General Purpose Robot Arm for Industry Use

ZERO

델타 로봇 사용설명서

지원 버전 R1.3.0 이상

문서 번호 : M-0301-220218

2022 년 02 월

산업용 로봇 "ZERO" 를 구입해주셔서 감사합니다 .

- 본 제품을 취급 시「산업안전보건교육」이나「전기기사 자격」및「Python」언어에 관한 지식과 기술이 필요합니다 .
- 사용 전에 사용설명서 등 매뉴얼을 충분히 읽고 올바르게 사용하십시오 .
- 제품의 성능 개선을 위해 예고없이 사양이 변경되는 경우가 있을 수 있습니다 .
- 사용설명서 등 매뉴얼은 ,
  - 제품을 사용하는 사용자가 잘 보관해주십시오 .
  - 제품 사양 변경에 따라 예고없이 개정될 수 있습니다 .
  - 내용의 일부 또는 전부를 무단으로 전재하는 것을 금지합니다 .

이 설명서는 다음 기종에 대응하고 있습니다 .

로봇	컨트롤러 (버전)	JOG 스틱	티칭 펜던트
ZERO 시리즈	ZC1*** (R1.3.0 이상)	ZJ1000	ZP1000

본 제품에는 다음 설명서가 있습니다 .

설치설명서

개봉부터 설치까지의 간이 설명서입니다 .

- 시작하기 전에
- 안전상의 주의
- 매니퓰레이터 및 컨트롤러 설치
- JOG 스틱
- 티칭 펜던트
- 컨트롤러와 PC 의 연결
- 배선 및 전원
- JOG 동작 및 ABS 원점 복귀
- 교시 및 오류 발생 시

안전설명서

반드시 지켜야할 안전 사항에 관한 내용입니다 .

- 준수 사항
- 안전에 관한 표시
- 위험 평가
- 산업안전보건교육
- 보수 · 점검
- 안전 대책
- 보증 및 면책

사용설명서 (본 문서)

제품과 프로그램에 관한 설명서입니다 .

- A 개요
- B 하드웨어
- C 교시
- D 소프트웨어
- Z 자료

A 개요

B 하드웨어

C 교시 (Teaching)

D 소프트웨어

1. 프로그래밍 가이드	PC 와 동작 환경 , 프로그래밍 가이드
2. 로봇 라이브러리	데이터형 , 모듈 , 메소드 요약 , 로봇 라이브러리
3. 메모리 맵	개요 , 공유 메모리 , 메모리 I/O
4. 프로그램 실행 단계	개요 , 실행 방법

Z 자료

1. 블록 다이어그램	시스템 블록 다이어그램 , 하드웨어 블록 다이어그램
2. 유지보수	점검 , 유지 보수
3. 용어집	용어집
4. 문제 해결	오류 로그 , 문제 해결

MEMO

A 개요

MEMO

1 시스템 개요

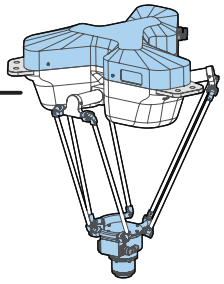
1. 제품 정보

1. 로봇 "ZERO" 의 구성

본 제품은 다음과 같은 구성으로 되어 있습니다

"ZERO" (= 로봇)

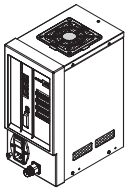
"ZERO" 는 매니퓰레이터와 로봇 컨트롤러로 구성되어 있습니다



매니퓰레이터

매니퓰레이터는 서보 모터로 구동하는 3 축 (4 축 옵션) 병렬형 로봇입니다 .

매니퓰레이터 끝에 다른 말단 장치를 사용하여 다양한 작업에 사용 가능합니다 .



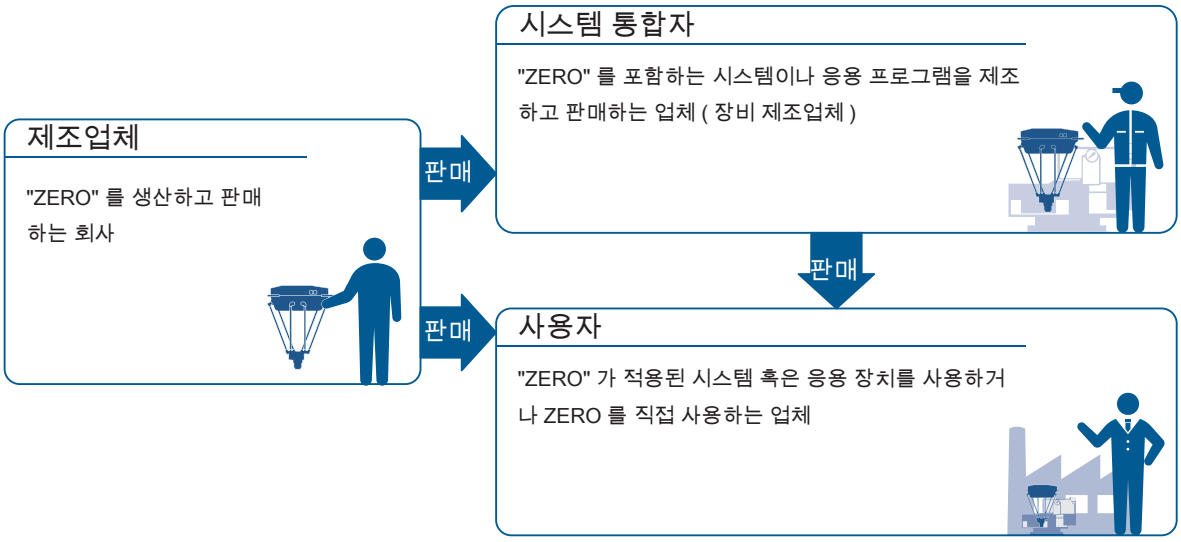
컨트롤러

제어 보드와 전원 기판 등을 수납한 제어장치입니다 .

상위 제어 장치와의 통신과 I/O 등의 인터페이스로 매니퓰레이터의 움직임을 종합적으로 제어합니다 .

2. 제조업체와 시스템 통합자 및 사용자

본 문서에서의 제조업체 , 시스템 통합자 그리고 사용자의 정의는 다음과 같습니다 .



3. 설명서의 기재 사항 중 주의점

- 사양값 (정격 , 성능) 은 단독 시험의 각 조건에서 얻은 값이며 , 복합 조건에서 얻어진 값은 보장할 수 없습니다 .
- 제품 개선이나 당사 사정으로 인해 예고 없이 사양을 변경하거나 생산을 종료할 수 있습니다 .

4. 용도

본 제품은 산업용 로봇입니다 . 공장에서 일반 공업 제품 생산용으로 설계 , 제조하고 있습니다 . 일반 가정이나 다음의 용도로의 사용은 적합하지 않습니다 . 당사는 다음 사항에 대해 어떠한 보증도 하지 않습니다 .

군사 또는 무기와 관련된 용도

최종 사용자 및 최종 용도가 군사나 무기 등 모든 군사에 관련된 용도

높은 안전성과 신뢰성이 필요한 용도

원자력 제어 설비 , 연소 설비 , 항공 우주 장비 , 수송 , 철도 시설 , 선박 제조 탑재 장치 , 승강 설비 , 오락 시설 , 의료장비 , 간호용 기기 , 안전 장비 , 자동차 제조 탑재 장비 ,
기타 인명에 위험을 미치는 장비

엄격한 조건이나 환경에서의 용도

야외시설 , 화학적 오염이 있는 시설 , 전자적 방해를 받는 시설 , 진동이나 충격을 받는 시설 , 분진이 있는 장소 , 갱내 채굴 작업

설명서 등에 기재되지 않은 조건이나 환경 등에서의 용도

이용 조건 등에 기재된 경고 및 주의 사항을 지키지 않으면 부상 (사망 또는 중상), 사고 , 고장 등이 발생할 수 있습니다 . 이에 당사는 책임을 지지 않습니다 .
당사는 위험 및 문제 발생에 대한 모든 상황을 다 예측할 수 없습니다 .
이용 조건 등에 기재된 경고 , 주의 , 기타 기재 사항은 당사가 예측할 수 있는 범위입니다 .

2. 해외 규격

1. 적합 규격

기계류 지침

. . . . 2006/42/EC

기계류의 안전성 - 기계의 전기 장비 - 제 1 부 : 일반 요구사항

. . . . EN60204-1:2018

로봇 및 로봇 장치 - 산업용 로봇의 안전에 대한 요구 사항 - 제 1 부 : 로봇

. . . . EN/ISO 10218-1 : 2011

EMC

. . . . EN61000-6-2:2005

. . . . EN55011 : 2009+A1:2010

KCs

. . . . . S2-W-5-2017

2. 환경 사양

규격	매니퓰레이터	컨트롤러
IP	IP40 (매니퓰레이터 선단부 , 컨트롤러) / IP20 (매니퓰레이터 상단부)	
진동 · 충격	—	JIS B3502

2 시스템 설치

1. 시작 절차

개봉

A 개요

- 운송 중에 손상된 흔적이 없는지 확인하십시오 .
- 플라스틱 커버에 부하가 걸리지 않도록 들어올려 꺼내주십시오 .
- 내용물이 손상되거나 변형되지 않았는지 확인하십시오 .
- 수령 제품이 주문 내용과 일치하는지 , 내용물이 모두 들어있는지 확인하십시오 .
- 컨트롤러와 매니퓰레이터가 포함되어 있는지 확인하십시오 .
- 평평한 장소에 적절한 공간을 확보해주십시오 .
- 헬멧 , 보호안경 , 안전화 , 장갑등 보호장구를 착용하고 , 스트랩 등 걸리는 것을 제거하고 꺼내십시오 .
- 옮기실 때를 대비해 , 개봉 자재나 고정 치구는 재사용을 위해 보관해 주십시오 .

운반 설치대까지 이동

A 개요

- 설치 작업은 반드시 2 명 이상 하십시오 .
- 설치할 가대 등에 최대한 가까이 크레인 및 대차 등으로 옮겨주십시오 .
- 가대로 이동할 때나 대차로의 이송 시에는 과도한 충격이나 진동이 가해지지 않도록 주의해주십시오 .
- 스위치 , 단자대 , 커넥터 , 방열 팬 등 돌출부에 무리한 힘이 가해지지 않도록 주의해주십시오 .
- 운반 중 플라스틱 커버에 부하가 걸리지 않도록 주의해주십시오 .
- 임시로 고정할 때는 적어도 볼트 1 개 이상 사용하여 체결해주십시오 .
- 가대에 고정될 때 까지 운반 자세를 유지한 채 작업해주십시오 .

설치

B 하드웨어

- 매니퓰레이터는 안전 보호 영역 내 (안전펜스 내) 의 수평인 설치면에 고정용 육각볼트 (M8) 로 단단히 고정하고 위치가 어긋나거나 넘어지지 않게 하십시오 .
- 매니퓰레이터는 반드시 강성이 충분히 높고 적절한 위치에 고정하십시오 . 설치면의 조도는 Rz25 이상을 권장합니다 . 설치면은 동작시 반발력 , 하중 등에 충분히 견딜수 있는 강도 , 강성을 확보하여 주십시오 .
- 다른 매니퓰레이터나 주변 장치 등과 간섭하지 않는 공간을 확보하여 주십시오 .
- 컨트롤러는 안전 보호 영역 외 (안전펜스 밖) 의 매니퓰레이터가 잘 보이는 곳에 나사를 사용하여 수평으로 고정하십시오 . 고무 다리는 분리하지 마십시오 .
- 향후 유지 보수 및 수리 점검에 필요한 장소나 작업 방법을 고려하여 설치하십시오 .
- 동작 확인과 시운전을 위해 임시로 설치하는 경우에도 반드시 고정하십시오 .

배선 , 전원

B 하드웨어

- 커넥터의 잘못된 삽입 , 잘못된 배선을 하지 않도록 주의하십시오 .
- 커넥터가 " 딸깍 " 소리가 날 때까지 확실히 삽입하십시오 .
- 나사식 커넥터는 확실히 삽입하고 단단히 고정하여 주십시오 .
- 모든 배선이 확실하게 완료된 후 전원을 연결하여 주십시오 .
- 케이블 및 배관을 부설하는 경우 작업자가 걸리거나 넘어지지 않게 유의하십시오 .
- 기기 간 케이블이나 외부 입출력 케이블은 다른 기기의 동력선이나 접지선과는 분리해 배선해주십시오 .
- 외부 입출력 케이블은 반드시 실드 케이블을 사용하십시오 .
- 각 단자에는 사용설명서에 정해진 전압 이외에는 인가하지 마십시오 .
- 단자의 연결 및 극성 (+ 와 -) 이 틀리지 않도록 주의하십시오 .
- 배선 공간을 확보하여 , 커넥터에 케이블 질량 , 반발력 , 하중 등이 걸리지 않도록 고정하십시오 . 필요에 따라 연결부를 보호하십시오 .
- 감전 방지 , 정전기 방지 , 노이즈 성능 향상 , 불필요한 전자파 방사 억제 등은 반드시 실시하십시오 .
- 접지용 전선은 지정된 사이즈의 전선을 사용하고 , 접지점과의 거리는 가능한 한 짧게 해주십시오 .
- 접지 방식은 전용 접지로 하고 , 다른 대형 기기의 설치에는 별도로 분리하여 접지하십시오 .

1 시작 절차

원점복귀 , 교시

C 교시

- 컨트롤러를 교시용 PC 와 연결하십시오 .
- 안전 보호 영역 외 (안전 펜스 밖) 에서 조그 스틱으로 조작하여 교시하십시오 .
- 매니퓰레이터를 동작시키고 싶을 때만 인가 스위치를 ON 으로 하십시오 .
- 부득이하게 안전 보호 영역 내 (안전 펜스 안) 에서 교시를 실시할 경우 반드시 안전을 확보한 후 하십시오 .
- 로봇 제어의 우선 순위를 확인하고 잠금 키 , 비상 정지 스위치 , 인터록 키를 사용하여 교시 작업 중 표시를 하십시오 .
- 만일을 대비해 비상 탈출구를 확보하십시오 .
- 작업 후에는 안전 보호 장치를 원래 상태로 되돌려주십시오 .
- 매니퓰레이터의 가동 범위 내에 주변 장치 등 장애물이 없는지 확인하고 동작시켜 주십시오 .

프로그래밍

D 소프트웨어

- Python 에 대한 기본적인 지식이 필요합니다 .
- 본 제품의「로봇 라이브러리」로 용이하게 프로그래밍할 수 있습니다 .
- 기본적인 동작의 샘플 프로그램이 준비되어 있습니다 .
- 메모리맵이 제공됩니다 .

동작확인 , 테스트

D 소프트웨어

- 안전 보호 영역 내 (안전 펜스 내) 에 사람이 없는지 , 매니퓰레이터 가동 범위에 장애물이 없는지 확인해주십시오 .
- 안전 보호 영역 외 (안전 펜스 밖) 에서 조작하십시오 . 안전 보호 영역 내 (안전펜스 내) 에서 작업할 경우 반드시 안전을 확보하고 작업하십시오 .
- 동작 확인은 모든 비상 정지 스위치가 동작하는 것을 확인한 후 실행해주십시오 .
- 비상 정지 스위치에 의해 주변기기를 포함한 매니퓰레이터가 정지하는지 확인하십시오 .
- 충분히 동작 확인을 하고 안전하게 장치를 작동할 수 있는지 확인하십시오 .

자동 운전 전에

D 소프트웨어

- 안전 보호 영역 내 사람이 없는지 , 매니퓰레이터 가동 범위에 장애물이 없는지 확인해주십시오 .
- 관련 주변 기기를 포함하여 시스템의 모든 것이 자동 운전이 가능한 상태인지 확인하십시오 .
- 운전 시작 시 우선 저속으로 운전하고 , 정상적으로 운전하는 것을 확인하십시오 .

시작 절차 완료

포인트 !

검사 방법과 문제 해결 및 기타 기술 자료는 Z 자료에 기재되어 있습니다 .

Z 자료

2. 개봉

수령

- 운송 중 손상된 흔적이 없는지 확인하십시오 .
  손상된 흔적이 있을 경우 , 수송업자 입회 하에 개봉하여 주십시오 .
  모든 포장 자재를 보관해주십시오 . 손해 청구를 할 때 필요할 수 있습니다 .
- 컨트롤러 , 매니퓰레이터 각각 상자 1 개씩입니다 . 2 상자가 모두 있는지 확인하십시오 .
- 수령품과 주문 내용이 일치하는지 확인하여 주십시오 .

개봉 준비

- 평탄한 장소에서 적절한 공간을 확보해 주십시오 . 불안정한 장소에서는 제품이 넘어질 우려가 있습니다 .
- 헬멧 , 안전화 , 장갑 등 보호구를 착용하고 , 스트랩 등 걸리는 것은 제외하고 작업하십시오 .

개봉

- 반드시 2 명 이상 작업하십시오 .
- 설치할 가대 등에 가능한 가까이 크레인이나 대차 등으로 옮겨주십시오 .
  사람이 운반할 때에는 두사람 간의 힘의 차이가 없게 하십시오 .
  또 , 도어 개폐 등으로 불안정한 상태가 되지 않게 하십시오 .
- 고정 치구 이외는 잡지 마십시오 . 특히 , 플라스틱 부분은 파손의 원인이 됩니다 .

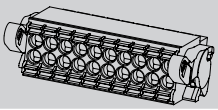
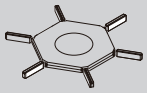
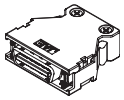
개봉이 끝나면

- 매니퓰레이터를 임시로 설치하는 경우 , 적어도 1 개 이상의 볼트로 체결하여 고정하십시오 .
- 고정 치구는 설치가 완료될 때까지 제거하지 마십시오 .
- 원래의 포장재나 고정 치구를 보관해주시고 이전 시나 운반 시에는 포장재를 사용하여 납품 때와 같은 상태로 다시 포장하여 주십시오 .

옮기기 전에 준비

- 본 제품에 관련된 설명서 등은 최종적으로 사용하는 사용자에게 모두 전달해주십시오 .
- 시스템 통합자는 본 제품을 포함한 시스템 전체의 설명서를 작성해 , 사용자에게 전달해주십시오 .
- 포장 자재는 모두 보관해 두었다가 납품 시와 같은 상태로 포장해 발송해주십시오 .
- 해외 발송에 나무 팔레트를 재사용하는 경우 , 국제기준 No.15「국제 무역에서의 목재 포장 자재 규제」를 확인해 주십시오 .

3. 동봉 · 부속품

동봉 · 부속품	형식 (제조업체)	개수	비고
매니퓰레이터	ZRC-*****	1 개	—
컨트롤러	ZC100*	1 개	—
사용설명서	—	1 부	PDF File
설치설명서	—	1 부	책자 / PDF File
안전설명서	—	1 부	책자 / PDF File
I/O 커넥터	DFMC 1,5/10-ST-3,5-LR (1790564) (PHOENIX CONTACT 사)	3 개	(20 pin) 
Safety 커넥터	위와 같음	1 개	위와 같음
오삽입 방지 키	CP-DMC 1,5 NAT (1790647) (PHOENIX CONTACT 사)	1 세트 (6 개)	— 
점퍼 커넥터	E2010101 (ZEUS CO., LTD.)	1 개	— 
매니퓰레이터 케이블	E2021701 (ZEUS CO., LTD)	1 개	· 길이 3 m · 제공된 페라이트 코어 2 개는 제거하지 마십시오 .
페라이트 코어	—	2 개	· 전원 케이블에 넣으십시오 · 대응 외경 4.5 - 8.5 mm

4. 운반

1. 설치대까지 이동

	작업할 때 , 반드시 2 인 이상 작업하십시오 . 설치할 가대 등에 가능한 가까이 대차 등으로 옮겨주십시오 . 인력으로 운반하는 경우 두사람 간의 힘의 차이가 없도록 해주십시오 . 문 개폐 등으로 한손으로 들지 마십시오 .	

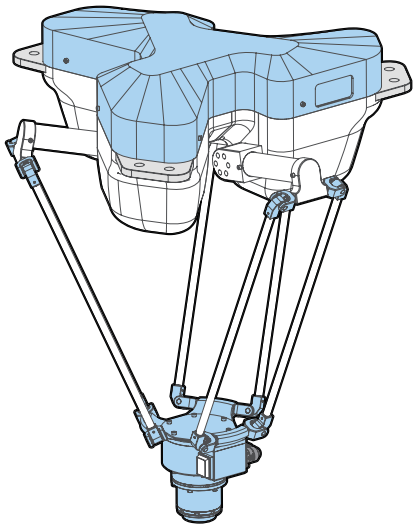
· 컨트롤러

앞면과 흡배기구에 충격이 가해지지 않도록 하부를 들고 옮기십시오 .

· 매니퓰레이터

조인트 부 (관절 부) 와 말단부는 잡지 마십시오 .

ZRC-*****



운반 자세

2. 이전 시 운반

	작업할 때 , 반드시 2 인 이상 작업하십시오 . 잘못된 포장으로 인한 파손에 대해서는 보증 대상에서 제외됩니다 .	
	구입 시의 포장재를 이용하여 안전하게 포장하십시오 . 「정밀기계」취급으로 운송하십시오 . 운송 시에 조인트부에 충격이나 부하가 걸리면 매니퓰레이터가 파손될 위험이 있습니다 .	

컨트롤러

앞면과 흡배기구를 손상시키지 않도록 주의해서 전용 골판지에 납품 시와 같은 상태로 넣어 주십시오 .

매니퓰레이터

운반자세에서 , 납품시의 포장자재를 사용하여 납품 시와 같은 상태로 포장해주십시오 . 지정된 이외의 상태로 운반시 , 사고나 고장의 원인이 됩니다 .

해외 발송 시 나무 팔레트를 재사용할 경우 국제 기준 No.15「국제 무역에서의 목재 포장 자재 규제」를 참고하십시오 .

매니퓰레이터 포장

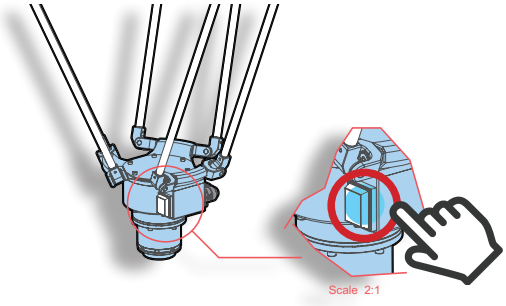
1. 매니퓰레이터 운반 자세

자세를 변경하려면 , 컨트롤러와 매니퓰레이터를 케이블로 연결하고 , 컨트롤러 전원을 ON 하십시오 . 전원이 연결된 상태에서 각 조인트에 있는 브레이크 해제 버튼을 누르면 브레이크가 해제됩니다 . (버튼을 누르고 있는 동안에만 브레이크가 해제됩니다 .)

브레이크 해제 스위치

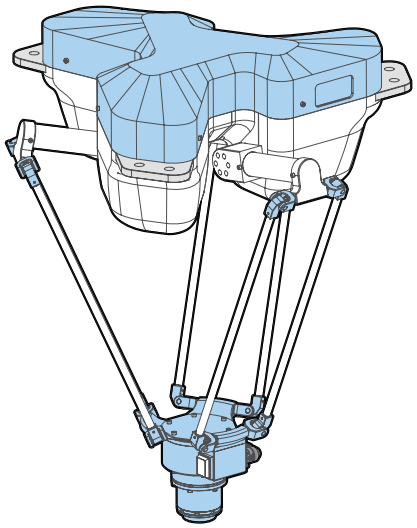
로봇에 전원이 연결되어 있지 않으면 브레이크가 해제되지 않습니다 .

조인트



브레이크 해제 버튼을 누르고 있는 동안 브레이크를 해제합니다 .

ZRC-*****

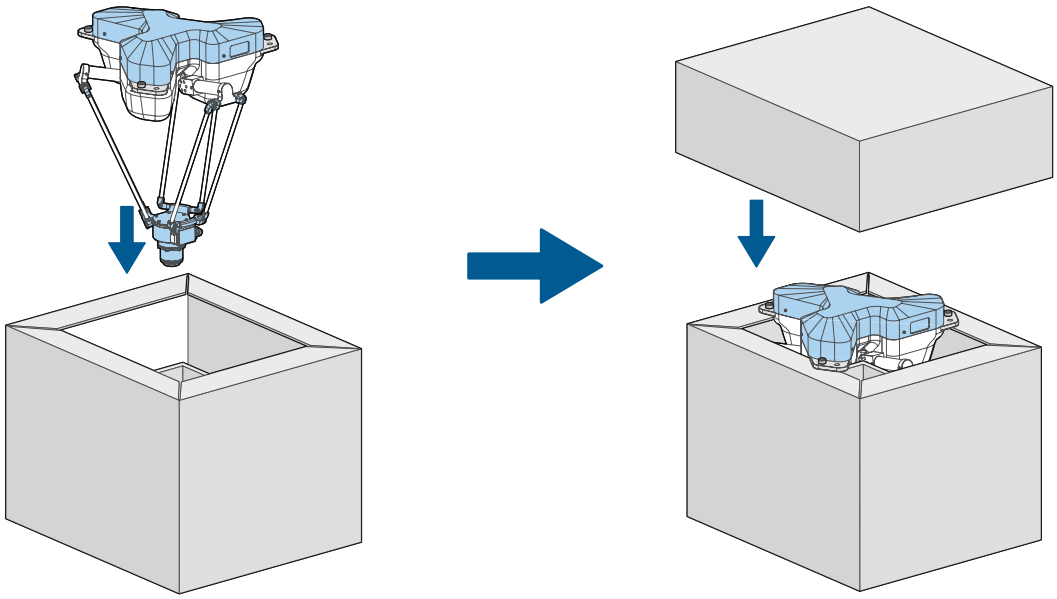


운반 자세

2. 전용 포장 상자 사용

매니퓰레이터를 포장 상자에 넣으십시오 .
완충재를 사용하여 확실히 보호되도록 넣으십시오 .

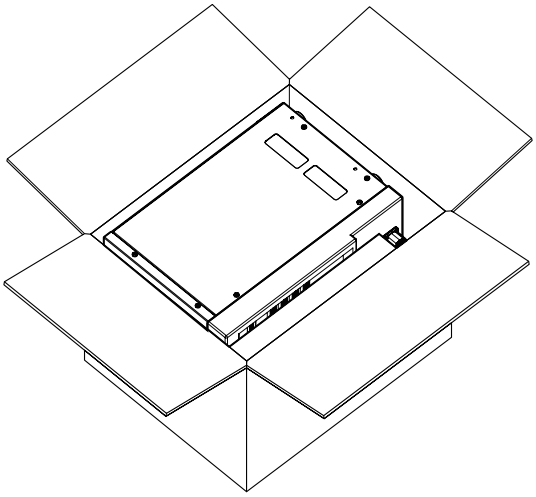
ZRC-*****



컨트롤러 포장

1. 전용 포장 상자 사용

컨트롤러를 전용 포장 상자에 옆으로 눕혀 넣으십시오 . 넣으실 때 , 커넥터는 모두 분리하십시오 .



B 하드웨어

MEMO

B 하드웨어

1 시스템 구성

1. 모델명

본 제품은 매니퓰레이터와 컨트롤러를 포함하여 제공됩니다 .

매니퓰레이터 모델명 : ZRC - 03 06 N

- ZRC : 제품 코드 — ZEUS Robot C type(세 번째)
- 03 : 최대 가반 중량

기호	사양
XX	XX kg

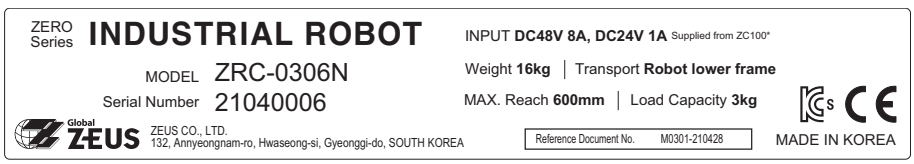
- 06 : 동작 범위 직경

기호	사양
XX	XX * 100 mm

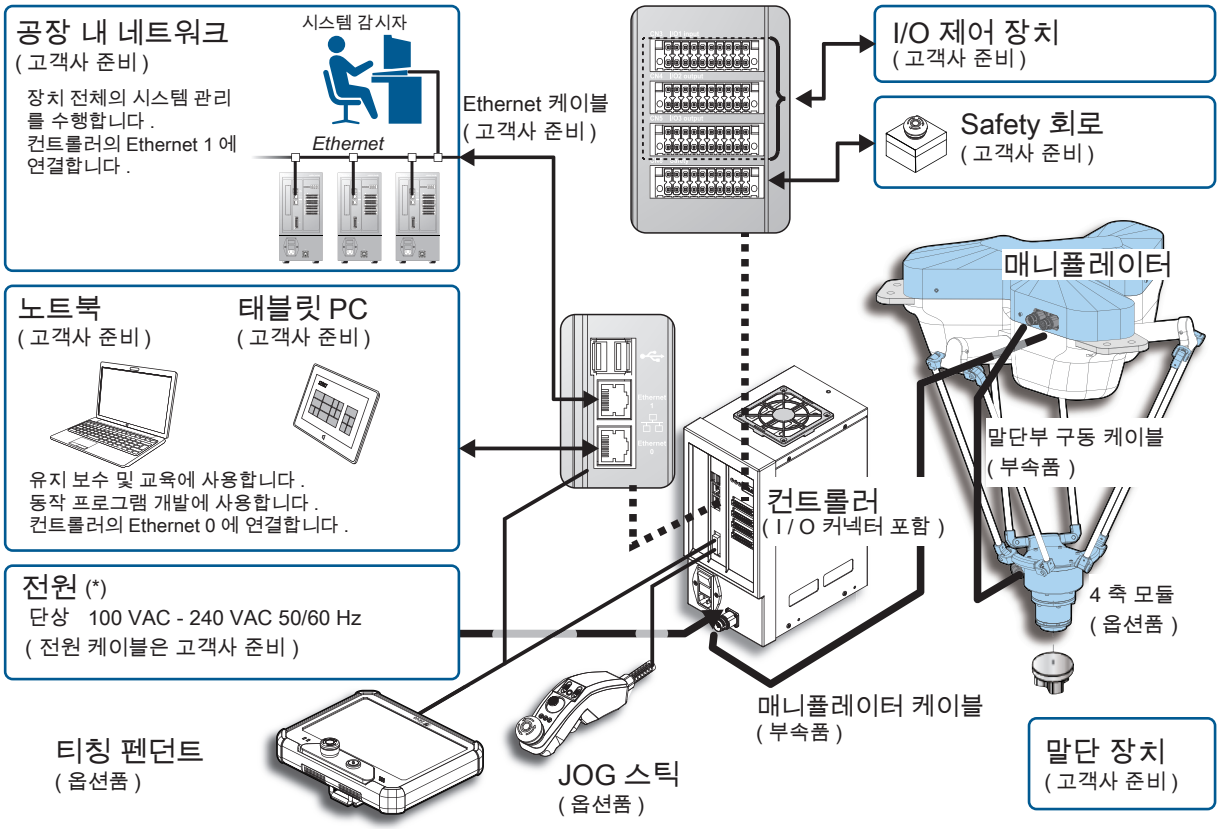
- N : 타입

기호	사양
N	3 자유도 Normal Type
R	4 자유도 Normal Type

예 : 매니퓰레이터의 라벨에 기재한 모델



2. 시스템 구성



*) 컨트롤러는 누전 차단기로 보호된 전원에 연결하십시오 .

