2

(BIN0: IN0 ~ IN7, BIN1: IN8 ~ 15, BIN2: IN16 ~ IN19)

- 확장 1 입력: BIN0 ~ 6

(BIN0: IN0 ~ IN7, BIN1: IN8 ~ 15, BIN2: IN16 ~ IN23, BIN3: IN24 ~ IN31,
BIN4: IN32 ~ IN39, BIN5: IN40 ~ IN47, BIN6: IN48 ~ IN51)

- 확장 2 입력: BIN0 ~ 10

(BIN0: IN0 ~ IN7, BIN1: IN8 ~15, BIN2: IN16 ~ IN23, BIN3: IN24 ~ IN31,
BIN4: IN32 ~ IN39, BIN5: IN40 ~ IN47, BIN6: IN48 ~ IN55,
BIN7: IN56 ~ IN63, BIN8: IN64 ~ IN71, BIN9: IN72 ~ IN79,
BIN10: IN80 ~ IN83)

〈대기시간〉 Data type: 〈정수형 변수/상수〉
설정 범위는 100 ~ 99,999 msec 입니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    SVON                    // 서보 ON
    JMOV GP0                // GP0까지 JMOV
    BIN0 = 255              // BIN0번 입력 값이 255일 때까지 대기
    JMOV GP1                // GP1까지 JMOV
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    SVON                    // 서보 ON
    INT SR                  // STROUT용 정수형 변수 선언
    WHILE 1                 // 반복문 시작
        BIN(0,1000) = 255   // BIN0번 입력 신호가 1일 때 까지 1초 대기
        IF BIN0 == 255 THEN // BIN0 신호 값이 255 일 때
            SR = STROUT(0,"BYTE SIGNAL IN")   // 문자열 전송
        ELSE                // 그 외의 경우
            OUT(0) = 1      // OUT0번에 1 값 출력
        ENDIF               // 조건문 종료
    ENDWL                   // 반복문 종료
EOP                         // 메인 함수 종료
```

Example 3

```
MAIN                        // 메인 함수 선언
    INT STATE, NUM          // 정수형 변수 선언
    NUM = 0                 // 정수형 변수 초기화
    STATE = BIN0            // BIN0 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
    STATE = BIN(NUM) & 0H0F // BIN0의 0 ~ 3번째 값 비트 AND 연산
    PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Related terms

IN, PIN

1) 신호 값 대기 시 사용에 주의하십시오.
2) 대기시간 입력 시 설정한 대기시간이 지나면 다음 구문을 실행하므로 프로그램 작성에 주의하십시오.

8.4 BOUT (디지털 바이트 출력)

Usage

지정된 바이트영역에 지정된 값을 출력합니다.

Syntax

BOUT[(]<바이트번호>[)] = <값>
<변수> = BOUT[(]<바이트번호>[)]

Description

BOUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 값을 출력합니다.
두 번째 방법은 지정된 바이트번호로 출력한 신호 값을 읽어 변수에 저장합니다.

1) <바이트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 BOUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <바이트번호>의 범위는 0 ~ 2까지이며, 확장 보드 사용 시 최대 10까지 입력 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 양의 정수 값(0 ~ 255)이 반환됩니다.
데이터 값은 우측에서 좌측으로 증가됩니다.

비트 / 정수 값	F E D C	B A 9 8	7 6 5 4	3 2 1 0
10	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 0
193	0 0 0 0	0 0 0 0	1 1 0 0	0 0 0 1
0HF4	0 0 0 0	0 0 0 0	1 1 1 1	0 1 0 0
0B11110011	0 0 0 0	0 0 0 0	1 1 1 1	0 0 1 1

Arguments

<값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 8비트로 0 ~ 255 입니다.

<바이트번호> Data type: <정수형 변수/상수>

- 일반 입력: BOUT0 ~ 2

(BOUT0: OUT0 ~ OUT7, BOUT1: OUT8 ~ OUT15, BOUT2: OUT16 ~ OUT19)

- 확장 1 입력: BOUT0 ~ 6

(BOUT0: OUT0 ~ OUT7, BOUT1: OUT8 ~ OUT15, BOUT2: OUT16 ~ OUT23, BOUT3: OUT24 ~ OUT31,BOUT4: OUT32 ~ OUT39, BOUT5: OUT40 ~ OUT47, BOUT6: OUT48 ~ OUT51)

- 확장 2 입력: BOUT0 ~ 10

(BOUT0: OUT0 ~ OUT7, BOUT1: OUT8 ~ OUT15, BOUT2: OUT16 ~ OUT23, BOUT3: OUT24 ~ OUT31,BOUT4: OUT32 ~ OUT39, BOUT5: OUT40 ~ OUT47, BOUT6: OUT48 ~ OUT55, BOUT7: OUT56 ~ OUT63, BOUT8: OUT64 ~ OUT71, BOUT9: OUT72 ~ OUT79, BOUT10: OUT80 ~ OUT83)

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    BOUT(0) = 112       // 0번 바이트 영역에 값 출력
    DLAY 1000           // 1초 대기
    BOUT0 = 0           // 0번 바이트 영역에 0 값 출력
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언

    STATE = BOUT1       // 출력한 BOUT1 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = BOUT(1)     // 출력한 BOUT1 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
    STATE = BOUT(I0)    // 전역 변수 I0번 값 참조하여 BOUT 신호 저장
    PRINT(2,STATE)      // 모니터링 로그 2번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Related terms

OUT, POUT

8.5 PIN (디지털 포트 입력)

Usage

포트 단위로 지정된 포트번호의 값을 확인 합니다.

Syntax

<변수> = PIN[(]<포트번호>[)]

Description

지정된 포트번호의 16비트 데이터 값을 한번에 확인 합니다.

1) <포트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 PIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <포트번호>의 범위는 0 ~ 1까지이며, 확장 보드 사용 시 최대 5까지 입력 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 16비트 정수 값이 반환됩니다. 데이터 값은 우측에서 좌측으로 증가됩니다.

비트 / 정수 값	F E D C	B A 9 8	7 6 5 4	3 2 1 0
31242	0 1 1 1	1 0 1 0	0 0 0 0	1 0 1 0
-63	1 1 1 1	1 1 1 1	1 1 0 0	0 0 0 1
0HF419	1 1 1 1	0 1 0 0	0 0 0 1	1 0 0 1
0B11110011	0 0 0 0	0 0 0 0	1 1 1 1	0 0 1 1

Arguments

<포트번호> Data type: <정수형 변수/상수>

- 일반 입력: PIN0 ~ 1

(PIN0: IN0 ~ IN15, PIN1: IN16 ~ 19)

- 확장 1 입력: PIN0 ~ 3

(PIN0: IN0 ~ IN15, PIN1: IN16 ~ 31, PIN2: IN32 ~ 47, PIN3: IN48 ~ 51)

- 확장 2 입력: PIN0 ~ 5

(PIN0: IN0 ~ IN15, PIN1: IN16 ~ 31, PIN2: IN32 ~ 47, PIN3: IN48 ~ 63, PIN4: 64 ~ 79, PIN5: 80 ~ 83)

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT STATE               // 정수형 변수 선언

    STATE = PIN0            // PIN0 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
    STATE = PIN(0)          // PIN0 신호 값 저장
    PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Example 2

- 비트 연산자 응용 예

```
MAIN                        // 메인 함수 선언
    DEFSTR STR              // 문자열 변수 선언
    INT AA, RES             // 정수형 변수 선언

    AA = PIN0               // PIN0 신호 값 저장
    RES = AA & 0H00F0       // 4 ~ 7번째 값 비트 AND 연산
    STR = BINS(RES)         // 문자열 변수에 이진 문자열 저장
    PRINT(0,STR)            // 모니터링 로그 0번 인덱스에 이진 문자열로 출력
    RES = PIN0 | 0H000F     // 0 ~ 3번째 값 비트 OR 연산
    STR = BINS(RES)         // 문자열 변수에 이진 문자열 저장
    PRINT(1,STR)            // 모니터링 로그 1번 인덱스에 이진 문자열로 출력
    RES = 0B11110000 ^ PIN0 // 4 ~ 7번째 값 비트 XOR 연산
    STR = BINS(RES)         // 문자열 변수에 이진 문자열 저장
    PRINT(2,STR)            // 모니터링 로그 2번 인덱스에 이진 문자열로 출력
EOP                         // 메인 함수 종료
```

Related terms

IN, BIN

8.6 POUT (디지털 포트 출력)

Usage

지정된 포트번호에 지정된 값을 출력합니다.

Syntax

POUT[(] <포트번호>[)] = <값>
<변수> = POUT[(] <포트번호>[)]

Description

POUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 값을 출력합니다.
두 번째 방법은 지정된 포트번호로 출력한 신호 값을 읽어 변수에 저장합니다.
1) <포트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 POUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <포트번호>의 범위는 0 ~ 1까지이며, 확장 보드 사용 시 최대 5까지 입력 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 16비트 정수 값이 반환됩니다. 데이터 값은 우측에서 좌측으로 증가됩니다.

비트 / 정수 값	F E D C	B A 9 8	7 6 5 4	3 2 1 0
65535	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
0HF419	1 1 1 1	0 1 0 0	0 0 0 1	1 0 0 1
0B11110011	0 0 0 0	0 0 0 0	1 1 1 1	0 0 1 1

Arguments

<값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 16비트로 0 ~ 65535 입니다.

<포트번호> Data type: <정수형 변수/상수>
- 일반 입력: POUT0 ~ 1

(POUT0: OUT0 ~ OUT15, POUT1: OUT16 ~ OUT19)
- 확장 1 입력: POUT0 ~ 3

(POUT0: OUT0 ~ OUT15, POUT1: OUT16 ~ OUT31, POUT2: OUT32 ~ OUT47, POUT3: OUT48 ~ 51)
- 확장 2 입력: POUT0 ~ 5

(POUT0: OUT0 ~ OUT15, POUT1: OUT16 ~ OUT31, POUT2: OUT32 ~ OUT47, POUT3: OUT48 ~ OUT63, POUT4: 64 ~ OUT79, POUT5: OUT80 ~ OUT83)

Examples

Example 1
```
MAIN                    // 메인 함수 선언
```

```
        POUT(0) = 6245          // 0번 포트에 값 출력
        DLAY 1000               // 1초 대기
        POUT0 = 0               // 0번 포트에 0 값 출력
    EOP                         // 메인 함수 종료
```

Example 2

```
    MAIN                        // 메인 함수 선언
        INT STATE               // 정수형 변수 선언

        STATE = POUT1           // 출력한 POUT1 신호 값 저장
        PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
        STATE = POUT(1)         // 출력한 POUT1 신호 값 저장
        PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
        STATE = POUT(I0)        // 전역 변수 I0번 값 참조하여 POUT 신호 저장
        PRINT(2,STATE)          // 모니터링 로그 2번 인덱스에 출력
    EOP                         // 메인 함수 종료
```

Related terms

OUT, BOUT

8.7 CIN (필드버스 비트 입력)

Usage

필드버스 카드에서 지정된 비트번호의 값을 확인 합니다.

Syntax

CIN[(] <비트번호>[)] = <0/1>
CIN(<비트번호>[, <대기시간>]) = <0/1>
<변수> = CIN[(]<비트번호>[)]

Description

CIN 명령어는 3가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 비트에 0 또는 1 값을 대입하는 경우로 해당 비트의 입력 신호가 대입한 값과 같아질 때까지 대기합니다.
두 번째 방법은 사용자가 지정한 시간까지만 대기합니다
마지막으로 지정된 비트번호의 신호 값을 읽어 변수에 저장합니다.

1) <비트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <비트번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.
3) <대기시간>의 설정은 괄호를 사용할 때만 가능하며, 설정 범위는 100 ~ 99,999 msec까지 입니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<0/1>	Data type: <정수형 변수/상수> 0(OFF) 또는 1(ON)만 입력할 수 있습니다.
<비트번호>	Data type: <정수형 변수/상수> CC-Link V1: CIN0 ~ CIN127 (리모트 입출력 영역) CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.
<대기시간>	Data type: <정수형 변수/상수> 설정 범위는 100 ~ 99,999 msec 입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV
    CIN0 = 1            // CIN0번 입력 신호가 1일 때까지 대기
    JMOV GP1            // GP1까지 JMOV
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    WHILE 1             // 반복문 시작
        CIN(0,1000) = 1 // CIN0번 입력 신호가 1일 때까지 1초 대기
        IF CIN0 == 0 THEN   // CIN0 신호 값이 0 일 때
            JMOV GP0        // GP0로 JMOV
        ELSE                // 그 외의 경우
            JMOV GP1        // GP1로 JMOV
        ENDIF               // 조건문 종료
    ENDWL               // 반복문 종료
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    INT STATE, NUM      // 정수형 변수 선언

    NUM = 2             // 정수형 변수 초기화
    STATE = CIN0        // CIN0 신호 값 저장
    PRINT(0, STATE)     // 모니터링 로그 0번 인덱스에 출력
    STATE = CIN(1)      // CIN1 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
    STATE = CIN(NUM)    // NUM 변수 값 참조하여 CIN 신호 저장
    PRINT(2,STATE)      // 모니터링 로그 2번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Related terms

CBIN, CWIN, CDIN, CFIN

1) 신호 값 대기 시 사용에 주의하십시오.
2) 대기시간 입력 시 설정한 대기시간이 지나면 다음 구문을 실행하므로 프로그램 작성에 주의하십시오.

8.8 COUT (필드버스 비트 출력)

Usage

지정된 필드버스 카드 비트번호에 지정된 값을 출력합니다.

Syntax

COUT[(]<비트번호>[)] = <0/1>
COUT[(]<비트번호>[)] = <0/1> <펄스 폭>
COUT[(]<비트번호>[)] = <0/1> <펄스 폭> <펄스 주기>
<변수> = COUT[(]<비트번호>[)]

Description

COUT 명령어는 4가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 비트번호에 0 또는 1 값을 출력합니다.
두 번째 방법은 지정된 비트번호에 임펄스 신호를 내보냅니다.
세 번째 방법은 지정된 비트 번호에 펄스 주기 파형을 내보냅니다.
마지막으로 지정된 비트번호로 출력한 신호 값을 읽어 변수에 저장합니다

1) <비트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 COUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <비트번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.
3) <펄스 폭>을 추가로 입력하는 경우 입력한 시간만큼 임펄스 신호를 내보내고 이전 값으로 돌아갑니다.
4) <펄스 주기>가 입력되면 입력한 시간을 주기로 하는 펄스 주기 파형을 내보냅니다. <펄스 주기>는 <펄스 폭>과 항상 함께 입력합니다.

※ <펄스 주기>는 <펄스 폭>보다 큰 값을 입력합니다.
※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<0/1>	Data type: <정수형 변수/상수> 0(OFF) 또는 1(ON)만 입력할 수 있습니다.
<비트번호>	Data type: <정수형 변수/상수> CC-Link V1: COUT0 ~ COUT127 (리모트 입출력 영역) CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.
<펄스 폭>	Data type: <정수형 변수/상수> 단위는 1ms 이나 2ms 이하 값은 무시되므로 2의 배수로 입력해 주십시오.
<펄스 주기>	Data type: <정수형 변수/상수> 단위는 1ms 이나 2ms 이하 값은 무시되므로 2의 배수로 입력해 주십시오.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    COUT(0) = 1         // COUT0번 출력 신호에 1값 출력
    DLAY 1000           // 1초 대기
    COUT0 = 0           // COUT0번 출력 신호에 0값 출력
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언

    STATE = COUT1       // COUT1 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = COUT(1)     // COUT1 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
    STATE = COUT(I0)    // 전역 변수 I0번 값 참조하여 COUT 신호 저장
    PRINT(2,STATE)      // 모니터링 로그 2번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    COUT0 = 0           // COUT0번 출력 신호에 0 값 출력
    COUT(0) = 1 100     // COUT0번 출력 신호에 0.1초 길이의 임펄스 출력
EOP                     // 메인 함수 종료
```

Example 4

```
MAIN                    // 메인 함수 선언
    COUT1 = 0           // COUT1번 출력 신호에 0 값 출력
    COUT1 = 1 1000 5000 // COUT1번 출력 신호에 5초 주기에 1초 펄스 폭의 파형 출력
EOP                     // 메인 함수 종료
```

Related terms

CBOUT, CWOUT, CDOUT, CFOUT

1) COUT명령어를 사용하여 임펄스 신호 또는 펄스 주기 파형을 생성하는 경우 <펄스 폭> 및 <펄스 주기> 값의 입력 단위는 1ms 입니다. 단, 2ms 이하 값은 무시하므로 입력 값은 2의 배수로 입력해 주십시오.
   - COUT0 = 1 1 → 펄스 출력 없음, 출력 값 계속 유지
   - COUT0 = 1 15 → 14ms 길이의 임펄스 출력
   - COUT0 = 1 10 15 → 14ms주기에 10ms의 펄스 폭을 갖는 파형 출력
   - COUT0 = 1 15 29 → 28ms 주기에 14ms의 펄스 폭을 갖는 파형 출력

8.9 CBIN (필드버스 바이트 입력)

Usage

필드버스 카드에서 지정된 바이트번호의 값을 읽어옵니다.

Syntax

CBIN [(]<바이트번호>[)] = <값>
CBIN (<바이트번호>[, <대기시간>]) = <값>
<변수> = CBIN [(]<바이트번호>[)]

Description

CBIN 명령어는 3가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 바이트번호 영역에 0 ~ 255값을 대입하는 경우로 해당 영역의 입력 신호가 대입한 값과 같아질 때까지 대기합니다.
두 번째 방법은 사용자가 지정한 시간까지만 대기합니다.
마지막으로 지정된 바이트번호 영역의 신호 값을 읽어 변수에 저장합니다

1) <바이트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CBIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <바이트번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.
3) <대기시간>의 설정은 괄호를 사용할 때만 가능하며, 설정 범위는 100 ~ 99,999 msec까지 입니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 양의 정수 값(0 ~ 255)이 반환됩니다.
데이터 값은 우측에서 좌측으로 증가됩니다.
BIN (디지털 바이트 입력) 설명을 참조하십시오.

Arguments

<값> Data type: <정수형 변수/상수>
- 8비트 양의 정수 값(0 ~ 255) 만 입력할 수 있습니다.

<바이트번호> Data type: <정수형 변수/상수>
CC-Link V1: CBIN0 ~ CBIN15 (리모트 입출력 영역)
CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

<대기시간> Data type: <정수형 변수/상수>
설정 범위는 100 ~ 99,999 msec 입니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT STATE               // 정수형 변수 선언

    STATE = CBIN0           // CBIN0 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
    STATE = CBIN(0)         // CBIN0 신호 값 저장
    PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    WHILE 1                 // 반복문 시작
        CBIN(0,1000) = 255  // CBIN0번 입력 신호가 1일 때 까지 1초 대기
        IF CBIN0 == 255 THEN // CBIN0 신호 값이 255 일 때
            OUT(0) = 0      // OUT0번에 0 값 출력
        ELSE                // 그 외의 경우
            OUT(0) = 1      // OUT0번에 1 값 출력
        ENDIF               // 조건문 종료
    ENDWL                   // 반복문 종료
EOP                         // 메인 함수 종료
```

Example 3

- 비트 연산자 응용 예

```
MAIN                        // 메인 함수 선언
    INT AA, RES             // 정수형 변수 선언
    DEFSTR STA              // 문자열 변수 선언

    AA = CBIN0              // CBIN0 신호 값 저장
    RES = AA | 0HF0         // 4 ~ 7번째 값 비트 OR연산
    STA = BINS(RES)         // 문자열 변수에 이진수 문자열 저장
    PRINT(0,STA)            // 모니터링 로그 0번 인덱스에 이진 문자열로 출력
    RES = 0B0110 & CBIN0    // 1 ~ 2번째 값 비트 AND 연산
    STA = BINS(RES)         // 문자열 변수에 이진수 문자열 저장
    PRINT(1,STA)            // 모니터링 로그 1번 인덱스에 이진 문자열로 출력
EOP                         // 메인 함수 종료
```

Related terms

CIN, CWIN, CDIN, CFIN

1) 신호 값 대기 시 사용에 주의하십시오.
2) 대기시간 입력 시 설정한 대기시간이 지나면 다음 구문을 실행하므로 프로그램 작성에 주의하십시오.

8.10 CBOUT (필드버스 바이트 출력)

Usage

지정된 필드버스 카드의 바이트번호에 지정된 값을 출력합니다.

Syntax

CBOUT[(]<바이트번호>[)] = <값>
<변수> = CBOUT[(]<바이트번호>[)]

Description

CBOUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 값을 출력합니다.
두 번째 방법은 지정된 바이트번호로 출력한 신호 값을 읽어 변수에 저장합니다.

1) <바이트번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CBOUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <바이트번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 양의 정수 값(0 ~ 255)이 반환됩니다. 데이터 값은 우측에서 좌측으로 증가됩니다. BOUT(디지털 바이트 출력) 설명을 참조하십시오.

Arguments

<값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 8비트로 -128 ~ 127 입니다.

<바이트번호> Data type: <정수형 변수/상수>
CC-Link V1: DeviceNet : CBOUT0 ~ CBOUT15 (리모트 입출력 영역)
CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT STATE               // 정수형 변수 선언
    STATE = CBOUT1          // 출력한 CBOUT1 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
    STATE = CBOUT(1)        // 출력한 CBOUT1 신호 값 저장
    PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
    CBOUT(0) = 62           // CBOUT 0번에 바이트 값 출력
    CBOUT0 = 0              // CBOUT 0번에 0 값 출력
EOP                         // 메인 함수 종료
```

Related terms

COUT, CWOUT, CDOUT, CFOUT

8.11 CWIN (필드버스 워드 입력)

Usage

필드버스 카드에서 지정된 워드번호의 값을 읽어 옵니다.

Syntax

〈변수〉 = CWIN[(]〈워드번호〉[)]

Description

지정된 워드번호의 신호 값을 읽어 변수에 저장합니다.

1) 〈워드번호〉는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CWIN과 〈번호〉간에는 공란이 있으면 안됩니다.
2) 〈워드번호〉의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

〈변수〉 Data type: 〈정수형 변수〉
결과 값은 16비트 정수 값이 반환됩니다.
데이터 값은 우측에서 좌측으로 증가됩니다.
PIN (디지털 포트 입력) 설명을 참조하십시오.

Arguments

〈워드번호〉 Data type: 〈정수형 변수/상수〉
CC-Link V1: CWIN0 ~ CWIN15 (리모트 레지스트 영역)
CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언
    STATE = CWIN0       // CWIN0 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = CWIN(0)     // CWIN0 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Example 2 비트 연산자 응용 예

```
MAIN                    // 메인 함수 선언
    INT RES             // 정수형 변수 선언
    RES = CWIN0 | 0HFF00    // CWIN0 신호 값 저장, 8 ~ 15번째 값 비트 OR연산
    PRINT(0,BINS(RES))  // 모니터링 로그 0번 인덱스에 이진 문자열로 출력
EOP                     // 메인 함수 종료
```

Related terms

CIN, CBIN, CDIN, CFIN

8.12 CWOUT (필드버스 워드 출력)

Usage

지정된 필드버스 카드 워드번호에 지정된 값을 출력합니다.

Syntax

CWOUT[(]<워드번호>[)] = <값>
<변수> = CWOUT [(]<워드번호>[)]

Description

CWOUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 값을 출력합니다.
두 번째 방법은 지정된 워드번호로 출력한 신호 값을 읽어 변수에 저장합니다.
1) <워드번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CWOUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <워드번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수> Data type: <정수형 변수>
결과 값은 16비트 정수 값이 반환됩니다. 데이터 값은 우측에서 좌측으로 증가됩니다.
POUT (디지털 포트 출력) 설명을 참조하십시오.

Arguments

<값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 16비트로 -32,768 ~ 32,767 입니다.

<워드번호> Data type: <정수형 변수/상수>
CC-Link V1: CWOUT0 ~ CWOUT15 (리모트 레지스트 영역)
CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

```
MAIN                        // 메인 함수 선언
    INT STATE               // CWOUT 0번에 16비트 정수 값 출력
    CWOUT(0) = -2456        // CWOUT 0번에 0 값 출력
    CWOUT0 = 0
                            // 정수형 변수 선언
    STATE = CWOUT1          // 출력한 CWOUT1 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
    STATE = CWOUT(1)        // 출력한 CWOUT1 신호 값 저장
    PRINT(1,STATE)          // 모니터링 로그 1번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Related terms

COUT, CBOUT, CDOUT, CFOUT

8.13 CDIN (필드버스 더블워드 입력)

Usage

필드버스 카드에서 지정된 더블워드번호의 값을 정수 형태로 읽어 옵니다.

Syntax

<변수> = CDIN [(]<더블워드번호>[)]

Description

지정된 더블워드번호의 신호 값을 읽어 변수에 저장합니다.

1) <더블워드번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CDIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <더블워드번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수>	Data type: <정수형 변수> 결과 값은 32비트 정수 값이 반환됩니다.

Arguments

<더블워드번호>	Data type: <정수형 변수/상수> CC-Link V1: CDIN0 ~ CDIN7 (리모트 입출력 영역) CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

```
MAIN                        // 메인 함수 선언
    INT DATA                // 정수형 변수 선언

    DATA = CDIN0            // CDIN0 신호 값 저장
    PRINT(0, DATA)          // 모니터링 로그 0번 인덱스에 정수 값으로 출력
    DATA = CDIN(0)          // CDIN0 신호 값 저장
    PRINT(1, DATA)          // 모니터링 로그 1번 인덱스에 정수 값으로 출력
EOP                         // 메인 함수 종료
```

Related terms

CIN, CBIN, CWIN

8.14 CDOUT (필드버스 더블워드 출력)

Usage

지정된 필드버스 카드 더블워드번호에 지정된 값을 정수 형태로 출력합니다.

Syntax

CDOUT [(]<더블워드번호>[)] = <값>
<변수> = CDOUT [(]<더블워드번호>[)]

Description

CDOUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 값을 정수 형태로 출력합니다.
두 번째 방법은 지정된 더블워드번호로 출력한 신호 값을 읽어 변수에 저장합니다.
1) <더블워드번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CDOUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <더블워드번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수>	Data type: <정수형 변수> 결과 값은 32비트 실수 값이 반환됩니다.

Arguments

<값>	Data type: <정수형 변수/상수> 입력할 수 있는 값의 범위는 32비트로 -2,147,483,648 ~ 2,147,483,647 입니다.
<더블워드번호>	Data type: <정수형 변수/상수> CC-Link V1: CDOUT0 ~ CDOUT7 (리모트 입출력 영역) CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    CDOUT(0) = 145748   // CDOUT 0번에 정수 값 출력
    CDOUT0 = 0          // CDOUT 0번에 0 값 출력
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    INT STATE           // 정수형 변수 선언
    STATE = CDOUT1      // 출력한 CDOUT1 신호 값 저장
    PRINT(0,STATE)      // 모니터링 로그 0번 인덱스에 출력
    STATE = CDOUT(1)    // 출력한 CDOUT1 신호 값 저장
    PRINT(1,STATE)      // 모니터링 로그 1번 인덱스에 출력
EOP                     // 메인 함수 종료
```

Related terms

COUT, CBOUT, CWOUT

8.15 CFIN (필드버스 더블워드 입력)

Usage

필드버스 카드에서 지정된 더블워드번호의 값을 실수 형태로 읽어 옵니다.

Syntax

<변수> = CFIN [(]<더블워드번호>[)]

Description

지정된 더블워드번호의 신호 값을 읽어 변수에 저장합니다.

1) <더블워드번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CFIN과 <번호>간에는 공란이 있으면 안됩니다.
2) <더블워드번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수>	Data type: <실수형 변수> 결과 값은 32비트 실수 값이 반환됩니다.

Arguments

<더블워드번호>	Data type: <정수형 변수/상수> CC-Link V1: CFIN0 ~ CFIN7 (리모트 레지스트 영역) CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

```
MAIN                        // 메인 함수 선언
    REAL DATA               // 실수형 변수 선언

    DATA = CFIN0            // CFIN0 신호 값 저장
    PRINT(0, DATA)          // 모니터링 로그 0번 인덱스에 출력
    DATA = CFIN(0)          // CFIN0 신호 값 저장
    PRINT(1, DATA)          // 모니터링 로그 1번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Related terms

CIN, CBIN, CWIN, CDIN

8.16 CFOUT (필드버스 더블워드 출력)

Usage

지정된 필드버스 카드 더블워드번호에 지정된 값을 정수 형태로 출력합니다.

Syntax

CFOUT [(]<더블워드번호>[)] = <값>
<변수> = CFOUT[(]<더블워드번호>[)]

Description

CFOUT 명령어는 2가지 사용방법을 제시합니다.
첫 번째 방법은 지정된 실수 값을 출력합니다.
두 번째 방법은 지정된 더블워드번호로 출력한 신호 값을 읽어 변수에 저장합니다.

1) <더블워드번호>는 괄호를 사용하여 입력하거나 숫자를 바로 입력하여 사용할 수 있습니다. 문자 CFOUT과 <번호>간에는 공란이 있으면 안됩니다.
2) <더블워드번호>의 범위는 사용되는 필드버스의 타입에 따라 달라질 수 있으므로 주의 바랍니다.

※ 필드버스 카드 설정은 PUB 파라미터의 1.HW_CONF(1/3)→4.OPTCARD에서 설정

Return value

<변수> Data type: <실수형 변수>
결과 값은 32비트 실수 값이 반환됩니다.

Arguments

<값> Data type: <실수형 변수/상수>
입력할 수 있는 값의 범위는 32비트로 3.402,823E-38 ~ 3.402,823E+38입니다.

<더블워드번호> Data type: <정수형 변수/상수>
CC-Link V1: CFOUT0 ~ CFOUT7 (리모트 레지스트 영역)
CC-Link V2, Ethernet IP, DeviceNet: 각 옵션 매뉴얼을 참고 하시기 바랍니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    CFOUT(0) = 14.5748      // CFOUT 0번에 실수 값 출력
    CFOUT0 = 0.0            // CFOUT 0번에 0 값 출력
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL STATE              // 실수형 변수 선언
    STATE = CFOUT1          // 출력한 CFOUT1 신호 값 저장
    PRINT(0,STATE)          // 모니터링 로그 0번 인덱스에 출력
EOP                         // 메인 함수 종료
```

Related terms

COUT, CBOUT, CWOUT, CDOUT

8.17 XIN (내부 필드버스 비트 입력)

Usage

지정한 디바이스 영역과 비트 번호를 참조하여 내부 필드버스 버퍼 값을 읽어옵니다.

Syntax

<변수> = XIN(<디바이스 영역>, <비트 번호>)

Description

1) CNET(디바이스 영역 “M”, “D”)
   - 비트 번호를 참조하여 수신 버퍼의 비트 입력 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Discrete input(1)), “W”(Input register(3))영역에 비트 번호를 참조하여 수신 버퍼의 비트 입력 값을 읽어옵니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<디바이스 영역>	Data type: <문자열> 또는 <문자열 변수> • CNET : “M” 또는 “D”문자 입력이 가능합니다. • MODBUS : “B” 또는 “W”문자 입력이 가능합니다.
<비트 번호>	Data type: <정수형 변수/상수> • CNET : “M” 영역: 0 ~ 383 (총 384개), “D” 영역: 0 ~ 2047 (총 2048개) • MODBUS : “B”, “W” 영역: 0 ~ 2047 (총 2048개)

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT DATA                // 정수형 변수 선언
    DATA = XIN(“M”,0)       // “M” 영역 0번 비트 값 읽어오기
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    IF XIN(“W”, 100) == 1 THEN  // “W” 영역 100번 비트 값이 1일 때
        JMOV GP1            // GP1 위치로 JMOV
    ENDIF                   // 조건문 종료
EOP                         // 메인 함수 종료
```

Related terms

XOUT, XBIN, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFIN, XFOUT

8.18 XOUT (내부 필드버스 비트 출력)

Usage

지정한 디바이스 및 비트 번호를 참조하여 내부 필드버스 발신 버퍼 값을 출력합니다.

Syntax

XOUT(<디바이스 영역>, <비트 번호>) = <0/1>
<변수> = XOUT(<디바이스 영역>, <비트 번호>)

Description

XOUT 명령어는 2가지 사용방법을 제시합니다.

1) CNET(디바이스 영역 “M”, “D”)
   - 비트 번호를 참조하여 발신 버퍼에 출력합니다.
   - 비트 번호로 출력한 신호 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Coil (0)), “W”(Holding register(4))영역에 비트 번호를 참조하여 발신 버퍼에 출력합니다.
   - 비트 번호로 출력한 신호 값을 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(OFF) 또는 1(ON)값이 반환됩니다.

Arguments

<디바이스 영역> Data type: <문자열> 또는 <문자열 변수>
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

<비트 번호> Data type: <정수형 변수/상수>
- CNET : “M” 영역: 0 ~ 383 (총 384개), “D” 영역: 0 ~ 2047 (총 2048개)
- MODBUS : “B”, “W” 영역: 0 ~ 2047 (총 2048개)

Examples

```
MAIN                                // 메인 함수 선언
    INT DATA                        // 정수형 변수 선언

    XOUT(“M”, 0) = 1                // “M” 영역 0번 비트에 1 값 출력
    XOUT(“D”, 100) = 1              // “D” 영역 100번 비트에 1 값 출력
    DATA = XOUT(“D”, 100)           // “D” 영역 100번 비트 출력 값 읽기
EOP                                 // 메인 함수 종료
```

Related terms

XIN, XBIN, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFIN, XFOUT

8.19 XBIN (내부 필드버스 바이트 입력)

Usage

지정한 디바이스 및 바이트 번호를 참조하여 내부 필드버스 버퍼 값을 읽어옵니다.

Syntax

<변수> = XBIN(<디바이스 영역>, <바이트 번호>)

Description

1) CNET(디바이스 영역 “M”, “D”)
   - 바이트 번호를 참조하여 수신 버퍼로 바이트 입력 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Discrete input(1)), “W”(Input register(3))영역에 바이트 번호를 참조하여 수신 버퍼로 바이트 입력 값을 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 정수 값이 반환되며, 데이터 값은 우측에서 좌측으로 증가됩니다.
BIN (디지털 바이트 입력) 설명을 참조하십시오.