MEMO

B 하드웨어 편

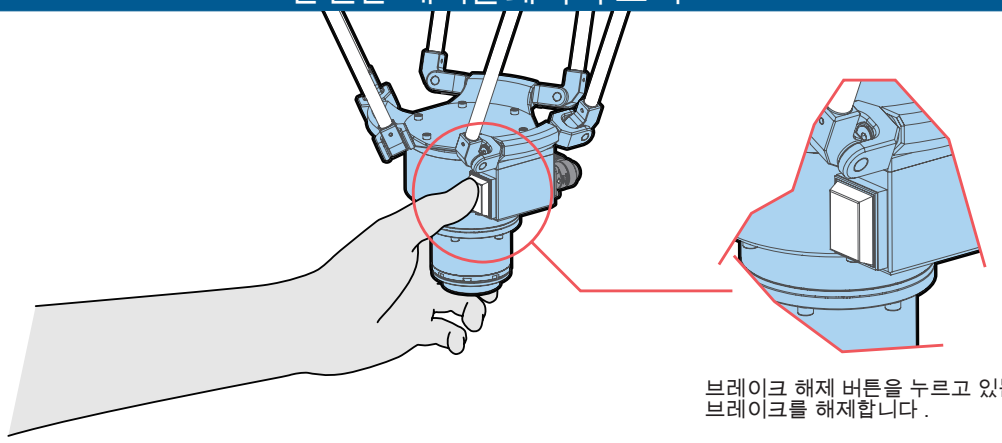
2 매니퓰레이터

1. 개요

1. 특징

반경 R300mm(Φ 600mm) 의 병렬형 로봇으로 , 엔드 이펙터 장착부 상단에 브레이크 해제 스위치가 있어 서보 전원이 차단된 상태에서도 손쉽게 원하는 위치로 이동 , 교시가 가능합니다 .
이외에도 엔드 이펙터의 Z 축 회전 옵션에 따라 "3 자유도 Normal Type" 과 "4 자유도 Normal Type" 으로 나눌 수 있습니다 . 자세한 내용은 문의하시기 바랍니다 .

간편한 매니퓰레이터 교시



브레이크 해제 버튼을 누르고 있는 동안 브레이크를 해제합니다 .

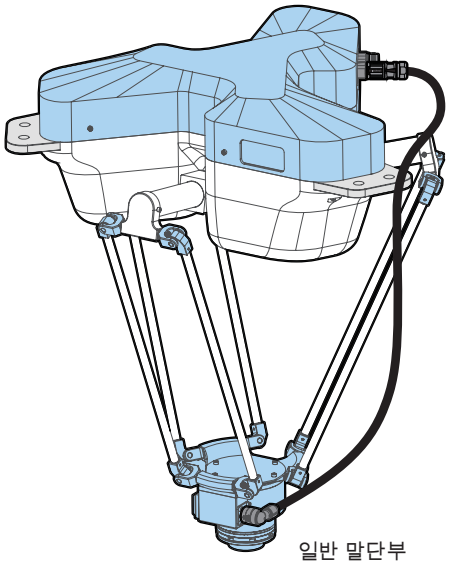
타입에 따른 차이점

3 자유도 Normal Type

XYZ

· XYZ 세 방향의 자유도를 가진다 .

ZRC-0306N



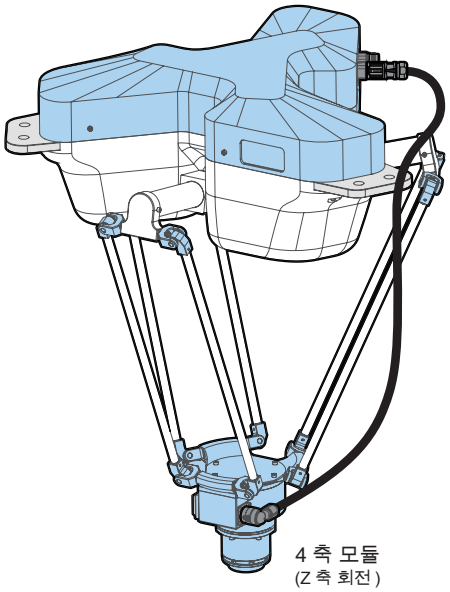
일반 말단부

4 자유도 Normal Type

XYZ + R_Z

· XYZ 세 방향에 더해 Z 축 방향 회전이 가능하다 .

ZRC-0306R

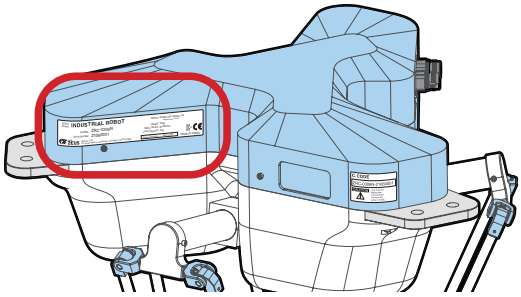


4 축 모듈
(Z 축 회전)

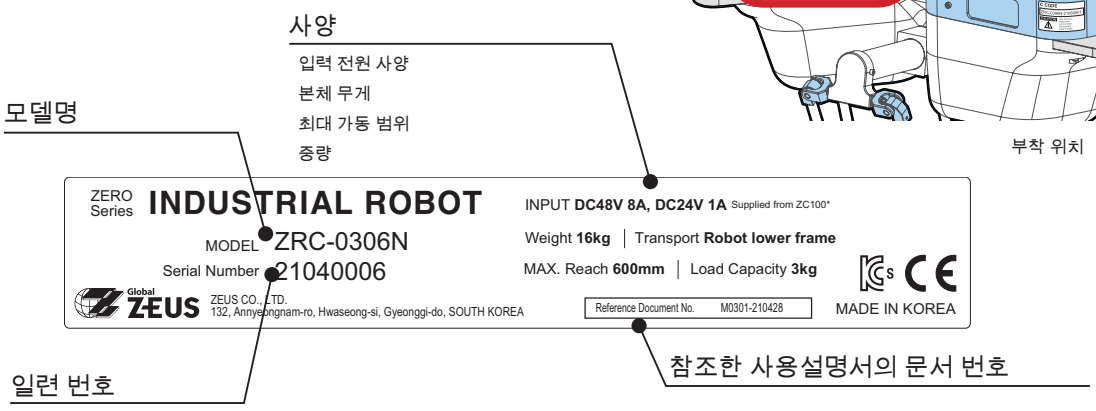
2. 라벨

매니퓰레이터에는 제품 라벨과 C. CODE 라벨이 부착되어 있습니다 .

제품 라벨



부착 위치



일련 번호 보는 방법

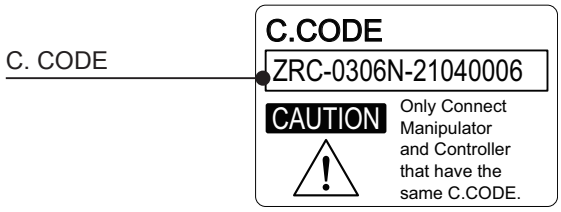
21	04	0006
제작 년도	제작 월	제조 번호

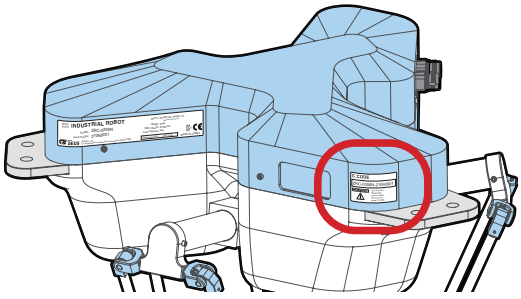
제작 년도 : "21"= 2021 년 (서기 아래 두 자리)
제조 월 : "01"= 1 월 ~ "12"= 12 월
제조 번호 : "0001"~ "9999"

이 제품 라벨은 매니퓰레이터의 기종명 "ZRC-0306N", 일련 번호 "21040006" 의 예입니다 .

C. CODE 라벨

	컨트롤러와 매니퓰레이터는 동일한 C. CODE (연결 코드) 라벨이 부착된 매니퓰레이터와 컨트롤러를 연결하십시오 .	

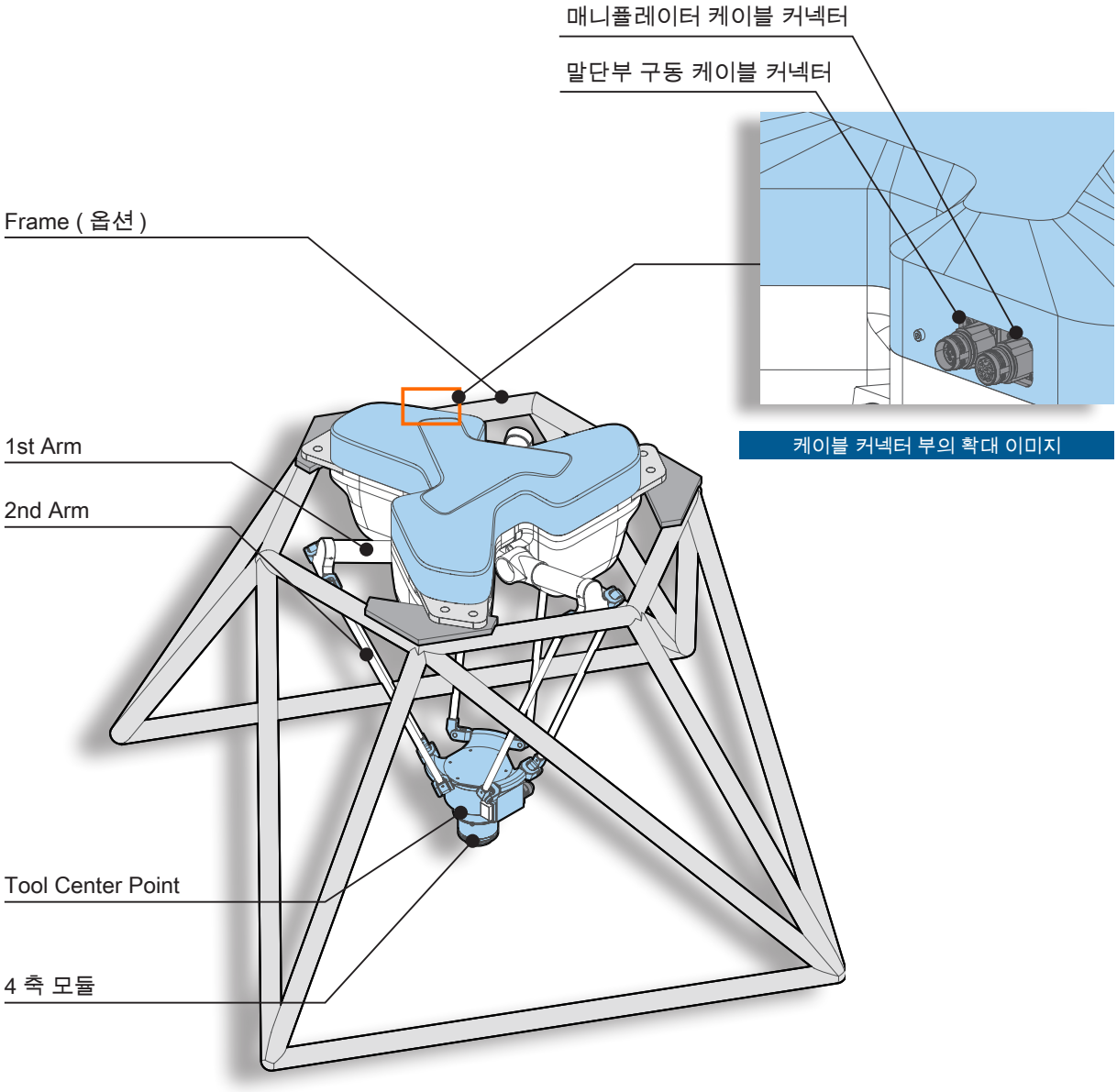




부착 위치

이 C. CODE 라벨은 매니퓰레이터의 기종명 "ZRC-0306N", 일련 번호 "21040006" 의 예입니다 .

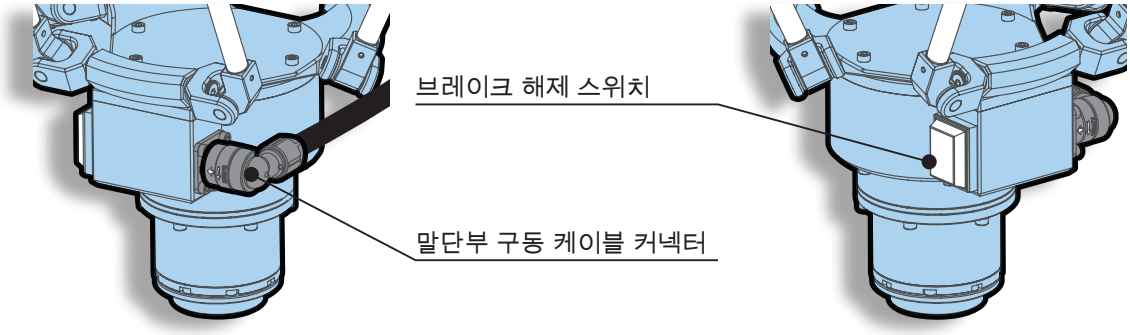
2. 각 부의 명칭



케이블 커넥터 부의 확대 이미지

프레임의 형상은 예시입니다 . 형상과 치수가 제품 개선이나 당사의 사정으로 변경될 수 있습니다 .

Tool Center Point 의 상세설명



3. 설치

주 의

설치 조건에 맞추어 바르게 설치하여 주십시오 .

설치 조건

항목	사양
사용 온도	0 ℃ - 40℃
사용 습도	30 %RH - 85 %RH (결로 주의)
사용 환경	건물 내부 (직사광선을 피할 것) 에서의 사용에 한한다 . 부식성 가스 , 인화성 가스 , 오일 미스트 , 물방울 , 분진 , 가연물 , 연삭재 등이 없는 것 . 통기성이 있고 환기가 잘됨
오염도	2 (IEC60664-1 준거)
진동 · 충격	IEC61131-2 준거 (컨트롤러 한정) 동작 중의 진동 0.5G 이하 (과도한 진동이나 충격이 없을 것)
보호 등급	IP40 (매니퓰레이터 선단부 , 컨트롤러) / IP20 (매니퓰레이터 상단부)
전원	전용 컨트롤러에서 공급
접지	D 종 (접지 저항 100 Ω 이하)
노이즈	주위에 강한 전자기장 (*) 을 발생시키는 것이 없을 것

*) 비정상적으로 강한 전자기장에 의해 로봇이 오작동할 소지가 있습니다 .

본서에 기재된 사양은 일반 사양입니다 상세한 내용은 납품사양서를 참조하여 주십시오 .

주 의

로봇을 파손시킬 가능성이 있는 기계와 로봇을 조합하거나 함께 사용하는 경우 , 다른 기계의 작업 공간 밖에서 , 모든 기능과 동작 프로그램을 , 개별적으로 시험하는 것을 추천합니다 .

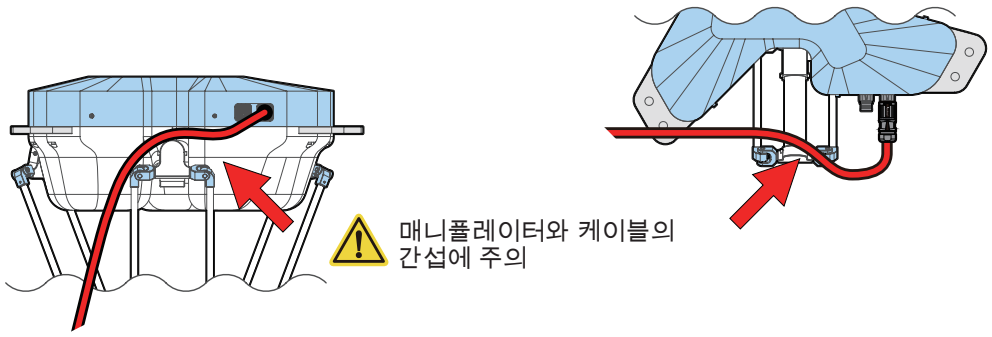
설치 형태

주 의

지정된 설치 형상을 준수하시고 , 바르게 설치하여 주십시오 .

주 의

매니퓰레이터의 가동 범위에 케이블이나 커넥터가 간섭받지 않게 해주세요 .



주 의

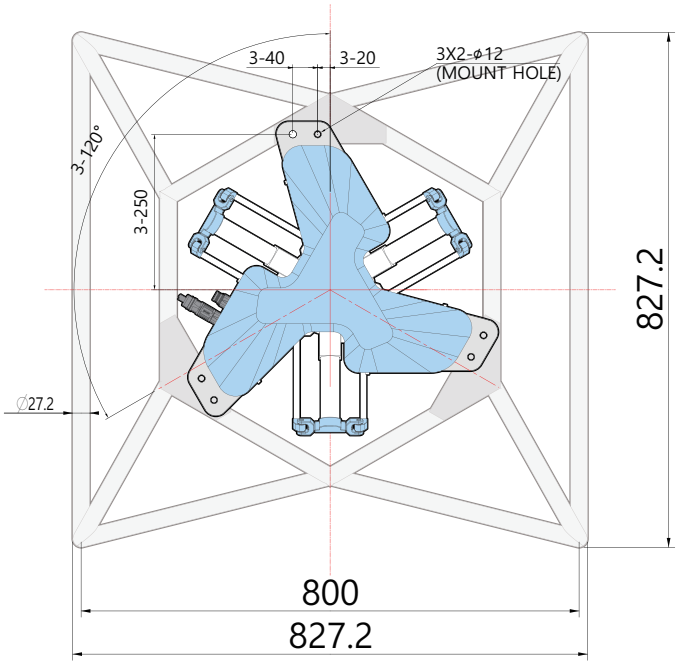
매니퓰레이터의 설치는 프레임의 설치 고정용 치수도를 , 말단 Tool 의 설치는 Tool center point 의 설치용 치수도를 참고해주십시오 .
프레임은 6 개의 나사를 전부 체결하는 것을 권장합니다 .

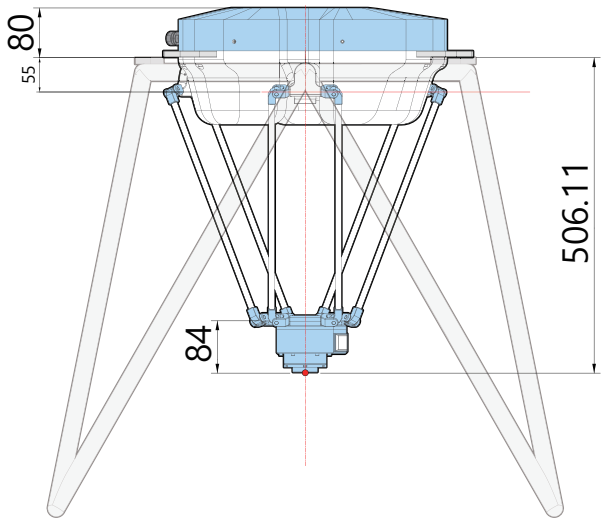
4. 치수도

ZRC-0306N 자유도 : 3 Normal Type

Not to Scale

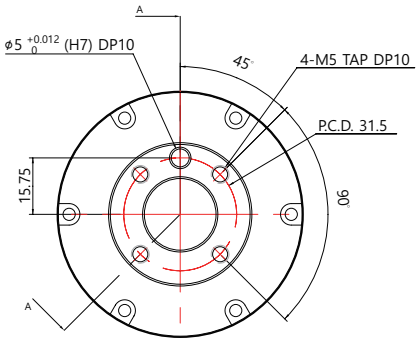
(mm)

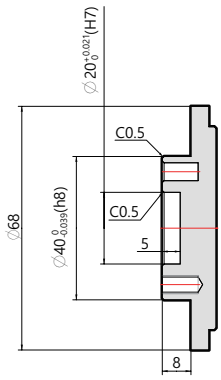




Tool center point

말단 Tool 설치 용





단면 A-A

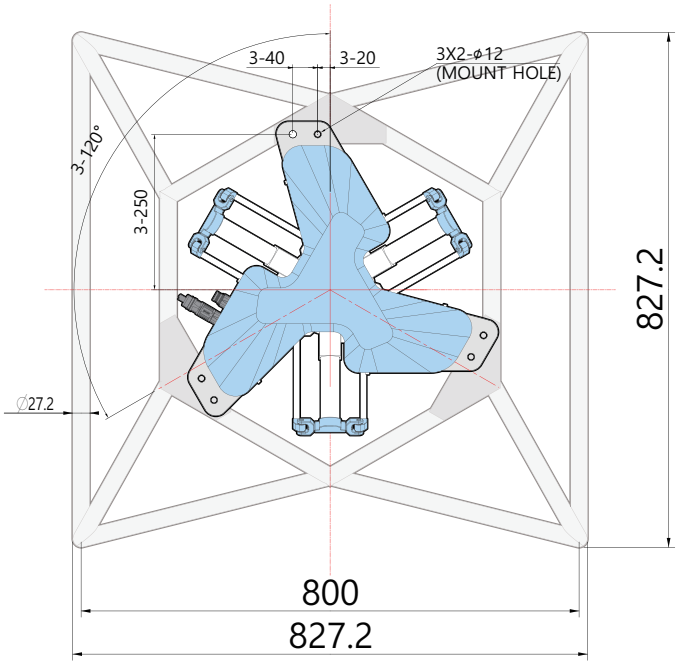
프레임의 형상은 예시입니다 . 형상과 치수가 제품 개선이나 당사의 사정으로 변경될 수 있습니다 .

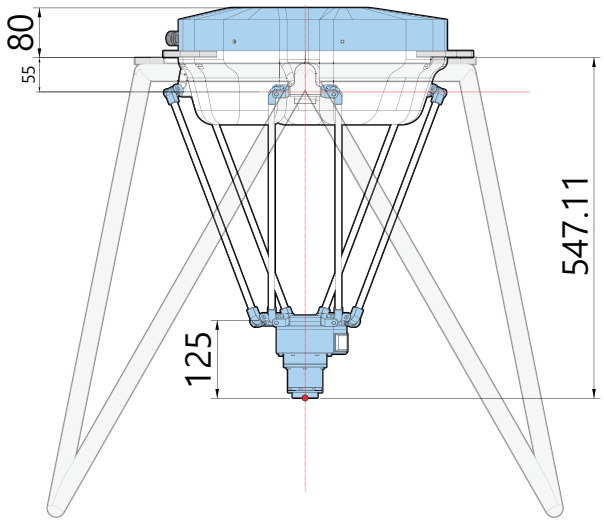
매니퓰레이터의 프레임 고정

프레임의 고정에는 길이 30 mm 이상의 육각렌치볼트 (M10) 를 사용해 주십시오 .

ZRC-0306R 자유도 : 4 Normal Type

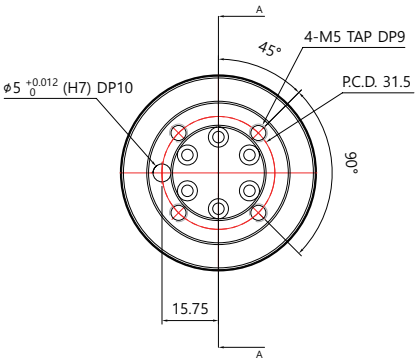
Not to Scale
(mm)

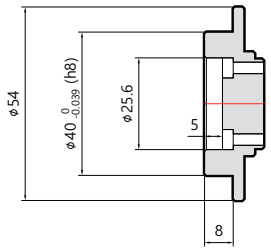




Tool center point

말단 Tool 설치 용





단면 A-A

프레임의 형상은 예시입니다 . 형상과 치수가 제품 개선이나 당사의 사정으로 변경될 수 있습니다 .

매니퓰레이터의 프레임 고정

프레임의 고정에는 길이 30 mm 이상의 육각렌치볼트 (M10) 를 사용해 주십시오 .

5. 사양

항 목		단위	ZRC-0306N	ZRC-0306R
구조		—	병렬형 로봇	
자유도(DOF)		—	3	4
설치 위치		—	바닥 (프레임 옵션), 천장	
구동 방식		—	BLDC 모터	
위치 검출 방식		—	Multi-turn Absolute Encoder (Battery Backup)	
위치 제어 방식		—	서보 제어	
브레이크		—	Electromagnetic Brake	
가반 중량 (*1)	정격 최대	kg	1 3	
동작범위		—	Φ 600 x H200	
가동 범위	J1 J2 J3 Roll	deg	160 (-60 ~ +100) 160 (-60 ~ +100) 160 (-60 ~ +100) —	720 (± 360)
합성 속도 (*2)	XYZ Roll	mm/s deg/s	1800 —	1000
반복 정밀도	XYZ Roll	mm deg	±0.1 —	±0.02
말단 허용 관성 (*3)	정격 최대	kg· m^2	0.025 0.05	
외형 치수			827 x 827 x 667	
본체 중량		kg	16 (본품), 26 (프레임 포함)	17 (본품), 27 (프레임 포함)
모터 전력 소비량		W	600	
전용 컨트롤러		—	ZC1***	
매니퓰레이터 케이블 길이		m	3	
매니퓰레이터 고정		—	M10 볼트 6곳 (치수도 참고)	
말단 장치 고정		—	(치수도 참고)	

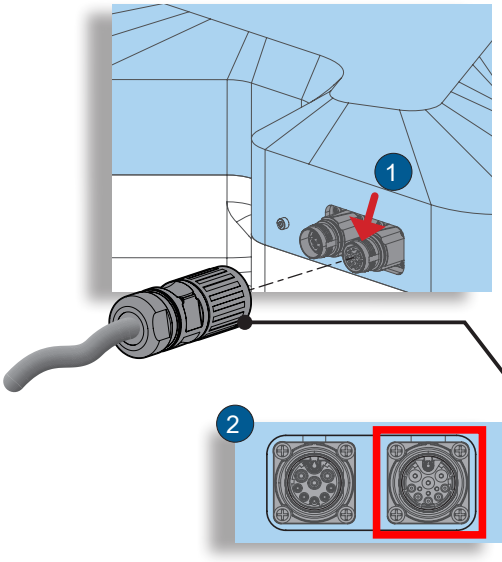
*1) 가반 중량은 작업, 도구 등 모두를 포함합니다. 로봇 동작의 자세, 속도, 가감속시간, 동작방향 등에 따라 사양 범위 내라도, 허용 토크 초과 오류 c13 및、과부화 오류 c14 가 발생할 수 있습니다. 동작 속도, 동작 자세, 가감속 시간 등을 변경 및 조절하십시오.
*2) 값은 참고값입니다.
*3) 가감속 등의 동작 조건에 따라 달라집니다.

보충) 본 제품은 정지 카테고리 "0"입니다. PL = d에 해당합니다.

6. 커넥터

1. 매니퓰레이터 케이블 연결 커넥터

매니퓰레이터에 연결하는 방법

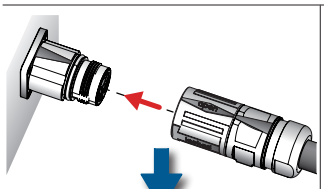
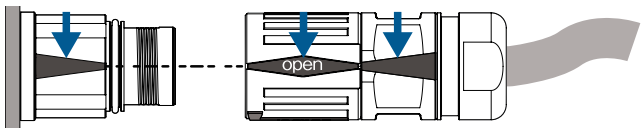
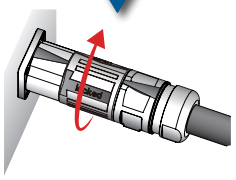
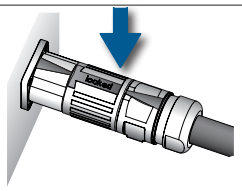
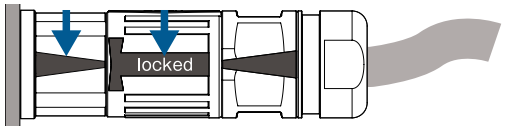


1. 매니퓰레이터의 케이블 커넥터 위치를 확인합니다 .
2. 컨트롤러와 연결하는 커넥터의 모양을 확인합니다 .
3. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .

3 매니퓰레이터 케이블

맞춤 기호가 "locked" 으로 되어있는지 확인합니다 .