Arguments

<디바이스 영역> Data type: <문자열> 또는 <문자열 변수>
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

<바이트 번호> Data type: <정수형 변수/상수>
- CNET : “M” 영역: 0 ~ 47 (총 48개), “D” 영역: 0 ~ 255 (총 256개)
- MODBUS : “B”, “W” 영역: 0 ~ 255 (총 256개)

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT DATA                            // 정수형 변수 선언
    DATA = XBIN(“M”, 0)                 // “M” 영역 0번 바이트 데이터를 읽기
EOP                                     // 메인 함수 종료
```

Example 2

- 비트 연산자 응용 예

```
MAIN                                    // 메인 함수 선언
    INT DATA                            // 정수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언
    DATA = XBIN("D",0) & 0H000F         // “D” 영역 0번 바이트의 0~3번째 비트 AND연산
    STR = BINS(DATA)                    // 결과 데이터 이진 문자열로 변환
    DATA = 0B11110000 ^ XBIN("D",0)     // “D” 영역 0번 바이트의 4~7번째 비트 XOR연산
EOP                                     // 메인 함수 종료
```

Related terms

XIN, XOUT, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFIN, XFOUT

8.20 XBOUT (내부 필드버스 바이트 출력)

Usage

지정한 디바이스 및 바이트 번호를 참조하여 내부 필드버스 버퍼 값을 출력합니다.

Syntax

XBOUT(<디바이스 영역>, <바이트 번호>) = <출력 값>
<변수> = XBOUT(<디바이스 영역>, <바이트 번호>)

Description

XBOUT 명령어는 2가지 사용방법을 제시합니다.

1) CNET(디바이스 영역 “M”, “D”)
   - 바이트 번호를 참조하여 발신 버퍼에 출력합니다.
   - 바이트 번호로 출력한 신호 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Coil(0)), “W”(Holding register(4))영역에 바이트 번호를 참조하여 발신 버퍼에 출력합니다.
   - 바이트 번호로 출력한 신호 값을 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 8비트 정수 값이 반환되며, 데이터 값은 우측에서 좌측으로 증가됩니다.
BOUT(디지털 바이트 출력) 설명을 참조하십시오.

Arguments

<출력 값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 8비트로 0 ~ 255 입니다.

<디바이스 영역> Data type: <문자열> 또는 <문자열 변수>
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

<바이트 번호> Data type: <정수형 변수/상수>
- CNET : “M” 영역: 0 ~ 47 (총 48개), “D” 영역: 0 ~ 255 (총 256개)
- MODBUS : “B”, “W” 영역: 0 ~ 255 (총 256개)

Examples

```
MAIN                          // 메인 함수 선언
    INT DATA                  // 정수형 변수 선언
    DATA = XBOUT(“B”, 0)      // “B”영역 0번 바이트에 값 쓰기
    XBOUT(“M”, 0) = 0HFF      // “M”영역 0번 바이트 출력 값 읽기
EOP                           // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XWIN, XWOUT, XDIN, XDOUT, XFIN, XFOUT

8.21 XWIN (내부 필드버스 워드 입력)

Usage

지정한 디바이스 및 워드 번호를 참조하여 내부 필드버스 수신 버퍼 값을 읽어옵니다.

Syntax

<변수> = XWIN(<디바이스 영역>, <워드 번호>)

Description

1) CNET(디바이스 영역 “M”, “D”)
   - 워드 번호를 참조하여 수신 버퍼로 워드입력 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Discrete input(1)), “W”(Input register(3))영역에 워드 번호를 참조하여 수신 버퍼로 워드입력 값을 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 16비트 정수 값이 반환되며, 데이터 값은 우측에서 좌측으로 증가됩니다.
PIN (디지털 포트 입력) 설명을 참조하십시오.

Arguments

<디바이스 영역> Data type: <문자열> 또는 <문자열 변수>
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

<워드 번호> Data type: <정수형 변수/상수>
- CNET : “M” 영역: 0 ~ 23 (총 24개), “D” 영역: 0 ~ 127 (총 128개)
- MODBUS : “B”, “W” 영역: 0 ~ 127 (총 128개)

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT DATA                // 정수형 변수 선언
    DATA = XWIN(“B”, 0)     // “B”영역 0번 워드 값 읽기
EOP                         // 메인 함수 종료
```

Example 2

- 비트 연산자 응용 예

```
MAIN                                    // 메인 함수 선언
    INT RES                             // 정수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언
    RES = XWIN(“D”, 0) | 0HFF00         // “D”영역 0번 워드의 8 ~ 15번째 비트 OR연산
    STR = BINS(RES)                     // 이진 문자열로 저장
    RES = 0B10000110 & XWIN(“D”, 0)     // “D”영역 0번 워드의 1 ~ 2, 7번째 비트 AND연산
EOP                                     // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWOUT, XDIN, XDOUT, XFIN, XFOUT

8.22 XWOUT (내부 필드버스 워드 출력)

Usage

지정한 디바이스 및 워드 번호를 참조하여 내부 필드버스 버퍼에 값을 출력합니다.

Syntax

XWOUT(<디바이스 영역>, <워드 번호>) = <출력 값>
<변수> = XWOUT(<디바이스 영역>, <워드 번호>)

Description

XWOUT 명령어는 2가지 사용방법을 제시합니다.

1) CNET(디바이스 영역 “M”, “D”)
   - 워드 번호를 참조하여 발신 버퍼에 출력합니다.
   - 워드 번호로 출력한 신호 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Coil(0)), “W”(Holding register(4))영역에 워드 번호를 참조하여 발신 버퍼에 출력합니다.
   - 워드 번호로 출력한 신호 값을 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 16비트 정수 값이 반환되며, 데이터 값은 우측에서 좌측으로 증가됩니다.
POUT (디지털 포트 출력) 설명을 참조하십시오.

Arguments

<출력 값> Data type: <정수형 변수/상수>
입력할 수 있는 값의 범위는 16비트로 -32,768 ~ 32,768 입니다.

<디바이스 영역> Data type: <문자열> 또는 <문자열 변수>
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

<워드 번호> Data type: <정수형 변수/상수>
- CNET : “M” 영역: 0 ~ 23 (총 24개), “D” 영역: 0 ~ 127 (총 128개)
- MODBUS : “B”, “W” 영역: 0 ~ 127 (총 128개)

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    INT DATA                  // 정수형 변수 선언
    XWOUT(“B”, 0) = 0HFFFF    // “B” 영역 0번 워드에 값 쓰기
    DATA = XWOUT(“D”, 0)      // “D” 영역 0번 워드 출력 값 읽기
EOP                           // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWIN, XDIN, XDOUT, XFIN, XFOUT

8.23 XDIN (내부 필드버스 더블 워드 입력)

Usage

지정한 디바이스 및 더블 워드 번호를 참조하여 내부 필드버스 버퍼 값을 읽어옵니다.

Syntax

<변수> = XDIN(<디바이스 영역>, <더블 워드 번호>)

Description

1) CNET(디바이스 영역 “M”, “D”)
   - 더블워드단위 번호를 참조하여 수신 버퍼로 워드입력 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Discrete input(1)), “W”(Input register(3))영역에 더블워드단위 번호를 참조하여 수신 버퍼로 워드입력 값을 읽어옵니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 32비트 정수 값이 반환됩니다.

Arguments

<출력 값>	Data type: <정수형 변수/상수> 입력할 수 있는 값의 범위는 16비트로 -32,768 ~ 32,768 입니다.
<디바이스 영역>	Data type: <문자열> 또는 <문자열 변수> • CNET : “M” 또는 “D”문자 입력이 가능합니다. • MODBUS : “B” 또는 “W”문자 입력이 가능합니다.
<더블 워드 번호>	Data type: <정수형 변수/상수> • CNET : “M” 영역: 0 ~ 11 (총 12개), “D” 영역: 0 ~ 63 (총 64개) • MODBUS : “B”, “W” 영역: 0 ~ 63 (총 64개)

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT DATA                // 정수형 변수 선언
    DATA = XDIN(“W”, 0)     // “W”영역 0번 더블 워드 값 읽기
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    INT RES                             // 정수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언
    RES = XDIN(“D”, 0) | 0HFFFF0000     // “D”영역 0번 더블 워드 16~31번째 비트 OR연산
    STR = BINS(RES)                     // 이진 문자열로 저장
    PRINT(0,STR)                        // 모니터링 로그 0번 인덱스에 문자열 출력
EOP                                     // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWIN, XWOUT, XDOUT, XFIN, XFOUT

8.24 XDOUT (내부 필드버스 더블 워드 출력)

Usage

지정한 디바이스 및 더블 워드 번호를 참조하여 내부 필드버스 버퍼 값을 출력합니다.

Syntax

XDOUT(<디바이스 영역>, <더블 워드 번호>) = <출력 값>
<변수> = XDOUT(<디바이스 영역>, <더블 워드 번호>)

Description

XDOUT 명령어는 2가지 사용방법을 제시합니다.

1) CNET(디바이스 영역 “M”, “D”)
   - 더블워드 단위 번호를 참조하여 발신 버퍼에 출력합니다.
   - 더블워드 단위 번호로 출력한 신호 값을 읽어 변수에 저장 합니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Coil(0)), “W”(Holding register(4))영역에 더블워드 단위 번호를 참조하여 발신 버퍼에 출력합니다.
   - 더블워드 단위 번호로 출력한 신호 값을 읽어 변수에 저장 합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 32비트 정수 값이 반환됩니다.

Arguments

<디바이스 영역>	Data type: <문자열> 또는 <문자열 변수> • CNET : “M” 또는 “D”문자 입력이 가능합니다. • MODBUS : “B” 또는 “W”문자 입력이 가능합니다.
<더블 워드 번호>	Data type: <정수형 변수/상수> • CNET : “M” 영역: 0 ~ 11 (총 12개), “D” 영역: 0 ~ 63 (총 64개) • MODBUS : “B”, “W” 영역: 0 ~ 63 (총 64개)

Examples

Example 1

```
MAIN                              // 메인 함수 선언
    XDOUT(“M”, 0) = 0HFFFFFFFF    // “M” 영역 0번 더블 워드에 값 쓰기
EOP                               // 메인 함수 종료
```

Example 2

```
MAIN                              // 메인 함수 선언
    INT DATA                      // 정수형 변수 선언
    DATA = XDOUT(“B” ,0)          // “B” 영역 0번 더블 워드 출력 값 읽기
EOP                               // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWIN, XWOUT, XDIN, XFIN, XFOUT

8.25 XFIN (내부 필드버스 실수 입력)

Usage

지정한 디바이스 및 더블 워드 번호를 참조하여 내부 필드버스 버퍼 값을 실수 형태로 읽어옵니다.

Syntax

〈변수〉 = XFIN(〈디바이스 영역〉, 〈더블 워드 번호〉)

Description

1) CNET(디바이스 영역 “M”, “D”)
   - 더블워드단위 번호를 참조하여 수신 버퍼로 실수 값을 읽어옵니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Discrete input(1)), “W”(Input register(3))영역에 더블워드단위 번호를 참조하여 수신 버퍼로 실수 값을 읽어옵니다.

Return value

〈변수〉 Data type: 〈실수형 변수〉
결과 값은 32비트 실수 값이 반환됩니다.

Arguments

〈디바이스 영역〉 Data type: 〈문자열〉 또는 〈문자열 변수〉
- CNET : “M” 또는 “D”문자 입력이 가능합니다.
- MODBUS : “B” 또는 “W”문자 입력이 가능합니다.

〈더블 워드 번호〉 Data type: 〈정수형 변수/상수〉
- CNET : “M” 영역: 0 ~ 11 (총 12개), “D” 영역: 0 ~ 63 (총 64개)
- MODBUS : “B”, “W” 영역: 0 ~ 63 (총 64개)

Examples

Example

```
MAIN                          // 메인 함수 선언
    REAL DATA1, DATA2         // 실수형 변수 선언

    DATA1 = XFIN(“M”, 0)      // “M” 영역 0번 더블 워드 영역의 실수 값 읽기
    DATA2 = XFIN(“W”, 63)     // “W” 영역 63번 더블 워드 영역의 실수 값 읽기
EOP                           // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFOUT

8.26 XFOUT (내부 필드버스 실수 출력)

Usage

지정한 디바이스 및 더블 워드 번호를 참조하여 내부 필드버스 버퍼 실수 값을 출력합니다.

Syntax

XFOUT(<디바이스 영역>, <더블 워드 번호>) = <실수 값>
<변수> = XFOUT(<디바이스 영역>, <더블 워드 번호>)

Description

XDOUT 명령어는 2가지 사용방법을 제시합니다.

1) CNET(디바이스 영역 “M”, “D”)
   - 더블워드 단위 번호를 참조하여 발신 버퍼에 출력합니다.
   - 더블워드 단위 번호로 출력한 신호 값을 읽어 변수에 실수 값을 저장 합니다.
2) MODBUS(디바이스 영역 “B”, “W”)
   - “B”(Coil(0)), “W”(Holding register(4))영역에 더블워드 단위 번호를 참조하여 발신 버퍼에 출력합니다.
   - 더블워드 단위 번호로 출력한 신호 값을 읽어 변수에 실수 값을 저장 합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값은 32비트 실수 값이 반환됩니다.

Arguments

<디바이스 영역>	Data type: <문자열> 또는 <문자열 변수> • CNET : “M” 또는 “D”문자 입력이 가능합니다. • MODBUS : “B” 또는 “W”문자 입력이 가능합니다.
<더블 워드 번호>	Data type: <정수형 변수/상수> • CNET : “M” 영역: 0 ~ 11 (총 12개), “D” 영역: 0 ~ 63 (총 64개) • MODBUS : “B”, “W” 영역: 0 ~ 63 (총 64개)

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    XFOUT(“M”, 0) = 12.45     // “M” 영역 0번 더블 워드에 실수 값 쓰기
EOP                           // 메인 함수 종료
```

Example 2

```
MAIN                          // 메인 함수 선언
    REAL DATA                 // 실수형 변수 선언
    DATA = XFOUT(“B”, 0)      // “B” 영역 0번 더블 워드 영역의 실수 값 읽기
EOP                           // 메인 함수 종료
```

Related terms

XIN, XOUT, XBIN, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFIN

9. Multithreading 제어

9.1 CTHREAD (Thread 생성)

Usage

함수 또는 프로그램을 동시에 실행합니다.

Syntax

[<변수> =] CTHREAD(<함수명>, <Thread 번호>, [<Thread 알람정지>])

Description

함수 또는 프로그램을 Thread에 할당하여 동시 실행시킵니다.
함수를 할당하는 경우, 반환 유형은 반드시 VOID형이고, 매개변수는 없어야 합니다.
프로그램을 할당하는 경우, INCLUDE 명령어를 사용해 반드시 파일을 미리 첨부해 주어야 합니다.

1) <함수명>에는 함수 또는 프로그램 이름을 써 넣습니다.
2) <Thread 번호>는 구동할 Thread의 번호를 입력합니다. 선택한 Thread 번호가 이미 다른 Thread에 의해 사용된 경우에는 Thread가 생성되지 않습니다.
3) <Thread 알람정지>는 알람이 발생하였을 때 Thread의 정지여부를 설정합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 음수 값(Thread 생성 실패) 또는 0(Thread 생성 성공)값이 반환됩니다.

Arguments

<함수명>	Data type: <함수 또는 프로그램 명> JOB 프로그램 명을 입력하는 경우 “_MAIN”을 반드시 프로그램 명 뒤에 추가해야 합니다. • 예시: TEST.JOB -> CTHREAD(TEST_MAIN, 2)
<Thread 번호>	Data type: <정수형 변수/상수> Thread 번호의 범위는 2 ~ 4(TH2 ~ TH4) 입니다.
<Thread 알람 정지>	Data type: <정수형 변수/상수> 알람 시 Thread 정지(1), 알람 시 Thread 미 정지(0) 중 하나를 입력합니다. 기본 값은 정지(1)입니다.

Examples

Example 1 • 프로그램 호출 예

```
INCLUDE “TEST”                          // TEST.JOB 파일 포함
MAIN                                    // 메인 함수 선언
    INT T_RET                           // 정수형 변수 선언

    T_RET = CTHREAD(TEST_MAIN,2)        // TEST.JOB의 메인 함수를 Thread 2번에 할당
    IF T_RET < 0 THEN                   // 실행 결과가 Thread 생성 실패 인 경우
        EXIT                            // JOB 프로그램 종료
    ENDIF                               // IF문 종료

    ...
    WHILE 1                             // 반복문 실행
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Example 2 • 함수 호출 예

```
MAIN                                    // 메인 함수 선언
    SVON                                // 서보 ON
    CTHREAD(MOVE, 2, 0)                 // MOVE 함수를 Thread 2번에 할당
                                        // 알람 발생 시 Thread 미종료
    WHILE 1                             // 반복문 실행
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료

FUNC VOID MOVE()                        // MOVE 함수 선언 (VOID 타입, 매개변수 없음)
    WHILE 1                             // 반복문 시작
        JMOV GP0                        // GP0로 JMOV
        DLAY 1000                       // 대기 1초
        JMOV GP1                        // GP1로 JMOV
        DLAY 1000                       // 대기 1초
    ENDWL                               // 반복문 종료
END                                     // 함수 종료
```

Related terms

ETHREAD, TH_STATE, TH_SUSPEND, TH_RESUME

9.2 ETHREAD (Thread 종료)

Usage

Thread를 강제로 종료합니다.

Syntax

[<변수> =] ETHREAD(<Thread 번호>)

Description

실행 중인 Thread를 완전히 종료합니다.
ETHREAD 명령어를 사용하여 Thread를 종료하면 TH_RESUME 명령어를 사용해 재시작 할 수 없습니다.

1) <Thread 번호>에 종료할 Thread의 번호를 입력합니다. CTHREAD 명령어를 사용할 때 입력한 Thread의 번호를 입력하여 Thread를 종료합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(Thread 종료 성공) 또는 -1(Thread 종료 실패)값이 반환됩니다.

Arguments

<Thread 번호>	Data type: <정수형 변수/상수> Thread 번호의 범위는 2 ~ 4(TH2 ~ TH4) 입니다.

Examples

```
INCLUDE “TEST”                  // TEST.JOB 파일 포함
MAIN                            // 메인 함수 선언
    INT T_RET                   // 정수형 변수 선언
    CTHREAD(TEST_MAIN,2,1)      // TEST.JOB의 메인 함수를 Thread 2번에 할당
                                // 알람 발생 시 Thread 종료
    IN0 = 1                     // 0번 입력 신호가 1일 때까지 대기
    T_RET = ETHREAD(2)          // Thread 강제 종료

    IF T_RET < 0 THEN           // 실행 결과가 Thread 종료 실패 인 경우
        EXIT                    // JOB 프로그램 종료
    ENDIF                       // IF문 종료
EOP                             // 메인 함수 종료
```

Related terms

CTHREAD, TH_STATE, TH_SUSPEND, TH_RESUME

9.3 TH_STATE (Thread 상태 읽기)

Usage

Thread의 현재 상태를 확인합니다.

Syntax

<변수> = TH_STATE(<실행 모드>, <Thread 번호>)

Description

각 Thread의 현재 상태를 반환합니다.

1) <실행 모드>를 지정하여 System 및 Robot 모드의 Thread 상태를 모두 확인할 수 있습니다.
2) <Thread 번호>를 지정하여 해당 Thread의 상태를 읽어옵니다.

Return value

<변수> Data type: <정수형 변수>
Thread의 현재 상태 정보를 반환합니다.
- 0 (TH_ERROR): Thread 번호 오류
- 1 (TH_IDLE): Thread 할당 없음, 신규 사용 가능 상태
- 2 (TH_STOP): Thread 정지
- 3 (TH_RUN): Thread 구동 중

Arguments

<실행 모드> Data type: <정수형 변수/상수>
System 모드(1) 또는 Robot 모드(0)를 입력합니다.

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
INCLUDE “MOVE”                          // MOVE.JOB 파일 포함
MAIN                                    // 메인 함수 선언
    INT ST                              // 정수형 변수 선언
    INT P_ID, T_ID

    P_ID = 0                            // Robot 모드 지정
    T_ID = 2                            // Thread 2번 지정

    CTHREAD(MOVE_MAIN,T_ID,1)           // MOVE.JOB의 메인 함수 Thread 할당
                                        // 알람 발생 시 Thread 종료
    WHILE 1                             // 무한 반복문 시작
        ST = TH_STATE(P_ID, T_ID)       // Robot 모드의 Thread 2번 상태 확인

        IF (ST == 2) THEN               // TH_STOP 상태라면
            TH_RESUME(T_ID)             // Thread 재 시작
        ELSEIF (ST == 3 && IN0 == 1) THEN // TH_RUN상태이고, 0번 입력 신호가 1인 경우
            TH_SUSPEND(T_ID)            // Thread 정지
        ENDIF                           // 조건문 종료

        DLAY 1000                       // 1초 대기

        IF (ST == 1 || ST == 0) THEN    // Thread가 할당되지 않거나 오류 인 경우
            ETHREAD(T_ID)               // Thread 종료
        ENDIF                           // 조건문 종료

    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

CTHREAD, ETHREAD, TH_SUSPEND, TH_RESUME

9.4 TH_SUSPEND (Thread 정지)

Usage

실행 중인 Thread를 정지합니다.

Syntax

[<변수> =] TH_SUSPEND(<Thread 번호>)

Description

실행 중인 Thread를 정지합니다.
정지상태의 Thread는 TH_RESUME 명령어로 재 실행할 수 있습니다.

1) <Thread 번호>에 정지할 Thread의 번호를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(Thread 정지 성공) 또는 -1(Thread 정지 실패)값이 반환됩니다.

Arguments

<Thread 번호>	Data type: <정수형 변수/상수> Thread 번호의 범위는 2 ~ 4(TH2 ~ TH4) 입니다.

Examples

```
INCLUDE “MOVE”                          // MOVE.JOB 파일 포함
MAIN                                    // 메인 함수 선언
    INT ST                              // 정수형 변수 선언
    INT P_ID, T_ID

    P_ID = 0                            // Robot 모드 지정
    T_ID = 2                            // Thread 2번 지정

    CTHREAD(MOVE_MAIN,T_ID,1)           // MOVE.JOB의 메인 함수 Thread 할당
                                        // 알람 발생 시 Thread 종료
    WHILE 1                             // 무한 반복문 시작
        ST = TH_STATE(P_ID, T_ID)       // Robot 모드의 Thread 2번 상태 확인

        IF (ST == 3 && IN0 == 1) THEN   // TH_RUN상태이고, 0번 입력 신호가 1인 경우
            TH_SUSPEND(T_ID)            // Thread 정지
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

CTHREAD, ETHREAD, TH_STATE, TH_RESUME

9.5 TH_RESUME (Thread 재시작)

Usage

정지한 Thread를 재 실행합니다.

Syntax

[<변수> =] TH_RESUME(<Thread 번호>)

Description

정지한 Thread를 재 시작합니다.
TH_SUSPEND로 정지한 Thread만 재 실행할 수 있습니다.

1) <Thread 번호>에 재 시작할 Thread의 번호를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(Thread 재시작 성공) 또는 -1(Thread 재시작 실패)값이 반환됩니다.

Arguments

<Thread 번호>	Data type: <정수형 변수/상수> Thread 번호의 범위는 2 ~ 4(TH2 ~ TH4) 입니다.

Examples

```
INCLUDE “MOVE”                          // MOVE.JOB 파일 포함
MAIN                                    // 메인 함수 선언
    INT ST                              // 정수형 변수 선언
    INT T_ID

    T_ID = 2                            // Thread 2번 지정

    CTHREAD(MOVE_MAIN,T_ID,1)           // MOVE.JOB의 메인 함수 Thread 할당
                                        // 알람 발생 시 Thread 종료
    WHILE 1                             // 무한 반복문 시작
        ST = TH_STATE(0, T_ID)          // Robot 모드의 Thread 2번 상태 확인

        IF (ST == 2) THEN               // TH_STOP 상태인 경우
            TH_RESUME(T_ID)             // Thread 재 시작
        ELSEIF (ST == 3 && IN0 == 1) THEN // TH_RUN상태이고, 0번 입력 신호가 1인 경우
            TH_SUSPEND(T_ID)            // Thread 정지
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

CTHREAD, ETHREAD, TH_STATE, TH_SUSPEND

10. 함수

10.1 TOOL (툴 좌표계 선택)

Usage

로봇에 부착된 사용 툴과 해당 좌표계를 선택합니다.

Syntax

TOOL <번호>

Description

작업 시 사용하는 툴의 번호를 지정합니다.
좌표 변환 시 지정한 해당 툴 파라미터 값을 참조하여 끝 단의 위치 및 방향을 계산합니다.
이 명령문에 의해 변경된 값은 프로그램 종료 시점까지 유효합니다.

이동 명령어에서 툴 값을 변경하는 경우 명령어에 삽입된 번호를 우선합니다.

1) <번호>를 입력하여 사용할 툴과 좌표계를 선택할 수 있습니다. 해당 툴 정보는 COORD-TOOL 파라미터에서 변경이 가능합니다.

Arguments

<번호>	Data type: <정수형 변수/상수> 툴 번호는 0 ~ 15까지 사용이 가능합니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV

    TOOL 1              // TOOL 1 번 지정
    LMOV GP1            // GP1까지 LMOV
    TOOL 2              // TOOL 2 번 지정
    LMOV GP2            // GP2까지 LMOV
    SVOF                // 서보 OFF
EOP                     // 메인 함수 종료
```

Related terms

USER

1) TOOL 0는 로봇 핸드의 끝점을 의미하며, 툴이 부착되지 않은 상태를 의미합니다. 해당 파라미터의 수정은 불가능합니다.
2) 작업 프로그램에서 TOOL 명령어를 사용하지 않는 경우 TOOL 0 값이 적용됩니다.

10.2 USER (사용자 좌표계 선택)

Usage

사용자 좌표계를 선택합니다.

Syntax

USER <번호>

Description

작업 시 사용하는 사용자 전용 번호를 지정합니다.
좌표 변환 시 지정한 해당 사용자 좌표계 파라미터 값을 참조하여 위치 및 방향을 계산합니다.
이 명령문에 의해 변경된 값은 프로그램 종료 시점까지 유효합니다.

이동 명령어에서 사용자 좌표계 값을 변경하는 경우 명령어에 삽입된 번호를 우선합니다.

1) <번호>를 입력하여 사용할 사용자 좌표계를 선택할 수 있습니다. 해당 좌표계 정보는 COORD-USER 파라미터에서 변경이 가능합니다.

Arguments

<번호>	Data type: <정수형 변수/상수> 사용자 좌표계 번호는 0 ~ 15까지 사용이 가능합니다.

Examples

```
MAIN                    // 메인 함수 선언
    SVON                // 서보 ON
    JMOV GP0            // GP0까지 JMOV

    USER 1              // USER 1 번 지정
    LMOV GP1            // GP1까지 LMOV
    USER 2              // USER 2 번 지정
    LMOV GP2            // GP2까지 LMOV
    SVOFF               // 서보 OFF
EOP                     // 메인 함수 종료
```

Related terms

TOOL

1) USER 0은 로봇 베이스를 원점으로 좌표계가 생성되어 있습니다. 해당 파라미터의 수정은 불가능합니다.
2) 작업 프로그램에서 USER 명령어를 사용하지 않는 경우 USER 0 값이 적용됩니다.

10.3 JTOB (좌표 데이터 계산, JOINT -> BASE)

Usage

조인트 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.

Syntax

<변수> = JTOB([<로봇 번호>,] <입력 변수>)

Description

입력 위치 변수의 툴 정보를 참고하여 조인트 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <조인트 좌표계 위치형 변수>

Examples

```
MAIN                                // 메인 함수 선언
    POS AP                          // 조인트 좌표계 위치형 변수 선언
    XPOS XP1, XP2                   // 베이스 좌표계 위치형 변수 선언
    AP = <0, 30, 30, 0, 30, 0>      // 조인트 위치 값 대입
    AP.T = 1                        // 툴 정보 대입
    XP1 = JTOB(AP)                  // AP를 베이스 좌표계 기준으로 변환
    PRINT(0, XP1)                   // 모니터링 로그 0번에 좌표 값 출력
    XP1.2 = -200                    // XP1의 Y값을 -200으로 설정
    XP2 = XP1                       // XP2에 XP1 대입
    XP2.2 = 200                     // XP2의 Y값을 200으로 설정
    SVON                            // 서보 ON
    JMOV AP                         // AP까지 JMOV
    LMOV XP1, T = 1                 // XP1까지 LMOV
    LMOV XP2, T = 1                 // XP2까지 LMOV
    SVOF                            // 서보 OFF
EOP                                 // 메인 함수 종료
```

Related terms

BTOJ, JTOU, UTOJ, BTOU, UTOB, TOOL, USER

10.4 BTOJ (좌표 데이터 계산, BASE -> JOINT)

Usage

베이스 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.

Syntax

<변수> = BTOJ([<로봇 번호>,] <입력 변수>)

Description

입력 위치 변수의 툴 및 로봇 자세 정보를 참고하여 베이스 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.
로봇 자세 정보가 지정되어 있지 않으면 현재 로봇의 자세를 기준으로 조인트 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <조인트 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <베이스 좌표계 위치형 변수>

Examples

```
MAIN                                          // 메인 함수 선언
    POS AP                                    // 조인트 좌표계 위치형 변수 선언
    XPOS XP                                   // 베이스 좌표계 위치형 변수 선언
    XP = <350, 0, 450, -180, 0, 0 T = 1, ARM = 212>  // 베이스 좌표 값 대입
    AP = BTOJ(XP)                             // XP를 조인트 좌표계로 변환
    PRINT(0, AP)                              // 모니터링 로그 0번 좌표 값 출력

    SVON                                      // 서보 ON
    JMOV AP                                   // AP까지 JMOV
    SVOF                                      // 서보 OFF
EOP                                           // 메인 함수 종료
```

Related terms

JTOB, JTOU, UTOJ, BTOU, UTOB, TOOL, USER

10.5 JTOU (좌표 데이터 계산, JOINT -> USER)

Usage

조인트 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.

Syntax

<변수> = JTOU([<로봇 번호>,] <입력 변수>)

Description

입력 위치 변수의 툴 및 사용자 좌표계 정보를 참고하여 조인트 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <사용자 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <조인트 좌표계 위치형 변수>

Examples

```
MAIN                                // 메인 함수 선언
    POS AP                          // 조인트 좌표계 위치형 변수 선언
    UPOS UP1, UP2                   // 사용자 좌표계 위치형 변수 선언
    AP = <0, 30, 30, 0, 30, 0>      // 조인트 좌표 값 대입
    AP.T = 1                        // AP1의 툴 좌표계 변호를 1로 설정
    AP.U = 1                        // AP1의 사용자 좌표계 변호를 1로 설정
    UP1 = JTOU(AP)                  // AP를 사용자 좌표계 기준으로 변환
    PRINT(0, UP1)                   // 모니터링 로그 0번 인덱스에 결과 출력
    UP1.2 = -200                    // UP1의 Y값을 -200으로 설정
    UP2 = UP1                       // UP2에 XP1 대입
    UP2.2 = 200                     // UP2의 Y값을 200으로 설정
    SVON                            // 서보 ON
    JMOV AP                         // AP까지 JMOV
    LMOV UP1, T = 1, U = 1          // UP1까지 LMOV
    LMOV UP2, T = 1, U = 1          // UP2까지 LMOV
    SVOF                            // 서보 OFF
EOP                                 // 메인 함수 종료
```

Related terms

JTOB, BTOJ, UTOJ, BTOU, UTOB, TOOL, USER

10.6 UTOJ (좌표 데이터 계산, USER -> JOINT)

Usage

사용자 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.

Syntax

<변수> = UTOJ([<로봇 번호>,] <입력 변수>)

Description

입력 위치 변수의 툴 및 사용자 좌표계 정보, 로봇 자세 정보를 참고하여 사용자 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.
로봇 자세 정보가 지정되어 있지 않으면 현재 로봇의 자세를 기준으로 조인트 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <조인트 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <사용자 좌표계 위치형 변수>

Examples

```
MAIN                                          // 메인 함수 선언
    POS AP                                    // 조인트 좌표계 위치형 변수 선언
    UPOS UP                                   // 사용자 좌표계 위치형 변수 선언
    UP = <10, 0, 10, -180, 0, 0 T = 1, ARM = 212,   // 사용자 좌표 값 대입
    U =1>
    AP = UTOJ(UP)                             // UP를 조인트 좌표계로 변환
    PRINT(0, AP)                              // 모니터링 로그 0번에 좌표 값 출력
    SVON                                      // 서보 ON
    JMOV AP                                   // AP까지 JMOV
    SVOF                                      // 서보 OFF
EOP                                           // 메인 함수 종료
```

Related terms

JTOB, BTOJ, JTOU, BTOU, UTOB, TOOL, USER

10.7 BTOU (좌표 데이터 계산, BASE -> USER)

Usage

베이스 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.

Syntax

<변수> = BTOU([<로봇 번호>,] <입력 변수>)

Description

입력 위치 변수의 사용자 좌표계 정보를 참고하여 베이스 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <사용자 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <베이스 좌표계 위치형 변수>

Examples

```
MAIN                                          // 메인 함수 선언
    XPOS XP1                                  // 베이스 좌표계 위치형 변수 선언
    UPOS UP1, UP2                             // 사용자 좌표계 위치형 변수 선언
    XP1 = <300, 0, 300, 180, 0, 0, ARM = 0, T = 1>  // 베이스 좌표 값 대입
    XP1.U = 1                                 // XP의 사용자 좌표계 번호 1로 설정
    UP1 = BTOU(XP1)                           // XP 사용자 좌표계를 기준으로 변환
    PRINT(0, UP1)                             // 모니터링 로그 0번에 좌표 값 출력
    UP2 = UP1                                 // UP2에 UP1 대입
    UP2.4 = 90                                // UP2의 X축 회전 각도 90도로 설정

    SVON                                      // 서보 ON
    JMOV UP1, T = 1, U = 1                    // UP1까지 JMOV
    LMOV UP2, T = 1, U = 1                    // UP2까지 LMOV
    SVOF                                      // 서보 OFF
EOP                                           // 메인 함수 종료
```

Related terms

JTOB, BTOJ, JTOU, UTOJ, UTOB, TOOL, USER

10.8 UTOB (좌표 데이터 계산, USER -> BASE)

Usage

사용자 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.

Syntax

<변수> = UTOB([<로봇 번호>,] <입력 변수>)

Description

입력 사용자 좌표계 정보를 참고하여 사용자 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <입력 변수>에 값을 입력할 때, 반드시 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<입력 변수>	Data type: <사용자 좌표계 위치형 변수>

Examples

```
MAIN                                        // 메인 함수 선언
    UPOS UP                                 // 사용자 좌표계 위치형 변수 선언
    XPOS XP1, XP2                           // 베이스 좌표계 위치형 변수 선언

    UP = <0, 0, 0, 180, 0, 0, ARM = 0, T = 1,U= 1>  // 사용자 좌표 값 대입
    XP1 = UTOB(UP)                          // UP를 베이스 좌표계를 기준으로 변환
    PRINT(0, XP1)                           // 모니터링 로그 0번에 좌표 출력

    XP2 = XP1                               // XP2에 XP1 대입
    XP2.4 = 90                              // XP2의 X축 회전 각도 90도로 설정

    SVON                                    // 서보 ON
    JMOV XP1                                // XP1까지 JMOV
    LMOV XP2, T = 1                         // XP2까지 LMOV
    SVOF                                    // 서보 OFF
EOP                                         // 메인 함수 종료
```

Related terms

JTOB, BTOJ, JTOU, UTOJ, BTOU, TOOL, USER

10.9 RELBASE (베이스 좌표계 기준 상대이동)

Usage

기준 위치에서 대입된 이동량만큼 베이스 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.

Syntax

<변수> = RELBASE([<로봇 번호>,] <기준 위치>, <이동량>)

Description

베이스 좌표계 기준 위치로부터 현재 베이스 좌표계 방향으로 상대 이동한 결과 값을 계산하여 베이스 좌표계 값으로 반환해 주는 함수입니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <기준 위치>에 값을 입력할 때 반드시 베이스 좌표계 위치형 변수 형태로 값을 입력해야 합니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <이동량>에 값을 입력할 때 반드시 위치형 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<기준 위치>	Data type: <베이스 좌표계 위치형 변수>
<이동량>	Data type: <모든 좌표계 위치형 변수>

Examples

```
MAIN                                    // 메인 함수 선언
    POS A_CUR, DELTA                    // 조인트 좌표계 위치형 변수 선언
    XPOS X_TAR, X_CUR                   // 베이스 좌표계 위치형 변수 선언

    DELTA = <50, 0, 50, 90, 0, 0>       // 상대 이동량 설정
    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    A_CUR.T = 1                         // A_CUR의 툴 좌표계 번호 설정

    X_CUR = JTOB(A_CUR)                 // A_CUR을 사용자 좌표계 기준으로 변환
    X_TAR = RELBASE(X_CUR, DELTA)       // X_CUR에서 DELTA만큼의 상대이동한 위치
                                        구함
    SVON                                // 서보 ON
    LMOV X_TAR, T = 1                   // X_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Related terms

RELTOOL, RELUSER

10.10 RELTOOL (툴 좌표계 기준 상대이동)

Usage

기준 위치에서 대입된 이동량만큼 툴 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.

Syntax

<변수> = RELTOOL([<로봇 번호>,] <기준 위치>, <이동량>)

Description

베이스 좌표계 기준 위치로부터 지정한 툴 좌표계 방향으로 상대 이동한 결과 값을 계산하여 베이스 좌표계 값으로 반환해 주는 함수입니다.
기준 위치의 툴 번호를 참조하여 툴 좌표계 상대 이동을 진행하게 됩니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <기준 위치>에 값을 입력할 때 반드시 베이스 좌표계 위치형 변수 형태로 값을 입력해야 합니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <이동량>에 값을 입력할 때 반드시 위치형 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<기준 위치>	Data type: <베이스 좌표계 위치형 변수>
<이동량>	Data type: <모든 좌표계 위치형 변수>

Examples

```
MAIN                                    // 메인 함수 선언
    POS A_CUR, DELTA                    // 조인트 좌표계 위치형 변수 선언
    XPOS X_TAR, X_CUR                   // 베이스 좌표계 위치형 변수 선언

    DELTA = <50, 0, 50, 90, 0, 0>       // 상대 이동량 설정
    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    A_CUR.T = 1                         // A_CUR의 툴 좌표계 번호 설정

    X_CUR = JTOB(A_CUR)                 // A_CUR을 사용자 좌표계 기준으로 변환
    X_TAR = RELTOOL(X_CUR, DELTA)       // X_CUR에서 DELTA만큼의 상대이동한 위치
                                        구함
    SVON                                // 서보 ON
    LMOV X_TAR, T = 1                   // X_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Related terms

RELBASE, RELUSER

10.11 RELUSER (사용자 좌표계 기준 상대이동)

Usage

기준 위치에서 대입된 이동량만큼 사용자 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.

Syntax

<변수> = RELUSER([<로봇 번호>,] <기준 위치>, <이동량>)

Description

사용자 좌표계 기준 위치로부터 지정한 사용자 좌표계 방향으로 상대 이동한 결과 값을 계산하여 사용자 좌표계 값으로 반환해 주는 함수입니다.
기준 위치의 사용자 좌표계 번호를 참조하여 사용자 좌표계 상대 이동을 진행하게 됩니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <기준 위치>에 값을 입력할 때 반드시 사용자 좌표계 위치형 변수 형태로 값을 입력해야 합니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <이동량>에 값을 입력할 때 반드시 위치형 변수 형태로 값을 입력해야 합니다. 상수를 입력하는 경우 알람이 발생합니다.

Return value

<변수>	Data type: <사용자 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<기준 위치>	Data type: <사용자 좌표계 위치형 변수>
<이동량>	Data type: <모든 좌표계 위치형 변수>

Examples

```
MAIN                                    // 메인 함수 선언
    POS A_CUR, DELTA                    // 조인트 좌표계 위치형 변수 선언
    UPOS U_TAR, U_CUR                   // 사용자 좌표계 위치형 변수 선언

    DELTA = <50, 0, 50, 90, 0, 0>       // 상대 이동량 설정
    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    A_CUR.T = 1                         // A_CUR의 툴 좌표계 번호 설정
    A_CUR.U = 1                         // A_CUR의 사용자 좌표계 번호 설정

    U_CUR = JTOU(A_CUR)                 // A_CUR을 사용자 좌표계 기준으로 변환
    U_TAR = RELUSER(U_CUR, DELTA)       // U_CUR에서 DELTA만큼의 상대이동한 위치
                                        구함
    SVON                                // 서보 ON
    LMOV U_TAR, T = 1, U = 1            // U_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP
                                        // 메인 함수 종료
```

Related terms

RELBASE, RELTOOL

10.12 GBASEDISP (베이스 좌표계 기준 이동량 검출)

Usage

입력된 두 좌표간 베이스 좌표계 기준의 이동량을 구하는 함수입니다.

Syntax

<변수> = GBASEDISP([<로봇 번호>,] <위치 1>, <위치 2>)

Description

입력된 <위치1>과 <위치2> 간의 이동량을 베이스 좌표계 기준 값으로 반환하는 함수입니다.
<위치2>는 <위치1>과 같은 타입의 위치형 변수여야 합니다. 또한, 베이스 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호가 동일해야 하며, 사용자 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호 및 사용자 좌표계 번호가 동일해야 합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <위치1>는 모든 위치형 변수 타입을 사용할 수 있습니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <위치2>가 <위치1>과 변수 타입 또는 좌표계 번호가 다를 경우 알람이 발생하므로 주의하시기 바랍니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<위치1>	Data type: <모든 좌표계 위치형 변수>
<위치2>	Data type: <모든 좌표계 위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS AP1, AP2                        // 조인트 좌표계 위치형 변수 선언
    XPOS DELTA                          // 베이스 좌표계 위치형 변수 선언
    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    AP1 = GP0                           // GP0를 AP1에 대입
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    AP2 = GP1                           // GP1을 AP2에 대입
    DELTA = GBASEDISP(AP1, AP2)         // AP1과 AP2간의 베이스 좌표계 이동량 구함
    PRINT(0, DELTA)                     // 모니터링 로그 0에 결과 값 출력
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    POS A_CUR                           // 조인트 좌표계 위치형 변수 선언
    XPOS XP1, XP2, X_CUR, X_TAR, DELTA  // 베이스 좌표계 위치형 변수 선언

    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    XP1 = JTOB(GP0)                     // GP0을 베이스 좌표계 기준으로 변환
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    XP2 = JTOB(GP1)                     // GP1을 베이스 좌표계 기준으로 변환
    DELTA = GBASEDISP(XP1, XP2)         // XP1, XP2의 베이스 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                     // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    A_CUR.T = 1                         // A_CUR의 툴 좌표계 번호 설정
    X_CUR = JTOB(A_CUR)                 // A_CUR을 베이스 좌표계 기준으로 변환
    X_TAR = RELBASE(X_CUR, DELTA)       // X_CUR에서 DELTA만큼의 상대이동한 위치
                                        구함
    SVON                                // 서보 ON
    LMOV X_TAR, T = 1                   // X_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Example 3

```
MAIN                                    // 메인 함수 선언
    POS A_CUR                           // 조인트 좌표계 위치형 변수 선언
    XPOS X_CUR, X_TAR, DELTA            // 베이스 좌표계 위치형 변수 선언
    UPOS UP1, UP2                       // 사용자 좌표계 위치형 변수 선언

    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    GP0.U = 1                           // GP0의 사용자 좌표계 번호 설정
    UP1 = JTOU(GP0)                     // GP0을 사용자 좌표계 기준으로 변환
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    GP1.U = 1                           // GP1의 사용자 좌표계 번호 설정
    UP2 = JTOU(GP1)                     // GP1을 사용자 좌표계 기준으로 변환
    DELTA = GBASEDISP(UP1, UP2)         // UP1,UP2의 베이스 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                     // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    A_CUR.T = 1                         // A_CUR의 툴 좌표계 번호 설정
    X_CUR = JTOB(A_CUR)                 // A_CUR을 베이스 좌표계 기준으로 변환
    X_TAR = RELBASE(X_CUR, DELTA)       // X_CUR에서 DELTA만큼의 상대이동한 위치
                                        구함
    SVON                                // 서보 ON
    LMOV X_TAR, T = 1                   // X_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Related terms

GTOOLDISP, GUSERDISP

10.13 GTOOLDISP (툴 좌표계 기준 이동량 검출)

Usage

입력된 두 좌표간 툴 좌표계 기준의 이동량을 구하는 함수입니다.

Syntax

<변수> = GTOOLDISP([<로봇 번호>,] <위치 1>, <위치 2>)

Description

입력된 <위치1>과 <위치2> 간의 이동량을 툴 좌표계 기준 값으로 반환하는 함수입니다.
<위치2>는 <위치1>과 같은 타입의 위치형 변수여야 합니다. 또한, 베이스 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호가 동일해야 하며, 사용자 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호 및 사용자 좌표계 번호가 동일해야 합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <위치1>는 모든 위치형 변수 타입을 사용할 수 있습니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <위치2>가 <위치1>과 변수 타입 또는 좌표계 번호가 다를 경우 알람이 발생하므로 주의하시기 바랍니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<위치1>	Data type: <모든 좌표계 위치형 변수>
<위치2>	Data type: <모든 좌표계 위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS AP1, AP2                        // 조인트 좌표계 위치형 변수 선언
    XPOS DELTA                          // 베이스 좌표계 위치형 변수 선언
    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    AP1 = GP0                           // GP0를 AP1에 대입
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    AP2 = GP1                           // GP1을 AP2에 대입
    DELTA = GTOOLDISP(AP1, AP2)         // AP1과 AP2간의 툴 좌표계 이동량 구함
    PRINT(0, DELTA)                     // 모니터링 로그 0에 결과 값 출력
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                        // 메인 함수 선언
    POS A_CUR                               // 조인트 좌표계 위치형 변수 선언
    XPOS XP1, XP2, X_CUR, X_TAR, DELTA      // 베이스 좌표계 위치형 변수 선언

    GP0.T = 1                               // GP0의 툴 좌표계 번호 설정
    GP0.U = 1                               // GP0의 사용자 좌표계 번호 설정
    XP1 = JTOB(GP0)                         // GP0을 베이스 좌표계 기준으로 변환
    GP1.T = 1                               // GP1의 툴 좌표계 번호 설정
    GP1.U = 1                               // GP1의 사용자 좌표계 번호 설정
    XP2 = JTOB(GP1)                         // GP1을 베이스 좌표계 기준으로 변환
    DELTA = GTOOLDISP(XP1, XP2)             // AP1과 AP2의 툴 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                         // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                            // A_CUR에 현재 위치 대입
    A_CUR.T = 1                             // A_CUR의 툴 좌표계 번호 설정
    X_CUR = JTOB(A_CUR)                     // A_CUR을 베이스 좌표계 기준으로 변환
    X_TAR = RELTOOL(X_CUR, DELTA)           // X_CUR에서 DELTA만큼의 상대이동한 위치
                                            구함
    SVON                                    // 서보 ON
    LMOV X_TAR, T = 1                       // X_TAR까지 LMOV
    SVOF                                    // 서보 OFF
EOP
                                            // 메인 함수 종료
```

Example 3

```
MAIN                                        // 메인 함수 선언
    POS A_CUR                               // 조인트 좌표계 위치형 변수 선언
    XPOS X_CUR, X_TAR, DELTA                // 베이스 좌표계 위치형 변수 선언
    UPOS UP1, UP2                           // 사용자 좌표계 위치형 변수 선언

    GP0.T = 1                               // GP0의 툴 좌표계 번호 설정
    GP0.U = 1                               // GP0의 사용자 좌표계 번호 설정
    UP1 = JTOU(GP0)                         // GP0을 사용자 좌표계 기준으로 변환
    GP1.T = 1                               // GP1의 툴 좌표계 번호 설정
    GP1.U = 1                               // GP1의 사용자 좌표계 번호 설정
    UP2 = JTOU(GP1)                         // GP1을 사용자 좌표계 기준으로 변환
    DELTA = GTOOLDISP(UP1, UP2)             // AP1과 AP2의 툴 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                         // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                            // A_CUR에 현재 위치 대입
    A_CUR.T = 1                             // A_CUR의 툴 좌표계 번호 설정
    X_CUR = JTOB(A_CUR)                     // A_CUR을 베이스 좌표계 기준으로 변환
    X_TAR = RELTOOL(X_CUR, DELTA)           // X_CUR에서 DELTA만큼의 상대이동한 위치
                                            구함
    SVON                                    // 서보 ON
    LMOV X_TAR, T = 1                       // X_TAR까지 LMOV
    SVOF                                    // 서보 OFF
EOP                                         // 메인 함수 종료
```

Related terms

GBASEDISP, GUSERDISP

10.14 GUSERDISP (사용자 좌표계 기준 이동량 검출)

Usage

입력된 두 좌표간 사용자 좌표계 기준의 이동량을 구하는 함수입니다.

Syntax

<변수> = GUSERDISP([<로봇 번호>,] <위치 1>, <위치 2>)

Description

입력된 <위치1>과 <위치2> 간의 이동량을 사용자 좌표계 기준 값으로 반환하는 함수입니다.
<위치2>는 <위치1>과 같은 타입의 위치형 변수여야 합니다. 또한, 베이스 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호가 동일해야 하며, 사용자 좌표계 위치형 변수를 대입하는 경우 <위치1>과 툴 좌표계 번호 및 사용자 좌표계 번호가 동일해야 합니다.

1) <로봇 번호>를 입력하여 JOB 프로그램 저장 시 선택한 로봇이 아닌 다른 로봇을 선택할 수 있습니다.
2) <위치1>는 모든 위치형 변수 타입을 사용할 수 있습니다. 상수 또는 그 외 다른 변수를 입력하는 경우 알람이 발생합니다.
3) <위치2>가 <위치1>과 변수 타입 또는 좌표계 번호가 다를 경우 알람이 발생하므로 주의하시기 바랍니다.

Return value

<변수>	Data type: <사용자 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<위치1>	Data type: <모든 좌표계 위치형 변수>
<위치2>	Data type: <모든 좌표계 위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS AP1, AP2                        // 조인트 좌표계 위치형 변수 선언
    UPOS DELTA                          // 사용자 좌표계 위치형 변수 선언
    GP0.U = 1                           // GP0의 사용자 좌표계 번호 설정
    AP1 = GP0                           // GP0를 AP1에 대입
    GP1.U = 1                           // G10의 사용자 좌표계 번호 설정
    AP2 = GP1                           // GP1을 AP2에 대입
    DELTA = GUSERDISP(AP1, AP2)         // GP0와 GP1간의 사용자 좌표계 이동량 구함
    PRINT(0, DELTA)                     // 모니터링 로그 0에 결과 값 출력
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    XPOS XP1, XP2                       // 베이스 좌표계 위치형 변수 선언
    UPOS U_CUR, U_TAR, DELTA            // 사용자 좌표계 위치형 변수 선언
    POS A_CUR                           // 조인트 좌표계 위치형 변수 선언

    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    GP0.U = 1                           // GP0의 사용자 좌표계 번호 설정
    XP1 = JTOB(GP0)                     // GP0을 베이스 좌표계 기준으로 변환
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    GP1.U = 1                           // GP1의 사용자 좌표계 번호 설정
    XP2 = JTOB(GP1)                     // GP1을 베이스 좌표계 기준으로 변환
    DELTA = GUSERDISP(XP1, XP2)         // XP1에서 XP2의 사용자 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                     // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    U_CUR = JTOU(A_CUR)                 // A_CUR을 베이스 좌표계 기준으로 변환
    U_TAR = RELUSER(U_CUR, DELTA)       // U_CUR에서 DELTA만큼의 상대이동한 위치 구함

    SVON                                // 서보 ON
    LMOV U_TAR, T = 1, U = 1            // U_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Example 3

```
MAIN                                    // 메인 함수 선언
    UPOS UP1,UP2, U_CUR, U_TAR, DELTA   // 사용자 좌표계 위치형 변수 선언
    POS A_CUR                           // 조인트 좌표계 위치형 변수 선언

    GP0.T = 1                           // GP0의 툴 좌표계 번호 설정
    GP0.U = 1                           // GP0의 사용자 좌표계 번호 설정
    UP1 = JTOU(GP0)                     // GP0을 사용자 좌표계 기준으로 변환
    GP1.T = 1                           // GP1의 툴 좌표계 번호 설정
    GP1.U = 1                           // GP1의 사용자 좌표계 번호 설정
    UP2 = JTOU(GP1)                     // GP1을 사용자 좌표계 기준으로 변환
    DELTA = GUSERDISP(UP1, UP2)         // UP1에서 UP2의 사용자 좌표계 기준 상대 이동량
    PRINT(0, DELTA)                     // 모니터링 로그 0번 인덱스에 결과 출력

    A_CUR = HERE                        // A_CUR에 현재 위치 대입
    U_CUR = JTOU(A_CUR)                 // A_CUR을 베이스 좌표계 기준으로 변환
    U_TAR = RELUSER(U_CUR, DELTA)       // U_CUR에서 DELTA만큼의 상대이동한 위치 구함

    SVON                                // 서보 ON
    LMOV U_TAR, T = 1, U = 1            // U_TAR까지 LMOV
    SVOF                                // 서보 OFF
EOP                                     // 메인 함수 종료
```

Related terms

GBASEDISP, GTOOLDISP

10.15 GET_TOOL_FRAME (툴 오프셋 출력)

Usage

지정한 툴의 오프셋 값을 출력하는 명령어 입니다.

Syntax

<문자열 변수> = GET_TOOL_FRAME(<로봇 번호>, <툴 번호>)

Description

지정한 툴의 오프셋 값을 출력합니다.

1) <로봇 번호>에 로봇 번호를 입력합니다.
2) <툴 번호>에 툴 번호를 입력합니다.

Return value

<변수>	Data type: <문자열 변수> 지정한 툴의 오프셋 값이 실수 형태의 문자열로 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.
<툴 번호>	Data type: <정수형 변수/상수> 툴 번호의 범위는 0 ~ 15 입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR T_OFFS                       // 오프셋 문자열 변수 선언
    INT R_CH                            // 로봇 번호 정수형 변수 선언
    INT T_N                             // 툴 번호 정수형 변수 선언

    R_CH = 1                            // 로봇 번호 정의
    T_N = 2                             // 툴 번호 정의

    T_OFFS = GET_TOOL_FRAME(R_CH, T_N)  // 툴 오프셋 출력
    PRINT(0, T_OFFS)                    // 모니터링 로그 0번 인덱스에 출력
EOP                                     // 메인 함수 종료
```

Related terms

SET_TOOL_FRAME

10.16 GET_USER_FRAME (사용자 좌표계 오프셋 출력)

Usage

지정한 사용자 좌표계의 오프셋 값을 출력하는 명령어 입니다.

Syntax

<문자열 변수> = GET_USER_FRAME(<로봇 번호>, <사용자 좌표계 번호>)

Description

지정한 사용자 좌표계의 오프셋 값을 출력합니다.

1) <로봇 번호>에 로봇 번호를 입력합니다.
2) <사용자 좌표계 번호>에 사용자 좌표계 번호를 입력합니다.

Return value

<변수>	Data type: <문자열 변수> 지정한 사용자 좌표계의 오프셋 값이 실수 형태의 문자열로 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.
<사용자 좌표계 번호>	Data type: <정수형 변수/상수> 사용자 좌표계 번호의 범위는 0 ~ 15 입니다.

Examples

```
MAIN                                        // 메인 함수 선언
    DEFSTR U_OFFS                           // 오프셋 문자열 변수 선언
    INT R_CH                                // 로봇 번호 정수형 변수 선언
    INT U_N                                 // 사용자 좌표계 번호 정수형 변수 선언

    R_CH = 1                                // 로봇 번호 정의
    U_N = 2                                 // 사용자 좌표계 번호 정의

    U_OFFS = GET_USER_FRAME(R_CH, U_N)      // 사용자 좌표계 오프셋 출력
    PRINT(0, U_OFFS)                        // 모니터링 로그 0번 인덱스에 결과 출력
EOP                                         // 메인 함수 종료
```

Related terms

SET_USER_FRAME

10.17 SET_TOOL_FRAME (툴 오프셋 입력)

Usage

지정한 툴에 오프셋 값을 입력하는 명령어 입니다.

Syntax

<정수형 변수> = SET_TOOL_FRAME(<로봇 번호>, <툴 번호>, <문자열 변수>)

Description

지정한 툴에 오프셋 값을 입력합니다.
SET_TOOL_FRAME 명령어는 로봇이 동작하지 않을 때에만 유효합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(정상) 또는 -1(오류)값이 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.
<툴 번호>	Data type: <정수형 변수/상수> 툴 번호의 범위는 1 ~ 15 입니다.
<문자열 변수>	Data type: <문자열> 또는 <문자열 변수> 문자열 변수는 순서대로 X,Y,Z,RX,RY,RZ 로 구성되어있습니다.

Examples

```
MAIN                                        // 메인 함수 선언
    DEFSTR T_OFFS                           // 오프셋 문자열 변수 선언
    INT R_CH, T_N                           // 로봇 번호,툴번호 정수형 변수 선언
    INT RES                                 // 오프셋 결과 정수형 변수 선언

    R_CH = 1                                // 로봇 번호 정의
    T_N = 3                                 // 툴 번호 정의
    T_OFFS = “-86,0,125,0,0,0”              // 툴 오프셋 값 정의

    RES = SET_TOOL_FRAME(R_CH, T_N, T_OFFS) // 툴 오프셋 생성
    IF RES == 0 THEN                        // 툴 좌표계 설정 값 이상 없을 경우
        SVON                                // 서보 ON
        JMOV GP0                            // GP0까지 JMOV
        LMOV GP1, T = T_N                   // GP1까지 LMOV
        SVOF                                // 서보 OFF
    ENDIF                                   // IF문 종료
EOP                                         // 메인 함수 종료
```

Related terms

GET_TOOL_FRAME

10.18 SET_USER_FRAME (사용자 오프셋 입력)

Usage

지정한 사용자 좌표계에 오프셋 값을 입력하는 명령어 입니다.

Syntax

<변수> = SET_USER_FRAME(<로봇 번호>, <사용자 좌표계 번호>, <문자열 변수>)

Description

지정한 사용자 좌표계에 오프셋 값을 입력합니다.
SET_USER_FRAME 명령어는 로봇이 동작하지 않을 때에만 유효합니다.

1) <로봇 번호>에 로봇 번호를 입력합니다.
2) <사용자 좌표계 번호>에 사용자 좌표계 번호를 입력합니다.
3) <문자열 변수>에 툴 오프셋으로 입력할 실수 형태의 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(정상) 또는 -1(오류)값이 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.
<사용자 좌표계 번호>	Data type: <정수형 변수/상수> 사용자 좌표계 번호의 범위는 1 ~ 15 입니다.
<문자열 변수>	Data type: <문자열> 또는 <문자열 변수> 문자열 변수는 순서대로 X,Y,Z,RX,RY,RZ 로 구성되어있습니다.

Examples

```
MAIN                                        // 메인 함수 선언
    DEFSTR U_OFFS                           // 오프셋 문자열 변수 선언
    INT R_CH, T_N                           // 로봇 채널, 툴 번호 변수 선언
    INT RES, U_N                            // 결과값, 사용자 좌표계 번호 선언
    UPOS UP                                 // 사용자 좌표계 위치형 변수 선언

    R_CH = 1                                // 로봇 채널 설정
    T_N = 1                                 // 툴 번호 설정
    U_N = 4                                 // 사용자 좌표계 번호 설정

    U_OFF = "400,0,-400,0,0,0"              // 사용자 좌표계 오프셋 문자열로 설정

    RES = SET_USER_FRAME(R_CH, U_N, U_OFF)  // 오프셋 문자열 입력으로 사용자 좌표
                                            계 생성
    GP1.U = U_N                             // GP1 의 사용자 좌표계 번호 설정
    GP1.T = T_N                             // GP1의 툴 좌표계 번호 설정
    UP = JTOU(GP1)                          // GP1을 사용자 좌표계 기준으로 변환

    IF RES == 0 THEN                        // 사용자 좌표계 설정값 이상 없을 경우
        SVON                                // 서보 ON
        JMOV GP0                            // GP0까지 JMOV
        LMOV UP, T = T_N, U = U_N           // UP까지 LMOV
        SVOF                                // 서보 OFF
    ENDIF                                   // IF문 종료
EOP                                         // 메인 함수 종료
```

Related terms

GET_USER_FRAME

10.19 READ_TOOL (TOOL 번호 읽기)

Usage

지정한 로봇에 사용 중인 툴 번호를 출력하는 명령어 입니다.

Syntax

<정수형 변수> = READ_TOOL(<로봇 번호>)

Description

지정한 로봇에 사용 중인 툴 번호를 출력합니다.

1) <로봇 번호>에 로봇 번호를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0 ~ 15 까지 툴 번호가 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.

Examples

```
MAIN                                // 메인 함수 선언
    INT T_N                         // 툴 번호 변수 선언
    TOOL 3                          // 사용할 툴 번호 선택
    T_N = READ_TOOL(1)              // 사용 중인 툴 번호 반환
    PRINT(0,T_N)                    // 모니터링 로그 0번 인덱스에 출력
EOP                                 // 메인 함수 종료
```

Related terms

TOOL, SET_TOOL_FRAME, GET_TOOL_FRAME

10.20 READ_USER (USER 번호 읽기)

Usage

지정한 로봇에 사용 중인 사용자 좌표계 번호를 출력하는 명령어 입니다.

Syntax

<정수형 변수> = READ_USER(<로봇 번호>)

Description

지정한 로봇에 사용 중인 사용자 좌표계 번호를 출력합니다.

1) <로봇 번호>에 로봇 번호를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0 ~ 15 까지 사용자 좌표계 번호가 반환됩니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> 로봇 번호의 범위는 1 ~ 3 입니다.

Examples

```
MAIN                                // 메인 함수 선언
    INT U_N                         // 사용자 좌표계 번호 변수 선언
    USER 3                          // 사용자 좌표계 번호 선택
    U_N = READ_USER(1)              // 사용 중인 사용자 좌표계 번호 반환
    PRINT(0,U_N)                    // 모니터링 로그 0번 인덱스에 출력
EOP                                 // 메인 함수 종료
```

Related terms

USER, SET_USER_FRAME, GET_USER_FRAME

10.21 ABS (절대값)

Usage

입력되는 값의 절대값을 반환하는 함수입니다.

Syntax

<변수> = ABS(<입력 값>)

Description

영점 위치에서부터 입력 값까지의 변위의 크기를 반환합니다.
즉, 정수형 데이터를 입력하는 경우 양의(+) 정수 값을 반환하며, 실수형 데이터 입력하는 경우 양의(+) 실수 값을 반환해 줍니다.

1) <입력 값>은 정수형 상수와 변수 또는 실수형 상수와 변수를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 또는 <실수형 변수> 변위의 크기를 반환합니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT AA,BB               // 정수형 변수 선언

    AA = -5                 // 정수형 변수 AA를 -5로 초기화
    I0 = ABS(-5)            // 전역 정수형 변수 I0에 5 저장
    BB = ABS(AA)            // 정수형 변수 BB에 5 저장
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL AA, BB             // 실수형 변수 선언

    AA = -1.5               // 실수형 변수 AA를 -1.5로 초기화
    F0 = ABS(-1.5)          // 전역 실수형 변수 F0에 1.5 저장
    BB = ABS(AA)            // 실수형 변수 BB에 1.5 저장
EOP                         // 메인 함수 종료
```

10.22 DEG (라디안 각을 각도 값으로 변환)

Usage

라디안 각(호도법)을 각도 값(육십분법)으로 변환하는 함수 입니다.

Syntax

<변수> = DEG(<입력 값>)

Description

라디안 각을 각도 값으로 환산 할 때 사용하는 함수 입니다.
아래 식과 같은 각도 변환 공식을 가집니다.

$$degree = (radian) \times \left(\frac{180^{\circ}}{\pi}\right)$$

1) <입력 값>은 라디안 각을 입력합니다.

Return value

<변수> Data type: <실수형 변수>
결과 값의 단위는 각도(°)입니다.

Arguments

<입력 값> Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 1              // 정수형 변수 AA에 라디안 각도(1)를 할당
    F0 = DEG(1)         // 정수 상수(1)를 각도 값(57.29,578)으로 변경하여 저장
    BB = DEG(AA)        // 정수 변수(AA)를 각도 값(57.29,578)으로 변경하여 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 3.1416         // 실수형 변수 AA에 라디안 각도(3.1,416)를 할당
    F0 = DEG(3.1416)    // 실수 상수(3.1416)를 각도 값(180)으로 변경하여 저장
    BB = DEG(AA)        // 실수형 변수(AA)를 각도 값(180)으로 변경하여 저장
EOP                     // 메인 함수 종료
```

Related terms

RAD

1) **주요각의 육십분법과 호도법**

육십분법	0°	30°	45°	60°	90°	180°	270°	360°
호도법	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3}{2}\pi$	2π

10.23 RAD (각도 값을 라디안 각으로 변환)

Usage

각도 값(육십분법)을 라디안 각(호도법)으로 변환하는 함수입니다.

Syntax

<변수> = RAD(<입력 값>)

Description

각도 값을 라디안 각으로 환산 할 때 사용하는 함수 입니다.
아래 식과 같은 라디안 각 변환 공식을 가집니다.

$$radian = (degree) \times \left(\frac{\pi}{180^{\circ}}\right)$$

1) <입력 값>은 각도 값(60분법)을 입력 합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 라디안(rad)입니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 180            // 정수형 변수 AA에 각도(180)를 할당
    F0 = RAD(180)       // 정수 상수(180)를 라디안 각도(3.14,159)로 변경하여 저장
    BB = RAD(AA)        // 정수 변수(AA)를 라디안 각도(3.14,159)로 변경하여 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 180.0          // 실수형 변수 AA에 각도(180.0)를 할당
    F0 = RAD(180.0)     // 실수 상수(180.0)를 라디안 각도(3.14,159)로 변경하여 저장
    BB = RAD(AA)        // 실수 변수(AA)를 라디안 각도(3.14,159)로 변경하여 저장
EOP                     // 메인 함수 종료
```

Related terms

DEG

1) **주요각의 육십분법과 호도법**

육십분법	0°	30°	45°	60°	90°	180°	270°	360°
호도법	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3}{2}\pi$	2π

10.24 POW (지수함수)

Usage

거듭제곱 값을 구할 때 사용하는 함수입니다.

Syntax

〈변수〉 = POW(〈밑수 값〉, 〈지수 값〉)

Description

거듭제곱(x^y)값을 계산하는 함수입니다.
예를 들어 2의 5승을 계산할 경우 $2^5 = 32$ 값을 반환합니다.

1) 〈밑수 값〉에 거듭제곱의 바탕이 되는 수를 입력합니다.
2) 〈지수 값〉에 거듭제곱의 지수 값을 입력합니다.

Return value

〈변수〉	Data type: 〈실수형 변수〉 거듭 제곱의 결과 값이 반환됩니다. 밑수 값이 0보다 작고 지수 값이 정수가 아닌 경우 NaN(Not-A-Number)을 반환합니다. 밑수 값이 0이고 지수 값이 0보다 작은 경우 무한대(Inf)를 반환합니다.

Arguments

〈밑수 값〉	Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉
〈지수 값〉	Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    INT AA, BB                // 정수형 변수 선언
    REAL CC                   // 실수형 변수 선언
    AA = 2                    // 밑수값을 2로 할당
    BB = 5                    // 지수값을 5로 할당
    F0 = POW(2, 5)            // 정수 상수(2^5)를 사용하여 F0에 저장
    CC = POW(AA, BB)          // 정수 변수(AA^BB)을 사용하여 CC에 저장
EOP                           // 메인 함수 종료
```

Example 2

```
MAIN                          // 메인 함수 선언
    REAL AA, BB, CC           // 실수형 변수 선언
    AA = 2.0                  // 밑수값을 2.0로 할당
    BB = 5.0                  // 지수값을 5.0로 할당
    F0 = POW(2.0, 5.0)        // 실수 상수(2^5)를 사용하여 F0에 저장
    CC = POW(AA, BB)          // 실수 변수(AA^BB)을 사용하여 CC에 저장
EOP                           // 메인 함수 종료
```

Related terms

SQRT

10.25 RND (반올림)

Usage

소수부를 반올림한 후에 정수 값으로 반환합니다.

Syntax

<변수> = RND(<입력 값>)

Description

소수점 이하를 반올림한 후에 정수 형태로 반환합니다.
예를 들어 12.8을 입력한 경우 13을 반환합니다.
아래 식과 같은 변환 공식을 가집니다.

$$result = (input\ \ value) + 0.5$$

1) <입력 값>에 반올림을 수행할 대상을 입력합니다.

Return value

<변수> Data type: <정수형 변수>
반올림을 수행한 결과 값을 반환합니다.