컨트롤러와 매니퓰레이터의 커넥터 연결 방법

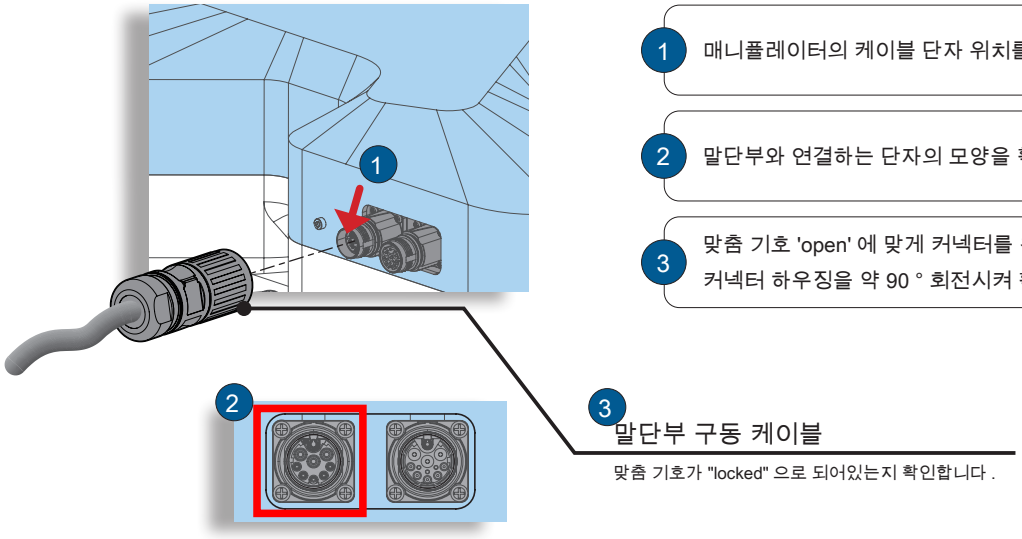
	맞춤 기호를 'open' 에 맞게 커넥터를 완전히 삽입합니다 . 
	커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .
	맞춤 기호가 "locked" 으로 되어 있는지 확인합니다 . 

컨트롤러와 매니퓰레이터는 올바른 조합으로 연결하시기 바랍니다 .

컨트롤러와 매니퓰레이터의 C.CODE 라벨을 확인하고 C.CODE 가 일치하도록 연결합니다 .

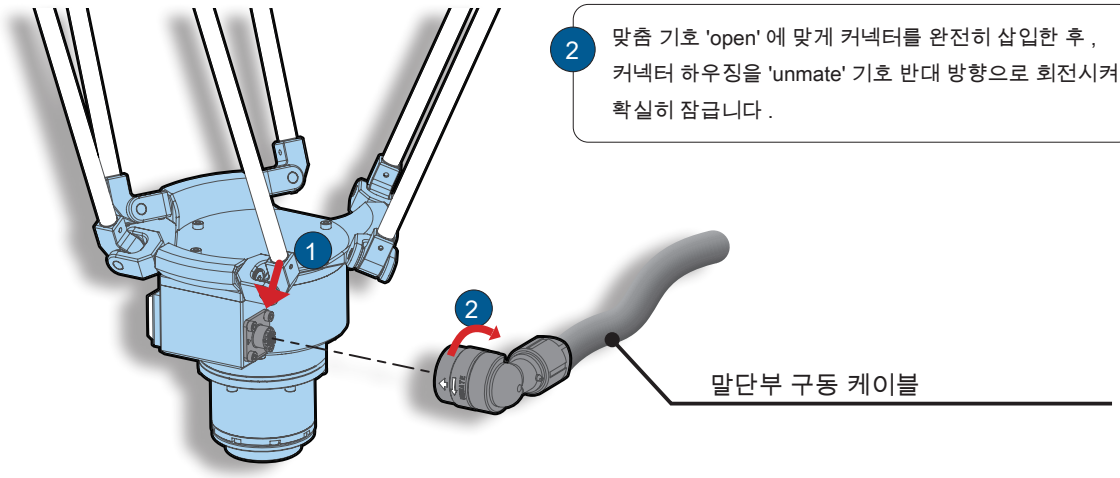
2. 말단부 구동 케이블 연결 커넥터

매니퓰레이터에 연결하는 방법



1. 매니퓰레이터의 케이블 단자 위치를 확인합니다 .
2. 말단부와 연결하는 단자의 모양을 확인합니다 .
3. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .

4 축 모듈에 연결하는 방법

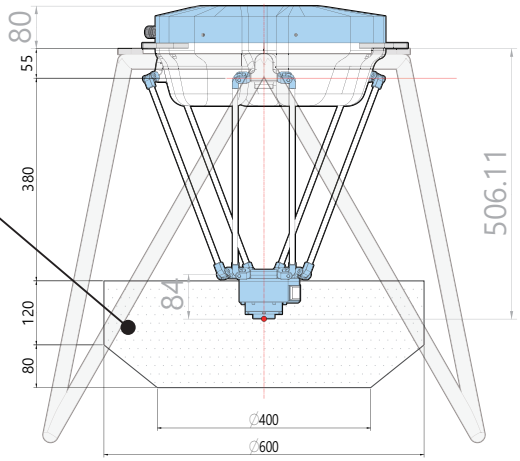


1. 말단부의 케이블 단자 위치를 확인합니다 .
2. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 'unmate' 기호 반대 방향으로 회전시켜 확실히 잠급니다 .

7. 동작 범위

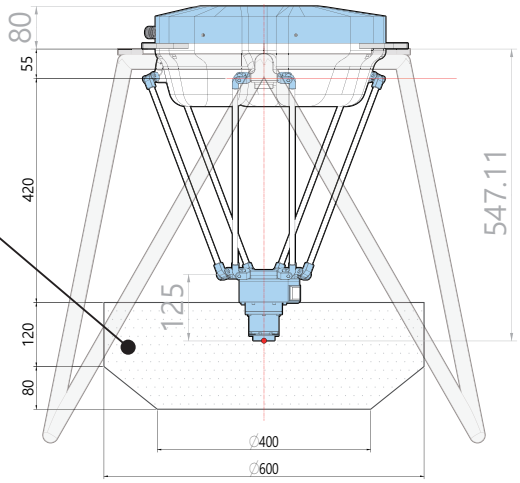
ZRC-0306N 동작 범위 직경 : 600 mm

Tool center point 최대 도달 범위 :
∅ 600, 높이 200 인 원주형



ZRC-0306R 동작 범위 직경 : 600 mm

Tool center point 최대 도달 범위 :
∅ 600, 높이 200 인 원주형



프레임의 형상은 예시입니다 . 형상과 치수가 제품 개선이나 당사의 사정으로 변경될 수 있습니다 .

8. 말단 장치 설계

주 의

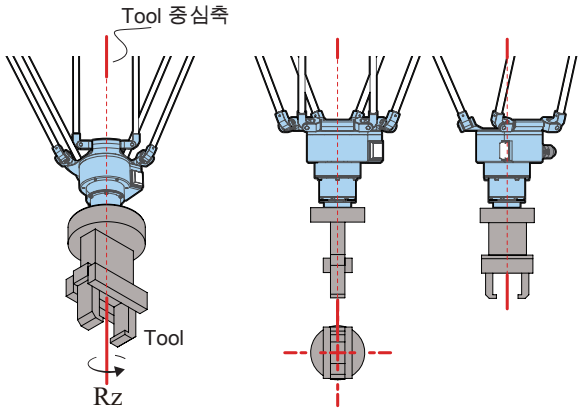
Tool center point 에 붙이는 Tool 의 설계는 , 아래의 예시를 참고로 하여 매니퓰레이터의 자세나 작동 범위에 충분한 검증을 실시해 주십시오 .

예시 1

권장합니다 .

Rz 회전축과 Tool 의 중심이 일치시킵니다 .

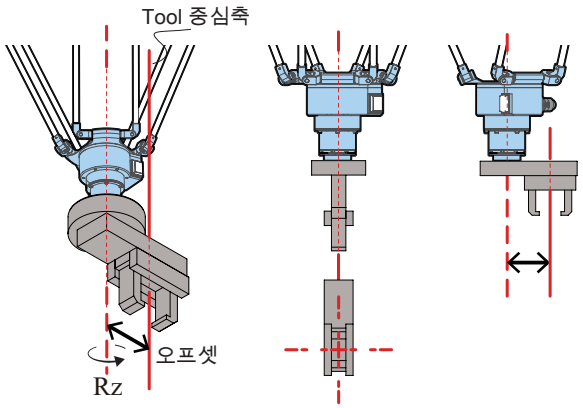
Tool center point 로부터 Tool 의 끝 까지의 거리가 멀어지면 , 매니퓰레이터에 걸리는 부하가 커지기 때문에 , 진동의 발생이나 동작 속도 저하의 원인이 될 수 있습니다 .



예시 2

권장하지 않습니다 .

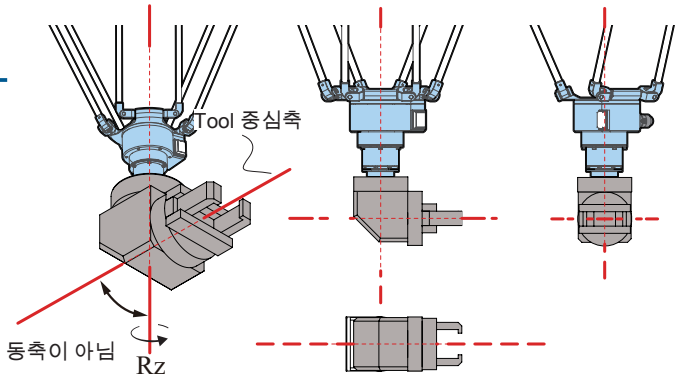
Tool 의 중심축과 Rz 회전축 사이에 오프셋이 있어 , 작업물을 핸들링하지 못할 가능성이 있습니다 .



예시 3

권장하지 않습니다 .

Tool 의 중심축과 Rz 회전축이 동축이 아니기 때문에 작업물을 핸들링하지 못할 가능성이 있습니다 .



MEMO

B 하드웨어

3 컨트롤러

1. 모델명과 라벨

1. 모델명

컨트롤러 모델명 : ZC 1001

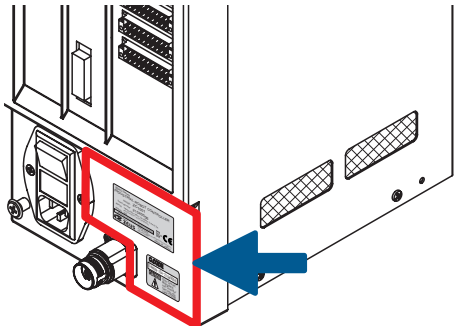
제품 코드

모델

제품 코드

기호	해당 로봇
1000	ZERO series
1001	ZERO series + ZP1000

2. 라벨

라벨	부착 위치
제품 라벨 ZERO Series **INDUSTRIAL ROBOT CONTROLLER** MODEL ZC1001 Serial Number 21040006 INPUT 1ΦAC100V-240V 50-60Hz 5.4A-2.7A Primary Fuses F1 and F2 8A/250VAC Weight 5kg Reference Document No. M0201-210408 Global ZEUS ZEUS CO., LTD. 132, Annyeongnam-ro, Hwaseong-si,Gyeonggi-do, SOUTH KOREA MADE IN KOREA C. CODE 라벨 C.CODE ZRC-0306N-21040006 CAUTION Only Connect Manipulator and Controller that have the same C.CODE.	

이 라벨은 매니퓰레이터의 일련 번호 "21040006", C. CODE "0440N-21040006" 의 경우의 예입니다 . 이러한 표기는 제품마다 다릅니다 .
일련 번호 체계는 매니퓰레이터와 동일합니다 .

컨트롤러와 매니퓰레이터를 같은 C.CODE 조합으로 연결하여 주시기 바랍니다 .

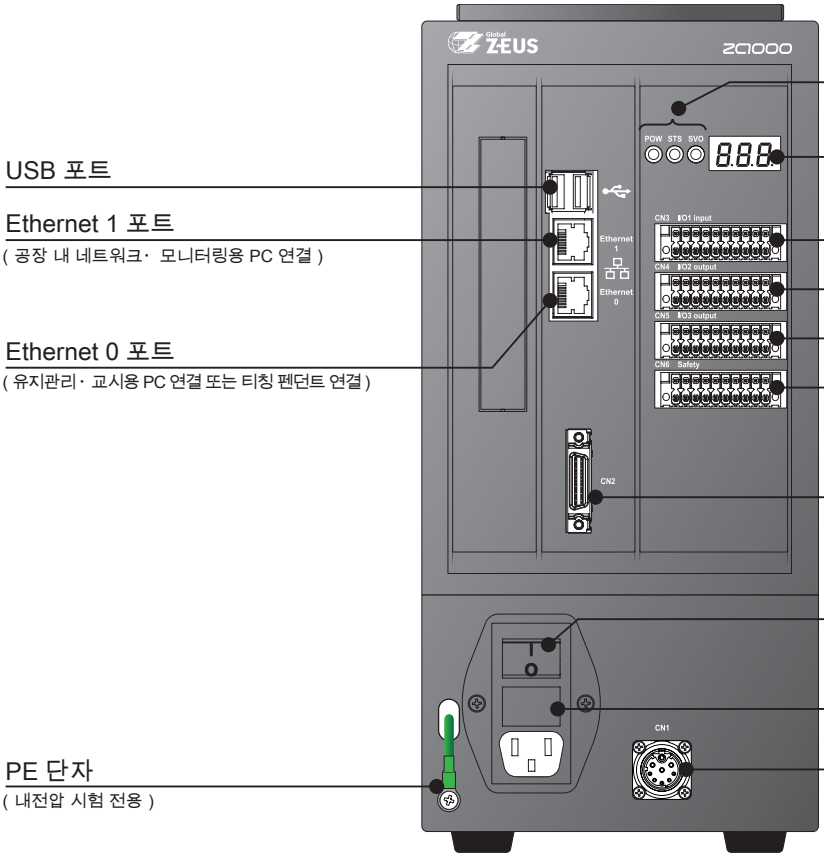
C.CODE 는 컨트롤러와 매니퓰레이터의 조합을 표시합니다 . 각각의 C. CODE 라벨을 확인하고 , C. CODE 가 일치하도록 연결해 주시기 바랍니다 .

2. 각 부의 명칭

⚠ 주 의

PE 단자는 내전압시험 전용입니다 .

- 나사를 제거하지 마십시오 .
- 아무 것도 연결하지 마시기 바랍니다 .



USB 포트

Ethernet 1 포트
(공장 내 네트워크· 모니터링용 PC 연결)

Ethernet 0 포트
(유지관리· 교시용 PC 연결 또는 티칭 펜던트 연결)

PE 단자
(내전압 시험 전용)

LED 표시등

7 세그먼트 LED 표시등

CN3 I/O 커넥터 1 (입력)

CN4 I/O 커넥터 2 (출력)

CN5 I/O 커넥터 3 (출력)

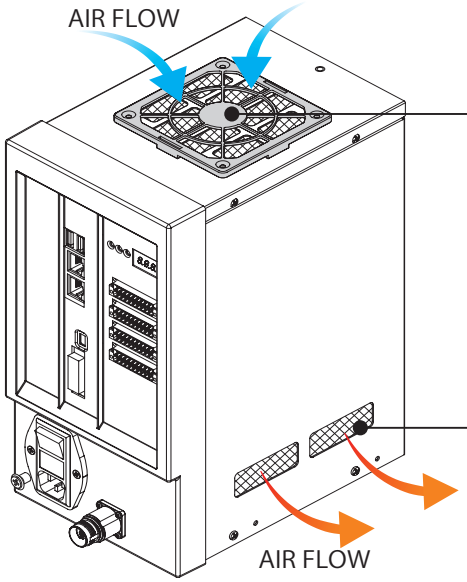
CN6 Safety 커넥터

CN2
(점퍼 커넥터 , JOG 스틱 또는
티칭 펜던트 연결)

AC 전원 스위치

퓨즈 박스

CN1
(매니퓰레이터 연결)



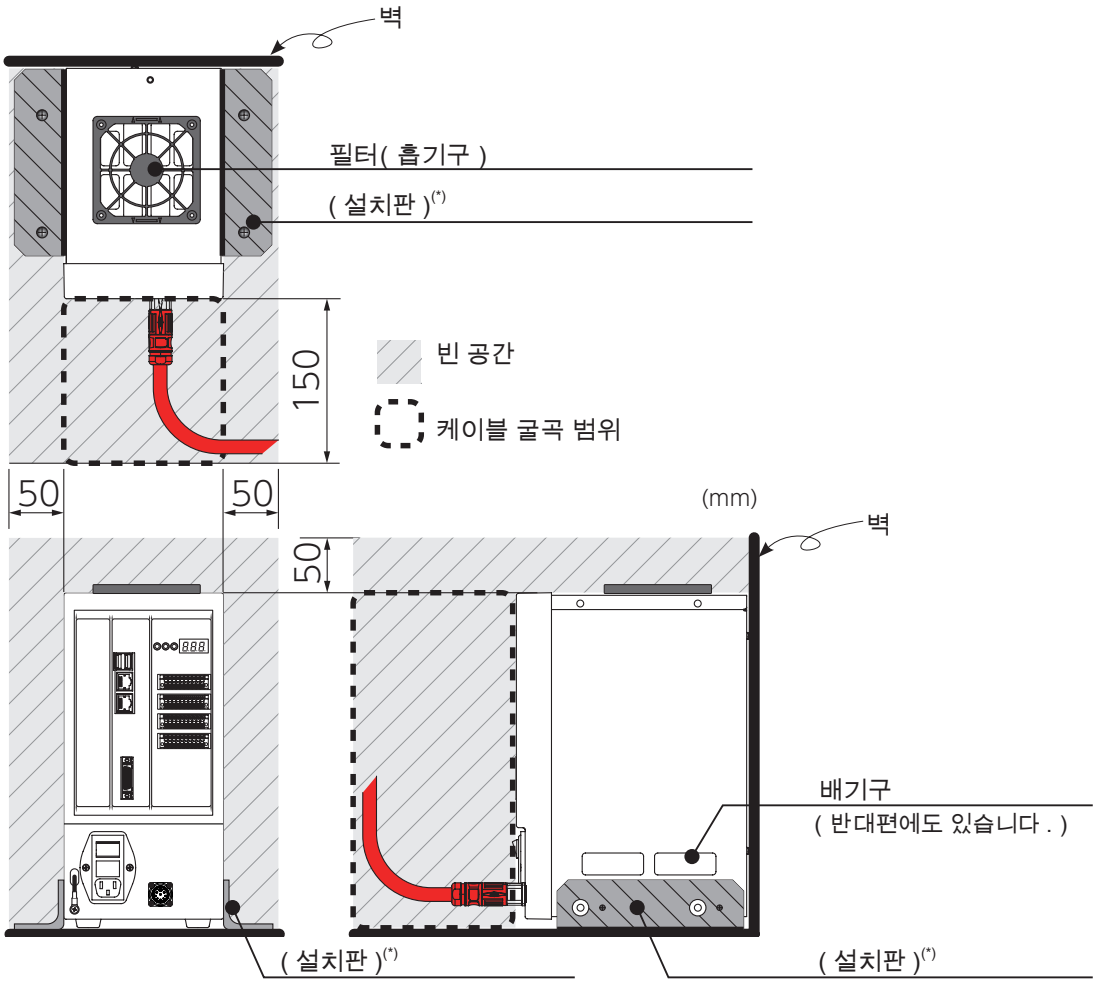
필터 유닛
(흡기)

배기구

배기구는 반대편에도 있습니다 .

3. 설치

⚠ 주 의		
	폐쇄된 공간에는 설치하지 마시기 바랍니다 . 배기구와 흡기구를 막지 마십시오 .	
	주변 온도가 40 ℃ 이하의 환경에서 사용하시기 바랍니다 .	
	컨트롤러는 아래의 그림을 참고하여 충분한 공간을 확보할 수 있는 편평한 장소에 설치하여 주시기 바랍니다 . 컨트롤러 측면의 고정용 나사 (M3X4 개) 를 사용하여 전도 방지 조치를 하는 것을 권장합니다 .	
	필터는 지정된 필터로 정기적으로 교환해 주십시오 .	

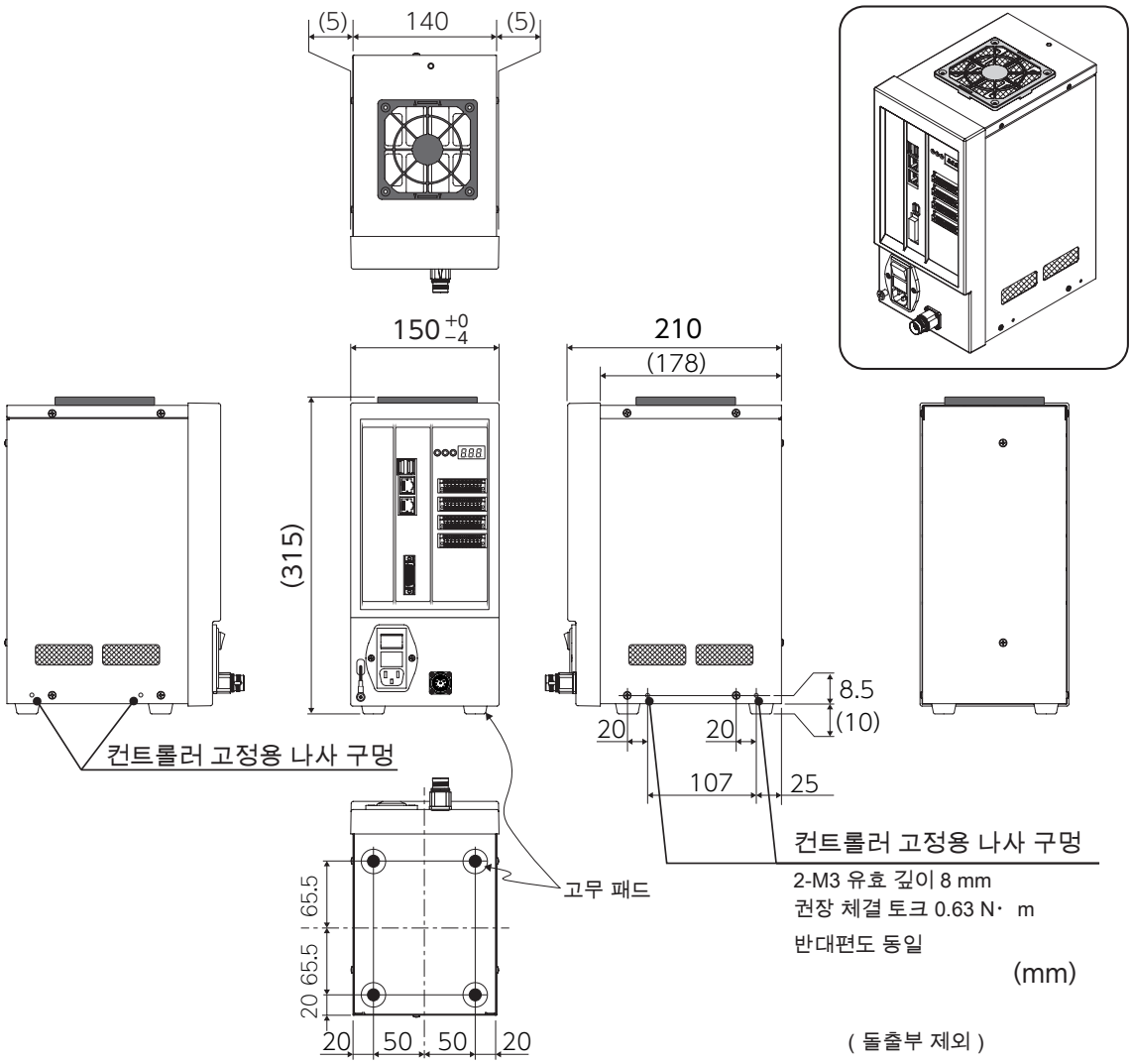


*) 설치판은 부속품이 아닙니다 .

4. 외관도

⚠ 주 의

고정 브라켓을 제작하는 경우 , 컨트롤러 고정용 나사 구멍에서 20 mm 위치에 있는 커버 고정 나사 머리에 간섭되지 않도록 하고 , 공기 흡입구를 막지 않도록 설계하여 주시기 바랍니다 .



5. 사양

1. 일반 사양

항목		ZC1000	ZC1001	비고
일반 사양	적용 로봇	ZERO series		티칭 펜던트 (ZP1000) 교시 시에는 ZC1001 필요
	외형	외관도 참조		돌출부 제외
	무게	5 kg		—
	제어 축 수	6 축		—
	프로그래밍 방법	PC 를 통한 오프라인 프로그래밍		어플리케이션 프로그램은 FTP 전송하여 실행
	프로그래밍 언어	Python		로봇 조작은 전용 라이브러리 사용
	저장 메모리	eMMC		—
	교시 방식	PC JOG 스틱 조작	PC JOG 스틱 조작 티칭 펜던트 조작	Web 브라우저를 사용하여 모니터링 및 데이터 저장· 제어
표시 기능	7 세그먼트 표시	3 자리		—
	상태 표시 LED	3 종		—
인터페이스 (컨트롤러)	매니퓰레이터 연결	1 Port		—
	입력	16 Bit		절연 상부 / 하부 선택 가능
	출력	16 Bit		절연 상부 / 하부 선택 가능
	안전	1 Port		EMS x 2 , Mode , 서보 ON 입력 , 서보 전원 모니터
	Ethernet	2 Port		—
	USB	2 Port		—
	JOG 스틱	1 Port		교시용 입력 장치 전용 I/F
인터페이스 (암 I/O)	디지털 입력	8 Bit		비절연 Comparator 입력
	디지털 출력	4 Bit		비절연 상부 스위치
	비동기 통신	1 Ch		RS422/RS485
	전원 출력	24 V		0.2 A max
전원 사양(*)	전압	단상 100 VAC - 240 VAC		—
	주파수	50 Hz - 60 Hz		—
	전류	2.7 A, 230 VAC / 5.4 A, 115 VAC		—
	돌입 전류	75 A, 230 VAC		—
	누설 전류	5.0 mA, 240 VAC		—
	단락 전류 정격	1,500 A		UL File No. E10480 기준
접지		3 종 접지 이상		접지 저항 100 Ω 이하
안전	규격	ISO 10218-1		준수
	내전압	1,500 VAC		1 차-FG, 1 분간
	절연 저항	1 M Ω 이상		I/P -FG 500VDC / 25℃ / 70%RH
EMC		EN61000-6-2:2005 EN55011 : 2009+A1:2010		중공업 수준

*) 전압 변동이 입력 전압 범위 내의 것
20 ms 이상의 순간 정전이 없을 것
돌입 전류를 포함하여 충분한 용량의 전원을 확보
퓨즈는 정격 전류 : 8A, 정격 차단 용량 : AC250 V / 1,500 A 를 사용

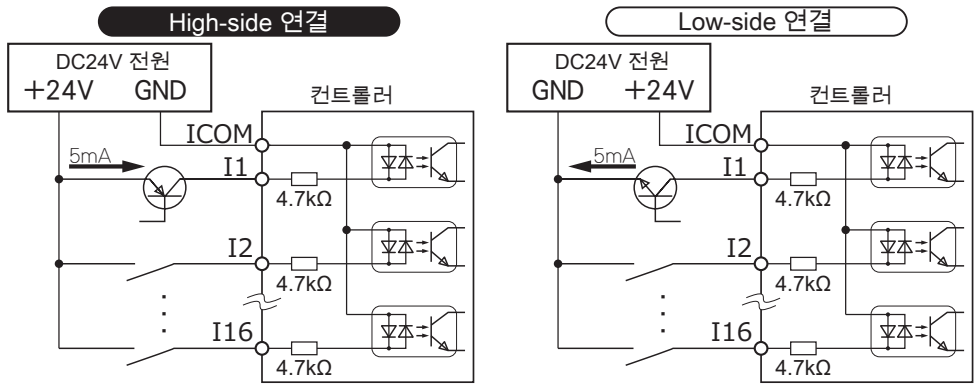
본 문서에 기재된 사양 항목과 그 내용은 일반 사양입니다 . 상세한 것은 납입 사양서를 참조해 주십시오 .

2. I/O 커넥터의 입출력 회로

입력 회로

항목	사양
방식	포토 커플러 입력 (16 점 공통 전원 입력형)
정격 전압	DC 24 V
입력 ON 전류	5 mA typ.(입력 OFF 전류 2.5mA 이하)

연결 예시

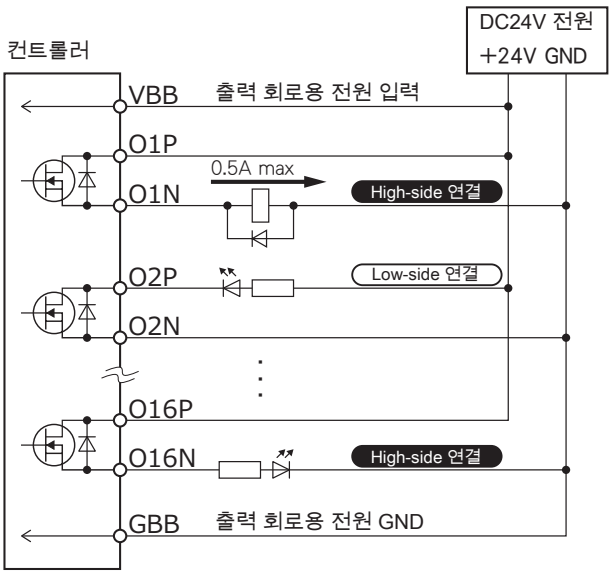


출력 회로

항목	사양
방식	드레인 , 소스 독립 출력 (상부 , 하부 개별 대응)
정격 전압	DC 24 V
정격 전류	0.5 A (출력 전류 제한 0.6 A - 1.2 A)

- 출력은 표시기 , 릴레이 , 부저 등 각종 기기를 연결할 수 있습니다 .
- 접속하는 기기의 사용설명서와 배선 예를 참고하십시오 .
- 릴레이 등 인덕턴스 성분을 갖는 기기를 연결할 때 보호 회로 (다이오드) 를 연결하십시오 .

연결 예시



6. 커넥터

1. I/O 커넥터

CN3 : I/O 커넥터 1 (입력)

단자	신호명	내용	단자	신호명	내용
1A	P24	컨트롤러 24 V 출력 (*1)	1B	–	—
2A	IN1	범용 입력	2B	IN2	범용 입력
3A	IN3	범용 입력	3B	IN4	범용 입력
4A	IN5	범용 입력	4B	IN6	범용 입력
5A	IN7	범용 입력	5B	IN8	범용 입력
6A	IN9	범용 입력	6B	IN10	범용 입력
7A	IN11	범용 입력	7B	IN12	범용 입력
8A	IN13	범용 입력	8B	IN14	범용 입력
9A	IN15	범용 입력	9B	IN16	범용 입력
10A	G24	컨트롤러 24 V GND (*1)	10B	ICOM	입력 공용

CN4 : I/O 커넥터 2 (출력)

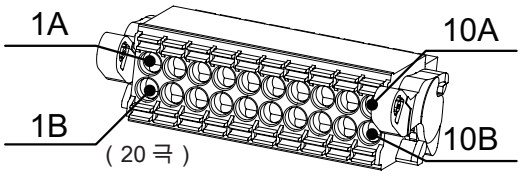
단자	신호명	내용	단자	신호명	내용
1A	P24	컨트롤러 24 V 출력 (*1)	1B	VBB	출력 회로 용 24 V 전원입력 (*2)
2A	O1P	범용 출력 Drain	2B	O1N	범용 출력 Source
3A	O2P	범용 출력 Drain	3B	O2N	범용 출력 Source
4A	O3P	범용 출력 Drain	4B	O3N	범용 출력 Source
5A	O4P	범용 출력 Drain	5B	O4N	범용 출력 Source
6A	O5P	범용 출력 Drain	6B	O5N	범용 출력 Source
7A	O6P	범용 출력 Drain	7B	O6N	범용 출력 Source
8A	O7P	범용 출력 Drain	8B	O7N	범용 출력 Source
9A	O8P	범용 출력 Drain	9B	O8N	범용 출력 Source
10A	G24	컨트롤러 24 V GND (*1)	10B	GBB	출력 회로 용 24 V 전원 GND (*2)

CN5 : I/O 커넥터 3 (출력)

단자	신호명	내용	단자	신호명	내용
1A	P24	컨트롤러 24 V 출력 (*1)	1B	VBB	출력 회로 용 24 V 전원입력 (*2)
2A	O9P	범용 출력 Drain	2B	O9N	범용 출력 Source
3A	O10P	범용 출력 Drain	3B	O10N	범용 출력 Source
4A	O11P	범용 출력 Drain	4B	O11N	범용 출력 Source
5A	O12P	범용 출력 Drain	5B	O12N	범용 출력 Source
6A	O13P	범용 출력 Drain	6B	O13N	범용 출력 Source
7A	O14P	범용 출력 Drain	7B	O14N	범용 출력 Source
8A	O15P	범용 출력 Drain	8B	O15N	범용 출력 Source
9A	O16P	범용 출력 Drain	9B	O16N	범용 출력 Source
10A	G24	컨트롤러 24 V GND (*1)	10B	GBB	출력 회로 용 24 V 전원 GND (*2)

I/O 커넥터 , Safety 커넥터 모델명

DFMC 1,5/10-ST-3,5-LR 1790564
(Phoenix Contact 주식회사)



I/O 의 기능 할당은「 D 소프트웨어 3 메모리맵」을 참조하십시오 .

*1) 입력 , 출력 , 말단 I / O 에서 사용하는 소비 전류는 총 100 mA 이하로 하십시오 .
*2) 출력 회로 전원 공급 장치 (VBB, GBB)
I / O 커넥터 2 와 3 의 VBB, GBB 는 독립적으로하지 않습니다 . 이 단자에 다른 계통의 전원을 연결하지 마십시오 .

2. Safety 커넥터

Safety 커넥터는 반드시 연결하십시오 .

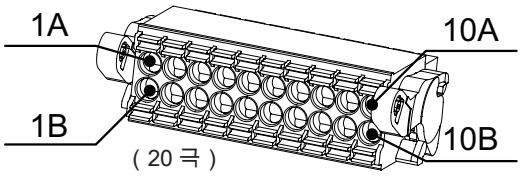
Safety 커넥터가 제대로 연결되어 있지 않으면 매니퓰레이터를 작동시킬 수 없습니다 .

「 5 배선과 전원 」을 참조하십시오 .

CN6 : Safety 커넥터

단자	신호명	내용	단자	신호명	내용
1A	EMS1_H+ (P24)	비상 정지 스위치 1a, 컨트롤러 24V 출력 (*3)	1B	E MS1_L+ (P24)	비상 정지 스위치 1a, 컨트롤러 24V 출력 (*3)
2A	EMS1_H–	비상 정지 스위치 1a (*3)	2B	EMS1_L–	비상 정지 스위치 1a (*3)
3A	EMS2_H+	비상 정지 스위치 2a (*3)	3B	EMS2_L+	비상 정지 스위치 2a (*3)
4A	EMS2_H–	비상 정지 스위치 2a (*3)	4B	EMS2_L–	비상 정지 스위치 2a (*3)
5A	MODE_H+	모드 스위치 (*3)	5B	MODE_L+	모드 스위치 (*3)
6A	MODE_H–	모드 스위치 (*3)	6B	MODE_L–	모드 스위치 (*3)
7A	SVON_MON+	서보 ON 모니터 출력	7B	SVON_MON_–	서보 ON 모니터 출력
8A	READY_H	레디 접점 출력	8B	READY_L	레디 접점 출력
9A	SVON_H+	서보 ON 입력	9B	SVON_H–	서보 ON 입력
10A	NC	사용하지 않는 단자	10B	G24	컨트롤러 24 V GND

Safety 커넥터 (*)
모델명 : DFMC 1,5/10-ST-3,5-LR 1790564
(Phoenix Contact 주식회사)
*) I/O 커넥터 1,2,3 도 동일함



포인트 !

I/O, Safety 커넥터의 연결

- 고압선과 모터 동력선으로부터 분리하여 실드선을 사용하십시오 .
- 노이즈를 고려하여 15 m 이하에서 사용하십시오 .
- 출력에 솔레노이드 및 릴레이 등 인덕턴스 성분을 갖는 기기를 연결하는 경우에는 반드시 보호 회로 (다이오드) 를 연결하여 서지에 대해 대처하십시오 .

*1), *2) : I/O 커넥터와 Safety 커넥터의 이름이 같은 터미널들은 컨트롤러 내부에서 연결되어 있습니다 .
내부 회로는 극성을 가집니다 .
입력 회로는 “(신호명)” 에 + 24V 가 출력되고 있습니다 .
FG 를 포함한 GND 로 단락하면 컨트롤러가 손상될 수 있습니다 .
출력 신호가 DC 24V 라인과 닿게되어 출력이 ON 되지 않도록 적절하게 배선 처리를 하십시오 .

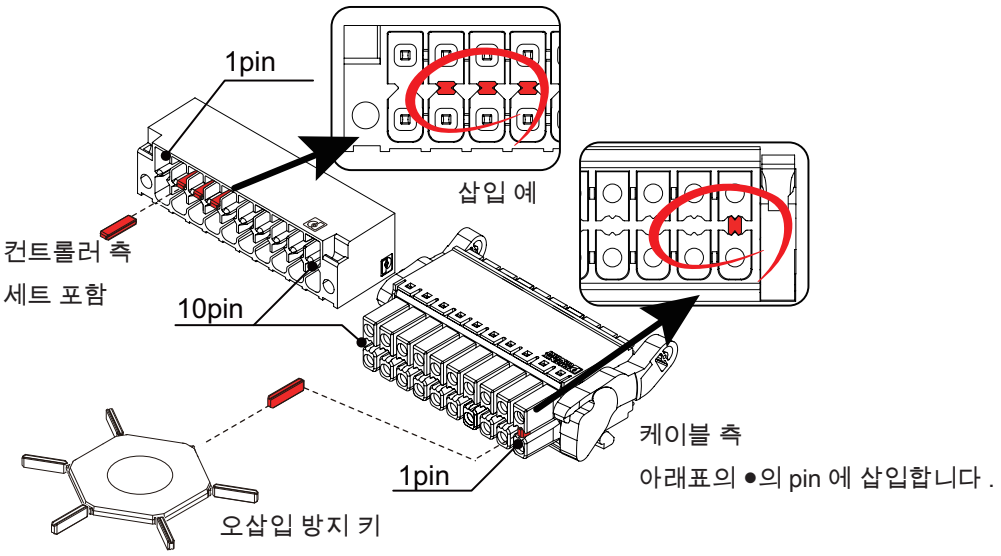
*3) : 커넥터 연결 예를 참고하여 반드시 연결하십시오 .

3. 오삽입 방지 키

주 의

커넥터의 잘못된 접속을 방지하기 위해 I / O 커넥터 1,2,3 와 ,
Safety 커넥터에 제공된 오삽입 방지 키를 각 1 개씩 삽입하십시오 .

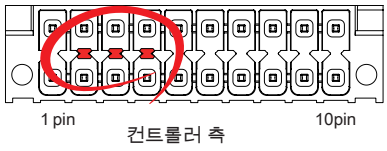
컨트롤러는 오삽입 방지 키가 설정되어 있습니다 .



커넥터		Pin No. 1	2	3	4	5	6	7	8	9	10
컨트롤러 측	CN3 I/O 커넥터 1		◎	◎	◎						
	CN4 I/O 커넥터 2	◎		◎	◎						
	CN5 I/O 커넥터 3	◎	◎		◎						
	CN6 Safety 커넥터	◎	◎	◎							

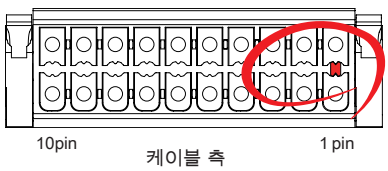
(◎의 위치에 오삽입 방지 키 삽입합니다 .)

예 : CN3 I/O 커넥터 1 의 경우



커넥터		Pin No. 1	2	3	4	5	6	7	8	9	10
케이블 측	CN3 I/O 커넥터 1	●									
	CN4 I/O 커넥터 2		●								
	CN5 I/O 커넥터 3			●							
	CN6 Safety 커넥터				●						

(●의 위치에 오삽입 방지 키 삽입합니다 .)



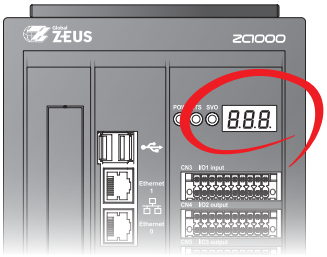
7. 컨트롤러의 상태 표시

컨트롤러의 7 세그먼트 LED 표시기와 LED 표시기에 로봇의 상태를 표시합니다 .

7 세그먼트 LED 의 표시

7 세그먼트 표시기에는 다음의 항목을 표시합니다 .

7 세그먼트 LED 표시기 오른쪽 아래 마침표의 점멸 주기는 컨트롤러 시스템 이 가동 중임을 나타냅니다 .



표시		의미
---	---	컨트롤러 가동중
ini	ini	컨트롤러 초기화중
rdy	rdy	준비완료 상태 (대기중)
inc	inc	ABS 원점 정보의 손실 발생 (*1)
tch	tch	교시 모드
JoG	JoG	JOG 조작 모드
run	run	사용자 프로그램 실행중
PAu	PAu	프로그램 일시 정지중
PoF	PoF	전원 오프 처리중
E88	E**	시스템 정의 오류 (*2, *4)
c88	c**	시스템 정의 오류 치명적 (*2, *5)
u88	u**	사용자 정의 오류 (*3, *4)
r88	r**	사용자 정의 오류 치명적 (*3, *5)

*1) 처음으로 매니퓰레이터를 가동할 경우 , ABS 정보가 손실됩니다 .
*2) 시스템 정의 오류의 자세한 내용은「문제 해결」을 참조하십시오 . 「 Z 자료 」
*3) 사용자 정의 오류는 가끔씩 Python 프로그램에서 발생합니다 . 「 D 소프트웨어 」
*4) 비치명적 오류는 오류의 원인을 제거하고 나서「에러 리셋 신호」와 함께 복구 가능합니다 .
*5) 치명적 오류는 오류의 원인을 제거하고 나서 전원을 재투입해 복구 가능합니다 .

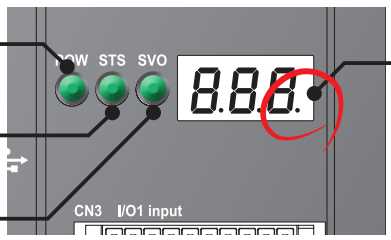
LED 표시등

LED

점등 : 본체 전원 ON

점등 : 사용자 프로그램 실행중
점멸 : 로그 취득중

점등 : 서보 ON 상태



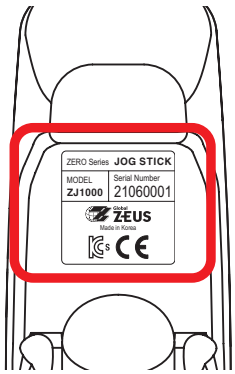
주기적인 점멸 :
시스템 가동 중

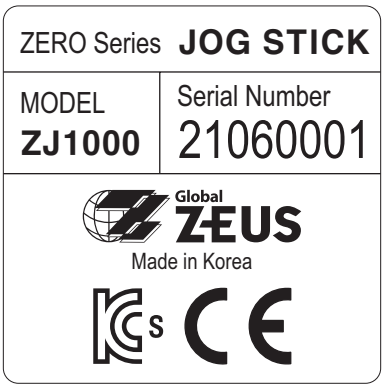
MEMO

B 하드웨어

4 JOG 스틱

1. 제품 라벨

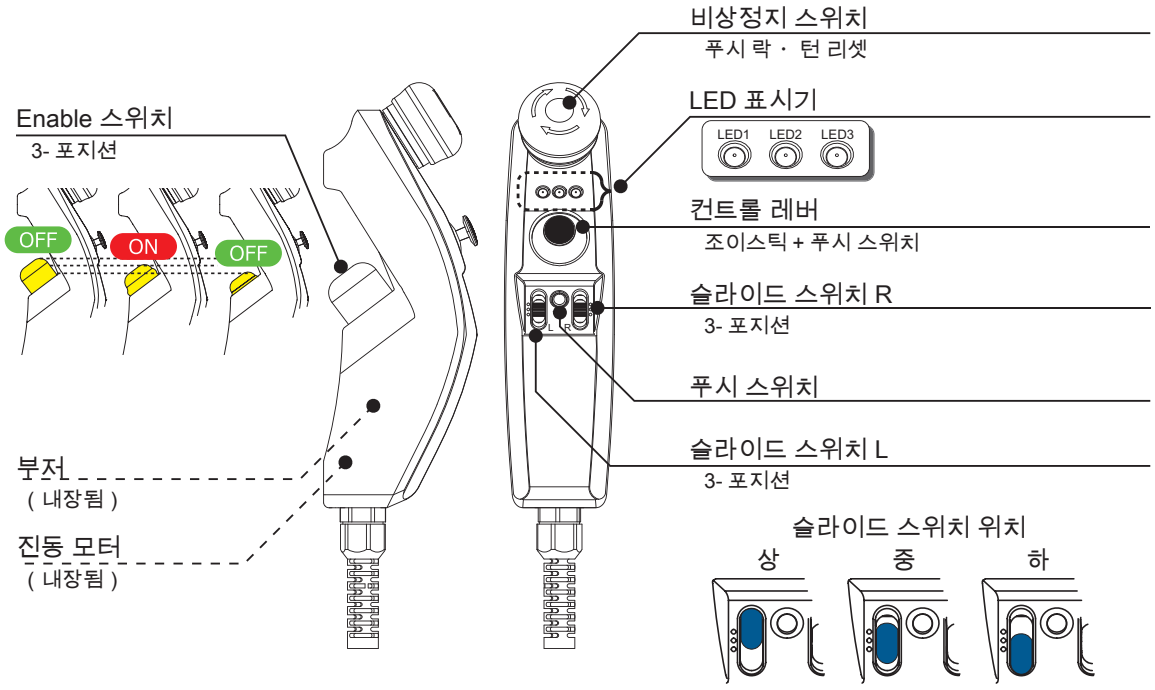
라벨	부착 위치
제품 라벨	  뒷면



위의 라벨은 일련 번호 "21060001" 의 경우의 예입니다 . 일련 번호는 제품마다 다릅니다 .
일련 번호 체계는 매니퓰레이터와 동일합니다 .

2. 각 부의 명칭

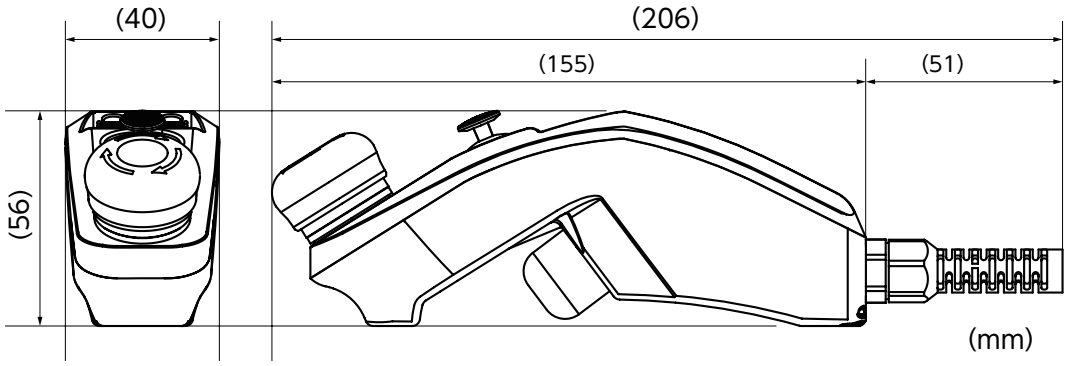
JOG 스틱 (옵션) 을 사용하여 매니퓰레이터의 각 축을 JOG 조작할 수 있습니다 . JOG 조작은 원점 위치로 이동하거나 교시 작업에 사용합니다 .



3. 설치

JOG 스틱을 사용하지 않는 경우에는 지정된 위치에 보관해 주시기 바랍니다 .

4. 외관도



5. 사양

항목		사양	비고
일반 사양	형식	ZJ1000	—
	외형 크기	H56 mm × D155 mm × W40 mm	본체만 케이블 제외
	무게	600 g 이하	—
	외관 재질	ABS 수지	색상 : 노랑색 , 검은색
	전원 전압	DC24 V ± 10%	—
	소비 전력	5 W 이하	—
	케이블 길이	5 m	—
환경 사양	사용 온도	0 ℃ – 40 ℃	—
	사용 습도	30 % – 85 %	—
	보관 온도	- 40 ℃ – 85 ℃	—
	보관 습도	10 % – 90 %	—
	냉각	자연 냉각	—

6. 기능

명칭	기능
비상정지 스위치	강하게 누르면 비상정지 상태가 됩니다 . 다시 서보 ON 하기 위해서는 시계 방향으로 돌려 비상정지를 해제하고 나서 Enable 스위치를 누릅니다 .
Enable 스위치	누르면서 서보 ON 을 합니다 . 손을 떼거나 더 깊게 누르면 , 서보 OFF 가 됩니다 .
슬라이드 스위치 L	조작 조인트를 변경합니다 . 스위치 위치 / 상 / 중 Joint 좌표계 / J1, J2 / J3, Rz 직교 좌표계 / X 축 , Y 축 / Z 축 , Rz 축
슬라이드 스위치 R	「JOG 스틱 동작」와 브라우저 화면 상의「조작패널 동작」을 변경합니다 . 스위치 위치 / 상 / 중 / 하 조작 / JOG 스틱 / 조작패널
푸시 스위치	누르면서 컨트롤러의 전원을 입력하면 ,「JOG 조작 모드」로 구동됩니다 .
컨트롤 레버	조이스틱 + 푸시 스위치입니다 . ・ 조이스틱 상하좌우로 기울여 매니퓰레이터를 조작합니다 . 슬라이드 스위치 L 로 조작하려는 조인트를 선택합니다 . ・ 푸시 스위치 【미사용】
LED1	녹색 LED 로 로봇의 상태를 표시합니다 . ・ JOG 스틱 전원 ON (녹색) / OFF(소등)
LED2	【미사용】
LED3	【미사용】
버저	버저음으로 상태를 알려줍니다 . ・ 교시 시에 울립니다 .
진동 모터	진동으로 상태를 알려줍니다 . ・ 매니퓰레이터의 말단이 이동 불가 지점에 가까워 지면 진동이 울립니다 .

버저음 패턴

버저음	의미
「삐」	다음의 상태에 대해 1 회 울립니다 . ・ 컨트롤러 구동 시 ・ 교시 동작의「Move To」에서「Direct Move」나「Hand Homing」동작의 시작 시

MEMO

B 하드웨어

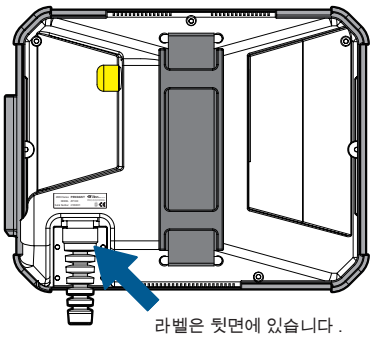
5 티칭 펜던트

1. 제품 라벨

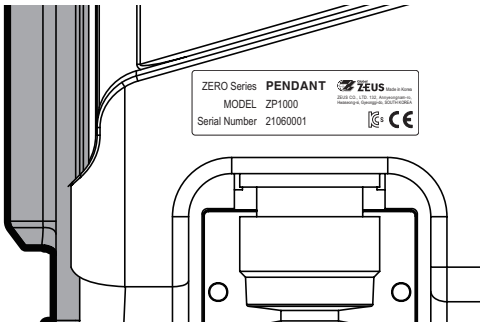
제품 라벨

ZERO Series **PENDANT**
MODEL ZP1000
Serial Number 21060001

Global ZEUS Made in Korea
ZEUS CO., LTD. 132, Annyeongnam-ro, Hwaseong-si, Gyeonggi-do, SOUTH KOREA



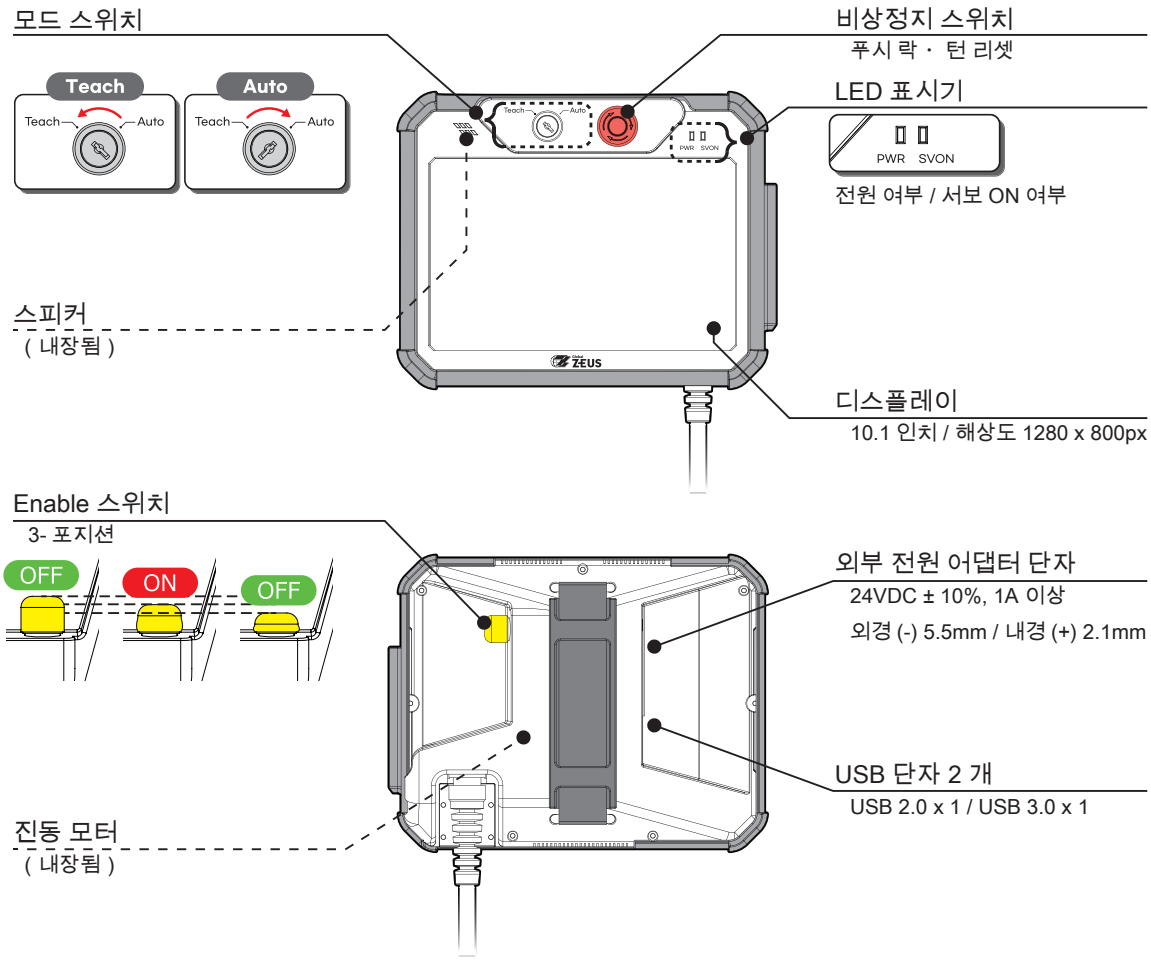
라벨은 뒷면에 있습니다 .



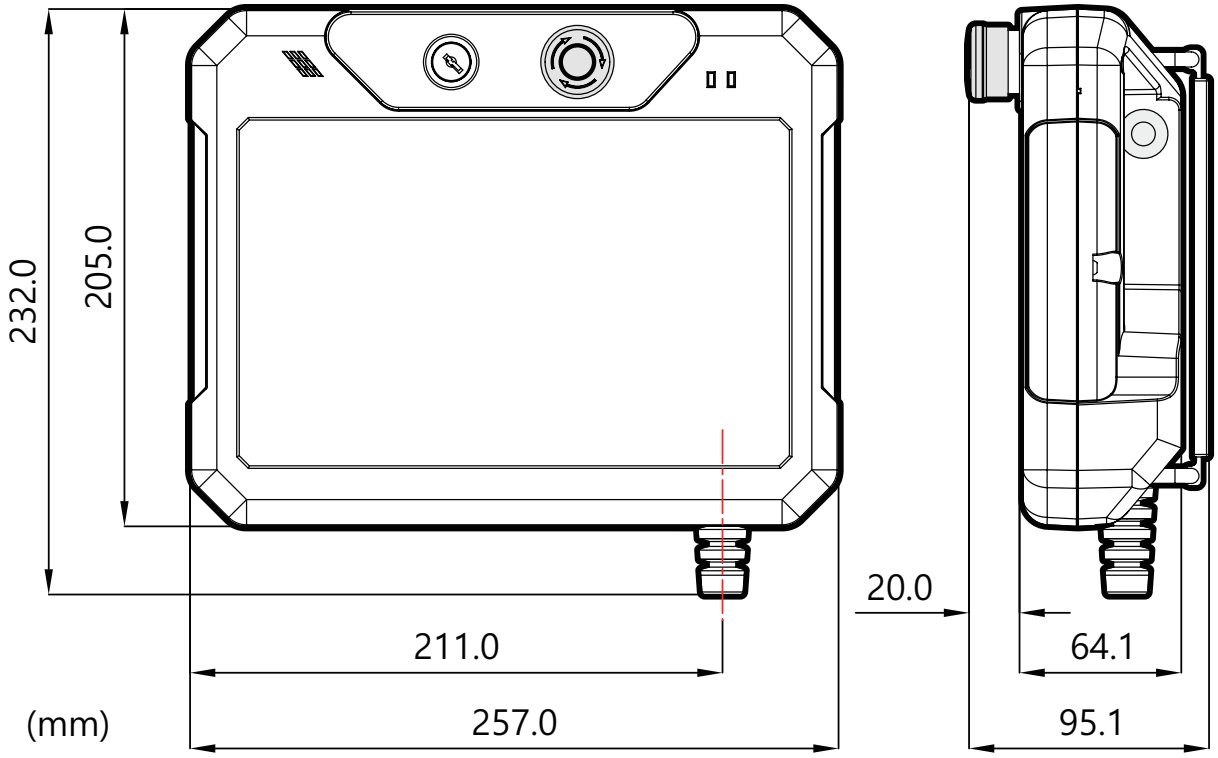
뒷면

위의 라벨은 일련 번호 "21060001" 의 경우의 예입니다 . 일련 번호는 제품마다 다릅니다 .
일련 번호 체계는 매니퓰레이터와 동일합니다 .