Arguments

<입력 값> Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA, BB          // 정수형 변수 선언
    AA = 12             // 정수형 변수에 12 할당
    I0 = RND(12)        // 정수 상수(12)을 입력할 경우 12 반환
    BB = RND(AA)        // 정수 변수(12)을 입력할 경우 12 반환
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA             // 실수형 변수 선언
    INT BB              // 정수형 변수 선언
    AA = 12.5           // 실수 변수에 12.5 할당
    I0 = RND(12.5)      // 실수 상수(12.5)을 입력 시 13 반환
    BB = RND(AA)        // 실수 변수(AA = 12.5)을 입력 시 13 반환
EOP                     // 메인 함수 종료
```

Example 3

```
MAIN                    // 메인 함수 선언
    REAL AA             // 실수형 변수 선언
    AA = RND(49.7/0.125)// 입력된 식의 결과 값의 반올림 값을 반환
EOP                     // 메인 함수 종료
```

10.26 EXP (지수 e^x)

Usage

무리수 e^x의 거듭제곱 값을 반환합니다. 로그 함수의 역함수 입니다.

Syntax

〈변수〉 = EXP(〈입력 값〉)

Description

무리수는 $e = 2.7182,818,284,590,452,353,602,874,713 \ldots$ 이며, e^x 의 결과값을 반환합니다.

예를 들어, $\mathrm{EXP}(1) = 2.718,282$ 일 경우
$\mathrm{EXP}(2) = 2.718,282 * 2.718,282 = 7.389,056$ 이며
$\mathrm{EXP}(3) = 2.718,282 * 2.718,282 * 2.718,282 = 20.08,554$ 입니다.

1) 〈입력 값〉에 무리수 e^x의 지수 x값을 입력합니다.

Return value

〈변수〉	Data type: 〈실수형 변수〉 무리수 e^x 의 결과값을 반환합니다.

Arguments

〈입력 값〉	Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 2              // 정수형 변수에 지수 값 2 할당
    F0 = EXP(2)         // e^2의 결과값을 전역 변수 F0에 저장
    BB = EXP(AA)        // e^(AA = 2)의 결과값을 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 2.0            // 실수형 변수에 지수 값 2 할당
    F0 = EXP(2.0)       // e^2.0의 결과값을 전역 변수 F0에 저장
    BB = EXP(AA)        // e^(AA = 2.0)의 결과값을 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Related terms

POW, LN, LOG

10.27 LN (자연로그)

Usage

자연로그 $\log_e x$의 결과 값을 반환합니다.

Syntax

<변수> = LN(<입력 값>)

Description

자연로그(Natural Logarithm)값을 구하는 함수 입니다.
예를 들어, $\mathrm{LN}(5) = 1.704{,}748$ 입니다.
아래 식과 같은 변환 공식을 가집니다.

$$y = \ln x = \log_e x$$

1) <입력 값>에 자연로그 식의 입력 값 x값을 입력합니다.

Return value

<변수>	Data type: <실수형 변수> 결과값 y를 반환합니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 15             // 정수형 변수에 15 할당
    F0 = LN(15)         // log_e x 값을(2.708) 전역 변수 F0에 저장
    BB = LN(AA)         // log_e x 값을(2.708) 실수 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 15.0           // 실수형 변수에 지수 값 15.0 할당
    F0 = LN(15.0)       // log_e x 값을(2.708) 전역 변수 F0에 저장
    BB = LN(AA)         // log_e x 값을(2.708) 실수 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Related terms

POW, EXP, LOG

10.28 LOG (상용로그)

Usage

상용로그 $\log_{10} x$의 결과 값을 반환합니다.

Syntax

<변수> = LOG(<입력 값>)

Description

상용로그(Common Logarithm)값을 구하는 함수 입니다.
예를 들어, $\mathrm{LOG}(10) = 1.0$, $\mathrm{LOG}(100) = 2.0$, $\mathrm{LOG}(1000) = 3.0$ 입니다.
아래 식과 같은 변환 공식을 가집니다.

$$y = \log_{10} x$$

1) <입력 값> 상용로그 식의 입력 값 x값을 입력합니다.

Return value

<변수>	Data type: <실수형 변수> 결과값 y를 반환합니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 100            // 정수형 변수에 100 할당
    F0 = LOG(100)       // log10 x의 결과값(2.0) 변수 F0에 저장
    BB = LOG(AA)        // log10 x의 결과값(2.0) 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 100.0          // 실수형 변수에 지수 값 100.0 할당
    F0 = LOG(100.0)     // log10 x의 결과값(2.0) 변수 F0에 저장
    BB = LOG(AA)        // log10 x의 결과값(2.0) 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Related terms

POW, EXP, LN

10.29 SQRT (제곱근)

Usage

제곱근을 반환합니다.

Syntax

〈변수〉 = SQRT(〈입력 값〉)

Description

입력 값(x)의 제곱근(y)을 반환합니다.
예를 들어, $\mathrm{SQRT}(1) = 1.00{,}000$, $\mathrm{SQRT}(2) = 1.41{,}421$, $\mathrm{SQRT}(3) = 1.73{,}205$, $\mathrm{SQRT}(4) = 2.00{,}000$ 입니다.
아래 식과 같은 변환 공식을 가집니다.

$y = \sqrt{x}$

1) 〈입력 값〉 제곱근 식의 입력 값 x를 입력합니다.

Return value

〈변수〉	Data type: 〈실수형 변수〉 결과값 y를 반환합니다. 입력 값이 음수인 경우 NaN(Not-A-Number)을 반환합니다.

Arguments

〈입력 값〉	Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 4              // 정수형 변수에 제곱근 입력값 4 할당
    F0 = SQRT(4)        // √4의 결과값 2를 전역 변수 F0에 저장
    BB = SQRT(AA)       // √(AA = 4)의 결과를 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 4.0            // 실수형 변수에 제곱근 입력값 4.0 할당
    F0 = SQRT(4.0)      // √4.0의 결과값 2를 전역 변수 F0에 저장
    BB = SQRT(AA)       // √(AA = 4.0)의 결과를 변수 BB에 저장
EOP                     // 메인 함수 종료
```

Related terms

POW

10.30 SIN (사인함수)

Usage

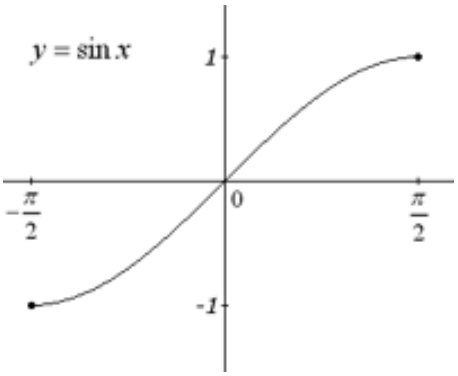
사인함수의 결과 값을 반환합니다.

Syntax

<변수> = SIN(<입력 값>)

Description

$\sin(x)$ 함수의 결과 값 y를 반환합니다.
$y = \sin(x)$



예를 들어, 삼각함수 특수각의 경우
$\mathrm{SIN}(\mathrm{RAD}(0)) = 0.0$,
$\mathrm{SIN}(\mathrm{RAD}(30)) = 0.5$,
$\mathrm{SIN}(\mathrm{RAD}(45)) = 0.70{,}711$,
$\mathrm{SIN}(\mathrm{RAD}(60)) = 0.86{,}603$,
$\mathrm{SIN}(\mathrm{RAD}(90)) = 1$ 입니다.

1) <입력 값>에 사인함수 식의 x값을 입력합니다. (x 단위: $radian$)

Return value

<변수>	Data type: <실수형 변수> 결과값 y를 반환합니다. 결과 값의 범위는 $-1 \le y \le 1$ 입니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 30             // 정수형 변수에 특수각(30)을 할당
    F0 = SIN(RAD(30))   // sin(30°)의 결과값 0.5를 F0에 저장
    BB = SIN(RAD(AA))   // 결과값 0.5를 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 30.0           // 실수형 변수에 특수각(30)을 할당
    F0 = SIN(RAD(30.0)) // sin(30°)의 결과값 0.5를 F0에 저장
    BB = SIN(RAD(AA))   // 결과값 0.5를 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Related terms

ASIN, COS, ACOS, TAN, ATAN, ATAN2

10.31 ASIN (아크 사인함수)

Usage

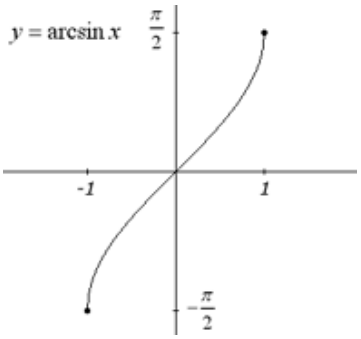
아크 사인함수의 결과 값을 반환합니다.

Syntax

〈변수〉 = ASIN(〈입력 값〉)

Description

$\text{asin}(x)$ 함수는 $\sin(x)$ 함수의 역함수이며, $-\frac{\pi}{2} \sim \frac{\pi}{2}$ 라디안 범위에 있는 asin값을 반환합니다.
기본적으로 x가 -1보다 작거나 1보다 큰 경우 asin는 무한대 값을 반환 합니다.



예를 들어,
$\text{ASIN}(-1) = -1.5708$,
$\text{ASIN}(0) = 0.0$,
$\text{ASIN}(1) = 1.5708$ 입니다.
1) 〈입력 값〉에 아크 사인함수 식의 x값을 입력합니다. $(-1 \le x \le 1)$

Return value

〈변수〉 Data type: 〈실수형 변수〉
결과값 y을 반환합니다. 결과 값의 범위는 $-\frac{\pi}{2} \le y \le \frac{\pi}{2}$ 입니다.

Arguments

〈입력 값〉 Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉
입력 값의 범위는 $-1 \le x \le 1$ 입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 1              // 정수형 변수에 입력값 1 할당
    F0 = ASIN(1)        // asin(1)의 결과값 1.5708를 F0에 저장
    BB = ASIN(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 1.0            // 실수형 변수에 입력값 1.0 할당
    F0 = ASIN(1.0)      // asin(1)의 결과값 1.5708를 F0에 저장
    BB = ASIN(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Related terms

SIN, COS, ACOS, TAN, ATAN, ATAN2

10.32 COS (코사인함수)

Usage

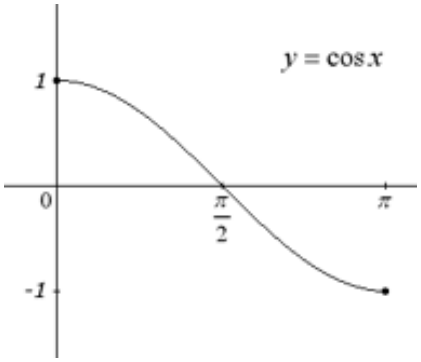
코사인함수의 결과 값을 반환합니다.

Syntax

<변수> = COS(<입력 값>)

Description

$\cos(x)$ 함수의 결과 값을 반환합니다.
$y = \cos(x)$



예를 들어, 삼각함수 특수각의 경우
$\mathrm{COS}(\mathrm{RAD}(0)) = 1.0,$
$\mathrm{COS}(\mathrm{RAD}(30)) = 0.86{,}603,$
$\mathrm{COS}(\mathrm{RAD}(45)) = 0.70{,}711,$
$\mathrm{COS}(\mathrm{RAD}(60)) = 0.5.$
$\mathrm{COS}(\mathrm{RAD}(90)) = 0.0$ 입니다.

1) <입력 값>에 코사인함수 식의 x값을 입력합니다. (x 단위: radian)

Return value

<변수>	Data type: <실수형 변수> 결과값 y를 반환합니다. 결과 값의 범위는 $-1 \le y \le 1$ 입니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT AA                  // 정수형 변수 선언
    REAL BB                 // 실수형 변수 선언
    AA = 60                 // 정수형 변수에 특수각(60)을 할당
    F0 = COS(RAD(60))       // cos(60°)의 결과값 0.5를 F0에 저장
    BB = COS(RAD(AA))       // 결과값 0.5를 실수형 변수 BB에 할당
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL AA, BB             // 실수형 변수 선언
    AA = 60.0               // 실수형 변수에 특수각(60)을 할당
    F0 = COS(RAD(60.0))     // cos(60°)의 결과값 0.5를 F0에 저장
    BB = COS(RAD(AA))       // 결과값 0.5를 실수형 변수 BB에 할당
EOP                         // 메인 함수 종료
```

Related terms

SIN, ASIN, ACOS, TAN, ATAN, ATAN2

10.33 ACOS (아크 코사인함수)

Usage

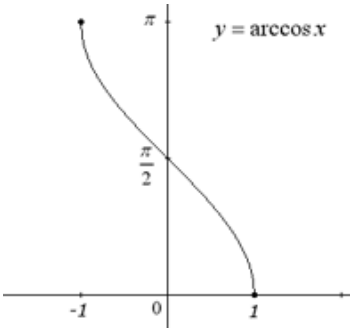
아크 코사인함수의 결과 값을 반환합니다.

Syntax

〈변수〉 = ACOS(〈입력 값〉)

Description

$\text{acos}(x)$ 함수는 $\cos(x)$함수의 역함수이며, $0 \sim \pi$ 라디안 범위에 있는 acos값을 반환합니다. 기본적으로 x가 -1보다 작거나 1보다 큰 경우 acos는 무한대 값을 반환합니다.



예를 들어,
$\text{ACOS}(-1) = 3.14{,}159.$
$\text{ACOS}(0) = 1.5708.$
$\text{ACOS}(1) = 0.0$ 입니다.

1) 〈입력 값〉에 아크 코사인함수 식의 x값을 입력합니다. $(-1 \le x \le 1)$

Return value

〈변수〉 Data type: 〈실수형 변수〉
결과값 y을 반환합니다. 결과 값의 범위는 $0 \le y \le \pi$ 입니다.

Arguments

〈입력 값〉 Data type: 〈정수형 변수/상수〉 또는 〈실수형 변수/상수〉
입력 값의 범위는 $-1 \le x \le 1$ 입니다.

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 0              // 정수형 변수에 입력값 0 할당
    F0 = ACOS(0)        // acos(0)의 결과값 1.5708를 F0에 저장
    BB = ACOS(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 0.0            // 실수형 변수에 입력값 0.0 할당
    F0 = ACOS(0.0)      // acos(0)의 결과값 1.5708를 F0에 저장
    BB = ACOS(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Related terms

SIN, ASIN, COS, TAN, ATAN, ATAN2

10.34 TAN (탄젠트함수)

Usage

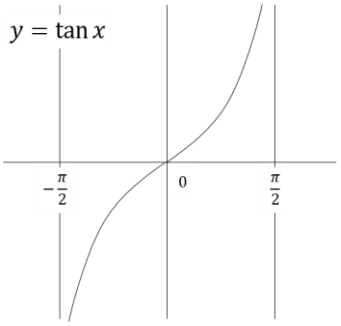
탄젠트함수의 결과 값을 반환합니다.

Syntax

<변수> = TAN(<입력 값>)

Description

$\tan(x)$ 함수의 결과 값을 반환합니다.
$y = \tan(x)$
예를 들어, 삼각함수 특수각의 경우
$\mathrm{TAN}(\mathrm{RAD}(0)) = 0.0,\ \mathrm{TAN}(\mathrm{RAD}(30)) = 0.57{,}735,$
$\mathrm{TAN}(\mathrm{RAD}(45)) = 1.0,\ \mathrm{TAN}(\mathrm{RAD}(60)) = 1.73{,}205,$
$\mathrm{TAN}(\mathrm{RAD}(90)) = *$ (표현 범위초과)입니다.
1) <입력 값>에 탄젠트함수 식의 x값을 입력합니다.
(x 단위: $radian$)



Return value

<변수>	Data type: <실수형 변수> 결과값 y를 반환합니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT AA                  // 정수형 변수 선언
    REAL BB                 // 실수형 변수 선언
    AA = 45                 // 정수형 변수에 특수각(45)을 할당
    F0 = TAN(RAD(45))       // tan(45°)의 결과값 1.0을 F0에 저장
    BB = TAN(RAD(AA))       // 결과값 1.0을 실수형 변수 BB에 할당
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL AA, BB             // 실수형 변수 선언
    AA = 45.0               // 실수형 변수에 특수각(45)을 할당
    F0 = TAN(RAD(45.0))     // tan(45°)의 결과값 1.0을 F0에 저장
    BB = TAN(RAD(AA))       // 결과값 1.0을 실수형 변수 BB에 할당
EOP                         // 메인 함수 종료
```

Related terms

SIN, ASIN, COS, ACOS, ATAN, ATAN2

1) $\tan(90^{\circ})$와 같이 무한대로 발산하는 경우 데이터 값이 크기 때문에 제어기 상에 데이터 값을 정량적으로 표기하기 어려우므로 Teaching Pendant에 “*”로 표기 됩니다. 이점 주의 하시기 바랍니다.

10.35 ATAN (아크 탄젠트함수)

Usage

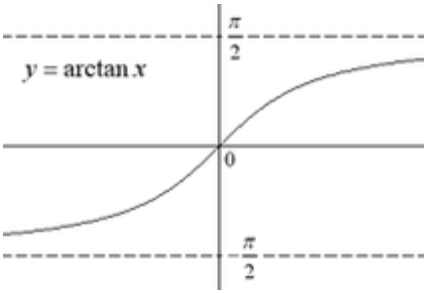
아크 탄젠트함수의 결과 값을 반환합니다.

Syntax

<변수> = ATAN(<입력 값>)

Description

$\text{atan}(x)$ 함수는 $\tan(x)$함수의 역함수이며 결과 값을 반환합니다.



예를 들어,
$\text{ATAN}(-1) = -0.7854$, $\text{ATAN}(0) = 0.0$,
$\text{ATAN}(1) = 0.7854$ 입니다.

1) <입력 값>에 아크 탄젠트 함수 식의 x값을 입력합니다. (x 단위: $radian$)

Return value

<변수> Data type: <실수형 변수>
결과값 y을 반환합니다. 결과 값의 범위는 $-\frac{\pi}{2} \le y \le \frac{\pi}{2}$ 입니다.

Arguments

<입력 값> Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA              // 정수형 변수 선언
    REAL BB             // 실수형 변수 선언
    AA = 1              // 정수형 변수에 1을 할당
    F0 = ATAN(1)        // atan(1)의 결과값 0.7854을 F0에 저장
    BB = ATAN(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB         // 실수형 변수 선언
    AA = 1.0            // 실수형 변수에 1.0을 할당
    F0 = ATAN(1.0)      // atan(1)의 결과값 0.7854을 F0에 저장
    BB = ATAN(AA)       // 결과값을 실수형 변수 BB에 할당
EOP                     // 메인 함수 종료
```

Related terms

SIN, ASIN, COS, ACOS, TAN, ATAN2

10.36 ATAN2(제2 아크 탄젠트함수)

Usage

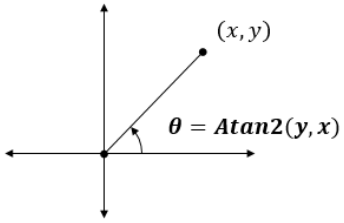
제2 아크 탄젠트함수의 결과 값을 반환합니다.

Syntax

<반환 값> = ATAN2(<Y 비율>, <X 비율>)

Description

$\text{atan2}(x)$ 함수의 결과 값을 반환합니다.
$y = \text{atan2}(x)$
예를 들어,
$\text{ATAN2}(1.0, 1.0) = 0.7854$,
$\text{ATAN2}(2.0, 1.0) = 1.10{,}715$,
$\text{ATAN2}(1.0, 2.0) = 0.46{,}365$ 입니다.



1) <Y 비율>에 직각 삼각형의 높이(y축) 값을 입력합니다.
2) <X 비율>에 직각 삼각형의 너비(x축) 값을 입력합니다.

Return value

<변수> Data type: <실수형 변수>
결과값 y을 반환합니다. 결과 값의 범위는 $-\pi \leq y \leq \pi$ 입니다.

Arguments

<Y 비율> Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

<X 비율> Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                    // 메인 함수 선언
    INT AA, BB          // 정수형 변수 선언
    REAL CC             // 실수형 변수 선언
    AA = 1              // 정수형 변수에 1을 할당
    BB = 1              // 정수형 변수에 1을 할당
    F0 = ATAN2(1,1)     // atan2(1,1)의 결과값 0.7854을 저장
    CC = ATAN2(AA,BB)   // 결과값을 실수형 변수 CC에 할당
EOP                     // 메인 함수 종료
```

Example 2

```
MAIN                    // 메인 함수 선언
    REAL AA, BB,CC      // 실수형 변수 선언
    AA = 1.0            // 실수형 변수에 1.0을 할당
    BB = 1.0            // 실수형 변수에 1.0을 할당
    F0 = ATAN2(1.0,1.0) // atan2(1.0,1.0)의 결과값 0.7854을 저장
    CC = ATAN2(AA,BB)   // 결과값을 실수형 변수 CC에 할당
EOP                     // 메인 함수 종료
```

Related terms

SIN, ASIN, COS, ACOS, TAN, ATAN

10.37 MIN (최솟값)

Usage

입력되는 두 개의 값을 비교하여 최솟값을 반환합니다.

Syntax

<변수> = MIN(<입력 값1>, <입력 값2>)

Description

입력 값1과 입력 값2를 비교하여 최솟값을 실수형 타입으로 반환합니다.

1) <입력 값1> 최솟값을 비교하기 위한 값1을 입력합니다.
2) <입력 값2> 최솟값을 비교하기 위한 값2를 입력합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값은 두 입력 값을 비교하여 최솟값을 반환합니다.

Arguments

<입력 값1>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>
<입력 값2>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    INT AA, BB              // 정수형 변수 선언
    REAL CC                 // 실수형 변수 선언
    AA = 2                  // 정수형 비교 변수1에 2을 할당
    BB = 5                  // 정수형 비교 변수2에 5을 할당
    F0 = MIN(2,5)           // 반환된 최솟값 2.0을 F0에 저장
    CC = MIN(AA,BB)         // 최솟값 2.0을 실수형 변수 CC에 저장
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    REAL AA, BB, CC         // 실수형 변수 선언
    AA = 2.0                // 실수형 비교 변수1에 2.0을 할당
    BB = 2.5                // 실수형 비교 변수2에 2.5을 할당
    F0 = MIN(2.0,2.5)       // 반환된 최솟값 2.0을 F0에 저장
    CC = MIN(AA,BB)         // 최솟값 2.0을 실수형 변수 CC에 저장
EOP                         // 메인 함수 종료
```

Related terms

MAX

10.38 MAX (최댓값)

Usage

입력되는 두 개의 값을 비교하여 최댓값을 반환합니다.

Syntax

<변수> = MAX(<입력 값1>, <입력 값2>)

Description

입력 값1과 입력 값2를 비교하여 최댓값을 실수형 타입으로 반환합니다.

1) <입력 값1> 최댓값을 비교하기 위한 값1을 입력합니다.
2) <입력 값2> 최댓값을 비교하기 위한 값2를 입력합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값은 두 입력 값을 비교하여 최댓값을 반환합니다.

Arguments

<입력 값1>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>
<입력 값2>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    INT AA, BB                // 정수형 변수 선언
    REAL CC                   // 실수형 변수 선언
    AA = 2                    // 정수형 비교 변수1에 2을 할당
    BB = 5                    // 정수형 비교 변수2에 5을 할당
    F0 = MAX(2,5)             // 반환된 최댓값 5.0을 F0에 저장
    CC = MAX(AA,BB)           // 최댓값 5.0을 실수형 변수 CC에 저장
EOP                           // 메인 함수 종료
```

Example 2

```
MAIN                          // 메인 함수 선언
    REAL AA, BB, CC           // 실수형 변수 선언
    AA = 2.0                  // 실수형 비교 변수1에 2.0을 할당
    BB = 2.5                  // 실수형 비교 변수2에 2.5을 할당
    F0 = MAX(2.0,2.5)         // 반환된 최댓값 2.5을 F0에 저장
    CC = MAX(AA,BB)           // 최댓값 2.5을 실수형 변수 CC에 저장
EOP                           // 메인 함수 종료
```

Related terms

MIN

10.39 RAND (난수 생성)

Usage

난수 값을 생성합니다.

Syntax

<변수> = RAND()

Description

난수 함수는 0부터 RAND_MAX 사이의 난수를 생성합니다. 32bit 리눅스 시스템에서 RAND_MAX값은 2,147,483,647입니다. 즉, 정수형 변수의 양의 최댓값입니다

- 1부터 10 사이의 난수 생성 방법: RAND()%10+1
- 1부터 100 사이의 난수 생성 방법: RAND()%100+1
- 1부터 1,000 사이의 난수 생성 방법: RAND()%1000+1

Return value

<변수>	Data type: <정수형 변수> 결과 값의 범위는 0 ~ RAND_MAX까지 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    REAL AA, BB                         // 실수형 변수 선언

    WHILE 1                             // WHILE ~ ENDWL 블록을 무한 실행
        AA = RAND()                     // 랜덤 값(0 ~ 7FFFFFFF)을 AA에 저장
        BB = AA%100+1                   // 1 ~ 100 사이 랜덤 값을 생성하여 BB에 저장
        DLAY 100                        // 시간지연 0.1초 지연
        PRINT(0,AA)                     // 모니터링 로그 0번 인덱스에 AA 값 출력
        PRINT(1,BB)                     // 모니터링 로그 1번 인덱스에 BB 값 출력
    ENDWL                               // WHILE문 종료
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    INT AA                              // 정수형 변수 선언

    AA = ((RAND())%100)+1               // 1 ~ 100 사이 랜덤 값을 생성하여 AA에 저장
EOP                                     // 메인 함수 종료
```

10.40 ASC (문자열 첫 문자의 문자 코드 반환)

Usage

문자열의 첫 문자에 해당 하는 문자 코드의 정수 값을 반환 합니다.

Syntax

<변수> = ASC(<문자열>)

Description

입력된 문자열의 첫 문자를 반환합니다

문자 코드(Character Code)란 문자에 해당하는 숫자(정수)를 말합니다.
예) 문자'A'는 정수(65)이고 문자 'B'는 정수(66)입니다.
아스키 코드표를 참조 하십시오.

1) <문자열>에 첫 문자를 검출하고자 하는 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값의 범위는 0 ~ 127까지 입니다.

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    I0 = ASC("STATEMENT")               // 첫번째 문자 "S"를 아스키 코드(83)로 변환
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR                          // 문자열 변수 선언
    INT VAL                             // 정수형 변수 선언

    STR = "STATEMENT"                   // 문자열 변수 STR에 "STATEMENT" 대입
    VAL = ASC(STR)                      // 첫번째 문자 "S"를 아스키 코드(83)로 변환
EOP                                     // 메인 함수 종료
```

Related terms

CHR

10.41 BINS (2진수 문자열 변환)

Usage

입력된 정수 값을 2진수 문자열로 변환하는 함수입니다.

Syntax

<변수> = BINS(<입력 값>)

Description

입력된 정수 값을 2진 변환 후 32 바이트 문자열로 반환합니다.

1) <입력 값>은 2진수 문자열로 변경할 정수 값을 입력합니다.

Return value

<변수>	Data type: <문자열 변수> 입력 값의 2진수 문자열을 반환합니다.

Arguments

<입력 값>	Data type: <정수형 변수/상수> -2^{31}(-2,147,483,648) ~ 2^{31}(2,147,483,647)까지 입력이 가능합니다.

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    DEFSTR STR1,STR2        // 문자열 변수 선언
    INT AA                  // 정수형 변수 선언

    AA = 16                 // 정수형 변수 초기화
    STR1 = BINS(16)         // 16의 2진수값에 해당하는 문자열을 STR 변수에
    STR2 = BINS(AA)         //“00000000000000000000000000010000”저장
EOP                         // 메인 함수 종료
```

Related terms

FTOS, HTOS, SVAL

10.42 CHR (정수 아스키 문자 변환)

Usage

입력된 정수 값에 해당하는 아스키 코드 문자를 반환합니다.

Syntax

<변수> = CHR(<입력 값>)

Description

입력된 정수에 해당하는 아스키 코드 문자를 반환합니다.

1) <입력 값>은 아스키 코드 값 내의 정수(0 ~ 127)를 입력합니다.

Return value

<변수>	Data type: <문자열 변수>

Arguments

<입력 값>	Data type: <정수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    DEFSTR STRA,STRB        // 문자열 변수 선언
    INT AA                  // 정수형 변수 선언

    AA = 66                 // 정수형 변수 초기화
    STRA = CHR(65)          // STRA에 아스키코드(65)에 해당하는 “A”할당
    STRB = CHR(AA)          // STRA에 아스키코드(66)에 해당하는 “B”할당
EOP                         // 메인 함수 종료
```

Related terms

ASC

10.43 FLUSH (통신 입출력 버퍼 클리어)

Usage

Ethernet/Serial 통신 입력, 출력 버퍼를 클리어하는 함수

Syntax

FLUSH <포트번호><버퍼 인덱스>

Description

통신 송수신 시 사용하는 입출력 버퍼를 클리어 합니다.

<포트번호>의 범위는 0 ~ 65,535이며 그 의미는 아래와 같습니다.
- 0 : RS-232 시리얼 통신 포트
- 10 ~ 12 : TCP/IP Server 포트
  (10 : 포트번호[38201], 11 : 포트번호[38202], 12 : 포트번호[38203])
- ~ 65,535: TCP/IP Client 포트

<버퍼 인덱스>의 범위는 1 ~ 2 이며 그 의미는 아래와 같습니다.
- 1: 입력 버퍼 클리어
- 2: 출력 버퍼 클리어
- 3: 입력, 출력 버퍼 클리어

Arguments

<포트번호>	Data type: <정수형 변수/상수> <버퍼 인덱스>의 범위는 0 ~ 65,535입니다.
<버퍼 인덱스>	Data type: <정수형 변수/상수> <버퍼 인덱스>의 범위는 1 ~ 3입니다.

Examples

Example 1

```
MAIN                                // 메인 함수 선언
    DEFSTR STR                      // 문자열 변수 선언

    STR = STRIN(0,1000)             // RS-232 시리얼 통신 문자열 수신(수신 대기 1초)
    FLUSH 0 1                       // RS-232 입력 버퍼 클리어
EOP                                 // 메인 함수 종료
```

Example 2

```
MAIN                                // 메인 함수 선언
    DEFSTR STR                      // 문자열 변수 선언
    INT AA                          // 정수형 변수 선언

    AA = 10                         // 정수형 변수 초기화
    STR = STRIN(10,1000)            // TCP/IP Server 통신 문자열 수신(수신 대기 1초)
    FLUSH AA 1                      // TCP/IP Server 입력 버퍼 클리어
EOP                                 // 메인 함수 종료
```

10.44 FTOS (정수/실수 값을 문자열로 변환)

Usage

정수나 실수 값을 문자열로 변환합니다.

Syntax

<변수> = FTOS(<입력 값>)

Description

정수나 실수 수치 데이터를 문자열로 변환합니다.

예) FTOS(1234) → "1234"
예) FTOS(-123.45) → "-123.45"

Return value

<변수>	Data type: <문자열 변수>

Arguments

<입력 값>	Data type: <정수형 변수/상수> 또는 <실수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    DEFSTR STR1,STR2        // 문자열 변수 선언
    INT AA                  // 정수형 변수 선언

    AA = 12345              // 정수형 변수 초기화
    STR1 = FTOS(1234)       // 1234를 문자열 "1234"로 변환 후 저장
    STR2 = FTOS(AA)         // 12345를 문자열 "12345"로 변환 후 저장
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                        // 메인 함수 선언
    DEFSTR STR1,STR2        // 문자열 변수 선언
    REAL AA                 // 실수형 변수 선언

    AA = -123.45            // 실수형 변수 초기화
    STR1 = FTOS(-12.345)    // -12.345를 문자열 "-12.345"로 변환 후
                            //저장
    STR2 = FTOS(AA)         // -123.45를 문자열 "-123.45"로 변환 후
                            //저장
EOP                         // 메인 함수 종료
```

Related terms

BINS, HTOS, SVAL

10.45 HTOS (정수를 16진수 문자열로 변환)

Usage

정수를 16진수의 문자열로 변환합니다.

Syntax

<변수> = HTOS(<입력 값>)

Description

입력된 정수를 16진수 문자열로 변환합니다.

예)HTOS(10) → “A”
예)HTOS(11) → “B”
예)HTOS(12) → “C”
예)HTOS(13) → “D”
예)HTOS(14) → “E”
예)HTOS(15) → “F”

Return value

<변수> Data type: <문자열 변수>.

Arguments

<입력 값> Data type: <정수형 변수/상수>

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    DEFSTR STR1,STR2        // 문자열 변수 선언
    INT AA                  // 정수형 변수 선언

    AA = 11                 // 정수형 변수 초기화
    STR1 = HTOS(10)         // 정수 10을 16진수 문자열(“A”)로 변환
    STR2 = HTOS(AA)         // 정수 11을 16진수 문자열(“B”)로 변환
EOP                         // 메인 함수 종료
```

Related terms

BINS, FTOS, SVAL

10.46 SLEFT (좌측 문자열 추출)

Usage

입력된 문자열의 좌측 문자열을 추출합니다.

Syntax

<변수> = SLEFT(<문자열>, <추출 개수>)

Description

입력된 문자열을 왼쪽에서부터 입력된 정수만큼 추출합니다.

1) <문자열>은 문자열을 추출할 대상입니다.
2) <추출 개수>는 문자열의 좌측에서부터 추출할 문자열 개수를 의미합니다.

Return value

<변수>	Data type: <문자열 변수> 선택한 문자열 개수만큼 추출 된 문자열을 반환합니다.

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>
<추출 개수>	Data type: <정수형 변수/상수>

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    DEFSTR STR0,STR1,STR2     // 문자열 변수 선언
    INT AA                    // 정수형 변수 선언

    STR0 = “TEST”             // 문자열 변수 “TEST” 할당
    AA = 2                    // 정수형 변수 초기화
    STR1 = SLEFT(“TEST”, 1)   // STR1 문자열 변수에 “T” 저장
    STR2 = SLEFT(STR0, AA)    // STR2 문자열 변수에 “TE” 저장
EOP                           // 메인 함수 종료
```

Example 2

```
MAIN                          // 메인 함수 선언
    DEFSTR STR0,STR1,STR2     // 문자열 변수 선언
    REAL AA                   // 실수형 변수 선언

    STR0 = “TEST”             // 문자열 변수 “TEST” 할당
    AA = 4.0                  // 실수형 변수 초기화
    STR1 = SLEFT(“TEST”, 3.0) // STR1 문자열 변수에 “TES” 저장
    STR2 = SLEFT(STR0, AA)    // STR2 문자열 변수에 “TEST” 저장
EOP                           // 메인 함수 종료
```

Related terms

SLEN, SMID, SPOS, SRIGHT

10.47 SLEN (문자열 길이 반환)

Usage

입력된 문자열의 길이를 반환합니다.

Syntax

<변수> = SLEN(<문자열>)

Description

입력된 문자열의 길이를 반환합니다.

1) <문자열>에 길이 값을 측정할 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 입력된 문자열 길이를 반환합니다.

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR1                         // 문자열 변수 선언
    INT AA                              // 정수형 변수 선언

    STR1 = “STATEMENT”                  // 문자열 변수 “STATEMENT” 할당
    I0 = SLEN(“STATEMENT”)              // 전역 변수 I0에 문자열 개수 저장
    AA = SLEN(STR1)                     // 내부 변수 AA에 문자열 개수 저장
EOP                                     // 메인 함수 종료
```

Related terms

SLEFT, SMID, SPOS, SRIGHT

10.48 SMID (지정 위치 문자열 추출)

Usage

문자열 지정위치부터 입력된 값만큼 문자열 추출합니다.

Syntax

<변수> = SMID(<문자열>, <시작 위치>, <추출 개수>)

Description

지정위치에서부터 사용자 입력 정수만큼 문자를 추출합니다. 문자열의 처음 위치는 0 부터 시작하여 1씩 증가합니다.

1) <문자열> 문자열을 추출할 대상입니다.
2) <시작 위치> 문자열 추출 시작 위치를 지정합니다.
   문자열의 처음 위치는 0입니다.
3) <추출 개수> 지정위치부터 읽어올 문자 개수를 입력합니다.

Return value

<변수>	Data type: <문자열 변수> 지정된 위치에서 설정한 문자 개수만큼 문자열을 반환합니다.

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>
<시작 위치>	Data type: <정수형 변수/상수>
<추출 개수>	Data type: <정수형 변수/상수>

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR0,STR1,STR2               // 문자열 변수 선언
    INT AA, BB                          // 정수형 변수 선언

    AA = 2                              // 정수형 변수 초기화(시작 위치)
    BB = 5                              // 정수형 변수 초기화(추출 개수)
    STR0 = “STATEMENT”                  // 문자열 변수 “STATEMENT” 할당
    STR1 = SMID(“STATEMENT”, 2, 3)      // 변수 STR1에 “ATE” 문자열 저장
    STR2 = SMID(STR0,AA,BB)             // 변수 STR2에 “ATEME” 문자열 저장
EOP                                     // 메인 함수 종료
```

Related terms

SLEFT, SLEN, SPOS, SRIGHT

10.49 SPOS (문자열 매칭 시작 위치 반환)

Usage

문자열1에서 문자열2가 매칭되는 시작 위치를 반환합니다.

Syntax

<변수> = SPOS(<문자열 1>, <문자열 2>)

Description

문자열1에서 문자열2와 매칭되는 시작 위치를 반환합니다.

1) <문자열 1> 비교 대상 문자열을 입력 합니다.
2) <문자열 2> 대상 문자열에서 검색할 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 입력된 문자열 1과 2를 비교하여 매칭 시작 위치를 반환합니다. 일치하는 문자열이 없는 경우 -1 값이 반환됩니다.

Arguments

<문자열 1>	Data type: <문자열> 또는 <문자열 변수>
<문자열 2>	Data type: <문자열> 또는 <문자열 변수>

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR1,STR2                    // 문자열 변수 선언
    INT AA                              // 정수형 변수 선언

    STR1 = “TEST LINE”                  // 문자열 변수 초기화(전체 문자열)
    STR2 = “LINE”                       // 문자열 변수 초기화(검색 문자열)
    I0 = SPOS(“TEST LINE”, “LINE”)      // 매칭 문자열 위치값 5를 전역 변수에 저장
    AA = SPOS(STR1,STR2)                // 매칭 문자열 위치값 5를 지역 변수에 저장
EOP                                     // 메인 함수 종료
```

Related terms

SLEFT, SLEN, SMID, SRIGHT

10.50 SRIGHT (우측 문자열 추출)

Usage

입력된 문자열의 우측 문자열을 추출합니다.

Syntax

<변수> = SRIGHT(<문자열>, <추출 개수>)

Description

입력된 문자열을 오른쪽에서부터 입력된 정수만큼 추출합니다.

1) <문자열>은 문자열을 추출할 대상입니다.
2) <추출 개수>는 문자열의 우측에서부터 추출할 문자열 개수를 의미합니다.

Return value

<변수>	Data type: <문자열 변수> 선택한 문자열 개수만큼 추출 된 문자열을 반환합니다.

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>
<추출 개수>	Data type: <정수형 변수>

Examples

Example 1

```
MAIN                                // 메인 함수 선언
    DEFSTR STR0,STR1,STR2           // 문자열 변수 선언
    INT AA                          // 정수형 변수 선언

    STR0 = “TEST”                   // 문자열 변수 “TEST” 할당
    AA = 2                          // 정수형 변수 초기화
    STR1 = SRIGHT(“TEST”, 1)        // STR1 문자열 변수에 “T” 저장
    STR2 = SRIGHT(STR0, AA)         // STR2 문자열 변수에 “ST” 저장
EOP                                 // 메인 함수 종료
```

Related terms

SLEFT, SLEN, SMID, SPOS

10.51 STOK (String Token)

Usage

구분자로 문자열을 추출 후에 지정 된 순번의 문자를 반환합니다.

Syntax

<변수>=STOK(<문자열>,<구분문자>,<번호>)

Description

입력된 문자 토큰으로 문자열 분리 시 사용할 수 있습니다.

Return value

<변수>	Data type: <문자열 변수>

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>
<구분문자>	Data type: <문자열> 또는 <문자열 변수>
<번호>	Data type: <정수형 변수/상수>

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR STR, TOK                     // 문자형 변수 선언
    DEFSTR STRTMP[4]                    // 문자형 변수 선언
    REAL REALPOS[4]                     // 실수형 변수 선언
    INT IVAL                            // 정수형 변수 선언

    STR = "12.345,-6.789,9876.543"      // 문자열 변수 할당
    TOK = ","                           // 문자열 변수","할당
    IVAL = 1                            // 정수형 변수 초기화

    FOR IVAL=1 TO 3 BY 1                // 1부터 3까지 1씩 증가
        STRTMP[IVAL] = STOK(STR,TOK,IVAL+1)  // STR변수의 문자열 중 TOK문자를
                                             구분자로 구분하여 IVAL번째 값 저장
        REALPOS[IVAL] = SVAL(STRTMP[IVAL])   // 구분된 문자를 숫자로 변환
    NEXT                                // FOR문 종료
EOP                                     // 메인 함수 종료
```

Related terms

10.52 STRIN (통신 문자열 수신)

Usage

시리얼 및 이더넷 통신을 통하여 입력 된 문자열을 읽어옵니다.

Syntax

<변수> = STRIN(<포트번호>, <대기시간>)

Description

입력한 대기시간(단위: ms)만큼 통신 입력을 기다립니다. 설정한 대기시간 동안 데이터가 들어오지 않으면, 시스템 영역에 대기 상태를 체크하고 다음 스텝의 명령을 수행합니다.
STRIN 함수를 통해 입력되는 데이터는 문자열의 끝을 판단하기 위해 CR+LF(기본 설정) 또는 CR, LF 문자를 사용하며 이를 새줄 문자라 합니다. 새줄 문자는 반드시 문자열의 끝에 위치해야 하며, 새줄 문자가 없는 경우 입력의 종료를 판단하지 못합니다.
RS-232 통신으로 문자열을 수신 받는 경우 한번에 최대 100개까지, TCP/IP 통신으로는 255개까지 받을 수 있습니다. 그 이상의 문자열을 수신 받는 경우 알람이 발생합니다.
시리얼 및 이더넷 통신 설정은 퍼블릭 파라미터 통신 설정 영역에서 가능합니다.
1) <포트번호>는 물리적 포트를 지정하고 10 ~ 12까지 입력 가능합니다.
   10: Port 번호 38201
   11: Port 번호 38202
   12: Port 번호 38203
2) <대기시간>은 수신대기 시간을 설정합니다.

Return value

<변수>	Data type: <문자열 변수> 수신대기 시간 동안 수신된 문자열을 반환합니다.

Arguments

<포트번호>	Data type: <정수형 변수/상수> RS-232(0) 또는 Ethernet(10,11,12)만 입력할 수 있습니다.
<대기시간>	Data type: <정수형 변수/상수>

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR RECVPK                       // 문자열 변수 선언
    INT LEN                             // 정수형 변수 선언

    RECVPK = STRIN(0,1000)              // RS-232 포트 문자열 수신을 1초 동안 대기
    LEN = SLEN(RECVPK)                  // 수신된 문자열 길이 저장
EOP                                     // 메인 함수 종료
```

Related terms

STROUT

1) 외부에서 이더넷 통신으로 제어기와 문자열을 송, 수신하기 위해서는 IP 주소 외에도 TCP 포트 번호가 추가로 필요합니다. 이 것은 STRIN, STROUT 명령어 사용 시 입력하는 <포트번호>와는 다른 것으로, 현재 고정되어 있으며 그 값은 38201, 38202, 38203 입니다.

10.53 STROUT (통신 문자열 송신)

Usage

시리얼 및 이더넷 통신을 통하여 문자열을 송신합니다.

Syntax

〈변수〉 = STROUT(〈포트번호〉, 〈문자열〉)

Description

입력한 문자열을 시리얼 통신을 통해 출력을 내보내며, 전송하지 못한 문자 수 반환합니다. 예를 들어, “ABCDEFGH”의 문자를 전송하는 경우 성공 시 반환 값은 0이며 전부 전송하지 못한 경우 반환 값은 8 입니다. 또, 전송 실패 시 음수를 반환합니다.

STROUT 함수를 통해 출력되는 데이터는 문자열의 끝을 판단하기 위해 CR+LF(기본 설정) 또는 CR, LF 문자를 자동으로 추가하여 전송하며, 이를 새줄 문자라 합니다.

시리얼 및 이더넷 통신 설정은 퍼블릭 파라미터 통신 설정 영역에서 가능합니다.

1) 〈포트번호〉는 물리적 포트를 지정하고 10 ~ 12까지 입력 가능합니다.
   10: Port 번호 38201
   11: Port 번호 38202
   12: Port 번호 38203
2) 〈문자열〉은 전송할 문자열을 입력합니다.

Return value

〈변수〉	Data type: 〈정수형 변수〉 전송이 정상적으로 완료된 경우 : 0 반환 전송이 완료되지 않은 경우 : 전송하지 못한 문자 개수 반환 전송 오류가 발생한 경우 : 음수 반환

Arguments

〈포트번호〉	Data type: 〈정수형 변수/상수〉 RS-232(0) 또는 Ethernet(10,11,12)만 입력할 수 있습니다.
〈문자열〉	Data type: 〈문자열〉 또는 〈문자열 변수〉

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR SENDPK                       // 문자열 변수 선언
    INT LEN                             // 정수형 변수 선언

    SENDPK = “SUCCESS”                  // 전송 문자열 초기화
    I0 = STROUT(10,“TEST”)              // 이더넷 포트로 “TEST” 문자열 전송
    LEN = STROUT(10,SENDPK)             // 이더넷 포트로 “SUCCESS”문자열 전송
EOP                                     // 메인 함수 종료
```

Related terms

STRIN

1) 외부에서 이더넷 통신으로 제어기와 문자열을 송, 수신하기 위해서는 IP 주소 외에도 TCP 포트 번호가 추가로 필요합니다. 이 것은 STRIN, STROUT 명령어 사용 시 입력하는 <포트번호>와는 다른 것으로, 현재 고정되어 있으며 그 값은 38201, 38202, 38203 입니다.

10.54 SVAL (문자열 숫자로 변환)

Usage

문자열을 숫자로 변환합니다.

Syntax

<변수> = SVAL(<문자열>)

Description

입력한 문자열을 정수 또는 실수 데이터로 변환합니다.

1) <문자열> 숫자로 변환할 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 또는 <실수형 변수>

Arguments

<문자열>	Data type: <문자열> 또는 <문자열 변수>

Examples

```
MAIN                        // 메인 함수 선언
    DEFSTR STR1,STR2        // 문자열 변수 선언
    INT IVAL1               // 정수형 변수 선언
    REAL FVAL               // 실수형 변수 선언

    STR1 = “1234”           // 정수로 변경할 문자열 “1234” 할당
    STR2 = “12.34”          // 실수로 변경할 문자열 “12.34” 할당
    I0 = SVAL(“1234”)       // 정수형 전역 변수 I0에 정수형 상수1234 저장
    IVAL1 = SVAL(STR1)      // 정수형 내부 변수 IVAL에 정수형 상수 1234 저장
    F0 = SVAL(“12.34”)      // 실수형 전역 변수 F0에 실수형 상수 12.34 저장
    FVAL = SVAL(STR1)       // 실수형 내부 변수 FVAL에 실수형 상수 12.34 저장
EOP                         // 메인 함수 종료
```

Related terms

BINS, FTOS, HTOS

11. 시스템

11.1 SETERR (System EMG 알람 발생)

Usage

사용자가 임의로 정의한 알람이며 System emergency를 발생시킵니다.

Syntax

SETERR(<사용자 알람 번호>[, <사용자 알람 메시지>])

Description

JOB 프로그램 실행 중 사용자의 의도에 의해 [2103] System emergency를 발생시키는 명령어로, 알람 메시지 창에 사용자 알람 번호와 메시지가 출력됩니다.

1) <사용자 알람 번호>는 정수 표현 범위를 벗어나지 않는 한도에서 사용자가 임의로 설정해야 합니다.
2) <사용자 알람 메시지>출력 기능은 사용자가 지정한 임의의 알람을 보다 알기 쉽게 식별하기 위해 제공됩니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Arguments

<사용자 알람 번호>	Data type: <정수형 변수/상수> 사용자가 임의로 설정한 알람 번호
<사용자 알람 메시지>	Data type: <문자열> 또는 <문자열 변수> 사용자가 임의로 출력하는 알람 메시지

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    IF (IN0 == 1) THEN      // 0번 입력 신호가 1인 경우
        SETERR(1)           // 사용자 알람 번호 1로 System emergency 발생
    ENDIF                   // 조건문 종료
EOP                         // 메인 함수 종료
```

Example 2

```
MAIN                                // 메인 함수 선언
    IF (IN0 == 1) THEN              // 0번 입력 신호가 1인 경우
        SETERR(1, “USER ERR 1”)     // 메시지와 함께 System emergency 발생
    ENDIF                           // 조건문 종료
EOP                                 // 메인 함수 종료
```

Related terms

RESET

11.2 RESET (알람 해제)

Usage

발생한 알람을 모두 해제합니다.

Syntax

RESET[(<로봇 번호>)]

Description

현재 발생된 알람을 모두 지웁니다.

1) <로봇 번호>를 입력하여 알람을 해지할 로봇을 선택합니다. 미 입력 시 모든 로봇의 알람을 해지합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    WHILE 1                             // 반복문 시작
        IF(IN0 == 1) THEN               // 0번 입력 신호가 1인 경우
            RESET                       // 모든 알람 해제
        ELSEIF (IN1 == 1) THEN          // 1번 입력 신호가 1인 경우
            RESET(1)                    // 로봇 1의 알람 해제
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

SETERR

11.3 RERROR (마지막 알람 코드 반환)

Usage

가장 마지막(최근)에 발생한 알람 코드를 반환합니다.

Syntax

<변수> = RERROR(<로봇 번호>)

Description

발생한 알람 중 가장 마지막(최근)에 알람 코드를 반환합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다. 0 입력 시 모든 로봇의 알람 중 가장 마지막(최근)에 발생한 알람 코드를 반환 합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Return value

<변수>	Data type: <정수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 0 ~ 3까지입니다. (0: All, 1 ~ 3: 로봇 선택)

Examples

```
MAIN                                // 메인 함수 선언
    INT A_ST, ECODE                 // 정수형 변수 선언

    WHILE 1                         // 반복문 실행
        A_ST = RSTATE(1,1)          // 알람 상태 확인
        IF( A_ST == 1) THEN         // 알람이 발생한 상태인 경우
            ECODE = RERROR(0)       // 마지막 알람 코드 반환
            PRINT(0,ECODE)          // 모니터링 로그 0번에 마지막 알람 코드 출력
        ENDIF                       // 조건문 종료
    ENDWL                           // 반복문 종료
EOP                                 // 메인 함수 종료
```

Related terms

RERRCNT, RERRCODE, RERRTEXT, READ_DERR

11.4 RERRCNT (발생한 알람 개수 반환)

Usage

총 발생한 알람 개수를 반환합니다.

Syntax

<변수> = RERRCNT(<로봇 번호>)

Description

발생한 알람의 총 개수를 반환합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다. 0 입력 시 전체 로봇에서 발생한 모든 알람을 더해 그 개수를 반환합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Return value

<변수>	Data type: <정수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 0 ~ 3까지입니다. (0: All, 1 ~ 3: 로봇 선택)

Examples

```
MAIN                                    // 메인 함수 선언
    INT A_ST, ECNT                      // 정수형 변수 선언

    WHILE 1                             // 반복문 실행
        A_ST = RSTATE(1,1)              // 알람 상태 확인
        IF( A_ST == 1) THEN             // 알람이 발생한 상태라면
            ECNT = RERRCNT(0)           // 전체 로봇에서 발생한 알람의 총 개수 반환
            PRINT(0,ECNT)               // 모니터링 로그 0번에 알람 총 개수 출력
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

RERROR, RERRCODE, RERRTEXT, READ_DERR

11.5 RERRCODE (인덱스 번호의 알람 코드 반환)

Usage

입력한 인덱스 번호에 해당하는 알람 코드를 반환합니다.

Syntax

<변수> = RERRCODE(<로봇 번호>. <인덱스 번호>)

Description

발생한 여러 알람 중 입력한 인덱스 번호의 알람 코드를 반환합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다. 0 입력 시 모든 로봇을 의미합니다.
2) <인덱스 번호>는 알람 발생 순서를 의미합니다. 따라서, 선택한 로봇에서 발생한 알람 중 원하는 순서에 해당하는 알람 코드를 반환합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Return value

<변수>	Data type: <정수형>

Arguments

<로봇 번호>	Data type: <정수형> <로봇 번호>의 범위는 0 ~ 3까지입니다. (0: All, 1 ~ 3: 로봇 선택)
<인덱스 번호>	Data type: <정수형> 인덱스 번호는 알람의 총 카운트를 초과해서는 안됩니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT A_ST, AA                        // 정수형 변수 선언
    INT ECNT, ECODE

    WHILE 1                             // 반복문 실행
        A_ST = RSTATE(1,1)              // 알람 상태 확인
        IF( A_ST == 1) THEN             // 알람이 발생한 상태라면
            ECNT = RERRCNT(0)           // 전체 로봇에서 알람의 총 개수 반환
            PRINT(0,ECNT)               // 모니터링 로그 0번에 알람 총 개수 출력

            FOR AA = 1 TO ECNT          // 알람 개수 만큼 알람 코드 확인
                ECODE = RERRCODE(0,AA)  // 지정한 인덱스의 알람 코드 값 반환
                PRINT(1+AA, ECODE)      // 모니터링 로그에 알람 코드 출력
            NEXT                        // 반복문 종료
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

RERROR, RERRCNT, RERRTEXT, READ_DERR

11.6 RERRTEXT (알람 내용 반환)

Usage

입력한 인덱스 번호에 해당하는 알람 내용을 반환합니다.

Syntax

<변수> = RERRTEXT(<로봇 번호>, <인덱스 번호>)

Description

발생한 여러 알람 중 입력한 인덱스 번호의 알람 내용을 반환합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다. 0 입력 시 모든 로봇을 의미합니다.
2) <인덱스 번호>는 알람 발생 순서를 의미합니다. 따라서, 선택한 로봇에서 발생한 알람 중 원하는 순서에 해당하는 알람 코드의 내용을 반환합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Return value

<변수>	Data type: <문자열 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 0 ~ 3까지입니다. (0: All, 1 ~ 3: 로봇 선택)
<인덱스 번호>	Data type: <정수형 변수/상수> 인덱스 번호는 알람의 총 카운트를 초과해서는 안됩니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT A_ST, AA                        // 정수형 변수 선언
    INT ECNT, ECODE
    DEFSTR E_STR                        // 문자열 변수 선언
    WHILE 1                             // 반복문 실행
        A_ST = RSTATE(1,1)              // 알람 상태 확인
        IF( A_ST == 1) THEN             // 알람이 발생한 상태라면
            ECNT = RERRCNT(0)           // 발생한 알람의 총 개수 반환
            PRINT(0,ECNT)               // 모니터링 로그 0번 알람 총 개수 출력
            FOR AA = 1 TO ECNT          // 알람 개수 만큼 알람 코드 확인
                ECODE = RERRCODE(0,AA)  // 지정한 인덱스의 알람 코드 값 반환
                E_STR = RERRTEXT(0,AA)  // 지정한 인덱스의 알람 내용 반환
                PRINT(1+AA,ECODE,E_STR) // 모니터링 로그에 알람 코드,알람 내용출력
            NEXT                        // 반복문 종료
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

RERROR, RERRCNT, RERRCODE, READ_DERR

11.7 READ_DERR (드라이버 알람 코드 반환)

Usage

지정한 축에서 발생한 드라이버의 알람 코드를 반환합니다.

Syntax

<변수> = READ_DERR(<로봇 번호>, <축 번호>)

Description

지정한 축 번호가 가리키는 드라이버의 알람 코드를 반환합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>는 로봇 관절의 연결 순서를 의미합니다.

※ 이 명령어는 System 모드의 메인 스레드에서 사용하시기바랍니다.

Return value

<변수>	Data type: <정수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> 축 번호는 축의 최대 개수를 초과하거나 최소 개수를 미달해서는 안됩니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT ECODE, AA                       // 정수형 변수 선언

    FOR AA = 1 TO 6                     // 1 ~ 6 축 까지 반복문 수행
        ECODE = READ_DERR(1,AA)         // 로봇 1의 각 축별 드라이버 알람 반환
        PRINT(AA, ECODE)                // 모니터링 버퍼에 출력
    NEXT                                // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

RERROR, RERRCNT, RERRCODE, RERRTEXT

11.8 RSTATE (로봇 상태 정보 획득)

Usage

로봇의 상태를 반환하는 명령어 입니다.

Syntax

<변수> = RSTATE(<로봇 번호>, <상태 인덱스>)

Description

지정한 로봇의 지정한 인덱스 정보를 가져옵니다.

1) <로봇 번호>는 상태를 읽어올 로봇의 번호를 의미합니다.
2) <상태 인덱스>는 로봇 특정 상태 정보를 사전에 정의한 내용입니다.

Return value

<변수> Data type: <정수형 변수>

Arguments

<로봇 번호> Data type: <정수형 변수/상수>
<로봇 번호>의 범위는 1 ~ 3까지입니다.

<상태 인덱스> Data type: <정수형 변수/상수>
<상태 인덱스>의 범위는 0 ~ 16까지입니다.

※ 아래 표에 설명되지 않은 인덱스 번호는 현재 사용되지 않는 번호입니다.

INDEX	STATE NAME	내용
0	ALL	전체 상태 정보를 반환
1	ALARM	알람 상태 반환 (0: 정상, 1: 알람 발생)
2	MC STATUS (READY)	MC 상태를 반환 (0: MC OFF, 1: MC ON)
3	ORIGIN	원점 완료 상태 반환 (0: 원점 미수행, 1: 원점 수행 완료) **※엔코더 타입이 ABS 타입인 경우 항상 ON**
4	INPOSITION	목표 위치 도달 완료 (0: 목표 위치 미도달, 1: 목표위치 도달 완료)
5	Reserved	예약됨
6	SERVO STATE	SERVO MOTOR ON/OFF 상태 반환 (0: Servo OFF, 1: Servo ON)
7	Reserved	예약됨

<table>
<tr><td>8</td><td>MODE MISMATCH</td><td>T/P 모드 스위치 정보와 상위설비 장치 (INDEX) 모드 상태 반환
<table><tr><th>반환 값</th><th>T/P MODE</th><th>INDEX MODE</th></tr><tr><td rowspan="2">0</td><td>SYSTEM</td><td>SYSTEM</td></tr><tr><td>ROBOT</td><td>ROBOT</td></tr><tr><td rowspan="2">1</td><td>SYSTEM</td><td>ROBOT</td></tr><tr><td>ROBOT</td><td>SYSTEM</td></tr></table>
※ IF F/W V1.2B 이상부터 지원합니다.</td></tr>
<tr><td>9</td><td>SYSTEM MODE</td><td>ROBOT / SYSTEM 상태 반환
(0: Robot 모드, 1: System 모드)</td></tr>
<tr><td>10</td><td>T/P MODE</td><td>T/P SYSTEM(1) / ROBOT(0) 상태 반환
(0: Robot 모드, 1: System 모드)</td></tr>
<tr><td>11</td><td>EMG</td><td>비상 정지 상태 반환
(0: 정상, 1: TP / FRONT/ System EMG 발생)</td></tr>
<tr><td>12</td><td>ROBOT ENB</td><td>ROBOT 활성화 상태 반환
(0: Robot Disable, 1: Robot Enable)</td></tr>
<tr><td>13</td><td>WARNING STATE</td><td>FAN/ ENC BATTERY 경고 상태 반환
(0: 정상, 1: FAN/ Encoder 배터리 경고 발생)</td></tr>
<tr><td>14</td><td>ENC BATTERY STATE</td><td>ENCODER BATTERY 경고 상태 반환
(0: 정상, 1: Encoder 배터리 경고 발생)</td></tr>
<tr><td>15</td><td>ROBOT MOVING</td><td>해당 로봇의 모션 수행 상태 알림
(0: 이동 없음, 1: 이동 중)</td></tr>
<tr><td>16</td><td>FAN STATE</td><td>팬 경고 상태 알림
(0: 정상, 1: FAN 경고 발생)</td></tr>
</table>

Examples

Example 1

- 전체 상태를 확인하여 I/O로 개별 출력

```
MAIN                                    // 메인 함수 선언
    INT TEMP                            // 정수형 변수 선언
    TEMP = RSTATE(1,0)                  // 로봇 1번 전체 상태 정보 읽기
    OUT10 = (TEMP>>6)&0H01              // 서보 상태 출력
    OUT9 = (TEMP>>9)& 0H01              // 시스템 상태 출력
    OUT8 = TEMP&0H01                    // 알람 상태 출력
EOP                                     // 메인 함수 종료
```

Example 2

- 개별 상태를 확인하여 I/O로 출력

```
MAIN                                    // 메인 함수 선언
    OUT10 = RSTATE(1,6)                 // 로봇 1번 서보 상태 출력
    OUT9 = RSTATE(1,9)                  // 로봇 1번 시스템 상태 출력
    OUT8 = RSTATE(1,1)                  // 로봇 1번 알람 상태 출력
EOP                                     // 메인 함수 종료
```

11.9 SWLIMIT (S/W LIMIT 파라미터 값을 읽음)

Usage

로봇의 축 별 이동 범위를 제한하는 파라미터 값을 읽어옵니다.

Syntax

<변수> = SWLIMIT(<로봇 번호>, <축 번호>, <최대/최소>)

Description

로봇 파라미터 중 BODY-RANG 에서 설정한 S/W Limit 값을 읽어옵니다.

1) <로봇 번호>를 입력하여 파라미터 값을 읽어올 로봇을 선택합니다.
2) <축 번호>를 입력하여 여러 축 중에 값을 읽어올 축 번호를 선택합니다.
3) <최대/최소>은 이동 범위 최솟값 또는 최댓값을 가져올 지 선택합니다.

Return value

<변수>	Data type: <실수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.
<최대/최소>	Data type: <정수형 변수/상수> 최솟값을 읽어오고자 하는 경우 0을, 최댓값을 읽어오고자 하는 경우는 1을 선택합니다.

Examples

```
MAIN                                    // 메인 함수 선언
    REAL LIMIT_MAX, LIMIT_MIN           // 실수형 변수 선언

    LIMIT_MIN = SWLIMIT(1,1,0)          // 1번 로봇 1번 축 이동 범위 최솟값 읽기
    LIMIT_MAX = SWLIMIT(1,1,1)          // 1번 로봇 1번 축 이동 범위 최댓값 읽기
EOP                                     // 메인 함수 종료
```

11.10 READ_RPM (RPM 값 읽기)

Usage

로봇의 축 별 현재 RPM 값을 가져옵니다.

Syntax

<변수> = READ_RPM(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 현재 RPM 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 현재 RPM을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 RPM입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT AA, R_VAL                       // 정수형 변수 선언
    REAL RPM[6]                         // 실수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언

    WHILE 1                             // WHILE 문 시작
        STR = “RPM = ”                  // 문자열 변수 초기화
        FOR AA = 0 TO 5                 // 0부터 5까지 FOR 문 시작
            RPM[AA] = READ_RPM(1,AA + 1)    // 축 별 RPM 읽어 배열에 저장
            STR = STR + FTOS(RPM[AA]) + “ ”     // 문자열 변수에 저장
        NEXT                            // FOR 문 종료
        R_VAL = STROUT(10, STR)         // 이더넷 포트로 문자열 데이터 전송
        DLAY 100                        // 100ms 대기
    ENDWL                               // WHILE 문 종료
EOP                                     // 메인 함수 종료
```

Related terms

READ_LOAD, READ_TRQMAX, READ_VEL, READ_TRQ, READ_MTEMP, READ_CTEMP

11.11 READ_LOAD (부하율 읽기)

Usage

로봇의 축 별 현재 부하율을 가져옵니다.

Syntax

<변수> = READ_LOAD(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 현재 부하 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 현재 부하율을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 %입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                        // 메인 함수 선언
    INT AA, R_VAL                           // 정수형 변수 선언
    REAL LOAD[6]                            // 실수형 변수 선언
    DEFSTR STR                              // 문자열 변수 선언

    WHILE 1                                 // WHILE 문 시작
        STR = “LOAD = ”                     // 문자열 변수 초기화
        FOR AA = 0 TO 5                     // 0부터 5까지 FOR 문 시작
            LOAD [AA] = READ_LOAD(1,AA+1)   // 축 별 로드 값 읽어 배열에 저장
            STR = STR + FTOS(LOAD [AA]) + “ ”   // 문자열 변수에 저장
        NEXT                                // FOR 문 종료
        R_VAL = STROUT(10, STR)             // 이더넷 포트로 문자열 데이터 전송
        DLAY 100                            // 100ms 대기
    ENDWL                                   // WHILE 문 종료
EOP                                         // 메인 함수 종료
```

Related terms

READ_RPM, READ_TRQMAX, READ_VEL, READ_TRQ, READ_MTEMP, READ_CTEMP

11.12 READ_TRQMAX (최대 토크 값 읽기)

Usage

로봇의 축 별 최대 토크 값을 가져옵니다.

Syntax

<변수> = READ_TRQMAX(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 최대 토크 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 최대 토크 값을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 %입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                        // 메인 함수 선언
    INT AA, R_VAL                           // 정수형 변수 선언
    REAL T_MAX[6]                           // 실수형 변수 선언
    DEFSTR STR                              // 문자열 변수 선언

    WHILE 1                                 // WHILE 문 시작
        STR = “TRQ_MAX = ”                  // 문자열 변수 초기화
        FOR AA = 0 TO 5                     // 0부터 5까지 FOR 문 시작
            T_MAX[AA] = READ_TRQMAX(1,AA+1) // 축 별 최대 토크 값 읽어 저장
            STR = STR + FTOS(T_MAX [AA]) + “ ”  // 문자열 변수에 저장
        NEXT                                // FOR 문 종료
        R_VAL = STROUT(10, STR)             // 이더넷 포트로 문자열데이터 전송
        DLAY 100                            // 100ms 대기
    ENDWL                                   // WHILE 문 종료
EOP                                         // 메인 함수 종료
```

Related terms

READ_RPM, READ_LOAD, READ_VEL, READ_TRQ, READ_MTEMP, READ_CTEMP

11.13 READ_VEL (속도 읽기)

Usage

로봇의 축 별 현재 속도 값을 가져옵니다.

Syntax

<변수> = READ_VEL(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 현재 속도 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 현재 속도 값을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 회전 축의 경우 degree/s, 직선 이동 축의 경우는 mm/s 입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT AA, R_VAL                       // 정수형 변수 선언
    REAL V_VAL[6]                       // 실수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언

    WHILE 1                             // WHILE 문 시작
        STR = “VEL = ”                  // 문자열 변수 초기화
        FOR AA = 0 TO 5                 // 0부터 5까지 FOR 문 시작
            V_VAL[AA] = READ_VEL(1,AA+1)    // 축 별 속도 값 읽어 배열에 저장
            STR = STR + FTOS(V_VAL[AA]) + “ ”   // 문자열 변수에 저장
        NEXT                            // FOR 문 종료
        R_VAL = STROUT(10, STR)         // 이더넷 포트로 문자열 데이터 전송
        DLAY 100                        // 100ms 대기
    ENDWL                               // WHILE 문 종료
EOP                                     // 메인 함수 종료
```

Related terms

READ_RPM, READ_LOAD, READ_TRQMAX, READ_TRQ, READ_MTEMP, READ_CTEMP

11.14 READ_TRQ (토크 값 읽기)

Usage

로봇의 축 별 현재 토크 값을 가져옵니다.

Syntax

<변수> = READ_TRQ(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 현재 토크 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 현재 토크 값을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 %입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT AA, R_VAL                       // 정수형 변수 선언
    REAL TRQ_VAL[6]                     // 실수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언
    WHILE 1
        STR = "TRQ = "                  // WHILE 문 시작
        FOR AA = 0 TO 5                 // 문자열 변수 초기화
            TRQ_VAL[AA] =               // 0부터 5까지 FOR 문 시작
            READ_TRQ(1,AA+1)            // 축 별 현재 토크 값 읽어 배열에 저장
            STR = STR +                 // 문자열 변수에 저장
            FTOS(TRQ_VAL[AA]) + " "
        NEXT                            // FOR 문 종료
        R_VAL = STROUT(10, STR)         // 이더넷 포트로 문자열 데이터 전송
        DLAY 100                        // 100ms 대기
    ENDWL                               // WHILE 문 종료
EOP                                     // 메인 함수 종료
```

Related terms

READ_RPM, READ_LOAD, READ_TRQMAX, READ_VEL, READ_MTEMP, READ_CTEMP

11.15 READ_MTEMP (모터 엔코더 온도 값 읽기)

Usage

로봇의 축 별 현재 모터 엔코더 온도 값을 가져옵니다.

Syntax

<변수> = READ_MTEMP(<로봇 번호>, <축 번호>)

Description

입력한 로봇과 축 번호에 해당하는 현재 모터 엔코더 온도 값을 가져옵니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 현재 모터 엔코더 온도 값을 읽어올 축을 선택합니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 섭씨온도(℃) 입니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지입니다.
<축 번호>	Data type: <정수형 변수/상수> <축 번호>의 범위는 1 ~ DOF까지입니다.

Examples

```
MAIN                                        // 메인 함수 선언
    INT AA, R_VAL                           // 정수형 변수 선언
    REAL TEMP[6]                            // 실수형 변수 선언
    DEFSTR STR                              // 문자열 변수 선언

    WHILE 1                                 // WHILE 문 시작
        STR = " ENCODER TEMP = "            // 문자열 변수 초기화
        FOR AA = 0 TO 5                     // 0부터 5까지 FOR 문 시작
            TEMP[AA] = READ_MTEMP(1,AA+1)   // 축 별 모터 엔코더 온도 값 저장
            STR = STR + FTOS(TEMP[AA]) + " "// 문자열 변수에 저장
        NEXT                                // FOR 문 종료
        R_VAL = STROUT(10, STR)             // 이더넷 포트로 문자열데이터 전송
        DLAY 100                            // 100ms 대기
    ENDWL                                   // WHILE 문 종료
EOP                                         // 메인 함수 종료
```

Related terms

READ_RPM, READ_LOAD, READ_TRQMAX, READ_VEL, READ_TRQ, READ_CTEMP

11.16 READ_CTEMP (제어기 내부 온도 값 읽기)

Usage

제어기 내부 온도 값을 가져옵니다.

Syntax

<변수> = READ_CTEMP()

Description

현재 제어기 내부 온도 값을 가져옵니다.