2. 각 부분의 명칭


모드 스위치
Teach
Auto
비상정지 스위치
푸시 락 · 턴 리셋
LED 표시기
PWR SVON
전원 여부 / 서보 ON 여부
스피커
(내장됨)
디스플레이
10.1 인치 / 해상도 1280 x 800px
Enable 스위치
3- 포지션
OFF
ON
OFF
외부 전원 어댑터 단자
24VDC ± 10%, 1A 이상
외경 (-) 5.5mm / 내경 (+) 2.1mm
USB 단자 2 개
USB 2.0 x 1 / USB 3.0 x 1
진동 모터
(내장됨)

3. 외관도



(고무 범퍼 , 케이블을 제외한 크기)

4. 사양

항목		사양	비고
일반 사양	형식	ZP1000	—
	외형 크기	H95.1 mm × D257 mm × W205 mm	본체만 케이블 제외
	무게	1.2 kg 이하	—
	외관 재질	PC + ABS 수지	색상 : 검은색
	전원 전압	DC24 V ± 10%	—
	소비 전력	12 W 이하	—
	케이블 길이	3 m	—
환경 사양	사용 온도	0 ℃ – 40 ℃	—
	사용 습도	30 % – 85 %	—
	보관 온도	- 40 ℃ – 85 ℃	—
	보관 습도	10 % – 90 %	—
	냉각	자연 냉각	—

5. 기능

명칭	기능
비상정지 스위치	강하게 누르면 비상정지 상태가 됩니다 . 다시 서보 ON 하기 위해서는 시계 방향으로 돌려 비상정지를 해제하고 나서 Enable 스위치를 누릅니다 .
Enable 스위치	누르면서 서보 ON 을 합니다 . 손을 떼거나 더 깊게 누르면 , 서보 OFF 가 됩니다 .
모드 스위치	동작 모드를 교시 모드와 원격 모드 (자동 운전 모드)를 전환합니다 스위치 위치 / 좌 / 우 모드 / 교시모드 / 원격 모드
외부 전원 어댑터 단자	전원 어댑터를 연결하면 티칭 펜던트의 전원이 켜집니다 . 일반적인 상황에서는 사용되지 않습니다 . 24VDC ± 10%, 1A 이상 외경 (-) 5.5mm / 내경 (+) 2.1mm ※ 전원이 인가된 상태에서 어댑터를 연결하지 마십시오 .
USB 단자	티칭 펜던트에 저장된 교시 표인트 , 오류 로그 등의 데이터를 가져옵니다 . 소프트웨어 업데이트 파일 등을 업로드합니다 . USB 2.0 x 1 / USB 3.0 x 1
LED 표시기	LED 로 티칭 펜던트와 로봇의 상태를 표시합니다 . ・ PWR: 티칭 펜던트 전원 ON (녹색) / OFF(소등) ・ SVON: 로봇 서보 전원 ON (녹색) / OFF(소등)
LCD	티칭 펜던트의 교시 화면을 나타냅니다 . 티칭 펜던트와 로봇의 상태를 확인할 수 있습니다 .
스피커	소리로 상태를 알려줍니다 . ・ 교시 시에 울립니다 .
진동 모터	진동으로 상태를 알려줍니다 . ・ 매니퓰레이터의 말단이 이동 불가 지점에 가까워 지면 진동이 울립니다 .

스피커 경고음 패턴

경고음	의미
「삐」	다음의 상태에 대해 1 회 울립니다 . ・ 특이점 구간 , 속도 리미트 , Joint angle 리미트 구간 접근 시

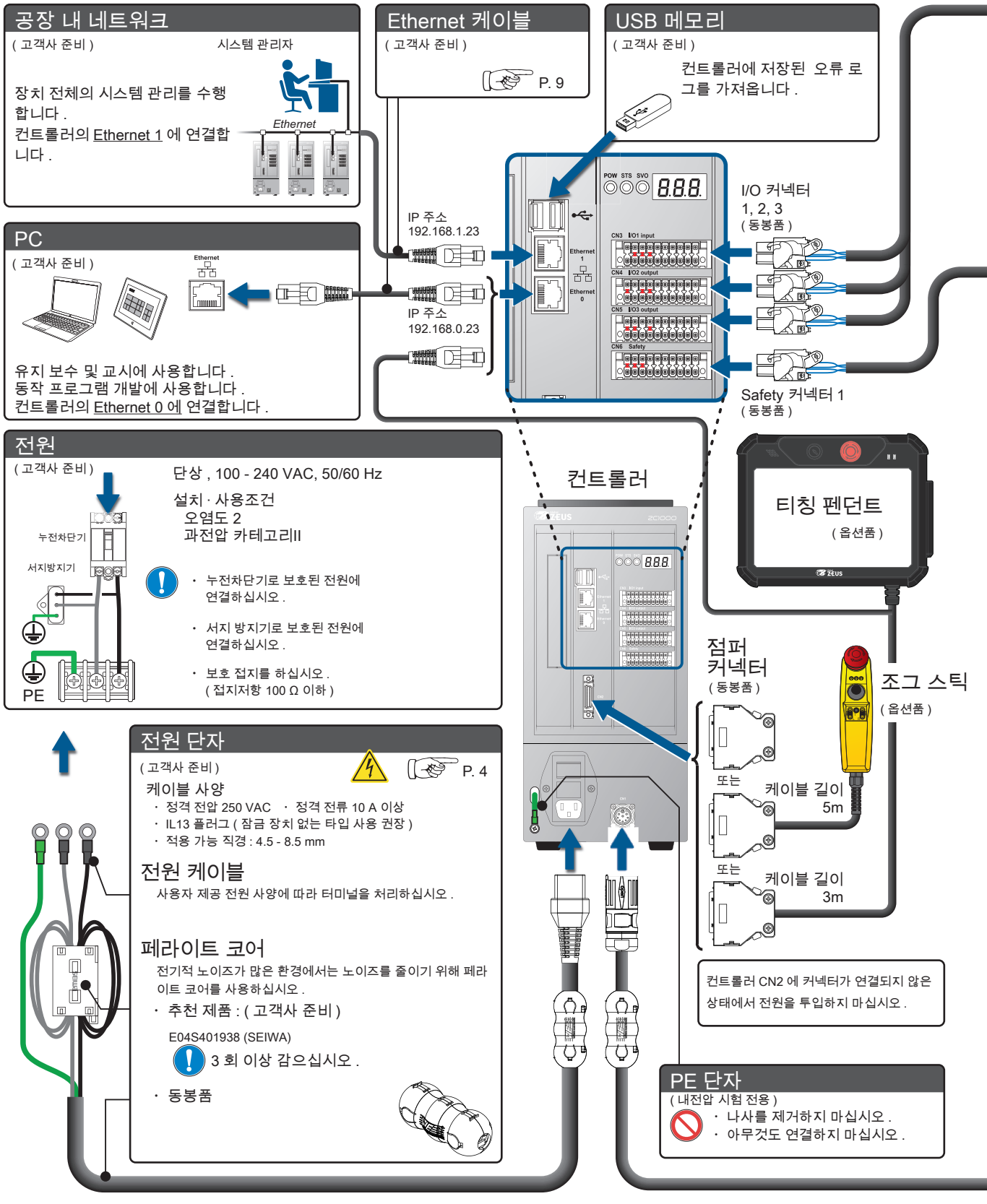
B 하드웨어

6 배선과 전원

1. 배선

1. 전체배선도

다음과 같이 확실하게 배선하십시오 .

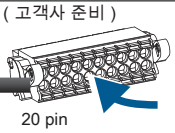


1 배선

I/O 제어 장치

(고객사 준비)

P. 7

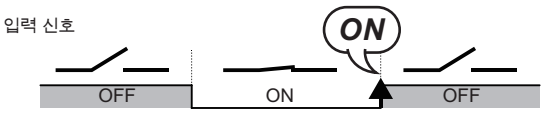


20 pin

I/O 커넥터 1, 2, 3 에 연결합니다 .
CN3 : I/O1 입력
CN4 : I/O2 출력
CN5 : I/O3 출력

신호의 할당 및 회로도를 참고하여 배선하십시오 .

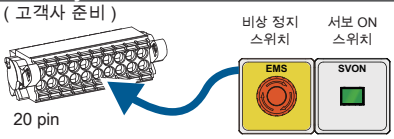
입력 신호는 Low 에서 High 의 입력 엣지를 감지합니다 .
출력이 ON 되면 해당 출력 포트가 High 가 됩니다 .



Safety 회로

(고객사 준비)

P. 8



20 pin

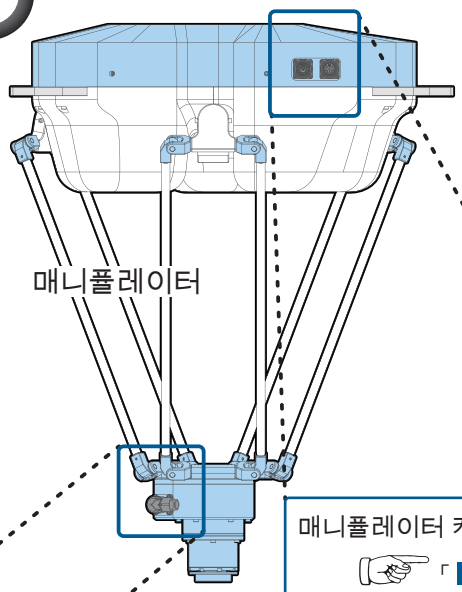
Safety 커넥터에 연결합니다 .

비상 정지 스위치와 서보 ON 스위치 등을 연결하십시오 .
제대로 연결하지 않으면 서보 ON 이 되지 않습니다 .

I/O 케이블

(고객사 준비)

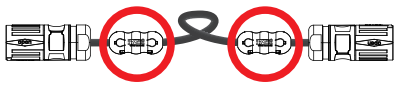
- 배선은 고전압선이나 동력선에서 멀리 떨어뜨리고 , 실드선을 사용하십시오 .
- AWG16-24
- 노이즈를 고려하여 15m 이내에서 사용하십시오 .



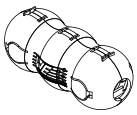
매니퓰레이터 케이블

부속
케이블 길이 3 m

P. 5



페라이트 코어 (동봉품)

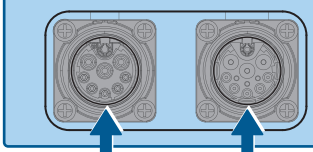


동봉된 페라이트 코어를 제거하지 마십시오 .

C. CODE 는 각 로봇마다 다릅니다 . 컨트롤러를 C. CODE 가 일치하는 매니퓰레이터와 연결하십시오 .
매니퓰레이터와 컨트롤러의 C. CODE 라벨을 확인하십시오 . C. CODE 가 일치하는 쌍만 연결할 수 있습니다 .

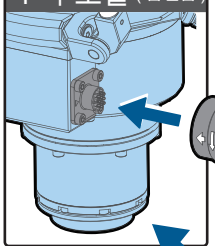
매니퓰레이터 케이블의 연결

「 2 매니퓰레이터 」



P. 6

4 축 모듈 (옵션품)



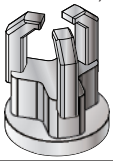
말단부 구동 케이블

부속

말단 장치 설치

말단 장치

(고객사 준비)



Arm I/O 커넥터 (옵션 사양) 에 배선을 연결합니다 .

Arm I/O 에 연결하는 장치의 소비 전류는 100 mA 이하로 사용하십시오 .

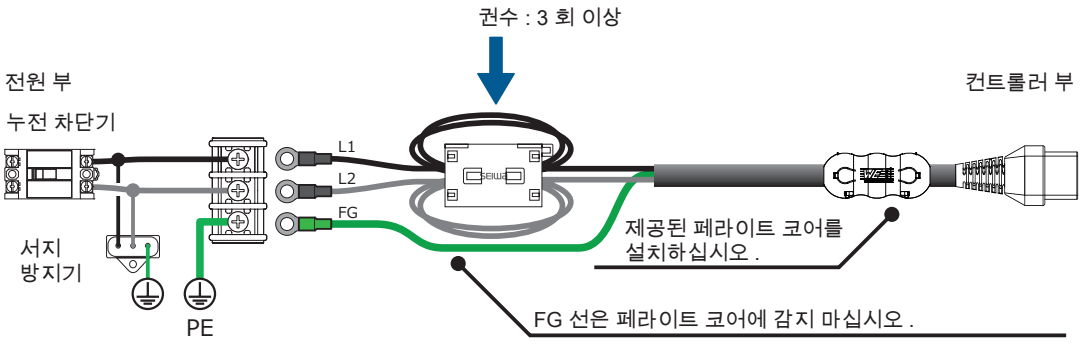
2. 전원 케이블

컨트롤러에 전원을 공급합니다 .

케이블은 권장 사양품을 사용하십시오 .
· 정격 전압 250 VAC
· 정격 전류 10 A 이상
· IL13 플러그 (잠금 장치 없음)
· 외경 4.5 - 8.5 mm

전원에 연결하는 방법

전원 노이즈가 많은 환경에서 사용하는 경우 , 전원에 연결하는 케이블 말단에는 아래 그림과 같이 페라이트 코어를 감고 절연 피복 원형 단자를 시공하십시오 . 시공 원형 단자는 사용하는 전원 설비에 적합한 크기나 모양으로 선정하십시오 .



페라이트 코어 추천품
E04SR401938 (SEIWA)

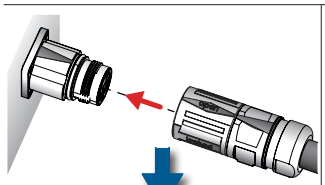
위 험

사용 전압 , 전류에 적합한 사양의 전원 케이블을 사용해야 합니다 .
전원의 배선 공사는 반드시 전문 자격 소지자가 수행해야 합니다 .
화재나 감전의 위험이 있습니다 .

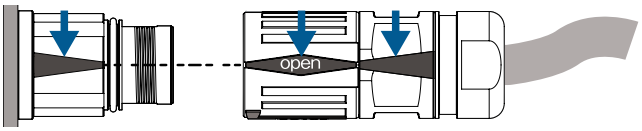
3. 매니퓰레이터 케이블 (부속품)

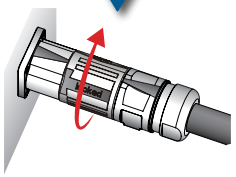
컨트롤러와 매니퓰레이터를 연결하는 케이블입니다 .
전원을 공급하고 통신을 수행합니다 .

컨트롤러와 매니퓰레이터의 커넥터 연결 방법

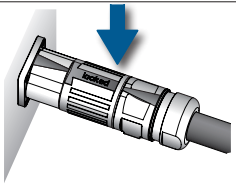


맞춤 기호를 'open' 에 맞게 커넥터를 완전히 삽입합니다 .

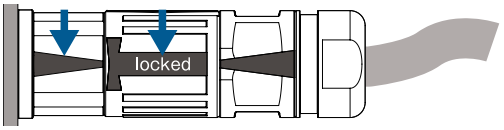




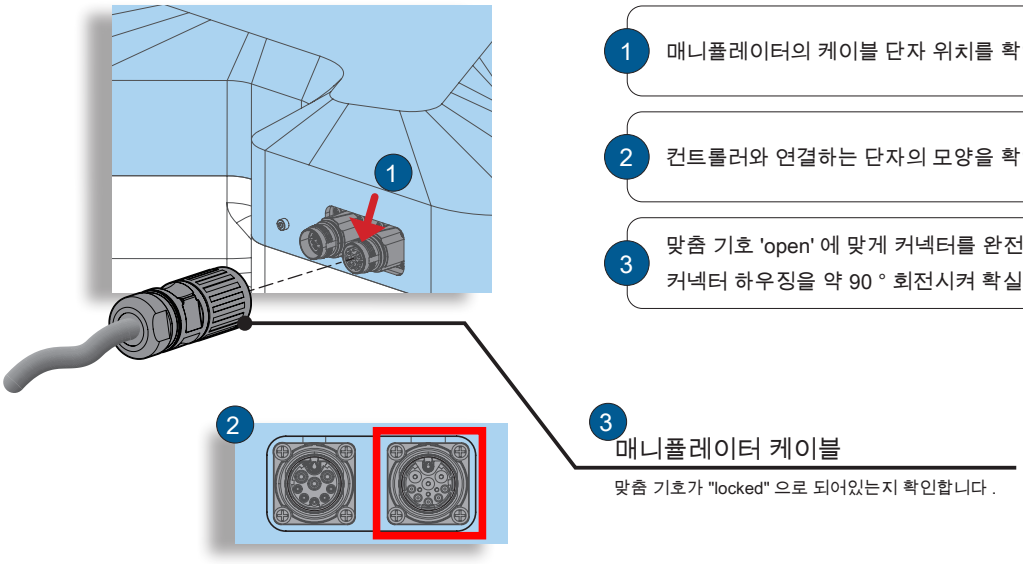
커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .



맞춤 기호가 "locked" 으로 되어 있는지 확인합니다 .



매니퓰레이터에 연결하는 방법



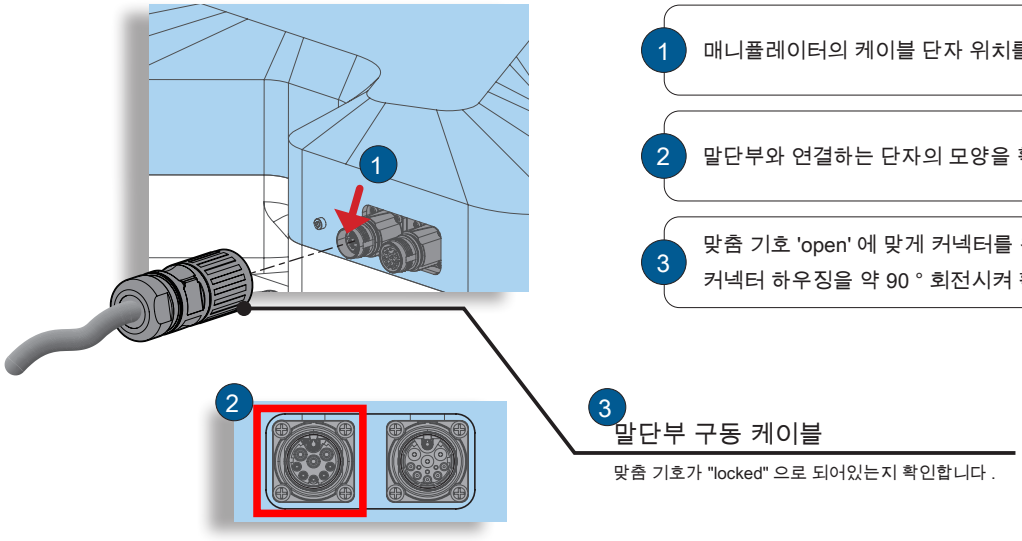
1. 매니퓰레이터의 케이블 단자 위치를 확인합니다 .
2. 컨트롤러와 연결하는 단자의 모양을 확인합니다 .
3. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .

컨트롤러와 매니퓰레이터는 올바른 조합으로 연결하시기 바랍니다 .

컨트롤러와 매니퓰레이터의 C.CODE 라벨을 확인하고 C.CODE 가 일치하도록 연결합니다 .

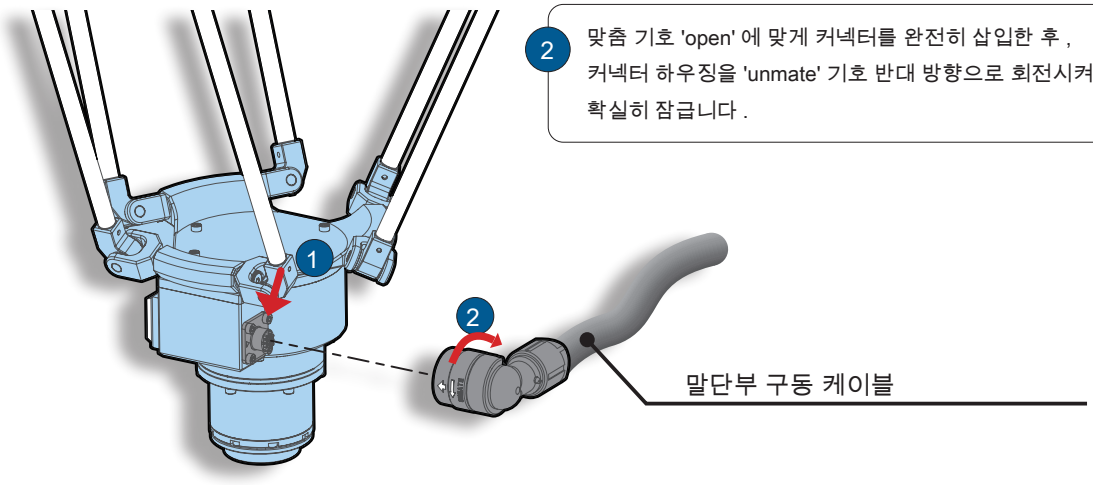
4. 말단부 구동 케이블 (부속품)

매니퓰레이터에 연결하는 방법



1. 매니퓰레이터의 케이블 단자 위치를 확인합니다 .
2. 말단부와 연결하는 단자의 모양을 확인합니다 .
3. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 약 90 ° 회전시켜 확실히 잠급니다 .

말단부에 연결하는 방법

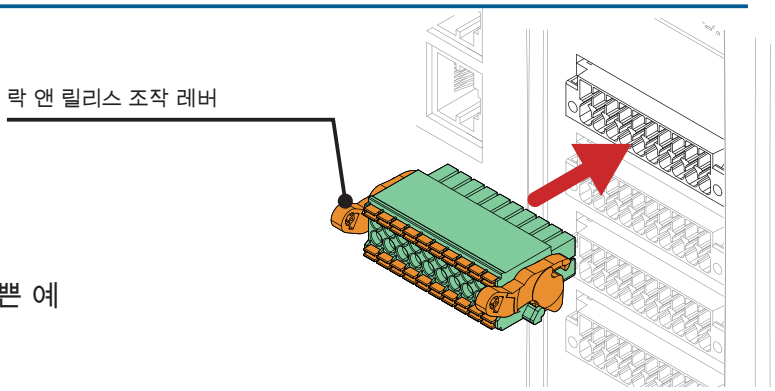


1. 말단부의 케이블 단자 위치를 확인합니다 .
2. 맞춤 기호 'open' 에 맞게 커넥터를 완전히 삽입한 후 , 커넥터 하우징을 'unmate' 기호 반대 방향으로 회전시켜 확실히 잠급니다 .

4. I/O 커넥터의 연결

커넥터는 락이 확실히 맞물릴 때까지 단단히 컨트롤러에 삽입하십시오 . 커넥터가 제대로 연결되면 좌우 2 개의 락 앤 릴리스 조작 레버는 자동으로 잠깁니다 .

컨트롤러에 연결하는 방법



커넥터 연결의 좋은 예와 나쁜 예

- 배선은 고전압선이나 모터 동력선에서 멀리하고 , 실드선을 사용하십시오 .
- AWG16-24 을 사용하십시오 .
- 노이즈를 고려하여 15 m 이내에서 사용하십시오 .

서보 ON 에 대해

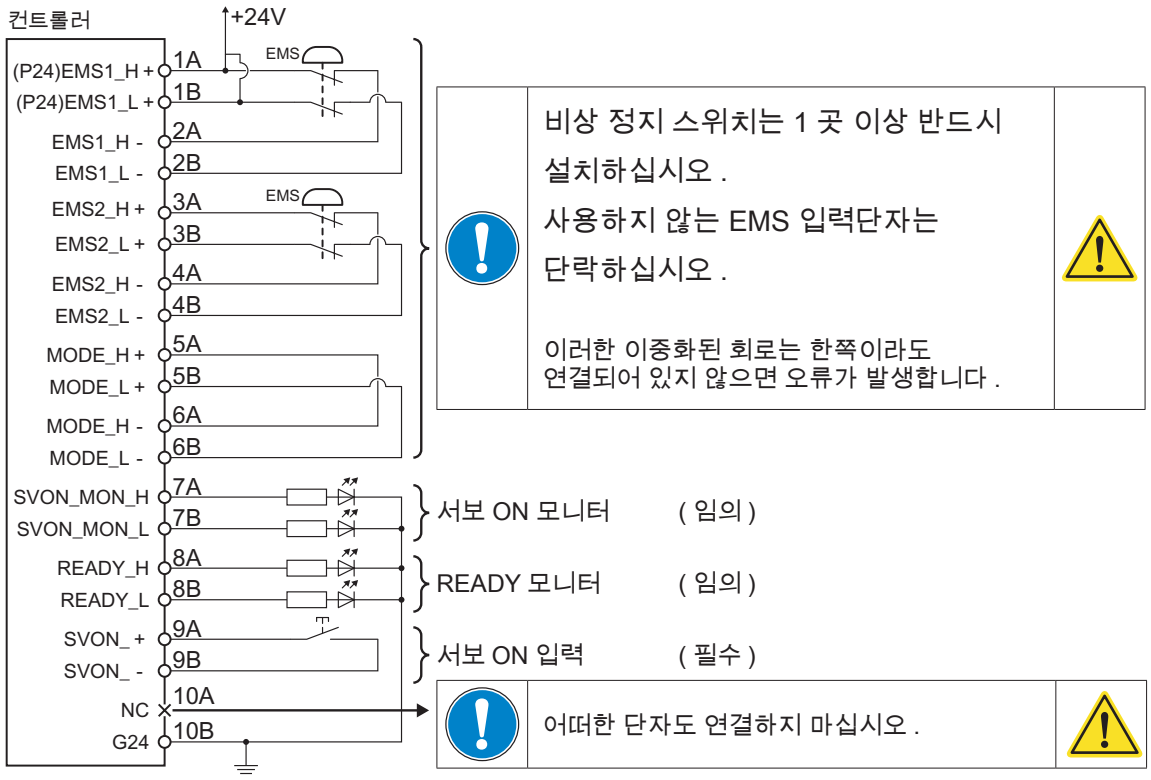
상승 엣지에서 서보 ON 합니다 .

기계식 순시동작 스위치 (a 접점) 를 사용하십시오 .

5. Safety 커넥터의 배선

주 의

각 스위치는 기계식 접점 스위치를 사용하십시오 .



비상 정지 스위치는 1 곳 이상 반드시 설치하십시오 .
사용하지 않는 EMS 입력단자는 단락하십시오 .

이러한 이중화된 회로는 한쪽이라도 연결되어 있지 않으면 오류가 발생합니다 .

어떠한 단자도 연결하지 마십시오 .

주 의

다음을 참조하여 케이블의 피복을 가공하십시오 .

피복 처리는 와이어 스트리퍼를 사용하십시오 . 적절하게 가공하지 않은 배선을 사용하면 접촉 불량이나 예기치 않은 동작이 발생할 수 있습니다 .

포인트 !

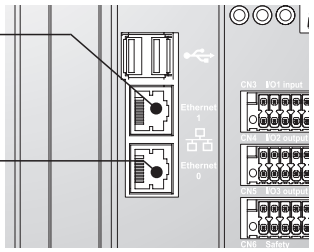
I/O 커넥터와 Safety 커넥터의 배선 - 피복 벗기는 방법

6. Ethernet 케이블

유지 보수용 PC, Tablet 과의 연결은 Ethernet 0 (아래) 를 사용하십시오 .
공장 내 네트워크 모니터링 용 PC 와의 연결은 Ethernet 1 (위) 를 사용하십시오 .
케이블은 직선 / 곡선 모두 사용할 수 있습니다 .
Ethernet CAT5 이상의 케이블을 사용하십시오 .

Ethernet 1 : 공장 내 네트워크 모니터링 용 PC 연결
IP 주소 : 192.168.1.23
서브넷 마스크 : 255.255.255.0

Ethernet 0 : 유지 보수 , 교시용 PC 연결 (전용)
IP 주소 : 192.168.0.23
서브넷 마스크 : 255.255.255.0



포인트 !

배선

케이블이 기름이나 물에 접촉된 상태에서 사용하지 마십시오 .

케이블의 접합 굴곡이나 자체 무게에 의한 스트레스가 가해지지 않도록 하십시오 .
케이블의 굴곡 반경은 가능한 한 크게 확보하십시오 .

케이블이 움직이도록 배선을 할 경우에는 반드시 동적 케이블을 사용하십시오 . 케이블은 케이블 베어 (체인) 에 넣고 굴곡에 의한 스트레스를 최소화하십시오 .

케이블 베어 (체인) 에 대전류 및 고전압선과 신호선을 배치한 경우에는 노이즈에 의한 오동작을 방지하기 위해 최대한 멀리 하십시오 .

7. JOG 스틱과 점퍼 커넥터

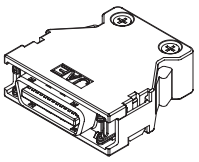
로봇의 운전 모드는 컨트롤러의 CN 2 커넥터에 연결하는 것으로 전환합니다 .

점퍼 커넥터를 연결하면 ...> 원격 모드 (자동 운전 모드) 입니다 .
JOG 스틱을 연결하면 ...> 교시 모드입니다 .

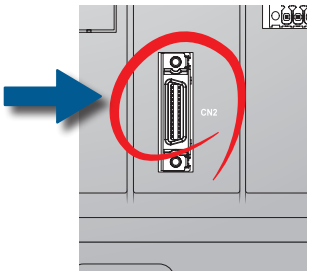
원격 모드 (자동 운전 모드)

로봇의 자동 운전을 하는 모드입니다 .

점퍼 커넥터를 연결하십시오 .



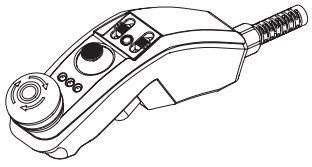
점퍼 커넥터
(부속품)



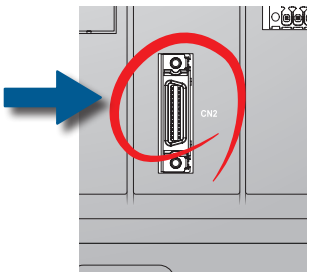
교시 모드

JOG 조작이나 교시를 하는 모드입니다 .

JOG 스틱을 연결하십시오 .