Return value

<변수>	Data type: <실수형 변수> 결과 값의 단위는 섭씨온도(℃) 입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT R_VAL                           // 정수형 변수 선언
    REAL C_TEMP                         // 실수형 변수 선언
    DEFSTR STR                          // 문자열 변수 선언

    WHILE 1                             // WHILE 문 시작
        STR = " CONTROLLER TEMP ="      // 문자열 변수 초기화
        C_TEMP = READ_CTEMP ()          // 현재 제어기 온도 값 읽어 배열에 저장
        STR = FTOS(C_TEMP)              // 문자열 변수에 저장
        R_VAL = STROUT(10, STR)         // 이더넷 포트로 문자열 데이터 전송
        DLAY 100                        // 100ms 대기
    ENDWL                               // WHILE 문 종료
EOP                                     // 메인 함수 종료
```

Related terms

READ_RPM, READ_LOAD, READ_TRQMAX, READ_VEL, READ_TRQ, READ_MTEMP

11.17 PRINT (모니터링 로그)

Usage

모니터링 용 로그를 기록합니다.

Syntax

PRINT(<인덱스>, <로그 내용1> [{, <로그 내용2>, … <로그 내용N>}])

Description

JOB 프로그램을 실행시키며 확인할 수 있는 로그를 기록하는 사용됩니다.

로그 기록 시 로그 내용에 모든 변수 및 상수의 대입이 가능합니다. 단, <로그 내용>에 함수를 대입하여 그 반환 값을 저장하는 사용방식은 현재 허용되지 않습니다.

- 사용 가능 예시: PRINT(0, 12, “STRING DATA”, F[12], I0, AA) ← (○)
- 사용 불가 예시: PRINT(0,FTOS(R_VAL)) ← (×)

PRINT 함수를 사용하여 남긴 기록은 팬던트 RUN 실행 화면의 DISP 창 9.PRINT에서만 확인이 가능합니다. 사용자가 지정한 인덱스 번호로 이동하여 데이터를 확인할 수 있으며, 입력된 로그 내용들의 구분은 “,” 기호로 이루어집니다. 위치형 데이터의 경우에만 “<”, “>” 기호를 사용하여 구분하며, 기호 안에 각 축 별 위치 값을 출력합니다.

- DISP 화면 9.PRINT 출력 예시

```
000:[0:0]<-11.860,8.636,18.761,57.015,12.979,-100.037>
001:[0:0]1,DATA,12.4000000,45
002:
003:
```

PRINT 함수 사용 시 사용자가 입력하는 로그 내용 앞에 JOB 프로그램이 실행되는 모드 및 Thread 번호를 추가로 표시해 줍니다.

- 화면 출력 정보 예시: [<실행 모드>:<Thread 번호>] <로그 내용1>, …, <로그 내용N>

실행 모드와 Thread 번호의 출력 의미는 아래 표를 참고해 주십시오.

분류	설명
<실행 모드>	System (1), Robot (0)
<Thread 번호>	0 ~ 3

1) <인덱스>를 지정하여 저장할 위치를 선택합니다.
2) <로그 내용>에 사용자가 저장하고 싶은 데이터를 기록합니다.

Arguments

<인덱스> Data type: <정수형>
<인덱스>의 범위는 0 ~ 999까지입니다.

<로그 내용> Data type: 모든 변수 및 상수

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS ACUR                            // 위치형 변수 선언

    ACUR = HERE                         // 현재 로봇 위치를 변수에 저장
    PRINT(0,ACUR)                       // 모니터링 로그 0번 인덱스에 위치 값 출력
EOP                                     // 메인 함수 종료
```

- T/P 출력 예시

```
000:[0:0]<-11.860,8.636,18.761,57.015,12.979,-100.037>
```

Example 2

```
MAIN                                    // 메인 함수 선언
    INT I_VAL                           // 정수형 변수 선언
    REAL R_VAL                          // 실수형 변수 선언

    I_VAL = 1                           // 정수형 변수 초기화
    R_VAL = 12.4                        // 실수형 변수 초기화
    PRINT(1,I_VAL, “DATA”, R_VAL,45)    // 모니터링 로그 1번 인덱스에 여러 값 출력
EOP                                     // 메인 함수 종료
```

- T/P 출력 예시

```
001:[0:0]1,DATA,12.400,45
```

Related terms

WLOG

11.18 WLOG (사용자 로그)

Usage

임의의 사용자 로그를 시간 정보를 포함하여 파일로 저장합니다.

Syntax

WLOG(<로그 내용1>, [{, <로그 내용2>, … <로그 내용N>}])

Description

JOB 프로그램을 실행 중 임의의 로그를 저장하는 경우에 사용됩니다. 시간 정보를 포함하여 파일의 형태로 저장되며 유니호스트 프로그램 또는 USB 백업 기능을 사용하여 다운로드 할 수 있습니다.

※ 로그 다운로드 기능에 대한 보다 자세한 설명 및 사용법은 N2 유니호스트 설명서 (N2-HM-K□□) 또는 N2 조작 및 운용 설명서(N2-OM-K□□)를 참고하여 주십시오.

로그 기록 시 로그 내용에 모든 변수 및 상수의 대입이 가능합니다. 단, <로그 내용>에 함수를 대입하여 그 반환 값을 저장하는 사용방식은 현재 허용되지 않습니다.

- 사용 가능 예시: WLOG(12, “STRING DATA”, F[12], I0, AA) ← (○)
- 사용 불가 예시: WLOG(FTOS(R_VAL)) ← (×)

WLOG 함수를 사용하여 남긴 기록은 9.INFO-> 3.LOG -> 3.USER 화면에서 확인이 가능합니다. 단, 소형 T/P를 사용하는 경우 가장 최근에 기록한 100줄까지의 기록만 확인이 가능합니다. 입력된 로그 내용들의 구분은 “,” 기호로 이루어집니다. 위치형 데이터의 경우에만 “<”, “>” 기호를 사용하여 구분하며, 기호 안에 각 축 별 위치 값을 출력합니다.

- USER LOG 화면 출력 예시

```
00:2021-01-12|09:45:41:869|726|USER|PROGRAM||MSG1|0|0|TEST|2
01:2021-01-12|09:45:42:156|789|USER|PROGRAM||MSG2|0|0|TEST|3
```

WLOG 함수 사용 시 우선 로그가 기록된 시간 정보를 기록합니다. 그 다음 줄에 사용자가 입력하는 로그 내용 앞에는 JOB 프로그램이 실행되는 모드 및 Thread 번호를 추가하여 기록합니다.

- 화면 출력 정보 예시:
  <기록 시간 정보>|<인덱스>|<USER>|<PROGRAM>||
  <로그 내용1>, …, <로그 내용N>|<실행모드>|<Thread번호>|<파일명>|<호출 라인>

분류	설명
<실행 모드>	System (1), Robot (0)
<Thread 번호>	0 ~ 3
<파일명>	JOB 파일명
<호출 라인>	WLOG함수를 호출한 라인

※ 구버전 USER LOG

- USER LOG 화면 출력 예시

```
00:[180901 15:00:06.085]
01:[0:0]WLOG DATA1,12.4000000,45
02:[180901 15:15:12.072]
03:[0:0]WLOG DATA2,9.548000,0
```

WLOG 함수 사용 시 우선 로그가 기록된 시간 정보를 기록합니다. 그 다음 줄에 사용자가 입력하는 로그 내용 앞에는 JOB 프로그램이 실행되는 모드 및 Thread 번호를 추가하여 기록합니다.

- 화면 출력 정보 예시: [<기록 시간 정보>]
  [<실행 모드>:<Thread 번호>] <로그 내용1>, …, <로그 내용N>

분류	설명
<실행 모드>	System (1), Robot (0)
<Thread 번호>	0 ~ 3

1) <로그 내용>에 사용자가 저장하고 싶은 데이터를 기록합니다.

Arguments

<로그 내용> Data type: 모든 변수 및 상수

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS ACUR                            // 위치형 변수 선언

    ACUR = HERE                         // 현재 로봇 위치를 변수에 저장
    WLOG("CUR POS", ACUR)               // 사용자 로그에 위치 값 저장
EOP                                     // 메인 함수 종료
```

- 기록 예시

```
00:2021-09-01|15:30:05.085|153|USER|PROGRAM||
CUR POS,<-11.860,8.636,18.761,57.015,12.979,-100.037>|0|0|MASTER|3
```

Example 2

```
MAIN                                    // 메인 함수 선언
    INT I_VAL                           // 정수형 변수 선언
    REAL R_VAL                          // 실수형 변수 선언

    I_VAL = 100                         // 정수형 변수 초기화
    R_VAL = 12.4                        // 실수형 변수 초기화
    WLOG("WLOG DATA", R_VAL, I_VAL)     // 사용자 로그에 여러 값 저장
EOP                                     // 메인 함수 종료
```

- 기록 예시

```
00:2021-09-01|15:45:05.085|153|USER|PROGRAM||
WLOG DATA,12.400,100|0|0|MASTER|6
```

Related terms

PRINT

11.19 CLOG (통신용 로그) ※ 현재 지원하지 않습니다.

11.20 WGPNT (전역 포인트 변수 파일 저장)

Usage

전역 포인트 변수를 파일로 저장합니다.

Syntax

[<변수> =] WGPNT(<동기화>)

Description

JOB 프로그램 실행 중 전역 포인트 변수를 수정하는 경우, 디스크에 저장하기 위해 WGPNT 함수를 사용합니다.

1) <동기화>에 값을 입력해 디스크에 동기화 될 때까지 대기할 것인지 아닌지 그 여부를 선택합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(쓰기 정상) 또는 -1(쓰기 오류)값이 반환됩니다.

Arguments

<동기화> Data type: <정수형 변수/상수>
0(디스크 동기화 대기 안 함) 또는 1(디스크 동기화 대기)만 입력할 수 있습니다.
※ 동기화 설정 값은 1(디스크 동기화 대기)를 권장합니다.

Examples

```
MAIN                                    // 메인 함수 선언
    POS ACUR                            // 위치형 변수 선언

    ACUR = <100,100,100,100,100,100>    // 위치형 변수 초기화
    GP0 = ACUR                          // GP0에 위치형 변수 대입
    WGPNT(1)                            // 전역 포인트 변수 파일로 저장
EOP                                     // 메인 함수 종료
```

Related terms

WGINT, WGFLT

1) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
2) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

11.21 WGINT (전역 정수 변수 파일 저장)

Usage

전역 정수 변수를 파일로 저장합니다.

Syntax

[<변수> =] WGINT(<동기화>)

Description

JOB 프로그램 실행 중 전역 정수 변수를 수정하는 경우, 디스크에 저장하기 위해 WGINT 함수를 사용합니다.

1) <동기화>에 값을 입력해 디스크에 동기화 될 때까지 대기할 것인지 아닌지 그 여부를 선택합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(쓰기 정상) 또는 -1(쓰기 오류)값이 반환됩니다.

Arguments

<동기화>	Data type: <정수형 변수/상수> 0(디스크 동기화 대기 안 함) 또는 1(디스크 동기화 대기)만 입력할 수 있습니다. **※ 동기화 설정 값은 1(디스크 동기화 대기)를 권장합니다.**

Examples

```
MAIN                        // 메인 함수 선언
    INT I_VAL               // 정수형 변수 선언

    I_VAL = 50 +25          // 정수형 변수 초기화
    I100 = 100              // I100에 값 대입
    I101 = I_VAL            // I101에 값 대입
    WGINT(1)                // 전역 정수 변수 파일로 저장
EOP                         // 메인 함수 종료
```

Related terms

WGPNT, WGFLT

1) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
2) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

11.22 WGFLT (전역 실수 변수 파일 저장)

Usage

전역 실수 변수를 파일로 저장합니다.

Syntax

[<변수> =] WGFLT(<동기화>)

Description

JOB 프로그램 실행 중 전역 실수 변수를 수정하는 경우, 디스크에 저장하기 위해 WGFLT 함수를 사용합니다.

1) <동기화>에 값을 입력해 디스크에 동기화 될 때까지 대기할 것인지 아닌지 그 여부를 선택합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 0(쓰기 정상) 또는 -1(쓰기 오류)값이 반환됩니다.

Arguments

<동기화>	Data type: <정수형 변수/상수> 0(디스크 동기화 대기 안 함) 또는 1(디스크 동기화 대기)만 입력할 수 있습니다. **※ 동기화 설정 값은 1(디스크 동기화 대기)를 권장합니다.**

Examples

```
MAIN                        // 메인 함수 선언
    REAL R_VAL              // 실수형 변수 선언

    R_VAL = RAD(90)         // 실수형 변수 초기화
    F100 = 3.14             // F100에 값 대입
    F101 = R_VAL            // F101에 값 대입
    WGFLT(1)                // 전역 실수 변수 파일로 저장
EOP                         // 메인 함수 종료
```

Related terms

WGPNT, WGINT

1) 프로그램 실행 중 전역 변수를 변경한 후 저장 명령을 실행하지 않는 경우, 전원차단 또는 재 부팅 시 변경된 데이터의 유지를 보장하지 않습니다.
2) 파일 저장 중 갑작스럽게 전원 공급이 차단되는 경우 데이터가 유실될 수 있습니다.

11.23 TACT_START (시간 측정 시작 위치 설정)

Usage

공정 시간 측정을 위한 측정 시작 위치를 설정합니다.

Syntax

TACT_START()

Description

JOB 프로그램 실행 중 이동 명령 또는 공정 전체에 대한 수행 시간을 측정해야 할 경우, 측정 시작 위치를 설정하기 위해 TACT_START 함수를 사용합니다.

Examples

```
MAIN                                // 메인 함수 선언
    INT VAL                         // 정수형 변수 선언

    TACT_START()                    // 시간 측정 시작 위치 설정
    JMOV GP0                        // GP0까지 JMOV
    LMOV GP1                        // GP1까지 LMOV
    VAL = TACT_STOP()               // 시간 측정 종료 위치 설정
    PRINT(0,VAL)                    // 모니터링 로그 0번 인덱스에 측정 시간 출력
EOP                                 // 메인 함수 종료
```

Related terms

TACT_START, TACT_STOP

1) TACT_STOP 함수를 사용하여 시간 측정의 종료 위치를 설정하지 않을 경우, 시간 측정 기능을 사용할 수 없습니다.

11.24 TACT_STOP (시간 측정 종료 위치 설정)

Usage

공정 시간 측정을 위한 측정 종료 위치를 설정합니다.

Syntax

<변수> = TACT_STOP()

Description

JOB 프로그램 실행 중 이동 명령 또는 공정 전체에 대한 수행 시간을 측정해야 할 경우, 측정 종료 위치를 설정하기 위해 TACT_STOP 함수를 사용합니다.

Return value

<변수>	Data type: <정수형 변수> 결과 값은 측정 시간이 반환됩니다. 결과 값의 단위는 ms 입니다.

Examples

```
MAIN                        // 메인 함수 선언
    INT VAL                 // 정수형 변수 선언

    TACT_START()            // 시간 측정 시작 위치 설정
    JMOV GP0                // GP0까지 JMOV
    LMOV GP1                // GP1까지 LMOV
    VAL = TACT_STOP()       // 시간 측정 종료 위치 설정
    PRINT(0,VAL)            // 모니터링 로그 0번 인덱스에 측정 시간 출력
EOP                         // 메인 함수 종료
```

Related terms

TACT_START, TACT_STOP

1) TACT_START 함수를 사용하여 시간 측정의 시작 위치를 설정하지 않을 경우, 시간 측정 기능을 사용할 수 없습니다.

11.25 INIT_POS (Initialization Point)

Usage

입력된 위치형 변수를 초기화합니다.

Syntax

INIT_POS (<변수>)

Description

위치형 변수의 USED 플래그를 NEW로 초기화 할 때 사용 할 수 있습니다.

Arguments

<변수>	Data type: <위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INIT_POS(P5)                        //5번 JOB Point 초기화
    INIT_POS(GP3)                       //3번 GPNT 초기화
EOP                                     // 메인 함수 종료
```

Example 2

```
XPOS $GA                                // XPOS 전역 위치변수 선언

MAIN                                    // 메인 함수 선언
    UPOS LA                             // UPOS 지역 위치변수 선언
    $GA=<-10,-20,-30,20,10,10,ARM=0,T=2>    // GA변수 값 할당
    LA=<100,100,100,100,100,100,ARM=111,T=2,U=3>    // LA변수 값 할당
    INIT_POS($GA )                      // GA변수 초기화
    INIT_POS(LA)                        // LA변수 초기화
EOP                                     // 메인 함수 종료
```

Related terms

11.26 GET_TWTIME (출하 이후 제어기 총 누적 통전 시간 읽기)

Usage

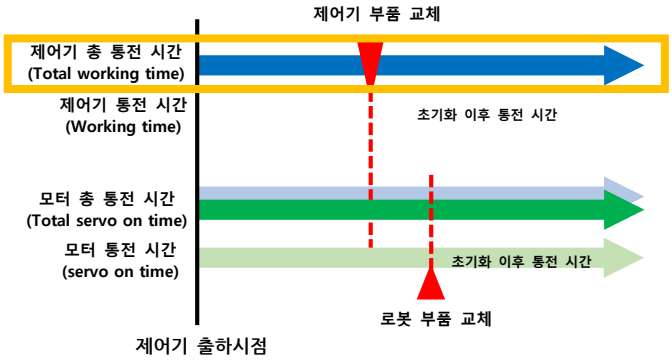
출하 이후 제어기 총 누적 통전시간을 확인 합니다.

Syntax

<변수> = GET_TWTIME()

Description

제어기 출하 이후 제어기 전원을 ON이후 총 시간입니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.

예) 1234일 12시간 34분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	1234일 12시간 34분 56초
표현 시간	1234:12:34:56

Examples

Example 1

```
MAIN                        // 메인 함수 선언
    DEFSTR TW_TIME          // 확인을 위한 변수 선언
    TW_TIME = GET_TWTIME()  // 총 누적 통전 시간 변수에 저장
    PRINT(0, TW_TIME)       // 저장된 변수 확인([0:0]1234:12:34:56)
EOP                         // 메인 함수 종료
```

Related terms

GET_TSVTIME, GET_TMVTIME, GET_TMVDIST

출하 이후 제어기 누적 통전 시간은 초기화 되지 않습니다.

11.27 GET_TSVTIME (출하 이후 모터 총 누적 통전 시간 읽기)

Usage

출하 이후 모터 총 누적 통전 시간을 확인 합니다.

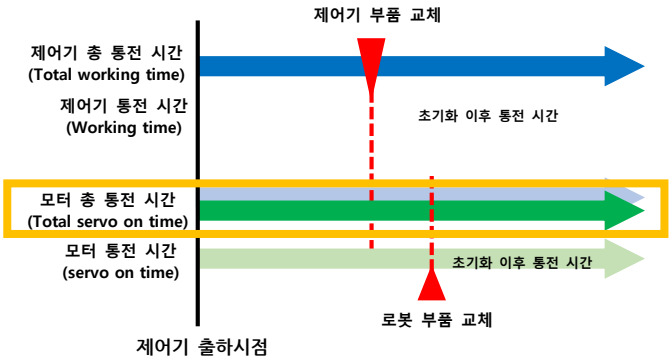
Syntax

<변수> = GET_TSVTIME(<로봇 번호>, <축 번호>)

Description

제어기 출하 이후 모터에 전원이 인가 된 총 시간을 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.
예) 1234일 12시간 34분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	1일 10시간 10분 25초
표현 시간	1:10:10:25

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    DEFSTR TSV_TIME                     // 확인을 위한 변수 선언
    INT AXIS                            // 모터 총 누적 통전 시간을 변수에 저장
    AXIS = 1                            // ([0:0]1:10:10:25)
    TSV_TIME = GET_TSVTIME(1, AXIS)     // 메인 함수 종료
EOP
```

Related terms

GET_TWTIME, GET_TMVTIME, GET_TMVDIST

출하 이후 모터의 누적 통전 시간은 대해서 초기화 되지 않습니다.

11.28 GET_TMVTIME (출하 이후 모터 총 누적 이동 시간 읽기)

Usage

출하 이후 모터 총 누적 이동 시간을 확인 합니다.

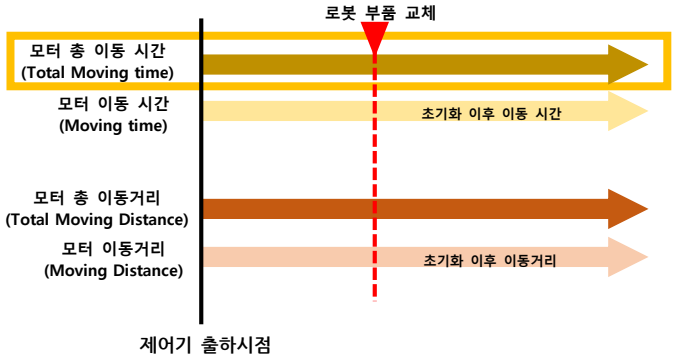
Syntax

<변수> = GET_TMVTIME(<로봇 번호>, <축 번호>)

Description

제어기 출하 이후 모터가 동작 된 총 누적 시간을 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.
예) 1234일 12시간 34분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	12일 3시간 4분 56초
표현 시간	12:23:45:67

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    DEFSTR TMV_TIME                     // 확인을 위한 변수 선언
    INT AXIS                            // 총 누적 시간 변수에 저장
    AXIS = 1                            // ([0:0]12:3:4:56)
    TMV_TIME = GET_TMVTIME(1, AXIS)     // 메인 함수 종료
EOP
```

Related terms

GET_TWTIME, GET_TSVTIME, GET_TMVDIST

출하 이후 모터의 누적 이동 시간은 초기화 되지 않습니다.

11.29 GET_TMVDIST (출하 이후 모터 총 누적 이동거리 읽기)

Usage

출하 이후 모터 총 누적 이동거리를 확인 합니다.

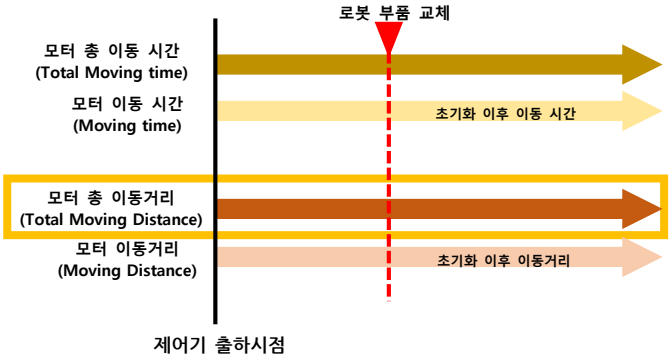
Syntax

<변수> = GET_TMVDIST(<로봇 번호>, <축 번호>)

Description

제어기 출하 이후 모터가 동작 된 총 누적 이동거리를 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 총 이동거리를 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <실수형 변수>
결과 값의 단위는 회전 축의 경우 degree, 직선 이동 축의 경우는 mm 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    REAL TMV_DIST                       // 확인을 위한 실수형 변수 선언

    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        TMV_DIST = GET_TMVDIST(1, AXIS) // 모터 총 누적 이동거리를 변수에 저장
        PRINT(AXIS, TMV_DIST)           // 저장된 변수 확인([0:0]55602.31)~ ([0:0]55602.31)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

GET_TWTIME, GET_TSVTIME, GET_TMVTIME

출하 이후 모터의 누적 거리는 초기화 되지 않습니다.

11.30 GET_WTIME (제어기 누적 통전 시간 읽기)

Usage

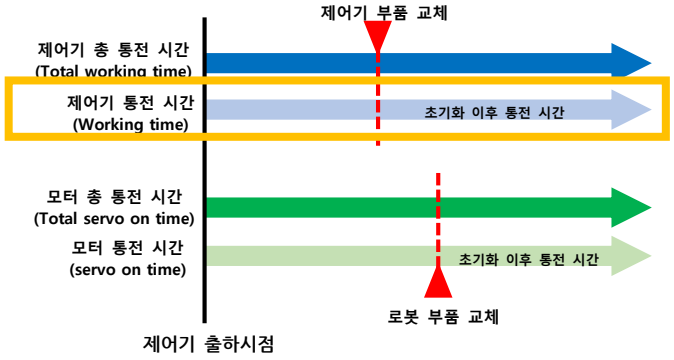
제어기 누적 통전시간을 확인 합니다.

Syntax

<변수> = GET_WTIME()

Description

제어기에 전원이 입력된 누적 시간을 확인합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.

예) 12일 3시간 4분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	12일 3시간 4분 56초
표현 시간	12:23:45:67

Examples

Example 1

```
MAIN                          // 메인 함수 선언
    DEFSTR W_TIME             // 확인을 위한 변수 선언
    W_TIME = GET_WTIME()      // 누적 통전 시간 변수에 저장(12일3시간4분56초)
    PRINT(0, W_TIME)          // 저장된 변수 확인([0:0]12:3:4:56)
EOP                           // 메인 함수 종료
```

Related terms

INIT_WTIME, GET_SVTIME, GET_MVTIME, GET_MVDIST

11.31 GET_SVTIME (모터 누적 통전 시간 읽기)

Usage

모터 누적 통전 시간을 확인 합니다.

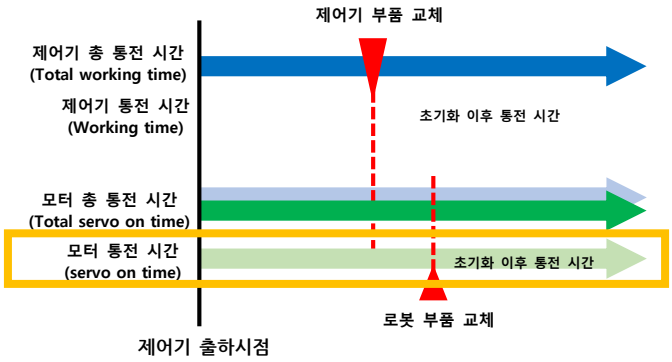
Syntax

<변수> = GET_SVTIME(<로봇 번호>, <축 번호>)

Description

모터에 전원이 인가 된 누적 시간을 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.

예) 1234일 12시간 34분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	1일 10시간 10분 25초
표현 시간	1:10:10:25

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    DEFSTR SV_TIME                      // 확인을 위한 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        SV_TIME = GET_SVTIME(1, AXIS)   // 모터 누적 통전 시간 변수에 저장
        PRINT(AXIS, SV_TIME)            // 저장 변수 확인([0:0]1:10:10:25) ~ ([0:0]1:10:10:25)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

INIT_SVTIME, GET_WTIME, GET_MVTIME, GET_MVDIST

11.32 GET_MVTIME (모터 누적 이동 시간 읽기)

Usage

모터 누적 이동 시간을 확인 합니다.

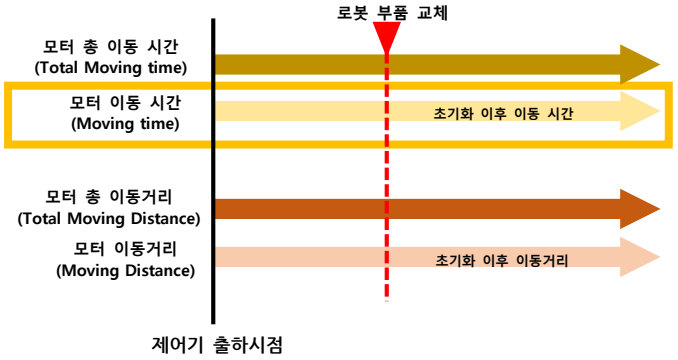
Syntax

<변수> = GET_MVTIME(<로봇 번호>, <축 번호>)

Description

모터가 동작 된 누적 시간을 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 이동 시간을 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 출력 됩니다.

예) 1234일 12시간 34분 56초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	12일 3시간 4분 56초
표현 시간	12:23:45:67

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    DEFSTR MV_TIME                      // 확인을 위한 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        MV_TIME = GET_MVTIME(1, AXIS)   // 모터 누적 이동 시간 변수에 저장
        PRINT(AXIS, MV_TIME)            // 저장 변수 확인([0:0]1:10:10:25) ~ ([0:0]1:10:10:25)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

INIT_MVTIME, GET_WTIME, GET_SVTIME, GET_MVDIST

11.33 GET_MVDIST (모터 누적 이동거리 읽기)

Usage

모터 누적 이동거리를 확인 합니다.

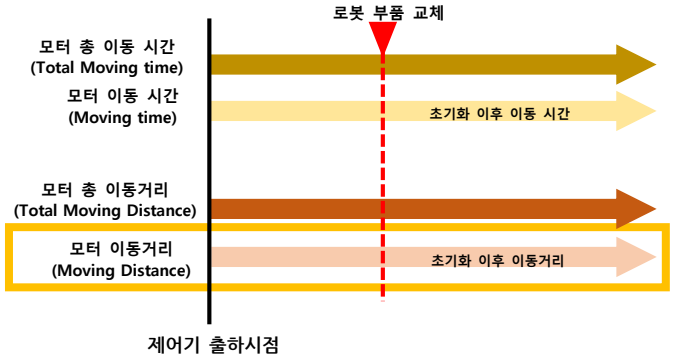
Syntax

<변수> = GET_MVDIST(<로봇 번호>, <축 번호>)

Description

모터가 동작 된 누적 이동거리를 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수>	Data type: <실수형 변수> 결과 값의 단위는 회전 축의 경우 degree, 직선 이동 축의 경우는 mm 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    REAL MV_DIST                        // 확인을 위한 실수형 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        MV_DIST = GET_MVDIST(1, AXIS)   // 모터 총 누적 이동거리를 변수에 저장
        PRINT(AXIS, MV_DIST)            // 저장 변수 확인([0:0]55602.31) ~ ([0:0]55602.31)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

INIT_MV_DIST, GET_WTIME, GET_SVTIME, GET_MVTIME

11.34 INIT_WTIME (제어기 누적 통전 시간 초기화)

Usage

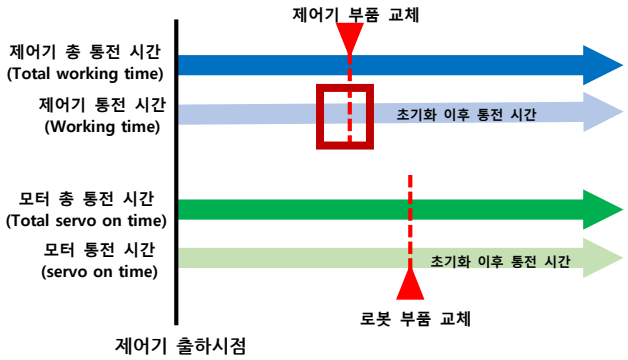
제어기 누적 통전 시간을 초기화 합니다.

Syntax

<변수> = INIT_WTIME()

Description

제어기 전원이 인가 된 누적 시간을 초기화 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 초기화된 값을 출력 합니다.

예) 0일 0시간 0분 0초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	0일 0시간 0분 0초
표현 시간	0:0:0:0

Examples

Example 1

```
MAIN                                // 메인 함수 선언
    DEFSTR W_TIME                   // 확인을 위한 변수 선언
    W_TIME = INIT_WTIME()           // 누적 통전 시간 변수에 저장(0일0시간0분0초)
    PRINT(0, W_TIME)                // 저장된 변수 확인([0:0]0:0:0:0)
EOP                                 // 메인 함수 종료
```

Related terms

GET_WTIME

출하 이후 제어기, 모터의 누적 통전시간, 이동시간, 거리에 대해서 초기화 되지 않습니다.

11.35 INIT_SVTIME (모터 누적 통전 시간 초기화)

Usage

모터 누적 통전 시간을 초기화 합니다.

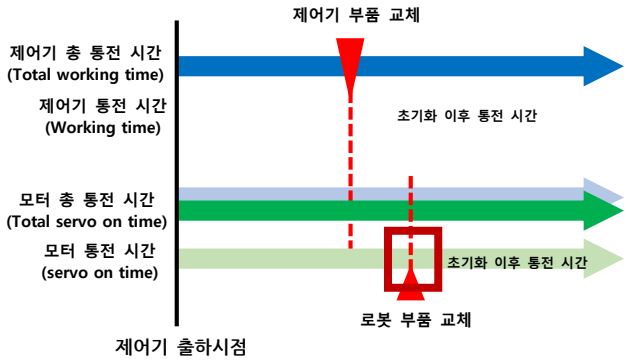
Syntax

<변수> = INIT_SVTIME(<로봇 번호>, <축 번호>)

Description

모터에 전원이 인가 된 누적 시간을 확인 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 초기화 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 초기화된 값을 출력 합니다.

예) 0일 0시간 0분 0초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	0일 0시간 0분 0초
표현 시간	0:0:0:0

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    DEFSTR SV_TIME                      // 확인을 위한 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        SV_TIME = INIT_SVTIME(1, AXIS)  // 모터 누적 통전 시간 변수에 저장
        PRINT(AXIS, SV_TIME)            // 저장된 변수 확인([0:0]0:0:0:0) ~ ([0:0]0:0:0:0)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

GET_SVTIME

11.36 INIT_MVTIME (모터 누적 이동 시간 초기화)

Usage

모터 누적 이동 시간을 초기화 합니다.

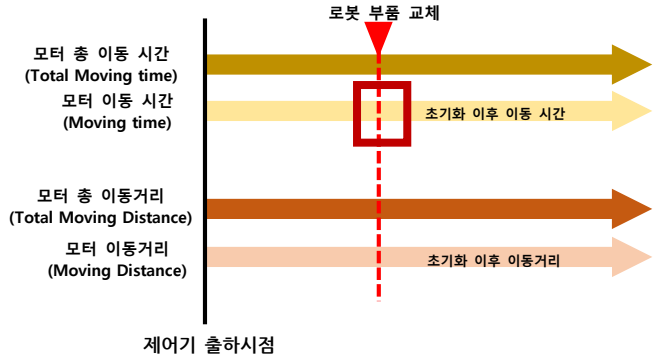
Syntax

<변수> = INIT_MVTIME(<로봇 번호>, <축 번호>)

Description

모터가 동작 된 총 누적 시간을 초기화 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수> Data type: <문자열 변수>
‘ : ’의 구분자로 “날짜:시간:분:초”의 문자열로 초기화된 값을 출력 합니다.

예) 0일 0시간 0분 0초의 출력 표시는 아래 표와 같습니다.

분류	설명
실제 시간	0일 0시간 0분 0초
표현 시간	0:0:0:0

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    DEFSTR MV_TIME                      // 확인을 위한 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        MV_TIME = INIT_MVTIME(1, AXIS)  // 동작 된 모터 시간을 변수에 저장
        PRINT(AXIS, MV_TIME)            // 저장된 변수 확인([0:0]0:0:0:0) ~ ([0:0]0:0:0:0)
    EXT                                 // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

GET_MVTIME

11.37 INIT_MVDIST (모터 누적 이동거리 초기화)

Usage

모터 누적 이동거리를 초기화 합니다.

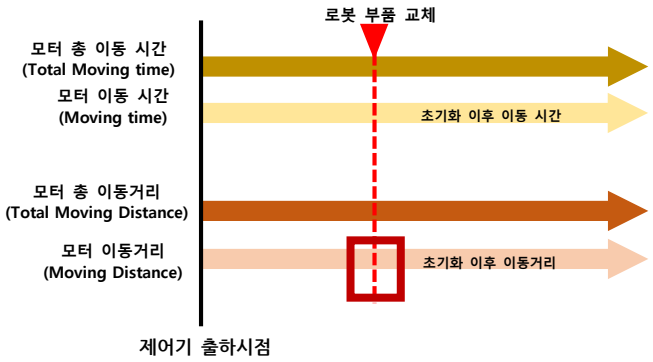
Syntax

<변수> = INIT_MVDIST(<로봇 번호>, <축 번호>)

Description

모터가 동작 된 총 누적 이동거리를 초기화 합니다.

1) <로봇 번호>를 입력하여 로봇을 선택합니다.
2) <축 번호>를 입력하여 통전 시간을 확인 할 축을 선택 합니다.