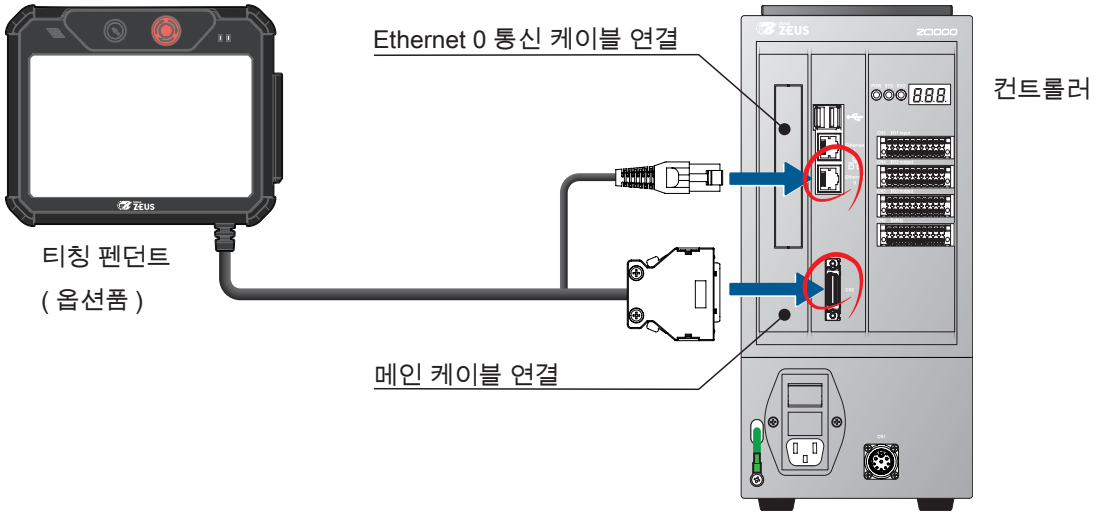
JOG 스틱
(옵션품)



컨트롤러의 CN2 는 JOG 스틱을 사용할 때를 제외하고는 항상 점퍼 커넥터 (부속품) 를 연결해야 합니다 .

8. 티칭 펜던트 케이블

티칭 펜던트 케이블은 두 가닥의 케이블로 구성됩니다 .
통신 케이블은 컨트롤러의 Ethernet 0 (아래) 에 연결하십시오 .
메인 케이블은 컨트롤러의 CN2 에 연결하십시오 .

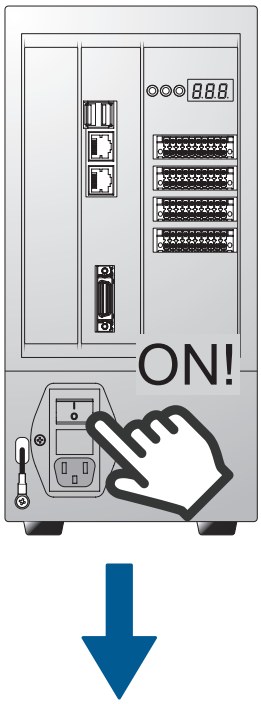


	컨트롤러에 전원이 투입된 상태에서 케이블을 연결하거나 분리하지 마십시오 . 티칭 펜던트를 컨트롤러의 CN2 와 이더넷 포트에 확실하게 고정하십시오 . 또한 케이블을 과도하게 잡아당기거나 구부러뜨리지 않도록 주의하십시오 . 교시가 끝나면 티칭 펜던트를 컨트롤러에서 분리하여 , 낙하 등에 의한 오동작이나 손상으로 부터 보호하십시오 . 교시할 때 이외에는 컨트롤러의 CN2 에 점퍼 커넥터를 연결하십시오 .	

2. 전원

⚠ 주 의		
	컨트롤러의 전원을 투입하기 전에 모든 배선이 완료되었는지 확인하십시오 .	
	모든 커넥터는 전원을 켠 상태로 연결하거나 제거하지 마십시오 .	

1. 전원투입



전원을 투입하면 컨트롤러의 7 세그먼트 표시기에 상태를 표시합니다 .

7 세그먼트 표시	
- - -	컨트롤러의 시작
↓ (약 10 초)	
ini	컨트롤러의 초기화
↓ (약 10 초) 초기화가 완료되면 이 중 하나가 표시됩니다 .	
inc	ABS 원점 소실 처음 시작할 때 또는 ABS 원점 소실시에 표시됩니다 . ABS 원점 복귀를 하십시오 (*)
rdy	준비 완료 (= 대기 상태) ABS 원점 복귀했습니다 . 로봇은 대기 상태입니다 .
c88 E88	오류 오류 코드를 확인하고 해결하십시오 .

2. 원점복귀 , 교시

사용설명서 「 C 교시 」 를 참조하십시오 .

*) 처음 시작할 때 매니퓰레이터의 ABS 정보가 소실되어 있습니다 .
ABS 원점 정보를 잃어버린 상태로 출하하고 있습니다 .

MEMO

C

교시 (Teaching)

MEMO

C 교시 (Teaching)

1 JOG 스틱 조작

1. JOG 조작 모드

1. JOG 조작 모드란

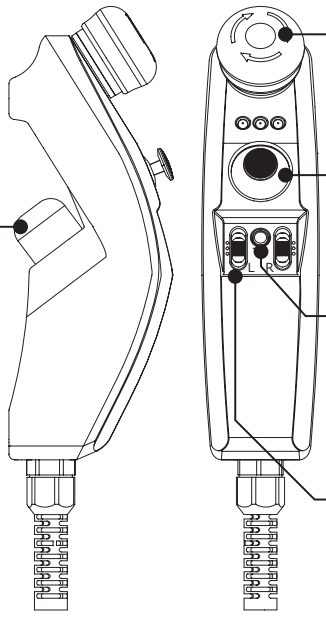
JOG 스틱을 조작하여 매니퓰레이터를 동작시키는 모드입니다 .

PC 와 접속하지 않고 로봇을 조작할 수 있습니다 .

- 작업자는 로봇으로부터 떨어진 위치에서 안전하게 로봇을 조작할 수 있습니다 .
- ABS 소실 중에도 로봇을 조작할 수 있습니다 .
- 로봇은 조인트 좌표계로 동작합니다 .
- 매니퓰레이터를 쉽게 원점 자세로 바꿀 수 있습니다 .
- 조작은 각 축 별로 가능하며 , 복수의 축을 조작할 수 없습니다 .

항목	사양
동작 속도	사양 최고 속도의 5% X : 90 mm/s, Y: 90 mm/s, Z : 90 mm/s, R_z : 50 deg/s
동작량	0.25deg 씩 동작 (컨트롤 레버를 누르고 있으면 , 5deg 씩 바뀝니다 .)

JOG 조작에서 사용하는 각 부분의 명칭과 기능



Enable 스위치

누르면 서보 ON 됩니다 .

떼어 내거나 , 더 깊게 누르면 서보 OFF 가 됩니다 .

비상정지 스위치

스위치를 누르면 비상정지 상태가 됩니다 .

다시 서보 ON 하기 위해서는 스위치를 돌려 비상정지 상태를 해제하고 Enable 스위치를 누릅니다 .

컨트롤 레버

조이스틱 + 푸쉬 스위치입니다 .

상하좌우로 움직여 매니퓰레이터를 움직입니다 .

푸쉬 스위치

JOG 조작 모드로 기동합니다 .

누른 상태로 전원을 연결하면 JOG 조작 모드로 기동합니다 . 컨트롤러의 7 세그먼트 표시가 JoG 로 될 때까지 계속 눌러 주시기 바랍니다 .

슬라이드 스위치 L

조작하는 조인트를 바꿉니다 .

상 : J1, J2

중 : J3, R_z

하 : -

1 JOG 조작 모드

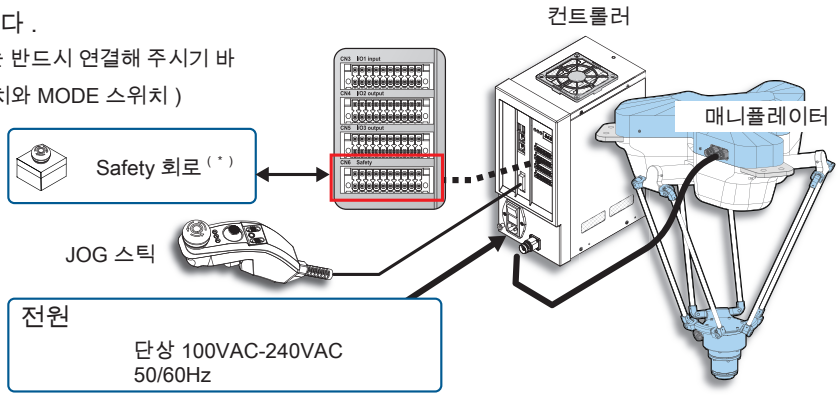
2. 기동과 종료

기동

순서 1 접속

그림과 같이 배선합니다 .

Safety 커넥터의 회로 (*) 는 반드시 연결해 주시기 바랍니다 . (비상정지 스위치와 MODE 스위치)

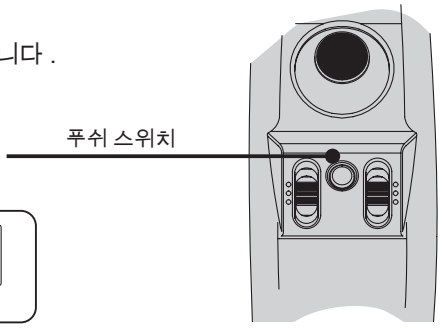


*) Safety 회로는「 B 하드웨어 5 배선과 전원 」의「 5 . Safety 커넥터의 배선」 참조하여 주시기 바랍니다 .

순서 2 기동

JOG 스틱의 푸쉬 스위치를 누르면서 컨트롤러의 전원을 켭니다 .

컨트롤러의 7 세그먼트 표시가 JoG 가 될 때까지 계속 눌러 주시기 바랍니다 .



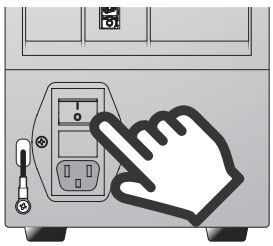
7 세그먼트 표시 ini → JoG

ini : 기동 중

JoG : 기동 완료

종료

컨트롤러의 전원을 차단합니다 .



포인트 !

에러가 발생했을 경우

c04

JOG 조작 에러

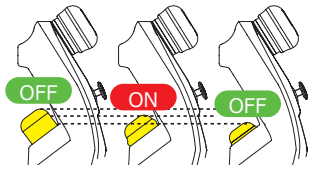
컨트롤러를 껐다가 다시 켜 주시기 바랍니다 .

1 JOG 조작 모드

3. 조작

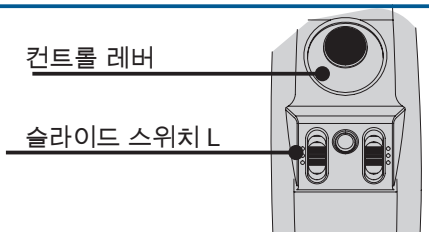
순서 1 서보 ON

Enable 스위치를 누릅니다 .
Enable 스위치에서 손을 떼거나 더 깊게 누르면 서보를 OFF 합니다 .

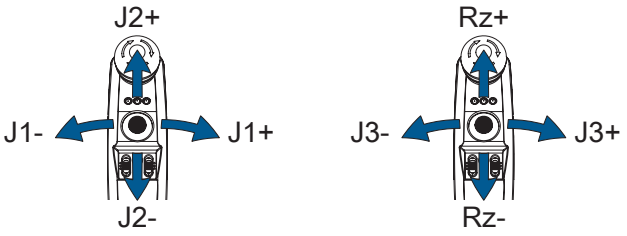


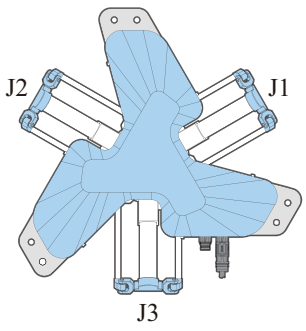
순서 2 조작

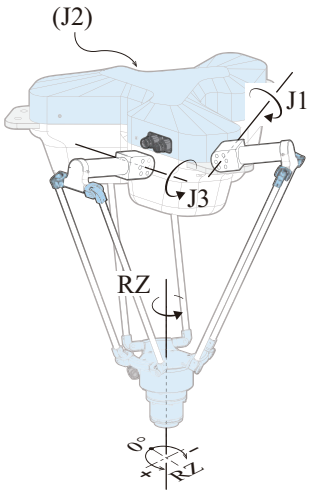
슬라이드 스위치 L 을 상 / 중 / 하로 바꾼 뒤 , 움직이고 싶은 조인트를 선택합니다 .
컨트롤 레버를 기울여 로봇을 움직입니다 .



	J1, J2 축	J3, Rz 축
슬라이드 스위치 L	상	중
컨트롤러의 7 SEG 표시	J12	J34





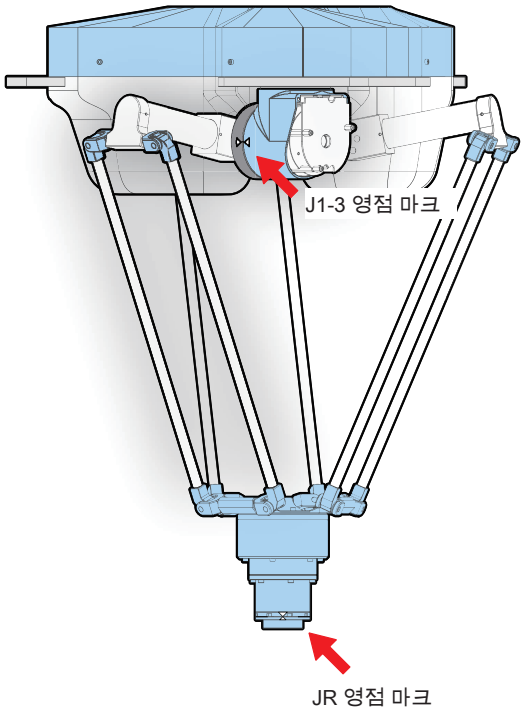


컨트롤러의 7 세그먼트 표시기에서 동작하는 조인트를 확인한 후에 컨트롤 레버를 조작하여 주시기 바랍니다 .

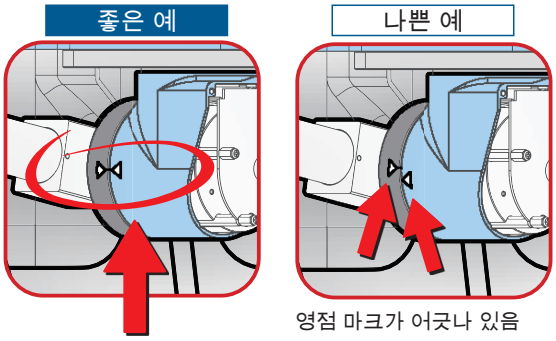
순서 3 원점 자세 맞추기

JOG 스틱을 조작하여 모든 조인트들을 영점 마크에 확실하게 맞추어 원점 자세로 만듭니다 .

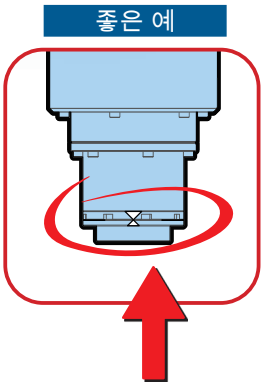
(ZRC-0306R 의 원점 자세 예시)



조인트 J1~3



조인트 R



*) 원점 자세는 기종마다 다릅니다 . 영점 마크의 위치를 잘 확인해 주시기 바랍니다 .
본 문서에서는 영점 마크를 강조해서 그려 놓았습니다 . 실제 위치 기준으로 진행하여 주시기 바랍니다 .

영점 마크는 1mm 이내에 맞춰 주시기 바랍니다 .

포인트 !

JOG 조작 도중에 JOG 스틱을 컨트롤러에서 떼어낸 경우

로봇의 동작은 정지합니다 .

재개하려면 JOG 스틱을 컨트롤러에 다시 연결합니다 .

(컨트롤러의 재부팅은 필요하지 않습니다 .)

MEMO

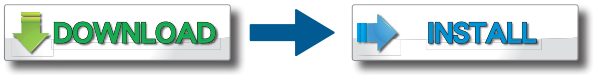
C 교시 (Teaching)

2 PC 접속

1. PC 와 컨트롤러의 연결

1. 소프트웨어 준비

아래 2 개의 소프트웨어를 준비합니다 .



「FFFTP」: FTP 클라이언트 소프트웨어

FTP (File Transfer Protocol) 를 이용하여 PC 와 컨트롤러 간의 파일 전송을 실시합니다 .

URL https://osdn.net/projects/ffftp/

사용하는 컴퓨터에 따라 32bit 혹은 64bit 버전을 선택합니다 .
설치 시의 설정은 초기 설정을 그대로 사용합니다 .

(FFFTP 는 소타 준 , FFFTP Project 의 저작물입니다 .)

「Tera Term」: 터미널 소프트웨어

원격 접속 클라이언트입니다 . Telnet 접속을 통해 로봇 구동 프로그램을 실행하는 등 , 컨트롤러를 조작하는 데에 사용합니다 .

URL https://osdn.net/projects/ttssh2/

설치 시의 설정은 초기 설정을 그대로 사용합니다 .

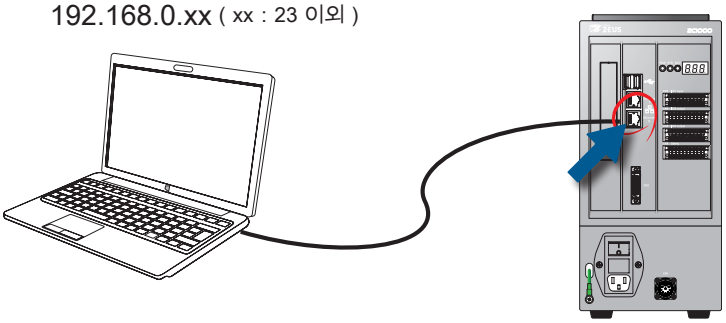
(Tera Term 은 테라니시 타카시 및 Tera Term Project 의 저작물입니다 .)

1 PC 와 컨트롤러의 연결

2. IP 주소 설정

컨트롤러와 접속하는 PC 의 네트워크를 설정합니다 .

PC 의 IP 주소
192.168.0.xx (xx : 23 이외)

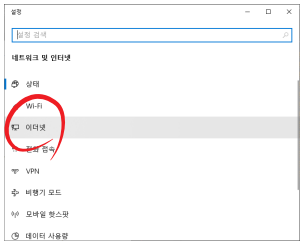


컨트롤러

아래 (Ethernet 0) 에 연결합니다 .
컨트롤러의 IP 주소는
192.168.0.23

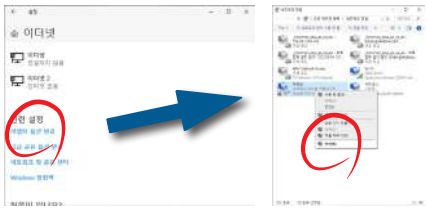
순서 1

제어판에 있는 " 네트워크와 인터넷 " 의 " 이더넷 " 을 클릭합니다 .



순서 2

「이더넷」의 ' 어댑터 옵션 변경 ' 을 클릭하고 이더넷 아이콘을 우클릭한 다음 속성을 클릭합니다 .



순서 3

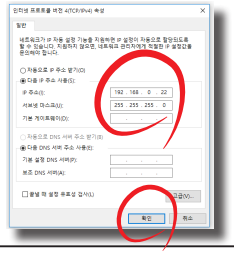
「인터넷 프로토콜 버전 4 (TCP/IPv4)」의 속성을 클릭합니다 .

순서 4

IP 주소와 서브넷 마스크를 다음과 같이 설정합니다 .

IP 주소 (I)	192. 168. 0. XX
서브넷 마스크 (U)	255. 255. 255. 0

(XX 는 23 외의 숫자로 설정해 주시기 바랍니다 .)



(상기 예는 Windows 10 입니다 .)

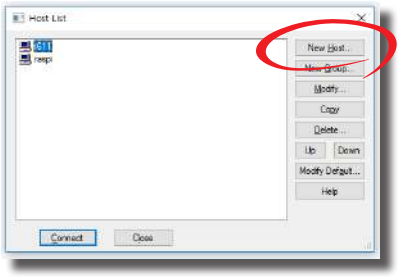
1 PC 와 컨트롤러의 연결

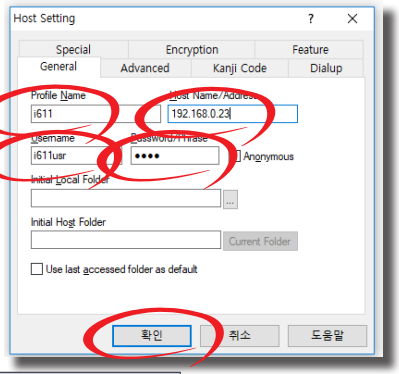
3. 접속 설정

아래 2 개의 소프트웨어에 대해 컨트롤러에 접속하기 위한 설정을 합니다 .

「FFFTP」를 실행합니다 .

New Host... 를 클릭하여 호스트 설정을 합니다 .





호스트 설정

Profile Name	i611 (임의)
Host Name/Address	192.168.0.23
Username	i611usr
Password/Phrase	i611

Connect 를 클릭합니다 .

「Tera Term」을 실행합니다 .

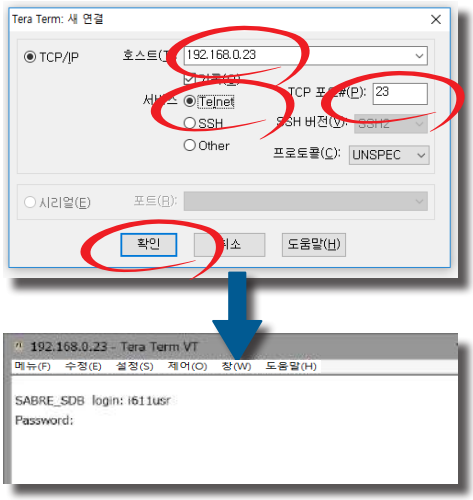
호스트 설정

호스트 (T)	192.168.0.23
서비스	Telnet
TCP 포트 # (P)	23

(컨트롤러의 전원이 켜져 있어야 함)

컨트롤러 인증

login	i611usr
Password	i611



(상기 예는 Windows 10 입니다 .)

C 교시 (Teaching)

3 ABS 원점 복귀

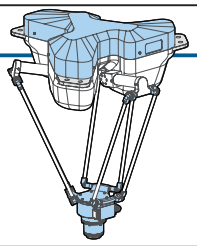
1. 주의 사항

- 로봇에 전원이 처음 연결되는 경우에는 반드시 ABS 원점 복귀를 시행해 주시기 바랍니다 .
- 모든 조인트를 JOG 조작을 통해 영점 마크 (시범마크) 에 확실하게 맞추어 주시기 바랍니다 .
- 영점 마크는 1mm 이내에 맞추어 주시기 바랍니다 .
- ABS 원점 복귀 실시 도중에 서보 ON 을 해야 합니다 . 미리 JOG 스틱과 서보 ON 을 할 수 있는 스위치를 연결해 주시기 바랍니다 .
- 티칭 펜던트는 별도의 JOG 동작 모드를 제공하지 않습니다 .

- 원점 복귀는 개봉 시에 1 회만 실시하는 것입니다 . 일상적으로 실시할 필요는 없습니다 .

2. 순서

순서 1 매니퓰레이터의 원점 자세 맞추기



JOG 스틱을 조작하여 모든 축을 영점 마크에 정확히 맞추어 원점 자세로 만듭니다 .

1 JOG 스틱에 의한 조작

순서 2 로봇 프로그램「 enc_reset.py 」실행

Telnet 으로 컨트롤러에 접속합니다 .

ABS 원점 복귀 프로그램을 실행합니다 .

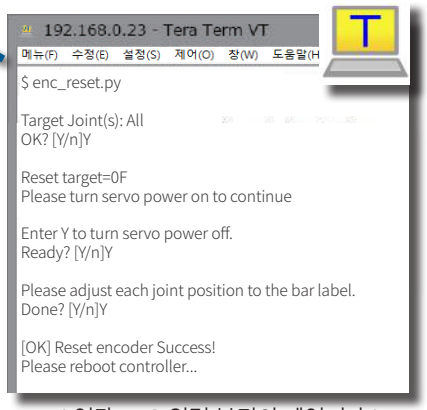
모든 조인트들을 일괄적으로 실행하는 경우

$enc_reset.py

조인트를 지정하는 경우

뒤에 공백을 준 후 , 조인트 번호를 추가합니다 .

$enc_reset.py 12 ← 조인트 1,2 의 경우



(일괄 ABS 원점 복귀의 예입니다)

확인 메세지가 표시됩니다 .

Target Joint(s) : All ← 모든 조인트들을 일괄적으로 실행하는 경우

OK? [Y/n] Y ←「 Y 」(대문자) 를 입력합니다 .

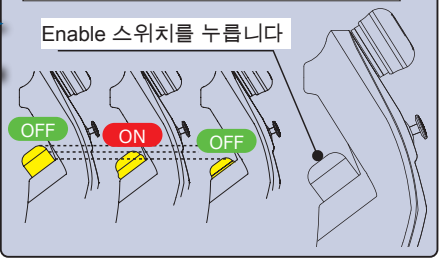
메세지가 표시됩니다 .

Reset target = 0F ← 모든 조인트들을 일괄적으로 실행하는 경우

Please turn servo power on to continue

서보 ON 해 주시기 바랍니다

Enable 스위치를 누릅니다



서보 ON 이 완료되면 확인 메세지가 표시됩니다 .

Enter Y to turn servo power off

Ready? [Y/n] Y ←「 Y 」(대문자) 를 입력합니다 .

메세지가 표시됩니다 .

Please adjust each joint position to the bar label.

Done? [Y/n] Y ←「 Y 」(대문자) 를 입력합니다 .

메세지가 표시됩니다 .

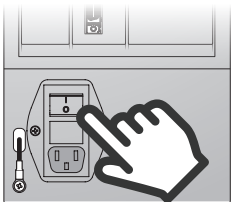
[OK] Reset encoder success!

Please reboot controller...

컨트롤러 재부팅을 실시합니다 .

순서 3 컨트롤러 재부팅

ABS 원점 복귀가 완료되면 컨트롤러의 7 세그먼트 표시가「rdy」가 됩니다 .



3. 확인

ABS 원점 복귀를 실시한 후에는 , 반드시 ABS 원점 복귀가 제대로 되어 있는지 아래의 방법을 통해 확인해 주시기 바랍니다 .

순서 1 확인 프로그램「 confirm_home.py 」의 실행

Telnet 으로 컨트롤러에 접속합니다 .

확인 프로그램을 실행합니다 .
$confirm_home.py

JOG 스틱의 Enable 스위치를 누른 상태로 ,「 confirm_home.py 」를 실행시켜 주십시오 .

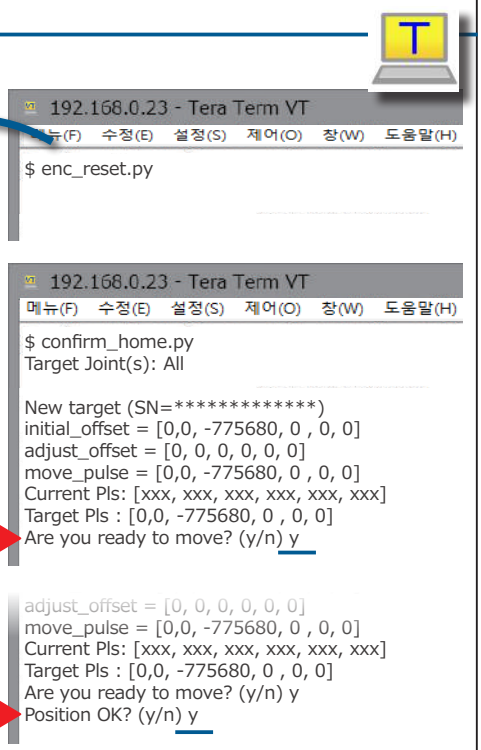
확인 메세지가 표시됩니다 .
Are you ready to move? (y/n) y

「 y 」(소문자) 를 입력합니다 .

「 y 」을 입력하고 Enter 를 누르면 모든 조인트가 각 관절의 영점 마크 위치로 되돌아가는 동작을 합니다 .
(confirm_home 의 매니퓰레이터 속도는 1% 입니다 .)
X : 18.0 mm/s Y : 18.0 mm/s
Z : 18.0 mm/s R_Z : 10.0 deg/s

동작이 완료되면 확인 메세지가 표시됩니다 .
Position OK? (y/n) y

「 y 」(소문자) 를 입력합니다 .

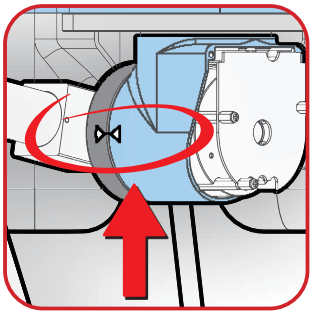


순서 2「 confirm_home.py 」실행 후 확인

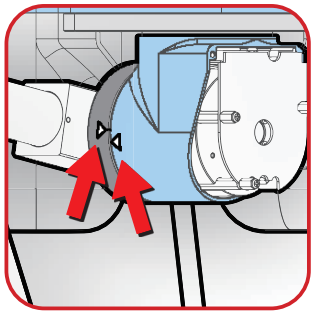
모든 조인트의 영점 마크 위치가 맞는지 확인해 주시기 바랍니다 .

confirm_home.py 실행 후의 예

좋은 예



나쁜 예



영점 마크가 어긋나 있음

C 교시 (Teaching)

4 교시 (Teaching)

1. 기본 조작

⚠ 주 의		
	매니퓰레이터를 처음 동작시키는 경우에는 , 반드시 조인트 좌표계를 선택해 주시기 바랍니다 . 매니퓰레이터의 동작 범위에 장애물이 없는 것을 확인한 후 , Jog 동작을 시행해 주시기 바랍니다 . Jog 동작 중에는 매니퓰레이터로부터 눈을 떼지 말아 주십시오 . 긴급 상황 시 , JOG 스틱 또는 티칭 펜던트의 비상 정지 스위치를 눌러 매니퓰레이터를 정지시켜 주시기 바랍니다 .	
	매니퓰레이터의 동작 중에는 컨트롤러의 전원을 차단하지 마시기 바랍니다 .	

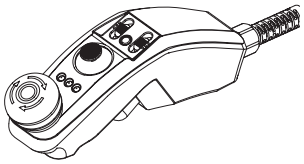
1. 조작 모드

로봇의 운전 모드는 컨트롤러의 CN2 커넥터 접속을 통해 전환합니다 .

교시 모드 (JOG 스틱)

Jog 동작이나 교시를 수행하는 모드입니다 .

JOG 스틱을 연결해 주십시오 .