Arguments

<변수>	Data type: <실수형 변수> 결과 값의 단위는 회전 축의 경우 degree, 직선 이동 축의 경우는 mm 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT AXIS                            // 반복문을 위한 변수 선언
    REAL MV_DIST                        // 확인을 위한 실수형 변수 선언
    FOR AXIS = 1 TO 6 BY 1              // 설정 1번 축부터 6번축 까지 반복 설정
        MV_DIST = INIT_MVDIST(1, AXIS)  // 모터 총 누적 이동거리를 변수에 저장
        PRINT(AXIS, MV_DIST)            // 저장된 변수 확인([0:0]0.000) ~ ([0:0]0.000)
    NEXT                                // 반복문
EOP                                     // 메인 함수 종료
```

Related terms

GET_MVDIST

> 출하 이후 제어기, 모터의 누적 통전시간, 이동시간, 거리에 대해서 초기화 되지 않습니다.

12. System 모드 전용 명령어

12.1 PSEL (프로그램 선택)

Usage

지정한 로봇 Thread 번호에 입력한 JOB 파일을 매칭합니다.

Syntax

[<변수> =] PSEL(<Thread 번호>, "<JOB 파일명>")

Description

입력한 JOB 파일을 읽어와 파일에 오류가 존재하는 지 확인하고, 지정한 로봇 Thread 에 JOB 파일을 매칭합니다.

※ 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용할 수 없습니다.

※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(선택 성공) 또는 음수 값(선택 실패)이 반환됩니다.

- 선택 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PSEL 명령어를 사용한 경우
-2	입력 데이터 오류
-3	입력한 JOB 파일이 존재하지 않는 경우
-4	입력한 JOB 파일 명이 "BGTD" 또는 "BGTF"인 경우
-5	선택한 Thread가 실행 중인 경우
-6	JOB 파일 로드가 실패한 경우

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

<JOB 파일명> Data type: <문자열> 또는 <문자열 변수>
<JOB 파일명>에는 파일 마침표 "." 와 확장자 "JOB"을 제외한 파일의 이름만 작성해 주십시오.

Examples

```
MAIN                                        // 메인 함수 선언
    INT R_VAL, STATE                        // 정수형 변수 선언

    R_VAL = PSEL(1,"MOVE1")                 // 로봇1번 Thread에 “MOVE1” 파일선택

    IF (R_VAL<0) THEN                       // 반환 값 확인 시 음수인 경우
        GOTO END_PROGRAM                    // 선택 시 실패, 분기점으로 이동
    ENDIF                                   // 조건문 종료
    WHILE 1                                 // 반복문 실행
        STATE = PSTATE(1)                   // 프로그램 실행 상태 읽기
        IF ((STATE == 2)&&(IN0 == 1))THEN   // 상태 값이 2(TH_STOP)이고, 0번 입력
                                            이 1인 경우
            R_VAL = PSEL(1,"MOVE2")         // 로봇1번 Thread에 “MOVE2” 파일선택
            IF (R_VAL<0) THEN               // 반환 값 확인 시 음수인 경우
                GOTO END_PROGRAM            // 선택 시 실패, 분기점으로 이동
            ENDIF                           // 조건문 종료
        ELSEIF ((STATE == 2)&&(IN1 == 1))THEN // 상태 값이 2(TH_STOP)이고, 1번 입력
                                            이 1인 경우
            R_VAL = PSEL(1,"MOVE1")         // 로봇1번 Thread에 “MOVE1” 파일선택
            IF (R_VAL<0) THEN               // 반환 값 확인 시 음수인 경우
              GOTO END_PROGRAM              // 선택 시 실패, 분기점으로 이동
            ENDIF                           // 조건문 종료
        ELSEIF ((STATE == 2)&&(IN2 == 1)) THEN // 상태 값이 2(TH_STOP)이고, 2번 입력
            PSTART(1)                       이 1인 경우, 프로그램 수행 시작
        ELSEIF ((STATE == 3)&&(IN3 == 1)) THEN // 상태 값이 3(TH_RUN)이고, 3번 입력이
            PSTOP(1)                        1인 경우, 프로그램 정지
        ELSEIF ((STATE == 2)&&(IN4 == 1)) THEN // 상태 값이 2(TH_STOP)이고, 4번 입력
            PRESTART(1)                     이 1인 경우, 프로그램 재 시작
        ELSEIF ((STATE == 2)&&(IN5 == 1)) THEN // 상태 값이 2(TH_STOP)이고, 5번 입력
            PEXIT(1)                        이 1인 경우, 프로그램 해제
        ELSEIF STATE == 0 THEN              // 상태 값이 0(TH_ERROR)인 경우
            GOTO END_PROGRAM                // 종료 분기점으로 이동
        ENDIF                               // 조건문 종료
    ENDWL                                   // 반복문 종료
    LABL END_PROGRAM                        // 프로그램 종료 분기점 선언
EOP                                         // 메인 함수 종료
```

Related terms

PSTART, PSTOP, PSTATE, PRESTART, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 “Cannot use PSEL at foretask.” 알람을 발생합니다.

12.2 PSTART (프로그램 시작)

Usage

지정한 로봇 Thread 번호에 선택된 JOB 프로그램을 실행합니다.

Syntax

[<변수> =] PSTART(<Thread 번호>)

Description

지정한 로봇 Thread 번호에 로드 된 프로그램을 첫 줄부터 실행합니다.

1) <Thread 번호>에 값을 입력해 JOB 프로그램을 실행합니다.
※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(실행 성공) 또는 음수 값(실행 실패)이 반환됩니다.

- 실행 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PSTART명령어를 사용한 경우
-2	입력 데이터 오류
-3	Thread 실행이 불가능하거나 이미 실행 중인 경우

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
MAIN                                  // 메인 함수 선언
    INT R_VAL                         // 정수형 변수 선언
    PSEL(1, “MOVE”)                   // 로봇 1번 Thread 에 “MOVE” 파일 선택
    R_VAL = PSTART(1)                 // 1번 Thread에 매칭된 프로그램 실행 시작
    PRINT(0, “RETURN VALUE: ”,R_VAL)  // 모니터링 로그 0번에 반환 값 출력
    WHILE 1                           // 반복문 시작
    ENDWL                             // 반복문 종료
EOP                                   // 메인 함수 종료
```

Related terms

PSEL, PSTOP, PSTATE, PRESTART, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 “Cannot use PSEL at foretask.” 알람을 발생합니다.

12.3 PSTOP (프로그램 실행 정지)

Usage

지정한 로봇 Thread 번호에 입력한 JOB 파일의 실행을 정지합니다.

Syntax

[<변수> =] PSTOP(<Thread 번호>)

Description

지정한 로봇 Thread 번호에서 실행되는 JOB 프로그램을 정지합니다.
1) <Thread 번호>에 값을 입력해 실행 중인 JOB 프로그램을 정지합니다.
※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(정지 성공) 또는 음수 값(정지 실패)이 반환됩니다.

- 정지 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PSTOP 명령어를 사용한 경우
-2	입력 데이터 오류

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
MAIN                        // 메인 함수 선언
    PSEL(1, "MOVE")         // 로봇 1번 Thread 에 "MOVE" 파일 선택
    PSTART(1)               // 로봇 1번 Thread에 매칭된 프로그램 실행 시작
    WHILE 1                 // 반복문 시작
        IF IN0 == 1 THEN    // 0번 입력 값이 1인 경우
            PSTOP(1)        // 로봇 1번 Thread에 매칭된 프로그램 정지
        ENDIF               // 조건문 종료
    ENDWL                   // 반복문 종료
EOP                         // 메인 함수 종료
```

Related terms

PSEL, PSTART, PSTATE, PRESTART, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 "Cannot use PSEL at foretask." 알람을 발생합니다.

12.4 PSTATE (프로그램 상태 확인)

Usage

지정한 로봇 Thread 번호의 상태를 읽어옵니다.

Syntax

<변수> = PSTATE(<Thread 번호>)

Description

지정한 로봇 Thread 번호의 현재 상태를 읽어옵니다.

1) <Thread 번호>에 값을 입력해 상태를 읽어옵니다.
※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Return value

<변수> Data type: <정수형 변수>
로봇 Thread의 현재 상태 정보를 반환합니다.

- 0 (TH_ERROR): Thread 번호 오류
- 1 (TH_IDLE): Thread 할당 없음, 신규 사용 가능 상태
- 2 (TH_STOP): Thread 정지
- 3 (TH_RUN): Thread 구동 중

상태 확인 실패 시 음수 값이 반환됩니다.

- 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PSTATE 명령어를 사용한 경우
-2	입력 데이터 오류

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT STATE                           // 정수형 변수 선언

    PSEL(1, “MOVE”)                     // 로봇 1번 Thread 에 “MOVE” 파일 선택
    PSTART(1)                           // 로봇 1번 Thread에 매칭된 프로그램 실행 시작
    WHILE 1                             // 반복문 시작
        STATE = PSTATE(1)               // 로봇 1번 Thread의 상태 확인
        IF STATE < 0 THEN               // 상태 확인에 실패한 경우
            PRINT(0,”ERROR:”, STATE)    // 모니터링 로그 0번에 실패 원인 출력
            BREAK                       // 반복문 강제 종료
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

PSEL, PSTART, PSTOP, PRESTART, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 “Cannot use PSEL at foretask.” 알람을 발생합니다.

12.5 PRESTART (프로그램 재 시작)

Usage

지정한 로봇 Thread 번호에 정지된 JOB 프로그램을 재실행합니다.

Syntax

[<변수> =] PRESTART(<Thread 번호>)

Description

지정한 로봇 Thread 번호에 로드 된 프로그램을 정지한 줄부터 실행합니다.

1) <Thread 번호>에 값을 입력해 JOB 프로그램을 재 실행합니다.
※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(실행 성공) 또는 음수 값(실행 실패)이 반환됩니다.

- 실행 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PRESTART 명령어를 사용한 경우
-2	입력 데이터 오류

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
MAIN                                        // 메인 함수 선언
    INT STATE                               // 정수형 변수 선언

    PSEL(1, "MOVE")                         // 로봇 1번 Thread 에 “MOVE”파일
    PSTART(1)                               선택
                                            // 1번 Thread에 매칭된 프로그램 시작
    WHILE 1                                 // 반복문 시작
        STATE = PSTATE(1)                   // 로봇 1번 Thread의 상태 확인
        IF ((STATE == 3)&&(IN0 == 1)) THEN  // 상태 값이 3(TH_RUN)이고, 0번 입력
            PSTOP(1)                        이 1인 경우, 프로그램 정지
        ELSEIF ((STATE == 2)&&(IN1 == 1)) THEN // 상태 값이 2(TH_STOP)이고, 1번 입
            PRESTART(1)                     력이 1인 경우, 프로그램 재 시작
        ELSEIF ((STATE == 2)&&(IN2 == 1)) THEN // 상태 값이 2(TH_STOP)이고, 2번 입
            PEXIT (1)                       력이 1인 경우, 프로그램 해제
        ENDIF                               // 조건문 종료
    ENDWL                                   // 반복문 종료
EOP                                         // 메인 함수 종료
```

Related terms

PSEL, PSTART, PSTOP, PSTATE, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 “Cannot use PSEL at foretask.” 알람을 발생합니다.

12.6 PEXIT (프로그램 해제)

Usage

지정한 로봇 Thread 번호의 JOB 을 해제합니다.

Syntax

[<변수> =] PEXIT(<Thread 번호>)

Description

지정한 로봇 Thread 번호에 로드 된 프로그램을 해제합니다.

Return value

<변수> Data type: <정수형 변수>
결과 값은 0(실행 성공) 또는 음수 값(실행 실패)이 반환됩니다.

- 실행 실패 원인

값	원인
-1	T/P 모드 스위치가 Manual 모드인 경우 또는, Robot 모드에서 PEXIT 명령어를 사용한 경우
-2	입력 데이터 오류

※ 현재 Thread 번호는 TH1(1)만 선택 가능합니다.

Arguments

<Thread 번호> Data type: <정수형 변수/상수>
Thread 번호의 범위는 1 ~ 4(TH1 ~ TH4) 입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT STATE                           // 정수형 변수 선언
    PSEL(1, "MOVE")                     // 로봇 1번 Thread에 “MOVE” 파일 선택
    PSTART(1)                           // 1번 Thread에 매칭된 프로그램 실행 시작
    WHILE 1                             // 반복문 시작
        STATE = PSTATE(1)               // 로봇 1번 Thread의 상태 확인
        IF ((STATE == 2)&&(IN1 == 1)) THEN  // 상태 값이 2(TH_STOP)이고, 1번 입력이 1
                                            인 경우
            PSTOP(1)                    // 프로그램 종료
            PEXIT(1)                    // 프로그램 해제
        ENDIF                           // 조건문 종료
    ENDWL                               // 반복문 종료
EOP                                     // 메인 함수 종료
```

Related terms

PSEL, PSTART, PSTOP, PSTATE, PEXIT

1) 이 명령어는 제어기의 System 모드 상태에서만 사용이 가능하며, Robot 모드에서 사용하는 경우 “Cannot use PSEL at foretask.” 알람을 발생합니다.

13. 외부 통신

13.1 NET_OPEN (TCP/IP Client 소켓 생성 및 접속 요청)

Usage

TCP/IP Client 소켓을 생성하고 Server로 접속을 요청하는 명령어 입니다.

Syntax

<변수> = NET_OPEN(<IP 번호>, <Port 번호>, <접속 요청 대기 시간>)

Description

TCP/IP Client 소켓을 생성하고 입력된 IP, Port 번호에 해당하는 TCP/IP Server로 접속 요청합니다. 동시에 3개의 Port가 연결 가능하며, 추가 생성이 불가능합니다.

Return value

<변수>	Data type: <정수형 변수> 정상 연결 소켓의 File descriptor를 반환하고 연결 실패 시 -1을 반환합니다.

Arguments

<IP 번호>	Data type: <문자열> 또는 <문자열 변수> Server의 IP 번호를 입력합니다.
<Port 번호>	Data type: <정수형 변수/상수> <Port 번호>의 범위는 1 ~ 65,535 입니다.
<접속 요청 대기 시간>	Data type: <정수형 변수/상수> <접속 요청 대기 시간>의 단위는 msec이며, 범위는 1 ~ 10,000 입니다.

Examples

```
MAIN                                    // 메인 함수 선언
    INT FD_NUM                          // 반환 값 정수형 변수 선언
    DEFSTR IP_NUM                       // IP 번호 문자열 변수 선언
    INT PORT_NUM                        // Port 번호 정수형 변수 선언
    INT WAIT_TIME                       // 접속 요청 대기 시간 정수형 변수 선언

    IP_NUM = “192.168.1.196”            // IP 번호 정의
    PORT_NUM = 10000                    // Port 번호 정의
    WAIT_TIME = 1000                    // 접속 요청 시간 정의

    FD_NUM = NET_OPEN(IP_NUM,           // 함수 실행
    PORT_NUM, WAIT_TIME)
EOP                                     // 메인 함수 종료
```

Related terms

NET_CLOSE, NET_STATUS, NET_KEEPALIVE, NET_DELIMITER, NET_FD

13.2 NET_CLOSE (TCP/IP 통신 종료)

Usage

제어기의 연결된 TCP/IP 통신을 종료하는 명령어 입니다.

Syntax

[<변수>] = NET_CLOSE(<FD>)

Description

제어기의 연결된 TCP/IP 통신을 종료합니다.

1) <FD>는 NET_FD 또는 NET_OPEN 명령어로 반환 받은 File Descriptor를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 통신이 정상 종료일 때 0, 오류 발생 시 -1을 반환합니다.

Arguments

<FD>	Data type: <정수형 변수/상수> <FD>의 범위는 -1 ~ 1,024 입니다.

Examples

```
MAIN                                // 메인 함수 선언
    INT FD_NUM                      // FD 번호 정수형 변수 선언

    ……                              // FD 번호는 NET_OPEN, 또는 NET_FD 로 반환 받은 값

    NET_CLOSE(FD_NUM)               // 함수 실행
EOP                                 // 메인 함수 종료
```

Related terms

NET_OPEN, NET_STATUS, NET_KEEPALIVE, NET_DELIMITER, NET_FD

1) FD(File descriptor) 는 시스템에서 할당 받은 소켓의 고유 번호를 의미합니다.
2) TCP/IP Server의 경우 NET_FD 명령어를 통해 반환 받을 수 있으며, Client 의 경우 NET_OPEN 명령어를 통해 반환 받을 수 있습니다.

13.3 NET_STATUS (TCP/IP 통신 상태 확인)

Usage

제어기의 TCP/IP 통신상태를 확인하는 명령어 입니다.

Syntax

<정수형 변수> = NET_STATUS(<FD>)

Description

제어기의 TCP/IP 통신상태를 확인해 반환해줍니다.

1) <FD>는 NET_FD 또는 NET_OPEN 명령어로 반환 받은 File Descriptor를 입력합니다.

Return value

<변수>	Data type: <정수형 변수> 통신이 정상일 때 0, 통신이 연결되지 않았을 때나 오류 발생 시 -1을 반환합니다.

Arguments

<FD>	Data type: <정수형 변수/상수> <FD>의 범위는 -1 ~ 1,024 입니다.

Examples

```
MAIN                                  // 메인 함수 선언
    INT FD_NUM                        // FD 번호 정수형 변수 선언
    INT RET_VAL                       // 반환 값 정수형 변수 선언

    ……                                // FD 번호는 NET_OPEN, 또는 NET_FD 로 반환 받은 값

    RET_VAL = NET_STATUS(FD_NUM)      // 함수 실행
EOP                                   // 메인 함수 종료
```

Related terms

NET_OPEN, NET_CLOSE, NET_KEEPALIVE, NET_DELIMITER, NET_FD

1) FD(File descriptor) 는 시스템에서 할당 받은 소켓의 고유 번호를 의미합니다.
2) TCP/IP Server의 경우 NET_FD 명령어를 통해 반환 받을 수 있으며, Client 의 경우 NET_OPEN 명령어를 통해 반환 받을 수 있습니다.

13.4 NET_KEEPALIVE (TCP/IP Client 세션체크 옵션 설정)

Usage

TCP/IP 통신 중 주기적으로 세션을 체크하여 LAN 케이블의 단선 혹은 비정상적인 세션 종료에 대해 감지하는 명령어 입니다.

Syntax

<정수형 변수> = NET_ KEEPALIVE(<FD>, <세션 체크 주기>, <반복 횟수>, <반복 주기>)

Description

<세션 체크 주기> 마다 세션 체크 패킷을 전송하고 응답이 없을 경우 <반복 주기> 마다 세션 체크 패킷을 전송, <반복 횟수> 동안 응답이 없으면 통신을 종료합니다.

1) <FD>는 NET_FD 또는 NET_OPEN 명령어로 반환 받은 File Descriptor를 입력합니다.
2) <세션 체크 주기>는 sec 단위로 소켓 생성 후 최초 세션 체크의 주기를 의미합니다.
3) <반복 횟수>는 세션 체크 패킷에 응답하지 않을 경우 반복 횟수를 의미합니다.
4) <반복 주기>는 sec 단위로 최초 세션 체크에 응답하지 않을 경우 세션 체크 반복 횟수 주기를 의미합니다.

EX) <세션 체크 주기: 10>, <반복 횟수: 20>, <반복 주기: 3> 입력 시 소켓 생성 후 10초마다 세션 체크 패킷을 전송하고 응답이 없을 경우 3초의 주기로 세션 체크 패킷을 전송합니다. 20회 동안 패킷을 전송하며 횟수를 초과하면 통신을 종료합니다.
디폴트로 10, 20, 3의 값이 설정되어 있으며 NET_KEEPALIVE 명령어를 통해 설정 값들을 변경할 수 있습니다.

Return value

<변수>	Data type: <정수형 변수> 정상 동작 시 0, 실패 시 -1을 반환합니다.

Arguments

<FD>	Data type: <정수형 변수/상수> <FD>의 범위는 -1 ~ 1,024 입니다.
<세션 체크 주기>	Data type: <정수형 변수/상수> <세션 체크 주기>의 단위는 sec 이며, 범위는 1 ~ 10,000 입니다.
<반복 횟수>	Data type: <정수형 변수/상수> <반복 횟수>의 단위는 정수이며, 범위는 1 ~ 10,000 입니다.
<반복 주기>	Data type: <정수형 변수/상수> <반복 주기>의 단위는 sec 이며, 범위는 1 ~ 10,000 입니다.

Examples

```
MAIN                                  // 메인 함수 선언
    INT RET_VAL                       // 반환 값 정수형 변수 선언
    INT FD_NUM                        // FD 번호 정수형 변수 선언
    INT KEEP_IDLE                     // 최초 세션 주기 정수형 변수 선언
    INT KEEP_CNT                      // 반복 횟수 정수형 변수 선언
    INT KEEP_INTERVAL                 // 반복 주기 정수형 변수 선언

    ……                                // FD 번호는 NET_OPEN, 또는 NET_FD 로 반환
                                      받은 값

    KEEP_IDLE = 10                    // 최초 세션 주기 10초로 정의
    KEEP_CNT = 20                     // 반복 횟수 20회로 정의
    KEEP_INTERVAL = 3                 // 반복 주기 3초로 정의

    RET_VAL = NET_KEEPALIVE(FD        // 함수 실행
    _NUM,KEEP_IDLE,KEEP_CNT,
     KEEP_INTERVAL)
EOP                                   // 메인 함수 종료
```

Related terms

NET_OPEN, NET_CLOSE, NET_STATUS, NET_DELIMITER, NET_FD

1) FD(File descriptor) 는 시스템에서 할당 받은 소켓의 고유 번호를 의미합니다.
2) TCP/IP Server의 경우 NET_FD 명령어를 통해 반환 받을 수 있으며, Client 의 경우 NET_OPEN 명령어를 통해 반환 받을 수 있습니다.

13.5 NET_FD (TCP/IP Server FD 반환)

Usage

제어기 TCP/IP Server에 접속한 Client Socket의 FD를 반환합니다.

Syntax

<정수형 변수> = NET_FD(<인덱스>)

Description

<인덱스>에 해당하는 TCP/IP Server Port에 접속한 Client Socket의 FD를 반환합니다. 연결이 되어 있다면 해당 Client Socket의 FD를, 연결되어 있지 않다면 -1를 반환합니다.

1) <인덱스>는 10 ~ 12의 값을 입력하며 아래의 값을 의미합니다.
   10: Port 번호 38201
   11: Port 번호 38202
   12: Port 번호 38203

Return value

<변수>	Data type: <정수형 변수/상수> 수신대기 시간 동안 수신된 문자열을 반환합니다.

Arguments

<인덱스>	Data type: <정수형 변수/상수> <인덱스>의 범위는 10 ~ 12 입니다.

Examples

```
MAIN                                // 메인 함수 선언
    INT FD_NUM                      // 반환 값 정수형 변수 선언
    INT INDEX                       // Index 값 정수형 변수 선언

    INDEX = 10                      // Index 값 설정

    FD_NUM = NET_FD(INDEX)          // 함수 실행
EOP                                 // 메인 함수 종료
```

Related terms

NET_OPEN, NET_CLOSE, NET_STATUS, NET_KEEPALIVE, NET_DELIMITER

13.6 NET_STRIN (통신 문자열 수신)

Usage

이더넷 통신을 통하여 입력 된 문자열을 읽어옵니다.

Syntax

<문자열 변수> = NET_STRIN(<파일 기술자 번호>, <대기시간>)

Description

입력한 대기시간(단위: ms)만큼 통신 입력을 기다립니다. 설정한 대기시간 동안 데이터가 들어오지 않으면, 시스템 영역에 대기 상태를 체크하고 다음 스텝의 명령을 수행합니다.

NET_STRIN 함수를 통해 입력되는 데이터는 문자열의 끝을 판단하기 위해 CR+LF(기본 설정) 또는 CR, LF 문자를 사용하며 이를 새줄 문자라 합니다. 새줄 문자는 반드시 문자열의 끝에 위치해야 하며, 새줄 문자가 없는 경우 입력의 종료를 판단하지 못합니다.

1) <파일 기술자 번호>에 NET_FD 또는 NET_OPEN 명령어로 반환 받은 파일 기술자(File descriptor)번호를 입력합니다.
2) <대기시간>은 수신대기 시간을 설정합니다.

Return value

<변수>	Data type: <문자열 변수/상수> 수신대기 시간 동안 수신된 문자열을 반환합니다.

Arguments

<파일 기술자 번호>	Data type: <정수형 변수/상수> 파일 기술자 번호의 범위는 -1 ~ 1,024 입니다.
<대기시간>	Data type: <정수형 변수/상수>

Examples

```
MAIN                                    // 메인 함수 선언
    DEFSTR RECVPK                       // 문자열 변수 선언
    INT FD_NUM                          // 정수형 변수 선언
    FD_NUM = NET_FD(10)                 // 10번 인덱스의 FD값 반환
    RECVPK = NET_STRIN(FD_NUM,1000)     // 문자열 수신을 1초 동안 대기
EOP                                     // 메인 함수 종료
```

Related terms

NET_OPEN, NET_CLOSE, NET_STATUS, NET_STROUT, NET_KEEPALIVE, NET_FD

13.7 NET_STROUT (통신 문자열 송신)

Usage

시리얼 및 이더넷 통신을 통하여 문자열을 송신합니다.

Syntax

<변수> = NET_STROUT(<파일 기술자 번호>, <문자열>)

Description

입력한 문자열을 시리얼 통신을 통해 출력을 내보내며, 전송하지 못한 문자 수 반환합니다. 예를 들어, “ABCDEFGH”의 문자를 전송하는 경우 성공 시 반환 값은 0이며 전부 전송하지 못한 경우 반환 값은 8 입니다. 또, 전송 실패 시 음수를 반환합니다.

NET_STROUT 함수를 통해 출력되는 데이터는 문자열의 끝을 판단하기 위해 CR+LF(기본 설정) 또는 CR, LF 문자를 자동으로 추가하여 전송하며, 이를 새줄 문자라 합니다.

1) <파일 기술자 번호>에 NET_FD 또는 NET_OPEN 명령어로 반환 받은 파일 기술자(File descriptor)번호를 입력합니다.
2) <문자열>은 전송할 문자열을 입력합니다.

Return value

<변수>	Data type: <정수형 변수/상수> 전송이 정상적으로 완료된 경우 : 0 반환 전송이 완료되지 않은 경우 : 전송하지 못한 문자 개수 반환 전송 오류가 발생한 경우 : 음수 반환

Arguments

<파일 기술자 번호>	Data type: <정수형 변수/상수> 파일 기술자 번호의 범위는 -1 ~ 1,024 입니다.
<문자열>	Data type: <문자열 변수/상수>

Examples

```
MAIN                                    // 메인 함수 선언
    INT FD_NUM                          // 정수형 변수 선언
    DEFSTR SENDPK                       // 문자열 변수 선언
    INT LEN                             // 정수형 변수 선언
    FD_NUM = NET_FD(10)                 // 10번 인덱스의 FD값 반환
    SENDPK = “SUCCESS”                  // 전송 문자열 초기화
    I0 = NET_STROUT(FD_NUM,“TEST”)      // “TEST” 문자열 전송
    LEN = NET_STROUT(FD_NUM,SENDPK)     // “SUCCESS”문자열 전송
EOP                                     // 메인 함수 종료
```

Related terms

NET_OPEN, NET_CLOSE, NET_STATUS, NET_STRIN, NET_KEEPALIVE, NET_FD

14. 영역 검출 전용 명령어

14.1 SET_INRANGE_ENB (In Range 기능 활성화 명령어)

Usage

In Range 기능을 활성화 또는 비활성화하는 명령어입니다.

Syntax

SET_IN RANGE_ENB(<로봇 번호>, <In Range 번호>, <활성화 Flag>)

Description

※ JOB 명령어를 통한 설정은 파라미터와 별도로 JOB 운용 시에만 적용됩니다.

로봇 파라미터 설정
4.PARA → (F1).ROBOT → 6.INRANGE(1/3) → #.INRNG# → 1.ENB
(#: 1~16 까지 숫자)

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<활성화 Flag>	Data type: <정수형 변수/상수> <활성화 Flag>의 범위는 0 ~ 1까지 입니다. <활성화 Flag> 는 0일 경우 비활성화, 1일 경우 활성화를 의미합니다.

Examples

```
MAIN                                    // 메인 함수 선언
    SET_INRANGE_ENB(1, 1, 0)            // In Range 기능 비활성화
    ......
    SET_INRANGE_ENB(1, 1, 1)            // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM,GET_INRANGE_OUT, GET_INRANGE_VECTOR

1) In Range 기능이 활성화 되어 있는 경우 알람이 발생합니다. SET_IN RANGE_ENB 명령어를 통해 In Range 비활성화 후 사용하기 바랍니다.

14.2 SET_INRANGE_POS (In Range 검출 영역 설정 명령어)

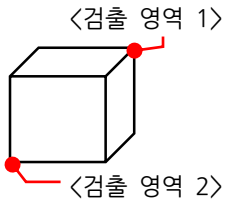
Usage

In Range 검출 영역을 설정하는 명령어입니다.

Syntax

SET_INRANGE_POS(<로봇 번호>, <In Range 번호>, <검출영역 1>, <검출영역 2>)

Description



<검출영역 1>에 검출영역의 한 꼭지점을 입력합니다.
<검출영역 2>에 가장 먼 꼭지점을 입력합니다.

※ JOB 명령어를 통한 설정은 파라미터와 별도로 JOB 운용 시에만 적용됩니다.

로봇 파라미터 설정 방법
4.PARA → (F1).ROBOT → 6.INRANGE(1/3) → #.INRNG# → 6.RANGE
(#: 1~16 까지 숫자)

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<검출 영역 1>	Data type: <베이스 좌표계 위치형 변수> 또는 <사용자 좌표계 위치형 변수>
<검출 영역 2>	Data type: <베이스 좌표계 위치형 변수> 또는 <사용자 좌표계 위치형 변수>

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    XPOS XP1, XP2                       // 베이스 좌표계 위치형 변수 선언
    XP1=<-10,-10,-10,0,0,0,ARM=0,T=1>   // 검출 영역 1 설정
    XP2=<10,10,10,0,0,0,ARM=0,T=1>      // 검출 영역 2 설정

    SET_IN RANGE_ENB(1, 1, 0)           // In Range 기능 비활성화
    SET_IN RANGE_POS(1, 1, XP1, XP2)    // In Range 범위를 XP1,XP2로 설정

    SET_IN RANGE_ENB(1, 1, 1)           // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    UPOS UP1, UP2                       // 사용자 좌표계 위치형 변수 선언
    UP1=<-10,-10,-10,0,0,0,ARM=0,T=1, U=3>  // 검출 영역 1 설정
    UP2=<10,10,10,0,0,0,ARM=0,T=1,U=3>      // 검출 영역 2 설정

    SET_IN RANGE_ENB(1, 1, 0)           // In Range 기능 비활성화
    SET_IN RANGE_POS(1, 1, UP1, UP2)    // In Range 범위를 UP1, UP2로 설정

    SET_IN RANGE_ENB(1, 1, 1)           // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

1) In Range 활성화 상태에서 영역 검출 전용 명령어를 설정하면, 알람이 발생합니다. SET_INRANGE_ENB 명령어를 통해 In Range 비활성화 후 설정하기 바랍니다.

14.3 SET_INRANGE_ALARM (In Range 검출 알람 설정 명령어)

Usage

In Range 검출 알람을 설정하는 명령어입니다.

Syntax

SET_INRANGE_ALARM(<로봇 번호>, <In Range 번호>, <알람 Flag>)

Description

알람 활성화 시 로봇 툴 끝이 검출 영역 안으로 도달할 경우 알람이 발생합니다.
검출 영역 안에서는 알람 해제가 가능합니다.
검출 영역 안에서 알람 해제 후, 검출 영역의 중심점 방향으로 움직이는 경우 재차 알람이 발생하고, 반대 방향으로 움직이는 경우 발생하지 않습니다.
※ JOB 명령어를 통한 설정은 파라미터와 별도로 JOB 운용 시에만 적용됩니다.

로봇 파라미터 설정 방법
4.PARA → (F1).ROBOT → 6.INRANGE(1/3) → #.INRNG# → 5.ALARM
(#: 1~16 까지 숫자)

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<알람 Flag>	Data type: <정수형 변수/상수> <알람 Flag>의 범위는 0 ~ 1까지 입니다. <알람 Flag>는 0일 경우 비활성화, 1일 경우 활성화를 의미합니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    SET_IN RANGE_ENB(1, 1, 0)           // In Range 기능 비활성화
    SET_IN RANGE_ALARM(1,1, 1)          // In Range 알람 기능 활성화

    SET_IN RANGE_ENB(1, 1, 1)           // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

1) In Range 활성화 상태에서 영역 검출 전용 명령어를 설정하면, 알람이 발생합니다. SET_INRANGE_ENB 명령어를 통해 In Range 비활성화 후 설정하기 바랍니다.

14.4 SET_INRANGE_OUT (In Range 검출 시 출력 접점 설정 명령어)

Usage

In Range 검출 시 출력 접점 번호를 설정하는 명령어입니다.

Syntax

SET_INRANGE_OUT(<로봇 번호>, <In Range 번호>, <접점 번호>)

Description

JOB 수행 중 In Range 검출 출력 접점 번호를 설정할 수 있습니다.
초기 값은 19로 고정되어 있으며, 최대 접점 번호보다 크게 설정할 경우 알람이 발생됩니다.
※ JOB 명령어를 통한 설정은 파라미터와 별도로 JOB 운용 시에만 적용됩니다.

로봇 파라미터 설정 방법
4.PARA → (F1).ROBOT → 6.INRANGE(1/3) → #.INRNG# → 2.I/O
(#: 1~16 까지 숫자)

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<접점 번호>	Data type: <정수형 변수/상수> <접점 번호>의 범위는 I/O Type 에 따라 달라질 수 있습니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    SET_INRANGE_ENB(1, 1, 0)            // In Range 기능 비활성화
    SET_INRANGE_OUT(1, 1, 19)           // In Range 출력 접점 번호 설정
    SET_INRANGE_ENB(1, 1, 1)            // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

1) In Range 활성화 상태에서 영역 검출 전용 명령어를 설정하면, 알람이 발생합니다. SET_INRANGE_ENB 명령어를 통해 In Range 비활성화 후 설정하기 바랍니다.

14.5 SET_INRANGE_MEASURE (In Range 검출 영역 데이터 타입 설정 명령어)

Usage

In Range 검출 영역의 데이터 타입을 설정하는 명령어입니다.

Syntax

SET_INRANGE_MEASURE(<로봇 번호>, <In Range 번호>, <데이터 타입>)

Description

JOB 수행 중 In Range 검출 영역의 <데이터 타입>을 선택합니다.
Feedback 좌표는 현재 값을 의미하며, Reference 좌표는 지령치를 의미합니다.
※ JOB 명령어를 통한 설정은 파라미터와 별도로 JOB 운용 시에만 적용됩니다.

로봇 파라미터 설정 방법
4.PARA → (F1).ROBOT → 6.INRANGE(1/3) → #.INRNG# → 3.MEASURE
(#: 1~16 까지 숫자)

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<데이터 타입>	Data type: <정수형 변수/상수> <데이터 타입>의 범위는 0 ~ 1까지 입니다. <데이터 타입>은 0일 경우 Feedback좌표, 1일 경우 Reference좌표를 설정합니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    SET_INRANGE_ENB(1, 1, 0)            // In Range 기능 비활성화
    SET_INRANGE_MEASURE(1, 1, 0)        // In Range 검출 영역 Feedback좌표 설정

    SET_INRANGE_ENB(1, 1, 1)            // In Range 기능 활성화
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM,
SET_INRANGE_OUT, GET_INRANGE_MIN, GET_INRANGE_MAX,
GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

1) In Range 활성화 상태에서 영역 검출 전용 명령어를 설정하면, 알람이 발생합니다.
SET_INRANGE_ENB 명령어를 통해 In Range 비활성화 후 설정하기 바랍니다.

14.6 GET_INRANGE_MIN (In Range 검출 영역 최소 좌표값 반환 명령어)

Usage

In Range 검출 영역의 최소 좌표값을 반환하는 명령어입니다.