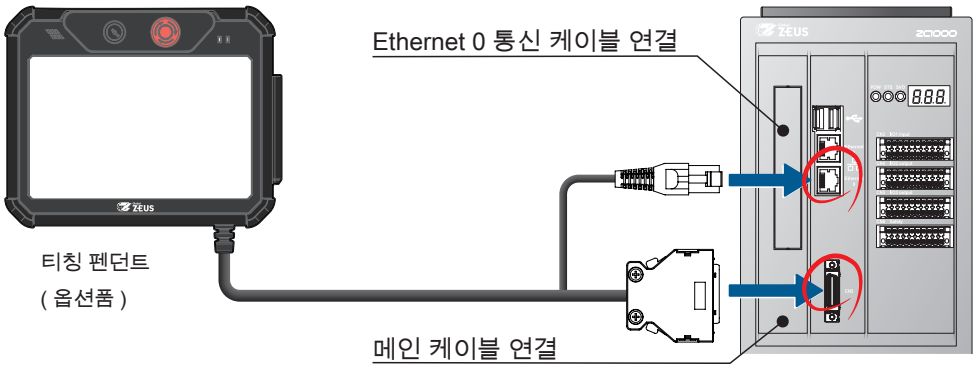


JOG 스틱
(옵션품)

컨트롤러

교시 모드 (티칭 펜던트)

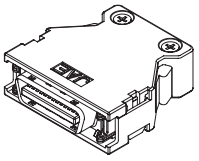
티칭 펜던트를 연결해 주십시오 .



리모트 모드 (자동운전 모드)

로봇의 자동운전을 수행하는 모드입니다 .

점퍼 커넥터를 연결해 주십시오 .



점퍼 커넥터
(부속품)

컨트롤러

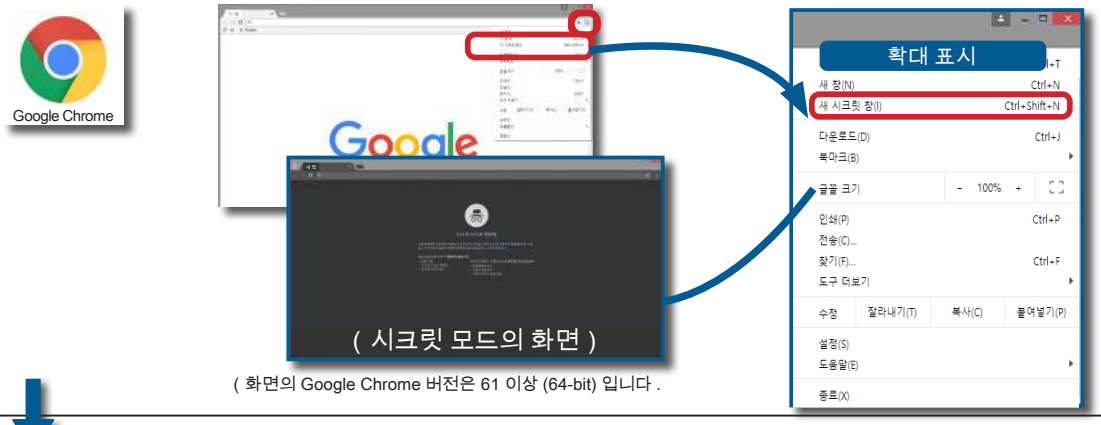
JOG 스틱이나 티칭 펜던트를 사용할 때 외에는 , 점퍼 커넥터를 상시 연결해 주시기 바랍니다 .

2. 준비

교시용 PC 에 접속

Web 브라우저 (Google Chrome) 를 시크릿 모드로 실행합니다 .
연결 주소를 입력하고 , 교시 화면을 실행합니다 .

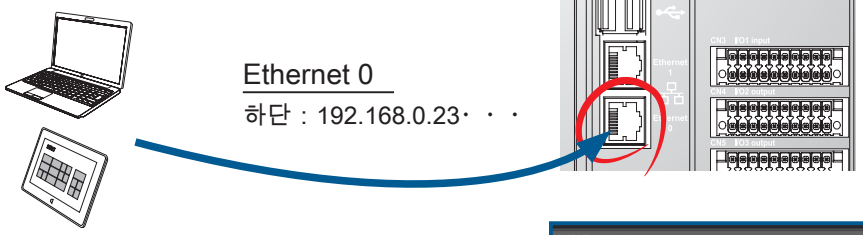
순서 1 브라우저를 시크릿 모드로 실행합니다 .



(화면의 Google Chrome 버전은 61 이상 (64-bit) 입니다 .

순서 2 컨트롤러와 PC 를 연결합니다 .

LAN 케이블을 Ethernet 0 에 연결합니다 .

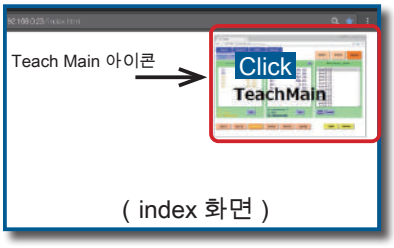


브라우저에 아래 주소를 입력합니다 .

컨트롤러 접속 주소

http://192.168.0.23

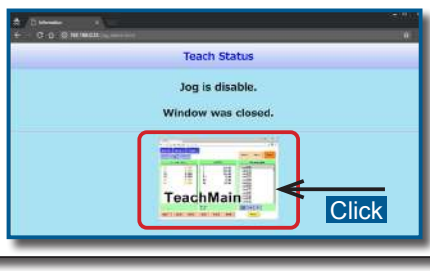
컨트롤러와 연결하면 index 화면이 나타납니다 .



포인트 !

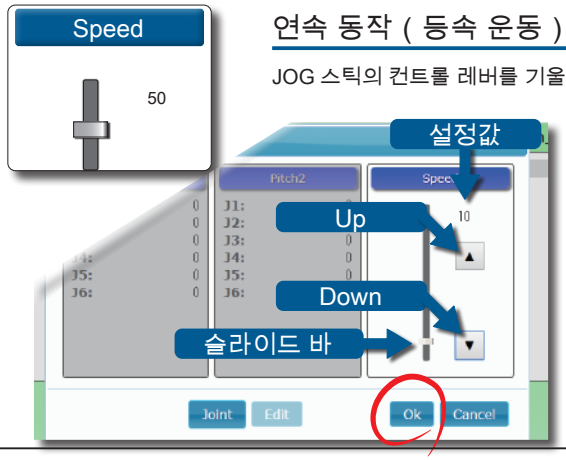
JOG 스틱을 연결하지 않은 경우

교시를 할 수 없습니다 .
index 화면으로 돌아갑니다 .
JOG 스틱을 연결한 후에 , Teach Main 아이콘을 클릭해 주시기 바랍니다 .



동작량과 속도 설정

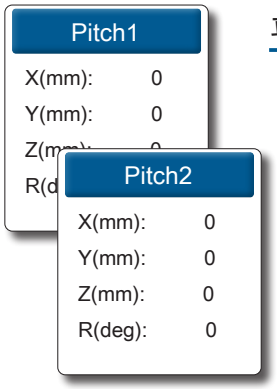
JOG 스틱을 1 회 조작할 때 이동하는 매니퓰레이터의 동작 속도 또는 동작량을 설정합니다 .



연속 동작 (등속 운동) 설정

JOG 스틱의 컨트롤 레버를 기울이고 있는 동안에는 연속해서 동작합니다 .

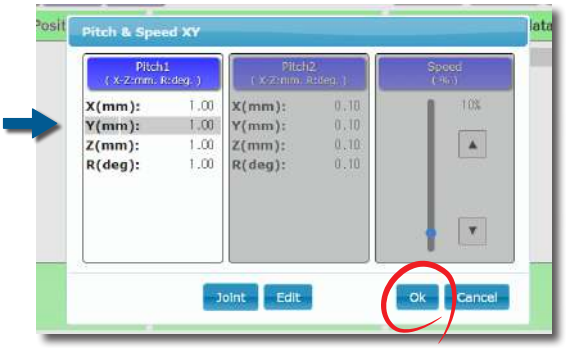
- Speed 버튼을 클릭하여 선택합니다 .
- 슬라이드 바 또는 ▲ ▼ 버튼으로 Jog 동작의 속도 (%) 를 조정합니다 .
  Speed 의 최대 (100%) 는 사양 최고 속도의 3% 입니다 .
  월드 좌표계 XY : X : 54.0 mm/s Y : 54.0 mm/s
  Z : 54.0 mm/s R_Z : 30.0 deg/s
  . . .
- OK 를 클릭하여 Teach Main 화면으로 돌아옵니다 .



피치 이동량 설정

JOG 스틱의 컨트롤 레버를 1 회 기울이면 일정량 동작합니다 .

- Pitch1 또는 Pitch2 버튼을 클릭하여 선택합니다 .
  피치 이동량은 월드 좌표계와 조인트 좌표계로 개별 설정이 가능합니다 .
  좌표계는 Joint / XY 버튼으로 변경 가능합니다 .
  Pitch1 및 Pitch2 로 2 종류의 피치 이동량을 각 조인트마다 설정이 가능합니다 .
- 설정하려는 조인트를 선택하고 , PC 의 키보드 또는 텐키 패널로 입력합니다 .
  설정 범위
  - 조인트 좌표계 : 0 ~ 2 [deg]
  - 월드 좌표계 : 0 ~ 10 [mm]
- OK 를 클릭하여 Teach Main 화면으로 돌아갑니다 .



보충

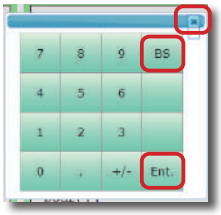
Pitch1 및 Pitch2 에서 설정한 피치 이동 속도는 Speed 의 설정값을 따릅니다 .

PC 의 키보드로 입력

설정하려는 조인트를 선택하고 PC 의 Enter 키를 눌러 값을 직접 입력합니다 .

텐키 패널로 입력

설정하려는 조인트를 선택하고 Edit 버튼을 클릭해 텐키로 값을 입력합니다 .



Close 버튼

BS 백스페이스 키

Ent. 엔터 키

3. 매니퓰레이터를 Jog 동작시키는 방법

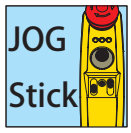
교시 화면이 표시되고 동작량과 속도의 설정이 완료되면 , 매니퓰레이터의 Jog 동작이 가능합니다 . 매니퓰레이터의 동작은「JOG 스틱」또는「조작 패널」로 할 수 있습니다 .

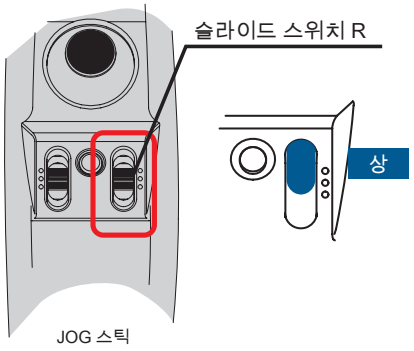
방법 1 JOG 스틱 사용

P.7

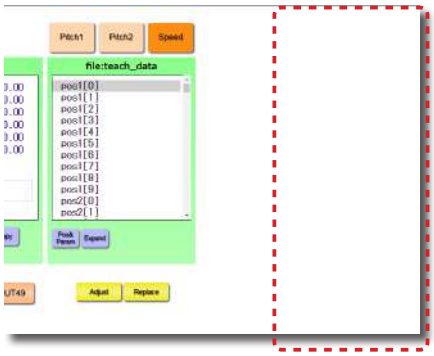
JOG 스틱의 슬라이드 스위치 R 을 " 상 " 위치로 설정합니다 .

(교시 화면의 조작 패널은 표시되지 않습니다 .)





조작 패널 미표시

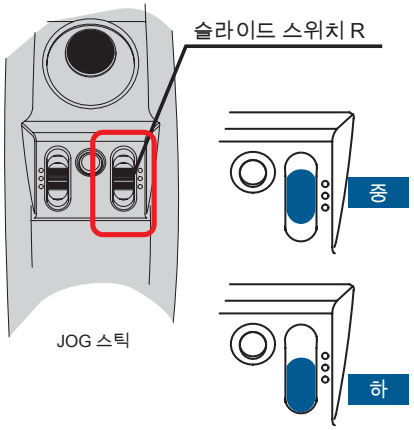


방법 2 조작 패널 사용

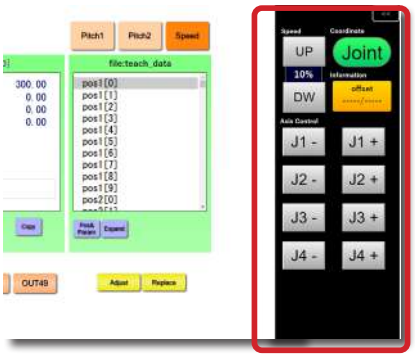
P.8

JOG 스틱의 슬라이드 스위치 R 을 " 중 " 또는 " 하 " 위치로 설정합니다 .

(교시 화면의 조작 패널이 표시됩니다 . JOG 스틱의 컨트롤 레버와 슬라이드 스위치 L 의 조작은 반영되지 않습니다 .)

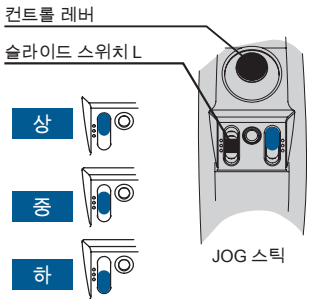


조작 패널 표시



방법 1 JOG 스틱 사용

JOG 스틱 (옵션) 을 사용하여 매니퓰레이터의 각 조인트를 Jog 동작합니다 . Jog 동작은 원점 자세로 이동하거나 교시에 사용합니다 .



순서 1 슬라이드 스위치 R 을「상」위치로 변경합니다 .

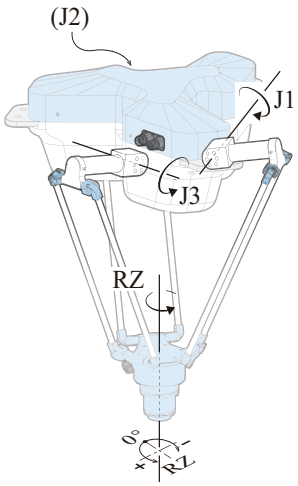
순서 2 Enable 스위치를 눌러 서보 ON 을 합니다 .

순서 3 컨트롤 레버를 기울여 Jog 동작을 합니다 .

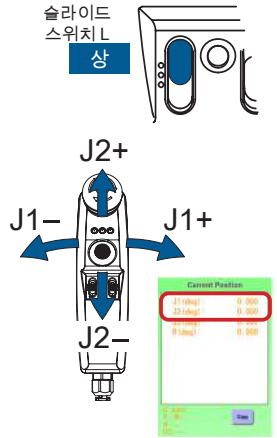
슬라이드 스위치 L 을 전환하여 조작하려는 조인트를 선택합니다 .

조인트 좌표계

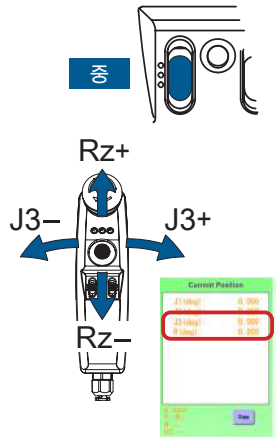
(교시 화면에서는 "Joint" 로 표시됩니다 .)



조인트 J1, J2

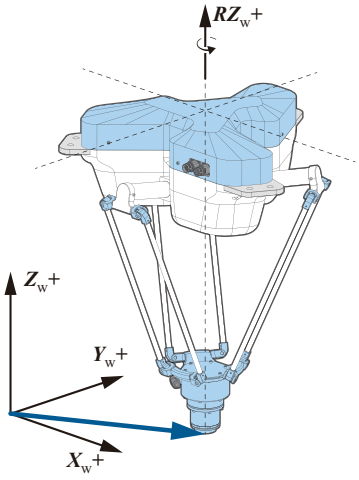


조인트 J3, Rz

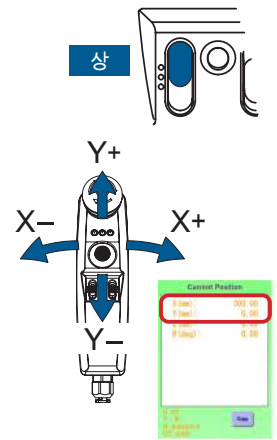


월드 좌표계

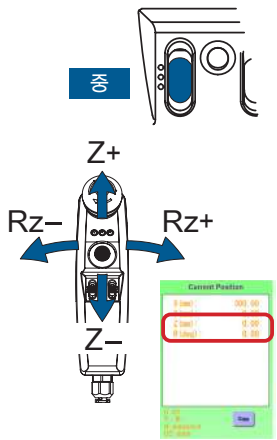
(교시 화면에서는 "XY" 로 표시됩니다 .)



X, Y 축

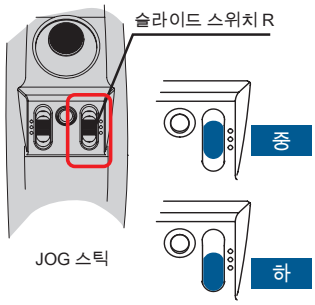


Z, Rz 축



방법 2 조작 패널 사용

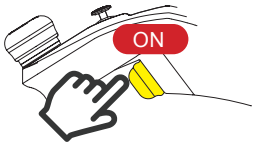
조작 패널은 교시용 PC 의 화면에 배치된 동작 버튼을 사용하여 각 축을 동작시킵니다 .



순서 1 슬라이드 스위치 R 을「중」또는「하」로 전환합니다 .

교시 화면에 조작 인터페이스가 나타납니다 .

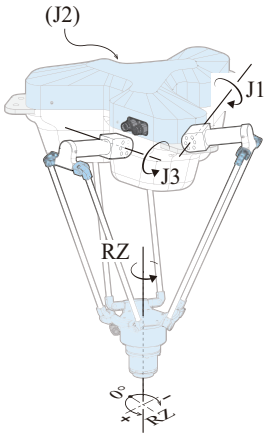
순서 2 Enable 스위치를 눌러 서보 ON 을 합니다 .

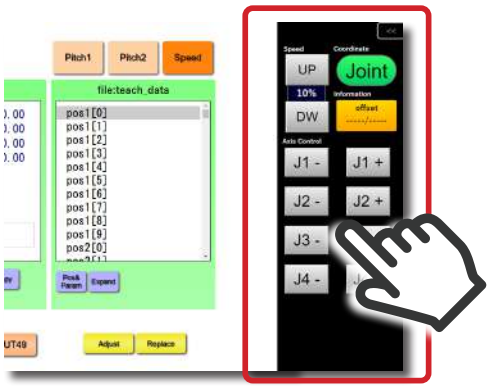


순서 3 조작하려는 좌표계를 선택하여 , 축의 버튼을 누르면 Jog 동작을 할 수 있습니다 .

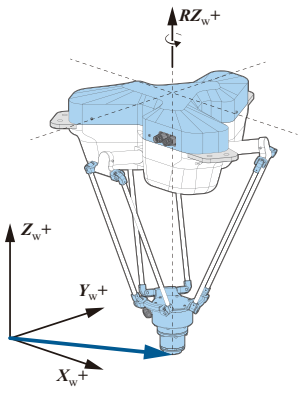
조작 패널이 활성화되면 , JOG 스틱의 컨트롤 레버와 슬라이드 스위치 L 의 조작이 적용되지 않습니다 .

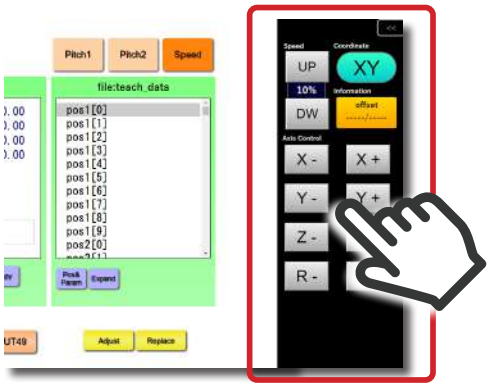
조인트 좌표계





월드 좌표계



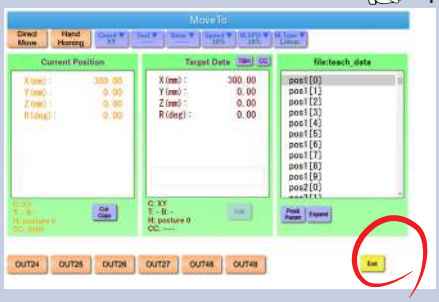


4. 교시 화면

교시는 Teach Main 화면에서 전환 버튼을 클릭하여 화면을 전환합니다 .

Move To 화면

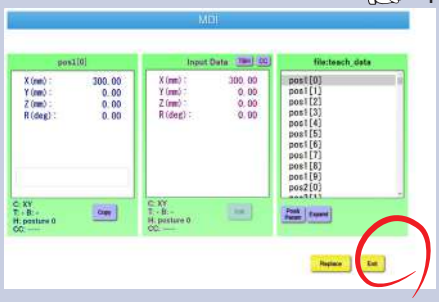
P.16



교시 포인트에 직접 이동하는 Direct Move 와 원점 자세로 이동하는 Hand Homing 이 가능합니다 .

MDI 화면

P.22



교시 데이터에 직접 수치를 입력합니다 .

Teach Main 화면

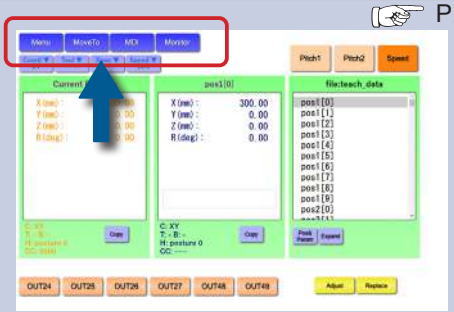
P.10

Exit

Move To

Menu

Close 버튼



교시 데이터의 설정과 편집 , Jog 동작 , 속도 설정 , 좌표계 설정 등이 가능합니다 .

Exit

MDI

Monitor

Close 버튼

Menu 화면

P.11

Tool 오프셋과 Base 오프셋을 설정합니다 .

Monitor 화면

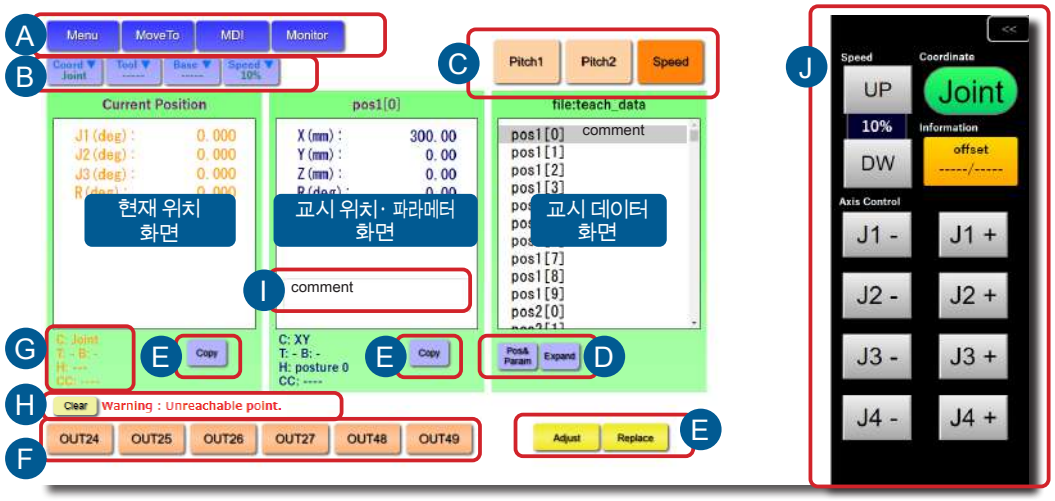
P.23

매니퓰레이터의 좌표나 I/O 등 각종 데이터에 대해 모니터링을 합니다 .
모니터링하면서 Jog 동작이나 포트 출력 조작이 가능합니다 .

Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면

Teach Main 화면

교시 조작의 메인 화면입니다 . 교시 데이터의 설정과 편집 , Jog 동작 속도 설정 , 좌표계의 설정 및 변경 , 오프셋 조정을 실시합니다 .



A 메뉴와 화면의 전환

Menu P.11 메뉴화면 표시

Move To P.16 Move To 화면으로 전환

MDI P.22 MDI 화면으로 전환

Monitor P.23 Monitor 화면 표시

B 표시와 설정 P.12

Coord ▾ Joint — 좌표계 전환

Tool ▾ ---- — Tool offset 선택

Base ▾ ---- — Base offset 선택

Speed ▾ 50% — Jog 동작 속도 설정

C Jog 동작 모드 전환 P.13

Pitch1 Pitch2 — 피치 이동 모드

Speed — 연속 동작 모드

D 교시 데이터 필터링 P.13

Pos& Param — 표시 항목 변경

Expand — 배열 전개 표시 변경

E 교시 데이터· 좌표 데이터의 조작 P.14

Copy — 표시된 좌표값을 클립보드에 복사

Adjust — 서보 OFF 직전 좌표값 표시

Replace — 포지션 데이터 저장

F 출력 포트의 조작 P.15

OUT24 • • • OUT49

유저 I/O 출력 제어

G 좌표 정보

C : 좌표계 T : Tool 오프셋 설정값

B : Base 오프셋 설정값

H : 자세 CC : 크로스오버 카운터 값

H 에러 / 경고 정보 P.15

메시지 상자에 내용 표시

Clear : 메시지 삭제

I 코멘트 표시

교시 (Teaching) 데이터에 입력한 코멘트 표시

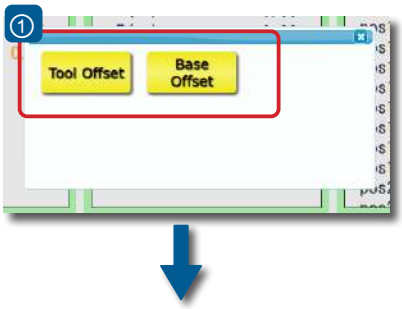
J 조작 패널 P.19

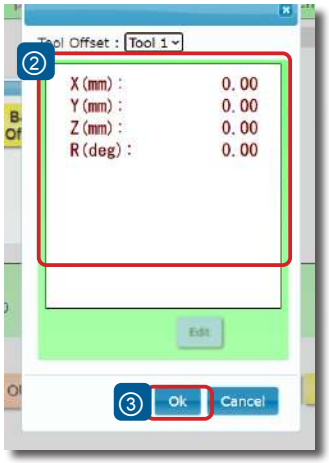
JOG 스틱의 슬라이드 스위치 R 을 " 중 " 또는 " 하 " 로 변경하면 표시됩니다 . 화면에 나타난 조작 버튼으로 매니퓰레이터를 Jog 동작할 수 있습니다 .

Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면
Teach Main 화면

A Menu 버튼

오프셋을 설정합니다 .





화면은 Tool Offset 의 예시입니다 .

① 설정한 오프셋 선택

Tool Offset : Tool Offset 설정

Tool 오프셋은 Tool center point 중심을 원점으로 하여 설치하는 말단 Tool 에 따라 설정합니다 .

Base Offset : Base Offset 설정

Bottom Flange 중심을 원점으로 하여 월드 좌표계부터 매니퓰레이터의 설치 상태에 맞추어 Base 오프셋 설정합니다 .

(설정한 오프셋은 Tool , Base 로 선택합니다 .)

② 설정하려는 항목을 선택하고 입력합니다 .

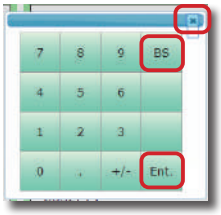
설정하려는 축을 선택하여 PC 의 키보드 또는 텐키 패널로 입력합니다 .

PC 의 키보드로 입력

설정하려는 축을 선택하여 , PC 의 키보드로 Enter 키를 눌러 값을 직접 입력합니다 .

텐키 패널로 입력

설정하려는 축을 선택하고 , Edit 버튼을 클릭하여 텐키로 값을 입력합니다 .



Close 버튼
BS 백스페이스 키
Ent. 엔터 키

③ OK 를 클릭하여 설정을 종료합니다 .

포인트 !

교시 화면 내 버튼 색상

기능 그룹 별로 색상을 구분합니다 .

- 주황색 : 동작에 관한 그룹
- 청색 : 설정· 메뉴에 관한 그룹
- 하늘색 : 편집에 관한 그룹 (편집만 가능)
- 보라색 : 선택에 관한 그룹
- 노랑색 : 중요한 조작에 관한 그룹

버튼 옆의 ▾ 마크

Coord ▾ Joint

▾ 마크가 있는 버튼을 클릭하면 팝업이 나타납니다 .
팝업 표시를 해제하려면 , 한번 더 ▾ 마크가 있는 버튼을 클릭합니다 .

보라색 버튼

Coord ▾ Joint

보라색 버튼의 2 번째 줄은 현재 선택되어 있는 항목을 표시합니다 .

Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면
Teach Main 화면

B Coord Joint | Tool ---- | Base ---- | Speed 50% 버튼

Coord Joint 좌표 변환

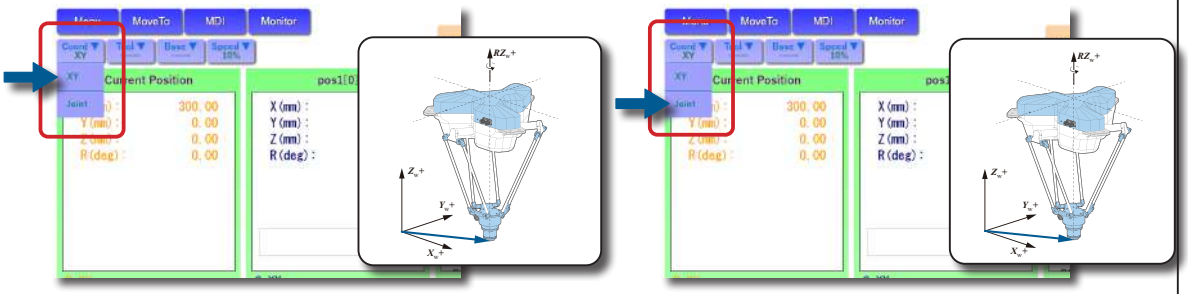
매니퓰레이터를 움직일 때의 좌표계를 선택합니다 .