Syntax

<변수> = GET_IN RANGE_MIN(<로봇 번호>, <In Range 번호>, <반환 좌표계>)

Description

JOB 수행 중 In Range 검출 영역 최소 좌표값을 반환합니다.
이 때 자료형은 <반환 좌표계> 입력 값에 따라 결정됩니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수> 또는 <사용자 좌표계 위치형 변수> 지정한 로봇 In Range에 설정되어 있는 검출 영역 최소 좌표값이 반환합니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<반환 좌표계>	Data type: <정수형 변수/상수> <반환 좌표계>의 범위는 0 ~ 1까지 입니다. <반환 좌표계>는 0일 경우 베이스 좌표계, 1일 경우 사용자 좌표계입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    XPOS XP                             // 베이스 좌표계 변수 선언
    UPOS UP                             // 사용자 좌표계 변수 선언
    XP = GET_INRANGE_MIN(1, 1, 0)       // In Range 검출 영역 최소 좌표값을 반환
    UP = GET_INRANGE_MIN(1, 1, 1)       // In Range 검출 영역 최소 좌표값을 반환
    PRINT(0, XP, UP)                    // 모니터링 0번 인덱스에 출력
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM,
SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MAX,
GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

14.7 GET_INRANGE_MAX (In Range 검출 영역 최대 좌표값 반환 명령어)

Usage

In Range 검출 영역의 최대 좌표값을 반환하는 명령어입니다.

Syntax

<변수> = GET_INRANGE_MAX(<로봇 번호>, <In Range 번호>, <반환 좌표계>)

Description

JOB 수행 중 In Range 검출 영역 최대 좌표값을 반환합니다.
이 때 자료형은 <반환 좌표계> 입력 값에 따라 결정됩니다.

Return value

<변수>	Data type: <베이스 좌표계 위치형 변수> 또는 <사용자 좌표계 위치형 변수> 지정한 로봇 In Range에 설정되어 있는 검출 영역 최대 좌표값이 반환합니다.

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.
<반환 좌표계>	Data type: <정수형 변수/상수> <반환 좌표계>의 범위는 0 ~ 1까지 입니다. <반환 좌표계>는 0일 경우 베이스 좌표계, 1일 경우 사용자 좌표계입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    XPOS XP                             // 베이스 좌표계 변수 선언
    UPOS UP                             // 사용자 좌표계 변수 선언
    XP = GET_INRANGE_MAX(1, 1, 0)       // In Range 검출 영역 최대 좌표값을 반환
    UP = GET_INRANGE_MAX(1, 1, 1)       // In Range 검출 영역 최대 좌표값을 반환
    PRINT(0, XP, UP)                    // 모니터링 0번 인덱스에 출력
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM,
SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN,
GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR

14.8 GET_INRANGE_ALARM (In Range 검출 알람 설정 값 반환 명령어)

Usage

In Range 검출 알람 설정 값을 반환하는 명령어입니다.

Syntax

<변수> = GET_INRANGE_ALARM(<로봇 번호>, <In Range 번호>)

Description

JOB 수행 중 In Range 검출 알람 설정 값을 반환 받을 수 있습니다.

Return value

<변수> Data type: <정수형 변수>
- 0: 알람 비활성화
- 1: 알람 활성화

Arguments

<로봇 번호> Data type: <정수형 변수/상수>
<로봇 번호>의 범위는 1 ~ 3까지 입니다.

<In Range 번호> Data type: <정수형 변수/상수>
<In Range 번호>의 범위는 1 ~ 16까지 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT RET_VEL                         // 정수현 변수 선언
    RET_VEL = GET_INRANGE_ALARM(1, 1)   // In Range 알람 설정 값을 반환.
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_OUT, GET_INRANGE_VECTOR

14.9 GET_INRANGE_OUT (In Range 출력 접점 반환 명령어)

Usage

In Range 검출 시 출력 접점 번호를 반환하는 명령어입니다.

Syntax

<변수> = GET_INRANGE_OUT(<로봇 번호>, <In Range 번호>)

Description

JOB 수행 중 In Range 검출 시 출력 접점 번호를 반환 받을 수 있습니다.

Return value

<변수>	Data type: <정수형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    INT RET_VEL                         // 정수형 변수 선언
    RET_VEL = GET_INRANGE_OUT(1, 1)     // In Range 출력 접점 번호를 반환.
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM, GET_INRANGE_VECTOR

14.10 GET_IN RANGE_VECTOR (In Range 중심점으로부터 TCP까지의 위치 반환 명령어)

Usage

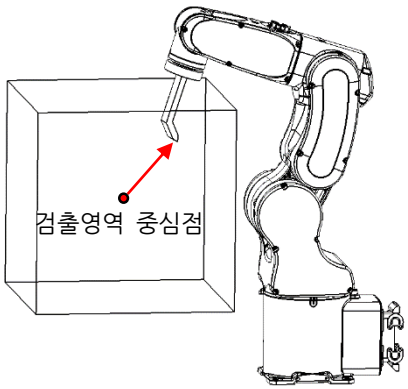
In Range 중심점으로부터 TCP까지의 위치를 반환하는 명령어.

Syntax

<변수> = GET_IN RANGE_VECTOR(<로봇 번호>, <In Range 번호>)

Description

In Range 중심점으로부터 TCP까지의 위치를 베이스 좌표계 위치형 변수 값으로 반환합니다.



Return value

<변수>	Data type: <베이스 좌표계 위치형 변수>

Arguments

<로봇 번호>	Data type: <정수형 변수/상수> <로봇 번호>의 범위는 1 ~ 3까지 입니다.
<In Range 번호>	Data type: <정수형 변수/상수> <In Range 번호>의 범위는 1 ~ 16까지 입니다.

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    XPOS XP                             // 베이스 좌표계 위치형 변수 선언
    XP = GET_IN RANGE_VECTOR(1, 1)      // 1번 로봇의 1번 In Range 중심점으로부터
                                        TCP까지의 위치를 반환
EOP                                     // 메인 함수 종료
```

Related terms

SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX, GET_INRANGE_ALARM, GET_INRANGE_OUT,

14.11 In Range 종합 예제

Examples

Example 1

```
MAIN                                    // 메인 함수 선언
    POS CUR_POS                         // 조인트 좌표계 위치형 변수 선언
    XPOS XMIN, XMAX, VECTOR             // 베이스 좌표계 위치형 변수 선언

    SET_INRANGE_ENB(1, 1, 0)            // In Range 기능 비활성화

    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지 JMOV

    CUR_POS = HERE                      // 현재 위치 저장
    XMIN = JTOB(CUR_POS)                // 현재 위치를 베이스 좌표계로 변환
    XMAX = JTOB(CUR_POS)                // 현재 위치를 베이스 좌표계로 변환

    XMIN.1 = 100                        // 최소 영역 설정
    XMIN.2 = -100
    XMIN.3 = -500

    XMAX.1 = 500                        // 최대 영역 설정
    XMAX.2 = 100
    XMAX.3 = 500

    SET_INRANGE_POS(1, 1, XMIN, XMAX)   // In Range 검출 영역 설정
    SET_INRANGE_MEASURE(1, 1, 0)        // In Range 검출 영역 Feedback 설정
    SET_INRANGE_ALARM(1, 1, 1)          // In Range 검출 시 알람 활성화
    SET_INRANGE_OUT(1, 1, 10)           // In Range 검출 시 출력 접점 설정
    SET_INRANGE_ENB(1, 1, 1)            // In Range 검출 활성화

    XMIN = GET_INRANGE_MIN(1, 1, 0)     // In Range 검출 영역을 베이스 좌표계로
    XMAX = GET_INRANGE_MAX(1, 1, 0)     반환
    PRINT(0, XMIN, XMAX)                // 모니터링 로그 0번 인덱스에 출력

    I0 = GET_INRANGE_OUT(1, 1)          // In Range 검출 시 출력 접점 반환
    I1 = GET_INRANGE_ALARM(1, 1)        // In Range 검출 시 알람 설정 상태 반환
    VECTOR = GET_INRANGE_VECTOR(1, 1)   // In Range 중심점으로부터 툴 끝단까지
                                        의 거리 반환(X, Y, Z 축 단위는 mm)
    PRINT(1, VECTOR)                    // 모티터링 로그 1번 인덱스에 출력
EOP                                     // 메인함수 종료
```

Example 2

```
MAIN                                    // 메인 함수 선언
    POS CUR_POS                         // 조인트 좌표계 위치형 변수 선언
    XPOS VECTOR                         // 베이스 좌표계 위치형 변수 선언
    UPOS UMIN, UMAX                     // 사용자 좌표계 위치형 변수 선언

    SET_INRANGE_ENB(1, 1, 0)            // In Range 기능 비활성화

    SVON                                // 서보 ON
    JMOV GP0                            // GP0까지 JMOV

    CUR_POS = HERE                      // 현재 위치 저장
    UMIN = JTOU(CUR_POS)                // 현재 위치를 사용자 좌표계로 변환
    UMAX = JTOU(CUR_POS)                // 현재 위치를 사용자 좌표계로 변환

    UMIN.1 = 100                        // 최소 영역 설정
    UMIN.2 = -100
    UMIN.3 = -500

    UMAX.1 = 500                        // 최대 영역 설정
    UMAX.2 = 100
    UMAX.3 = 500

    SET_INRANGE_POS(1, 1, UMIN, UMAX)   // In Range 검출 영역 설정
    SET_INRANGE_MEASURE(1, 1, 1)        // In Range 검출 영역Reference좌표설정
    SET_INRANGE_ALARM(1, 1, 1)          // In Range 검출 시 알람 활성화
    SET_INRANGE_OUT(1, 1, 10)           // In Range 검출 시 출력 접점 설정
    SET_INRANGE_ENB(1, 1, 1)            // In Range 검출 활성화

    UMIN = GET_INRANGE_MIN(1, 1, 1)     // In Range 검출 영역 사용자 좌표계로
    UMAX = GET_INRANGE_MAX(1, 1, 1)     반환
    PRINT(0, UMIN, UMAX)                // 모니터링 로그 0번 인덱스에 출력

    I0 = GET_INRANGE_OUT(1, 1)          // In Range 검출 시 출력 접점 반환
    I1 = GET_INRANGE_ALARM(1, 1)        // In Range 검출 시 알람 설정 상태 반환
    VECTOR = GET_INRANGE_VECTOR(1, 1)   // In Range 중심점으로부터 툴 끝단까지
                                        의 거리 반환(X, Y, Z 축 단위는 mm)
    PRINT(1, VECTOR)                    // 모니터링 로그 1번 인덱스에 출력
EOP                                     // 메인함수 종료
```

제 5장 개정

개정	날짜	개정내용
1	2019.01.21	최초 배포
2	2019.06.19	1. 외부 통신 신규 명령어 추가 13 외부 통신(TCP/IP Client 전용 명령어)
3	2020.01.07	1. 입출력 제어 신규 명령어 추가 BIN, BOUT 2. 입출력 제어 명령어 문법 수정 IN, BIN, CIN, CBIN 명령어 타임아웃 기능 추가 3. 시스템 명령어 문법 수정 RSTATE 명령어 상태 인덱스 정의 내용 추가
4	2020.01.08	1. 외부통신 기존 명령어 삭제 NET_DELIMITER 2. 함수 신규 명령어 추가 GET_TOOL_FRAME, GET_USER_FRAME, SET_TOOL_FRAME, SET_USER_FRAME, READ_TOOL, READ_USER 3. 입출력 신규 명령어 추가 XIN, XOUT, XBIN, XBOUT, XWIN, XWOUT, XDIN, XDOUT, XFIN, XFOUT, 4. 외부 통신 신규 명령어 추가 NET_STRIN, NET_STROUT 5. 함수 명령어 문법 수정 FLUSH
5	2020.01.13	1. 시스템 명령어 사양 변경 RSTATE
6	2020.05.07	1. 로봇 동작 조건 명령어 문법 사양 변경 FOS 2. 입출력 명령어 사양 변경 OUT ,COUT
7	2021.01.12	1. 영역 검출 전용 신규 명령어 추가 SET_INRANGE_ENB, SET_INRANGE_POS, SET_INRANGE_ALARM, SET_INRANGE_OUT, SET_INRANGE_MEASURE, GET_INRANGE_MIN, GET_INRANGE_MAX,GET_INRANGE_ALARM, GET_INRANGE_OUT, GET_INRANGE_VECTOR 2. 함수 명령어 신규 추가 STOK 3. 시스템 명령어 신규 추가 INIT_POS
8	2022.01.17	1. Maintenance 전용 신규 명령어 추가 GET_TWTIME, GET_TSVTIME, GET_IMVTIME, GET_TMVDIST, GET_WTIME, GET_SVTIME, GET_MVTIME, GET_MVDIST, INIT_WTIME, INIT_SVTIME, INIT_MVTIME, INIT_MVDIST 2. POUT 명령어 사양 변경

A. 문서 참조

이 장에서는 본 제품을 사용하는 모든 로봇 시스템의 서비스, 수리 또는 설치에 필요한 모든 문서를 지정합니다.
모든 문서 ID의 첫 단어는 제어기 명칭을 나타내며, 두 번째 단어는 해당 문서의 약자를 의미합니다. 마지막으로 언어와 해당 문서의 버전을 표기합니다.

언어는 아래의 규칙을 따라 표기합니다.

- 한국어: K
- 영어: E
- 중국어: C
- 베트남어: V

문서 ID	설명
N2-IM-K□□	**설치 및 취급 설명서 (Installation and handling manual)** 제어기 구성 및 설치, 외부 기기와의 인터페이스 방법에 관해 설명합니다.
N2-OM-K□□	**조작 및 운용 설명서 (Operation manual)** 제어기 및 티치 팬던트의 사용 방법 및 파라미터 설정, JOB 프로그램의 편집, 부가 기능 등에 대하여 설명합니다.
N2-PM-K□□	**프로그래밍 설명서 (Programming manual)** 로보스타 로봇 프로그램인 RRL(Robostar robot language) 작성방법 및 제공되는 명령어에 대하여 설명합니다.
N2-HM-K□□	**유니호스트 설명서 (Unihost manual)** 로보스타 온라인 PC 프로그램인 유니호스트에 대하여 설명합니다.
N2-AM-K□□	**알람 및 유지보수 설명서 (Alarm and maintenance manual)** 해당 제어기 기반의 로봇 시스템에서 발생한 문제에 대한 정보 및 해결 방법 및 절차에 대한 설명합니다

B. 위험 단계 및 기호

기호	지정	의미
	DANGER	지침을 준수하지 않으면 사고가 발생하여 중상 또는 치명적 부상 및/또는 심각한 제품 손상을 초래할 수 있음을 경고합니다.
	WARNING	지침을 준수하지 않으면 사고가 발생할 수 있으며, 이로 인해 심각한 부상, 치명적 또는 제품 손상이 발생할 수 있음을 경고합니다.
	CAUTION	지침을 준수하지 않으면 사고가 발생하여 제품이 손상될 수 있다는 경고입니다.
	ELECTRICAL SHOCK	감전 또는 감전 기호는 중상 또는 사망에 이를 수 있는 전기적 위험을 나타냅니다.
	NOTE	중요한 사실 및 조건을 알려줍니다.
	PROHIBITION	정상적인 제품의 사용을 위하여 금지되는 사항임을 알려줍니다.

C. 명령어 리스트 (알파벳 순서)

No.	명령어	설명	Ref.
1	ABS	입력되는 값의 절대값을 반환하는 함수입니다.	10.21
2	ACC	가속 시간의 백분율(%)을 설정합니다.	7.3
3	ACOS	아크 코사인함수의 결과 값을 반환합니다.	10.33
4	AMOV	현재 위치에서 경유점 1, 2 를 원호를 그리며 이동합니다.	6.6
5	ASC	문자열의 첫 문자에 해당 하는 문자 코드의 정수 값을 반환 합니다.	10.40
6	ASIN	아크 사인함수의 결과 값을 반환합니다.	10.31
7	ATAN	아크 탄젠트함수의 결과 값을 반환합니다.	10.35
8	ATAN2	제 2 아크 탄젠트함수의 결과 값을 반환합니다.	10.36
9	BIN	바이트 단위로 지정된 바이트영역의 값을 읽어 옵니다.	8.3
10	BINS	입력된 정수 값을 2 진수 문자열로 변환하는 함수입니다.	10.41
11	BOUT	지정된 바이트영역에 지정된 값을 출력합니다.	8.4
12	BREAK	반복문 루프 내에서 반복문을 탈출하기 위해 사용합니다.	5.9
13	BTOJ	베이스 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.	10.4
14	BTOU	베이스 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.	10.7
15	CBIN	필드버스 카드에서 지정된 바이트번호의 값을 읽어 옵니다.	8.9
16	CBOUT	지정된 필드버스 카드의 바이트번호에 지정된 값을 출력합니다.	8.10
17	CDIN	필드버스 카드에서 지정된 더블워드번호의 값을 정수 형태로 읽어 옵니다.	8.11
18	CDOUT	지정된 필드버스 카드 더블워드번호에 지정된 값을 정수 형태로 출력합니다.	8.12

19	CFIN	필드버스 카드에서 지정된 더블워드번호의 값을 실수 형태로 읽어 옵니다.	8.15
20	CFOUT	지정된 필드버스 카드 더블워드번호에 지정된 값을 실수 형태로 출력합니다.	8.16
21	CHR	입력된 정수 값에 해당하는 아스키 코드 문자를 반환합니다.	10.42
22	CIN	필드버스 카드에서 지정된 비트번호의 값을 읽어 옵니다.	8.7
23	CLOG	**※ 현재 지원하지 않습니다.**	11.19
24	CMOV	현재 위치에서 경유점 1, 2 를 잇는 원을 그리며 이동합니다.	6.5
25	Comment	프로그램 실행에 어떠한 효과도 갖지 않도록 처리합니다.	3.2
26	CONTINUE	반복문 루프 내에서 뒤의 구문을 수행하지 않고 바로 반복문 후처리 구문 및 조건식으로 분기할 때 사용합니다.	5.8
27	COS	코사인함수의 결과 값을 반환합니다.	10.32
28	COUT	지정된 필드버스 카드 비트번호에 지정된 값을 출력합니다.	8.8
29	CTHREAD	함수 또는 프로그램을 동시에 실행합니다.	9.1
30	CWIN	필드버스 카드에서 지정된 워드번호의 값을 읽어 옵니다.	8.13
31	CWOUT	지정된 필드버스 카드 워드번호에 지정된 값을 출력합니다.	8.14
32	DEC	감속 시간의 백분율(%)을 설정합니다.	7.4
33	DEFSTR	프로그램 내부에서 사용할 수 있는 문자열 변수를 선언합니다.	2.3
34	DEG	라디안 각(호도법)을 각도 값(육십분법)으로 변환하는 함수 입니다.	10.22
35	DLAY	지정된 시간 동안 프로그램의 처리를 일시 중단합니다.	7.6
36	ETHREAD	Thread 를 강제로 종료합니다.	9.2
37	EXIT	JOB 프로그램 실행을 정지합니다.	5.12

38	EXP	지수함수의 결과 값을 반환합니다. 로그 함수의 역함수 입니다.	10.26
39	F	시스템에서 미리 선언한 실수형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다.	2.9
40	FLUSH	시리얼 통신의 입력, 출력 버퍼를 클리어 합니다.	10.43
41	FOR … TO … [BY] … NEXT	조건이 false 가 될 때까지 블록을 반복 수행합니다.	5.4
42	FOS	축 선단이 목표점에 도달하기 전에 다음 목표점으로 궤적을 변경합니다.	7.5
43	FTOS	정수나 실수 값을 문자열로 변환합니다.	10.44
44	FUNC … END	함수의 시작 및 종료를 나타냅니다.	5.2
45	GAIN	게인 그룹을 전환합니다.	7.11
46	GBASEDISP	두 좌표간 베이스 좌표계 기준의 이동량을 구하는 함수입니다.	10.12
47	GET_INRANGE_ALARM	In Range 검출 알람 설정 값을 반환합니다	14.8
48	GET_INRANGE_MAX	In Range 검출 영역의 최대 좌표값을 반환합니다.	14.7
49	GET_INRANGE_MIN	In Range 검출 영역의 최소 좌표값을 반환합니다.	14.6
50	GET_INRANGE_OUT	In Range 검출 시 출력 접점 번호를 반환합니다.	14.9
51	GET_INRANGE_VECTOR	In Range 중심점으로부터 TCP 까지의 위치를 반환합니다.	14.10
52	GET_MVDIST	모터 누적 이동거리를 반환합니다.	11.33
53	GET_MVTIME	모터 누적 이동 시간을 반환합니다.	11.32
54	GET_SVTIME	모터 누적 통전 시간을 반환합니다.	11.31
55	GET_TMVDIST	출하 이후 모터 총 누적 이동거리를 반환합니다.	11.29
56	GET_TMVTIME	출하 이후 모터 총 누적 이동 시간을 반환합니다.	11.28

60	GET_TOOL_FRAME	지정한 툴 좌표계의 오프셋 값을 출력합니다.	10.15
61	GET_TSVTIME	출하 이후 모터 총 누적 통전 시간을 반환합니다.	11.27
62	GET_USER_FRAME	지정한 사용자 좌표계의 오프셋 값을 출력합니다.	10.16
63	GET_TWTIME	출하 이후 제어기 총 누적 통전시간 반환합니다.	11.26
64	GET_WTIME	제어기 누적 통전시간을 반환합니다.	11.30
65	GP	시스템에서 미리 선언한 조인트 좌표계 위치형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다.	2.10
66	GTOOLDISP	두 좌표간 툴 좌표계 기준의 이동량을 구하는 함수입니다.	10.13
67	GUSERDISP	두 좌표간 사용자 좌표계 기준의 이동량을 구하는 함수입니다.	10.14
68	HERE	시스템에서 미리 선언한 변수로 현재 위치를 조인트 각도 값으로 반환합니다.	2.13
69	HERE_REF	시스템에서 미리 선언한 변수로 현재 레퍼런스 위치를 조인트 각도 값으로 반환합니다.	2.14
70	HTOS	정수를 16 진수의 문자열로 변환합니다.	10.45
71	I	시스템에서 미리 선언한 정수형 전역 변수로 전체 프로그램에서 공통으로 사용 가능한 변수입니다	2.8
72	IF …THEN … ENDIF	조건식의 연산 결과에 따라 블록을 실행합니다.	5.6
73	IN	1 비트 단위로 지정된 비트번호의 값을 읽어 옵니다.	8.1
74	INCLUDE	특정 프로그램 파일을 현재 위치에 첨부하여 하나의 프로그램처럼 사용할 수 있게 해 줍니다.	3.1
75	INT	프로그램 내부에서 사용할 수 있는 정수형 변수를 선언합니다.	2.1
76	INIT_MVDIST	모터 누적 이동거리를 초기화 합니다.	11.37
77	INIT_MVTIME	모터 누적 이동 시간을 초기화 합니다.	11.36
78	INIT_POS	모든 위치형 변수를 초기화 합니다.	11.25

79	INIT_SVTIME	모터 누적 통전 시간을 초기화 합니다.	11.35
80	INIT_WTIME	제어기 누적 통전 시간을 초기화 합니다.	11.34
81	IPE	목표 위치 도달까지의 허용 시간 최대 값을 설정합니다.	7.12
82	JCALL	JOB 파일을 호출해 즉시 실행합니다.	5.10
83	JMOV	현재 위치에서 목표점으로 이동합니다.	6.3
84	JTOB	조인트 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.	10.3
85	JTOU	조인트 좌표계 위치 값으로부터 사용자 좌표계 위치 값을 계산합니다.	10.5
86	LABL … GOTO	분기점 위치를 지정하고, 해당 분기 위치로 이동하는 명령어 입니다.	5.7
87	LIMT	각 축의 이동범위를 제한합니다.	7.9
88	LMOV	현재 위치에서 목표점으로 직선 보간 이동합니다.	6.4
89	LN	자연로그의 결과 값을 반환합니다.	10.27
90	LOG	상용로그의 결과 값을 반환합니다.	10.28
91	MAIN … EOP	주 프로그램 시작 및 종료를 나타냅니다.	5.1
92	MAX	입력되는 두 개의 값을 비교하여 최댓값을 반환합니다.	10.38
93	MIN	입력되는 두 개의 값을 비교하여 최솟값을 반환합니다.	10.37
94	MVR	시스템에서 미리 선언한 변수로, 전체 이동거리에 대한 이동거리 비율을 반환합니다.	2.12
95	NET_CLOSE	제어기의 연결된 TCP/IP 통신을 종료하는 명령어 입니다.	13.2
96	NET_FD	제어기 TCP/IP Server 에 접속한 Client Socket 의 FD 를 반환합니다.	13.5
97	NET_KEEPALIVE	TCP/IP 통신 중 주기적으로 세션을 체크하여 LAN 케이블의 단선 혹은 비정상적인 세션 종료에 대해 감지하는 명령어 입니다.	13.4

98	NET_OPEN	TCP/IP Client 소켓을 생성하고 Server 로 접속을 요청하는 명령어 입니다.	13.1
99	NET_STATUS	제어기의 TCP/IP 통신상태를 확인하는 명령어 입니다.	13.3
100	NET_STRIN	이더넷 통신을 통하여 입력 된 문자열을 읽어옵니다.	13.6
101	NET_STROUT	이더넷 통신을 통하여 문자열을 송신합니다.	13.7
102	OFFS	목표 지점을 지정 된 값만큼 가감해 이동 시킵니다.	7.7
103	OFFSCLR	설정된 오프셋 값을 해지합니다.	7.8
104	Operators	특정 수학적 또는 논리적 조작을 수행하도록 지시하는 기호들입니다.	4
105	OUT	지정된 비트번호에 지정된 값을 출력합니다.	8.2
106	P	시스템에서 미리 선언한 조인트 좌표계 위치형 변수로 프로그램 파일에 저장되며 해당 프로그램에서 전역 변수로 사용됩니다.	2.7
107	PEXIT	지정한 로봇 Thread 번호의 JOB 을 해제합니다.	12.6
108	PIN	포트 단위로 지정된 포트번호의 값을 읽어 옵니다.	8.5
109	POS	프로그램 내부에서 사용할 수 있는 조인트 좌표계 위치형 변수를 선언합니다.	2.4
110	POUT	지정된 포트번호에 지정된 값을 출력합니다.	8.6
111	POW	거듭제곱 값을 구할 때 사용하는 함수입니다.	10.24
112	PRESTART	지정한 로봇 Thread 번호의 JOB 을 재 실행합니다.	12.5
113	PRINT	모니터링 용 로그를 기록합니다.	11.17
114	PSEL	지정한 로봇 Thread 번호에 입력한 JOB 을 매칭합니다.	12.1
115	PSTART	지정한 로봇 Thread 번호의 JOB 을 실행합니다.	12.2
116	PSTATE	지정한 로봇 Thread 번호의 상태를 읽습니다.	12.4

117	PSTOP	지정한 로봇 Thread 번호의 JOB 을 정지합니다.	12.3
118	RAD	각도 값(육십분법)을 라디안 각(호도법)으로 변환하는 함수입니다.	10.23
119	RAND	난수 값을 생성합니다.	10.39
120	READ_CTEMP	제어기 내부의 온도를 가져옵니다.	11.16
121	READ_DERR	지정한 축에 해당하는 드라이버의 알람 코드를 반환합니다.	11.7
122	READ_LOAD	로봇의 축 별 현재 부하율을 가져옵니다.	11.11
123	READ_MTEMP	로봇의 축 별 현재 모터 엔코더의 온도 값을 가져옵니다.	11.15
124	READ_RPM	로봇의 축 별 현재 RPM 값을 가져옵니다.	11.10
125	READ_TOOL	지정한 로봇에 사용 중인 툴 좌표계 번호를 출력합니다.	10.19
126	READ_TRQ	로봇의 축 별 현재 토크 값을 가져옵니다.	11.14
127	READ_TRQMAX	로봇의 축 별 최대 토크 값을 가져옵니다.	11.12
128	READ_USER	지정한 로봇에 사용 중인 사용자 좌표계 번호를 출력합니다.	10.20
129	READ_VEL	로봇의 축 별 현재 속도 값을 가져옵니다.	11.13
130	REAL	프로그램 내부에서 사용할 수 있는 실수형 변수를 선언합니다.	2.2
131	RELBASE	기준 위치에서 대입된 이동량만큼 베이스 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.	10.9
132	RELTOOL	기준 위치에서 대입된 이동량만큼 툴 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.	10.10
133	RELUSER	기준 위치에서 대입된 이동량만큼 사용자 좌표계 정보를 기준으로 상대 이동한 위치 값을 계산하는 함수입니다.	10.11
134	RERRCNT	총 발생한 알람 개수를 반환합니다.	11.4
135	RERRCODE	지정한 인덱스 번호에 해당하는 알람 코드를 반환합니다.	11.5

136	RERROR	가장 마지막에 발생한 알람 코드를 반환합니다.	11.3
137	RERRTEXT	지정한 인덱스 번호에 해당하는 알람 내용을 반환합니다.	11.6
138	RESET	발생한 알람을 모두 해제합니다.	11.2
139	RET	함수의 실행을 마치고 값을 반환하며 함수를 호출한 곳으로 복귀합니다.	5.3
140	RND	소수부를 반올림한 후에 정수 값으로 반환합니다.	10.25
141	RSTATE	로봇의 상태를 반환하는 명령어 입니다.	11.8
142	SET_INRANGE_ALARM	In Range 검출 알람을 설정합니다.	14.3
143	SET_INRANGE_ENB	In Range 기능을 활성화 또는 비활성화합니다.	14.1
144	SET_INRANGE_MEASURE	In Range 검출 영역의 데이터 타입을 설정합니다.	14.5
145	SET_INRANGE_OUT	In Range 검출 시 출력 접점 번호를 설정합니다.	14.4
146	SET_INRANGE_POS	In Range 검출 영역을 설정합니다.	14.2
147	SET_TOOL_FRAME	지정한 툴 좌표계에 오프셋 값을 입력하는 명령어입니다.	10.17
148	SET_USER_FRAME	지정한 사용자 좌표계에 오프셋 값을 입력하는 명령어입니다.	10.18
149	SETERR	사용자가 임의로 System emergency 알람을 발생시킵니다.	11.1
150	SIN	사인함수의 결과 값을 반환합니다.	10.30
151	SLEFT	입력된 문자열의 좌측 문자열을 추출합니다.	10.46
152	SLEN	입력된 문자열의 길이를 반환합니다.	10.47
153	SMID	문자열 지정위치부터 입력된 값만큼 문자열 추출합니다.	10.48
154	SPOS	문자열 1 에서 문자열 2 가 매칭되는 시작위치를 반환합니다.	10.49

155	SQRT	제곱근을 반환합니다.	10.29
156	SRIGHT	입력된 문자열의 우측 문자열을 추출합니다.	10.50
157	STOK	구분자로 문자열 추출 후 지정 된 순번의 문자를 반환합니다.	10.51
158	STOP	로봇의 동작을 정지합니다.	5.11
159	STRIN	시리얼 및 이더넷 통신을 통하여 입력 된 문자열을 읽어옵니다.	10.52
160	STROUT	시리얼 및 이더넷 통신을 통하여 문자열을 송신합니다.	10.53
161	SVAL	문자열을 숫자로 변환합니다.	10.54
162	SVOF	서보 모터 전원을 끕니다.	6.2
163	SVON	서보 모터 전원을 켭니다.	6.1
164	SWLIMIT	로봇의 축 별 이동 범위를 제한하는 파라미터 값을 읽어 옵니다.	11.9
165	TACT_START	공정 시간 측정을 위한 측정 시작 위치를 설정합니다.	11.23
166	TACT_STOP	공정 시간 측정을 위한 측정 종료 위치를 설정합니다.	11.24
167	TAN	탄젠트함수의 결과 값을 반환합니다.	10.34
168	TH_RESUME	정지한 Thread 를 재 실행 합니다.	9.5
169	TH_STATE	Thread 의 현재 상태를 확인합니다.	9.3
170	TH_SUSPEND	실행 중인 Thread 를 정지합니다.	9.4
171	TMR	시스템에서 미리 선언한 타이머 변수로, 카운트 값을 지정하거나 현재 카운트 값을 읽어오는데 사용하는 변수입니다.	2.11
172	TOOL	로봇에 부착된 사용 툴과 해당 좌표계를 선택합니다.	10.1
173	TRQ	현재 토크 값이 입력된 제한 토크 이상일 경우 알람을 발생시키며 정지합니다.	7.10

174	UPOS	프로그램 내부에서 사용할 수 있는 사용자 좌표계 위치형 변수를 선언합니다.	2.6
175	USER	사용자 좌표계를 선택합니다.	10.2
176	UTOB	사용자 좌표계 위치 값으로부터 베이스 좌표계 위치 값을 계산합니다.	10.8
177	UTOJ	사용자 좌표계 위치 값으로부터 조인트 좌표계 위치 값을 계산합니다.	10.6
178	VEL	축 이동속도의 천분율(%)을 설정합니다.	7.2
179	WGFLT	전역 포인트 변수를 파일로 저장합니다.	11.22
180	WGINT	전역 정수 변수를 파일로 저장합니다.	11.21
181	WGPNT	전역 포인트 변수를 파일로 저장합니다.	11.20
182	WHILE … ENDWL	조건식이 참(true)인 동안만 블록을 반복 수행합니다	5.5
183	WITH … ENDWT	로봇 동작과 함께 블록 내 수행문을 동시처리 합니다.	7.1
184	WLOG	임의의 사용자 로그를 시간 정보를 포함하여 파일로 저장합니다.	11.18
185	XBIN	지정한 디바이스 및 바이트 번호를 참조하여 CNET 수신 버퍼에서 값을 읽어옵니다.	8.19
186	XBOUT	지정한 디바이스 및 바이트 번호를 참조하여 CNET 디바이스 발신 버퍼에 값을 출력합니다.	8.20
187	XDIN	지정한 디바이스 및 더블 워드 번호를 참조하여 CNET 수신 버퍼에서 값을 읽어옵니다.	8.23
188	XDOUT	지정한 디바이스 및 더블 워드 번호를 참조하여 CNET 디바이스 발신 버퍼에 값을 출력합니다.	8.24
189	XFIN	지정한 디바이스 및 더블 워드 번호를 참조하여 CNET 수신 버퍼에서 값을 실수 형태로 읽어옵니다.	8.25
190	XFOUT	지정한 디바이스 및 더블 워드 번호를 참조하여 CNET 디바이스 발신 버퍼에 실수 값을 출력합니다.	8.26
191	XIN	지정한 디바이스 및 비트 번호를 참조하여 CNET 수신 버퍼에서 값을 읽어옵니다.	8.17
192	XOUT	지정한 디바이스 및 비트 번호를 참조하여 CNET 디바이스 발신 버퍼에서 값을 출력합니다.	8.18

193	XPOS	프로그램 내부에서 사용할 수 있는 베이스 좌표계 위치형 변수를 선언합니다.	2.5
194	XWIN	지정한 디바이스 및 워드 번호를 참조하여 CNET 수신 버퍼에서 값을 읽어옵니다.	8.21
195	XWOUT	지정한 디바이스 및 워드 번호를 참조하여 CNET 디바이스 발신 버퍼에 값을 출력합니다.	8.22

N2 Series Controller

Programming Manual
Seventh edition, May 03, 2021

ROBOSTAR CO., LTD.
ROBOT R&D CENTER