「XY」,「Joint」중에서 선택합니다 . Current Position 에 표시되는 매니퓰레이터의 현재 위치도 선택한 좌표계 화면으로 변환됩니다 .

XY : 월드 좌표계 (직교 좌표계)

(Base 좌표계 , Tool 좌표계 , 유저 좌표계 포함)

Joint : 조인트 좌표계



Tool ---- Base ---- 오프셋 선택

설정한 오프셋을 변경합니다 .

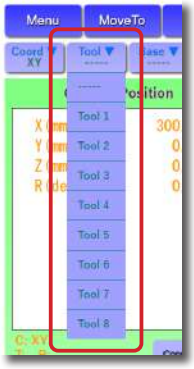
Tool ---- : Tool 오프셋 선택

Tool center point 의 중심을 원점으로 하여 , 설치한 말단 Tool 에 따라 선택합니다 .

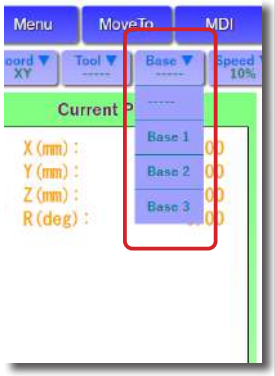
Base ---- : Base 오프셋 선택

Bottom Flange 의 중심을 원점으로 하여 , 월드 좌표계로부터 매니퓰레이터의 설치 상태에 따라 선택합니다 .

Tool 오프셋



Base 오프셋



- Tool 오프셋이나 Base 오프셋을 선택하면 자동적으로 Tool 좌표계나 Base 좌표계로 전환됩니다 .
  오프셋을 해제하는 경우에는「----」를 선택하여 주시기 바랍니다 .
- 이 버튼은 Teach Main 화면과 Move To 화면에서 사용할 수 있습니다 .

Speed 50% 속도 설정

JOG 스틱을 1 회 조작할 때 이동하는 매니퓰레이터의 동작 속도 또는 동작량을 설정합니다 .

예를 들면 , Pitch1 을 큰 동작용 , Pitch2 를 미소 동작(정확한 위치 결정) 용으로 설정하여 , 구분하여 사용이 가능합니다 .

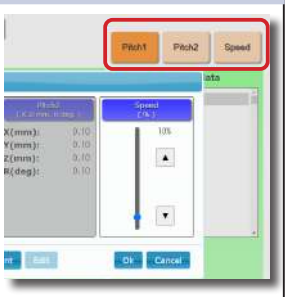
Speed 의 최대 (100%) 는 사양 최고 속도의 3% 입니다 .

월드 좌표계 XY : X : 54.0 mm/s Y : 54.0 mm/s
Z : 54.0 mm/s R_Z : 30.0 deg/s

조인트 좌표계 Joint : J1 : 54.0 deg/s J2 : 54.0 deg/s
J3 : 54.0 deg/s R_Z : 30.0 deg/s

Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면
Teach Main 화면

C Pitch1 Pitch2 Speed 버튼



Jog 동작 모드를 변경합니다 .

Speed : 연속 동작 모드

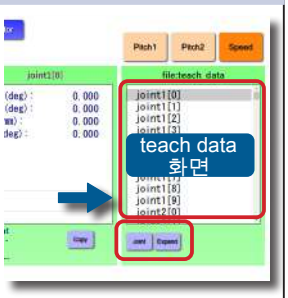
JOG 스틱의 컨트롤 레버를 기울이고 있는 동안 연속해서 동작합니다 .
속도는 Speed 의 설정값을 따릅니다 .

Pitch1 Pitch2 : 피치 이동 모드

JOG 스틱의 컨트롤 레버를 1 회 기울였을 때 , 일정량의 동작을 합니다 .
속도는 각각의 Pitch1 Pitch2 의 설정값을 따릅니다 .

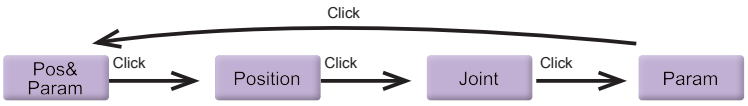
동작량과 속도 설정 방법은 P.5 를 참조하여 주시기 바랍니다 .

D Pos& Param Expand 버튼



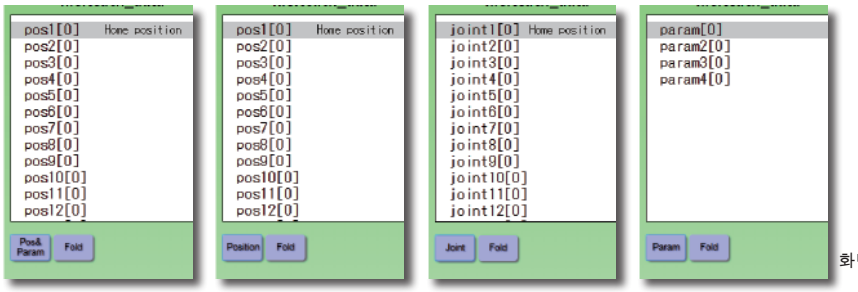
teach data 화면 의 표시 항목을 변경합니다 .

필터 1 : Pos& Param 표시 데이터의 전환



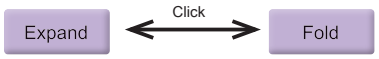
표시 데이터
- Position 데이터
- Joint 데이터
- Param 데이터

- Position 데이터
- Joint 데이터
- Param 데이터

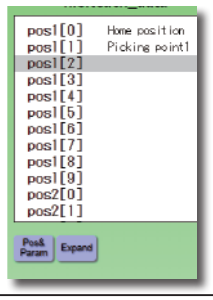


화면 예시

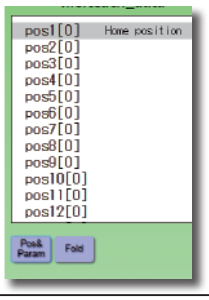
필터 2 : Expand 배열 표시의 전환



배열이 확장된 상태입니다 .



배열이 접힌 상태입니다 .



Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면

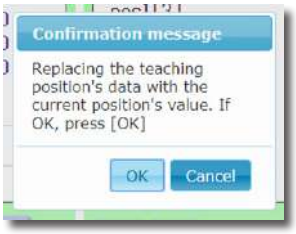
Teach Main 화면

E Copy Adjust Replace 버튼

Copy 표시 중인 좌표값을 클립보드에 복사합니다 .

Replace 포지션 데이터를 저장합니다 .

현재 위치 화면 의 좌표를 교시 데이터 화면 에서 선택한 교시 데이터에 저장합니다 .

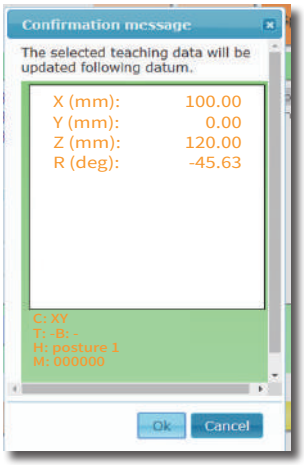


확인 메시지 창

Adjust 포지션 데이터를 보정하고 저장합니다 .

현재 위치 보정 기능 (*) 을 사용하여 교시 데이터 화면 에서 선택한 교시 데이터에 저장합니다 .

*) 서보 OFF 직전의 좌표값으로 보정하는 기능입니다 .



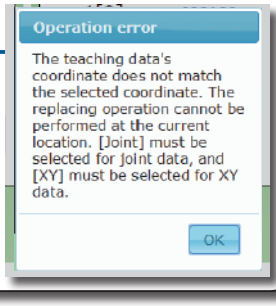
확인 메시지 창

데이터를 저장하는 중에는 컨트롤러의 전원을 차단하지 말아 주십시오 .

포인트 !

좌표계가 일치하지 않는 경우

현재 위치 화면 에서 표시된 좌표계가 저장된 교시 포인트의 좌표계와 다를 경우에는 에러 메시지가 나타납니다 .
좌표계를 일치시킨 후 , 다시 저장하여 주시기 바랍니다 .



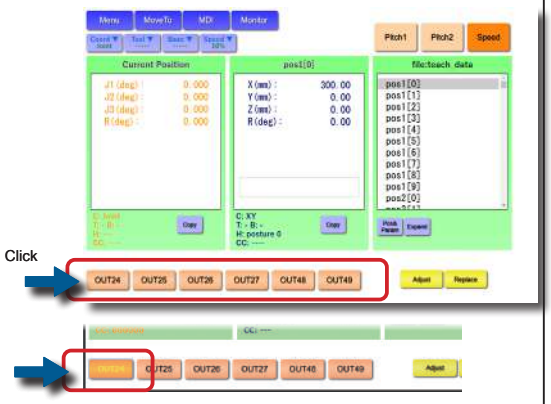
Teach Main 화면 | Move To 화면 | 조작 패널 | MDI 화면 | Monitor 화면

F OUT24 ... OUT49 버튼

출력 포트를 조작합니다 .

미리 등록한 출력 포트를 제어합니다 .
(포트 No.16 ~ 31)

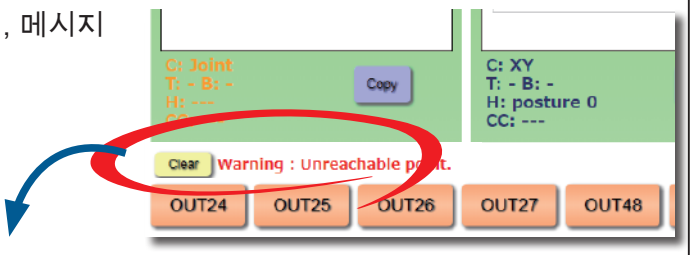
글자 색으로 출력 상태를 표시합니다 .
검은색 문자 : 출력 OFF 상태
노랑색 문자 : 출력 ON 상태



H 에러 / 경고 정보

교시 (Teaching) 시에 경고가 발생하면 , 메시지가 표시됩니다 .

Clear : 메시지를 제거합니다 .

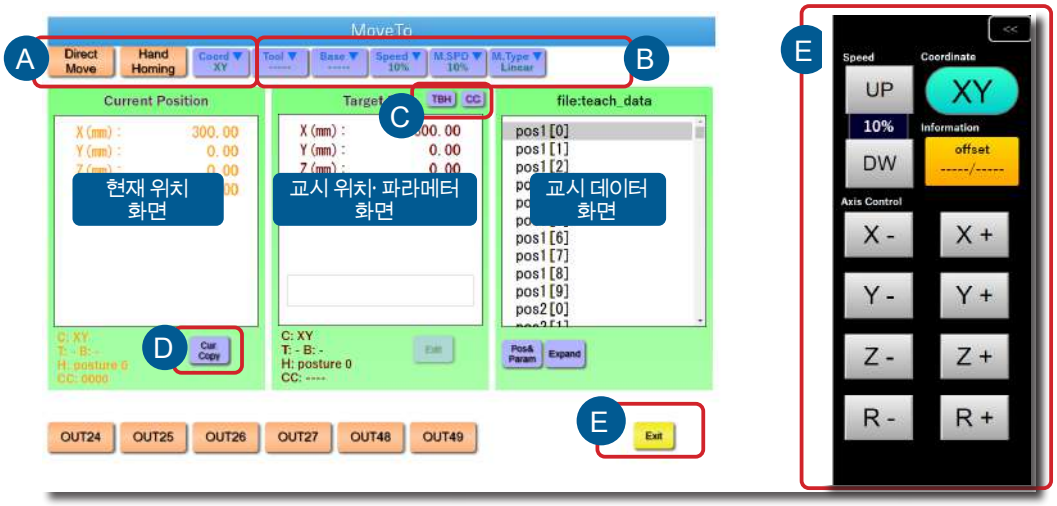


표시	의미
Warning - Angle limit over	동작 목표 위치가 가동 범위 (-60° ~ +100°) 를 넘었음
Warning - Unreachable point	Direct Move (선형 보간 동작) 중 이동 불가 지점을 통과하려 했음
Warning - Area over	Direct Move (선형 보간 동작) 중 가동 범위 (-60° ~ +100°) 를 넘었음
Warning - Speed over	교시 모드의 상한 속도를 넘었음

Teach Main 화면 | 조작 패널 | MDI 화면 | Monitor 화면

Move To 화면

저장된 교시 포인트로 직접 이동하는「Direct Move 동작」와 원점 위치로 이동하는「Hand Homing 동작」이 있습니다 .



A 동작 선택 ☞ P.17

Direct Move
교시위치· 파라메터 화면에 표시된 위치로 동작합니다 .

Hand Homing
각 축을 원점으로 복귀합니다 .
각 축 0°의 위치로 이동합니다 .

B 표시와 설정 ☞ P.17

Coord ▾ Joint
좌표계를 전환합니다 .

Base ▾ ----
Base 오프셋을 선택합니다 .

M.SPD ▾ 50%
Direct Move 의 동작 속도를 설정합니다 .

Tool ▾ ----
Tool 오프셋을 선택합니다 .

Speed ▾ 50%
Jog 동작을 설정합니다 .

M.Type ▾ Linear
Direct Move 의 동작 방법을 선택합니다 .

C 좌표 위치 설정 ☞ P.18

TBH
Tool 오프셋 , Base 오프셋 , 자세를 설정합니다 .

CC
크로스오버 카운터를 설정합니다 .

D 교시 (Teaching) 데이터 조작 ☞ P.18

Cur. Copy
현재의 좌표값을 복사합니다 .

E Move To 화면 종료

Exit
Teach Main 화면으로 돌아갑니다 .

F 조작 패널 ☞ P.19

JOG 스틱의 슬라이드 스위치 R 을 " 중 " 또는 " 하 " 에 위치하면 표시됩니다 . 화면에 배치된 조작 버튼으로 매니퓰레이터를 Jog 동작할 수 있습니다 . Teach main 화면의 조작 패널도 동일합니다 .

Teach Main 화면 | 조작 패널 | MDI 화면 | Monitor 화면

Move To 화면

A Direct Move Hand Homing 버튼

Direct Move Target Data 화면에 표시된 좌표로 이동합니다 .

P.26 「Direct Move」에 의한 로봇의 동작

Hand Homing 매니퓰레이터를 원점 위치로 이동합니다 .

P.25 「Hand Homing」에 의한 원점 좌표로의 이동

B M.SPD 50% M.Type Linear 버튼

M.SPD 50% Direct Move Hand Homing 에서 사용하는 동작 속도를 설정합니다 .

M.SPD 의 100% 는 가장 긴 거리를 이동하는 축의 속도를 기준으로 하며 ,

사양 최고 속도의 3% 입니다 .

M.Type Linear : 54mm/s M.Type PTP

M.Type Linear 동작 모드를 설정합니다 .

월드 좌표계 Coord XY 에서 2 점 사이를 이동하는 동작 모드를 선택합니다 .

- M.Type Linear : 선형 보간 동작
- M.Type PTP : PTP 동작
- M.Type PTP(C2) : PTP 동작 (C2= 크로스오버 카운터 정보 사용)

포인트 !

목표 위치의 좌표계와 선택한 좌표계가 일치하지 않는 경우

대화 상자를 표시합니다 . 좌표계가 불일치한 상태로 Direct Move 버튼을 눌러도 , 안전 상의 이유로 매니퓰레이터를 동작시킬 수 없습니다 . Coord Joint 버튼으로 좌표계를 선택한 후 , 다시 Direct Move 버튼을 눌러 주시기 바랍니다 .

목표 위치로 이동 시에 · · ·

1) 목표 위치에 도달한 경우

도착 메시지 대화 상자가 표시됩니다 .

(Direct Move 와 Hand Homing 에서 표시되는 메시지가 다릅니다 .)

2) 동작 중에 JOG 스틱의 컨트롤 레버 또는 Enable 스위치를 놓친 경우

동작이 취소됩니다 .

(Abort 메시지 대화 상자가 표시됩니다 .)

3) 이동 중에 이동 불가 지점을 통과하는 경우

동작을 중단합니다 .(Unreach 메시지 대화 상자가 표시됩니다 .)

Teach Main 화면 | 조작 패널 | MDI 화면 | Monitor 화면
Move To 화면

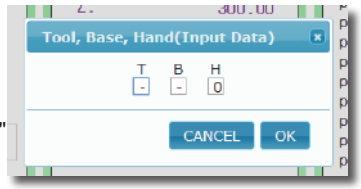
C 좌표 위치 설정

TBH Tool 오프셋 , Base 오프셋 , 자세를 설정합니다 .

T : Tool 오프셋
말단 Tool 에 따라 선택합니다 . 설정값 : "–"(설정 해제), "1" ~ "7"

B : Base 오프셋
매니퓰레이터의 설치 상태에 따라 선택합니다 . 설정값 : "-"(설정 해제), "1" ~ "3"

H : 자세
매니퓰레이터의 자세에 따라 선택합니다 . 설정값 : "0" ~ "1"

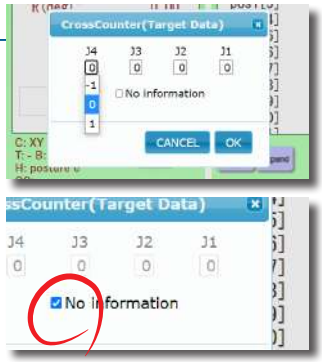


CC 크로스오버 카운터를 설정합니다 .

관절의 회전 정보를 축의 0°를 기준으로 하여 , - 방향 및 + 방향으로 회전 여부를 기록합니다 .

–1 : - 방향으로 180°이상 회전
0 : 무회전
1 : + 방향으로 180°이상 회전

크로스오버 카운터 정보를 사용하지 않는 경우에는「 ☑ No Information 」에 체크 표시를 하십시오 .

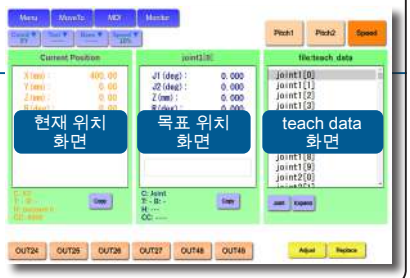


크로스오버 카운터 정보를 사용하지 않는 경우

D 교시 데이터 조작

Cur. Copy 현재 좌표값 복사

- 현재 위치 화면 의좌표값을 목표위치 화면 에 복사합니다 .



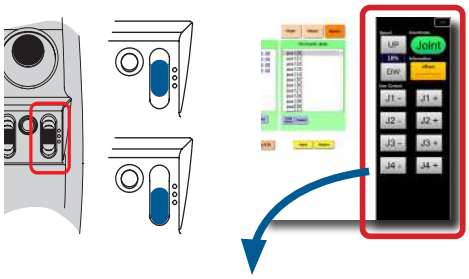
교시 데이터로 저장하는 경우 , MDI 화면에서 편집하십시오 .

Move To 화면에서 수정된 목표 위치는 동작 확인용입니다 . 교시 데이터에는 반영되지 않습니다 .

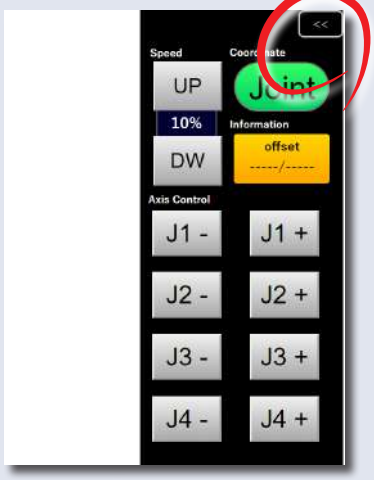
Teach Main 화면 | Move To 화면 | MDI 화면 | Monitor 화면

조작 패널

조작 패널로 JOG 스틱의 컨트롤 레버를 대신하여 매니퓰레이터를 Jog 동작할 수 있습니다 . 조작 패널 설정 , 에러 코드 확인 , 매니퓰레이터의 " 자세 " 확인 , 좌표계와 조인트 축의 정의를 확인할 수 있습니다 .



조작 패널



<< 버튼을 클릭하면 패널이 확장 됩니다 .

Settings

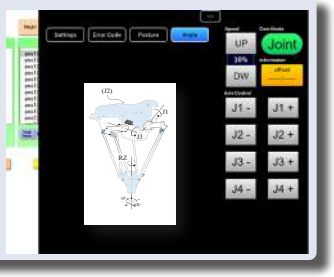
조작 패널을 설정합니다 .

Error Code

에러 코드 목록을 표시합니다 .

Angle

매니퓰레이터의 좌표나 조인트 축의 정의를 표시합니다 .

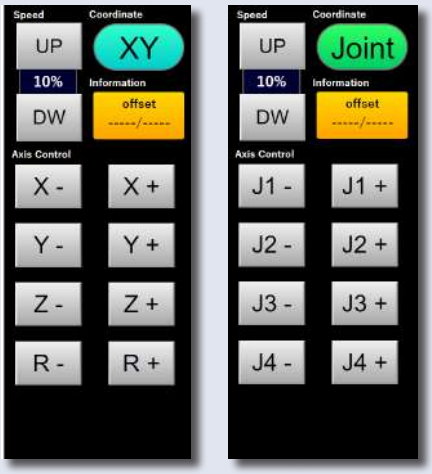


1 기본 조작

Teach Main 화면 | Move To 화면 | MDI 화면 | Monitor 화면

조작 패널

조작 패널 설명



직교 좌표계 조인트 좌표계

XY Joint 좌표계의 변환

버튼을 길게 눌러서 (약 1 초 간) 전환합니다 .

XY 버튼 ⟷ Joint 버튼

길게 누름

(약 1 초 간)

Coord Joint 버튼으로 설정한 좌표계와 연동됩니다 .

UP DW 동작 속도의 변경

설정은 버튼을 누를 때마다「1 %, 3%, 5%, 10% 15%, 20%, 30%, 50%」로 변경됩니다 .

설정 100% 일 때의 동작 속도는 사양 최고 속도의 3% 입니다 .

월드 좌표계 XY : X : 54.0 mm/s Y : 54.0 mm/s Z : 54.0 mm/s R_Z : 30.0 deg/s

조인트 좌표계 Joint : J1 : 54.0 deg/s J2 : 54.0 deg/s J3 : 54.0 deg/s R_Z : 30.0 deg/s

UP 버튼

10%

DW 버튼

현재 설정값을 표시합니다 .

「동작량과 속도의 설정」의 설정과 연동됩니다 .

Speed : 연속 동작 모드

Pitch1 Pitch2 : 피치 이동 모드

X- … R+

J1- … J4+ 축 제어 버튼

선택한 좌표계에 따라 버튼의 표시가 바뀝니다 .

조인트 좌표계 J1- … J4+ 버튼

직교 좌표계 X- … R+ 버튼

버튼 동작

짧게 누름 : 피치 이동 1 회분의 동작

길게 누름 : 연속 동작

offset 오프셋 정보 표시

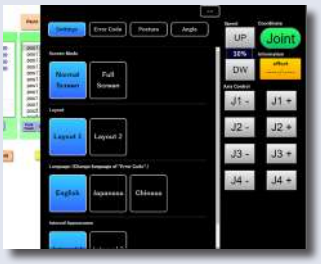
Tool 오프셋과 Base 오프셋의 설정값을 표시합니다 .

오프셋은 Tool , Base 버튼에서 설정해 주시기 바랍니다 .

Teach Main 화면 | Move To 화면 | MDI 화면 | Monitor 화면

조작 패널

Settings 조작 패널을 설정합니다 .



Screen:

전체 화면 표시와 일반 화면 표시를 전환합니다 .

Layout:

버튼 배치를 변경합니다 .

Language:

에러 코드 목록의 표시 언어를 변경합니다 .

「English」「Japanese」「Chinese」

Interval Appearance:

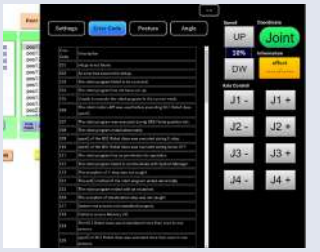
조작이 발생한 경우에 , 다음 조작을 받아들이기 전 조작 무효 시간 동안의 표시 방법을 설정합니다 .

Interval1: 팝업「Data in Process…」를 표시합니다 .

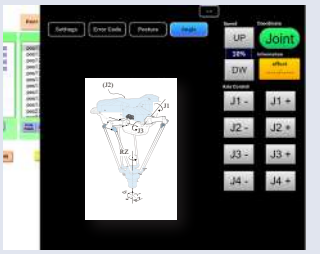
Interval2: 어둡게 표시합니다 .

Error Code 에러 코드 목록을 표시합니다 .

에러에 대한 상세 정보는 영어 , 일본어 및 중국어 (간체) 로 변경할 수 있습니다 .



Angle 매니퓰레이터의 좌표 축이나 조인트 축의 정의를 표시합니다 .

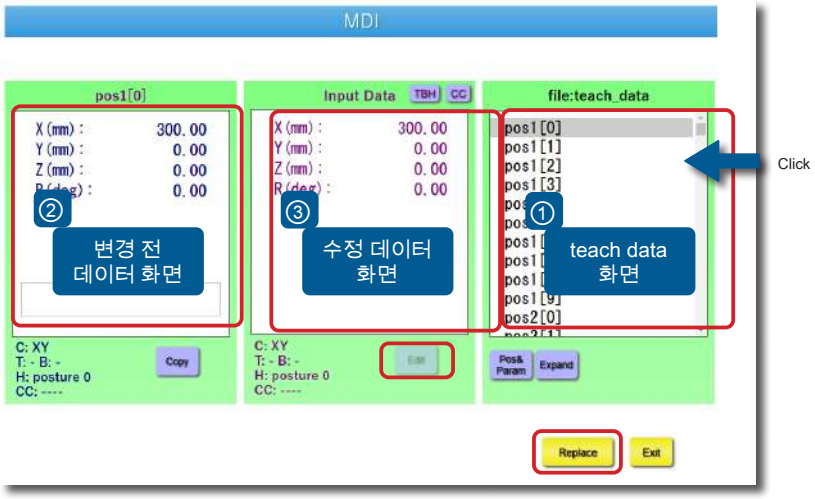


Teach Main 화면 | Move To 화면 | 조작 패널 | Monitor 화면

MDI 화면

MDI = 「Manual Data Input」

선택한 교시 데이터를 편집합니다 .



순서 1 teach data 화면 에서 편집하려는 교시 데이터를 선택합니다 .

클릭 : 데이터를 선택합니다 .
배경 더블 클릭 : 설명을 입력할 수 있습니다 .(최대 32 글자)

순서 2 포지션 데이터를 입력합니다 .

변경 전 데이터 화면 에서 선택한 교시 데이터의 좌표값이 표시됩니다 .
편집하려는 축을 선택하여 , PC 의 키보드 또는 텐키 패널로 입력합니다 .

PC 의 키보드로 입력

설정하려는 축을 선택하여 , PC 의 키보드로 Enter 키를 눌러 값을 직접 입력합니다 .

텐키 패널로 입력

설정하려는 축을 선택하여 , Edit 버튼을 클릭하여 텐키로 값을 입력합니다 .

순서 3 위치 데이터를 등록합니다 .

Replace 위치 데이터의 저장

선택한 교시 데이터를 수정 데이터 화면 의 위치 데이터로 대체합니다 .

포인트 !

설정 가능한 교시 포인트 수

데이터	표시	포인트 수
직교 좌표계 데이터	pos1[0] ~ [9] … pos20[0] ~ [9]	200
조인트 좌표계 데이터	joint1[0] ~ [9] … joint20[0] ~ [9]	200
파라메터 데이터	param[0] ~ [9] … param4[0] ~ [9]	40

Teach Main 화면 | Move To 화면 | 조작 패널 | MDI 화면

Monitor 화면

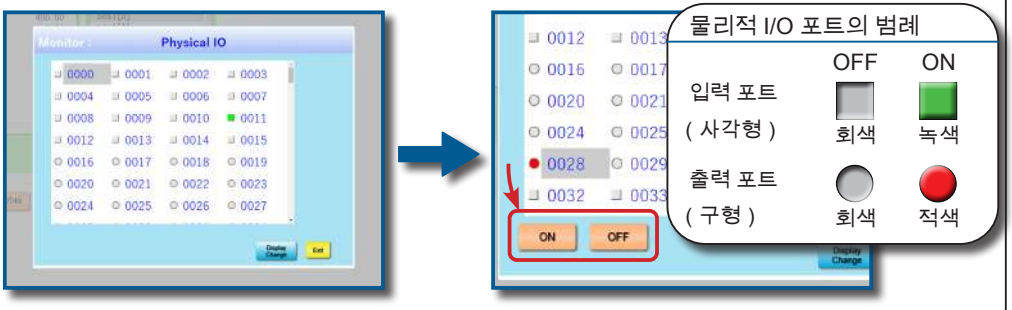
매니퓰레이터의 좌표나 I/O 등의 각종 데이터를 모니터링합니다 .
모니터링을 하는 도중에 Jog 동작이나 포트의 출력 조작이 가능합니다 .

Display Change 표시 항목의 전환

물리적 I/O, 매니퓰레이터 현재 위치 , 버전 정보 , OSS 라이센스 정보 중에서 표시 항목을 선택합니다 .

Physical I/O 물리적 I/O

물리적 I/O 포트의 모니터링을 합니다 .
출력 포트의 상태를 ON OFF 버튼으로 조작합니다 .

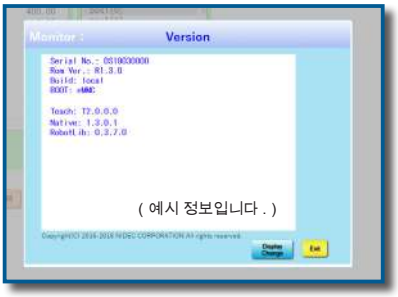


Current Position 매니퓰레이터 현재 위치

매니퓰레이터의 현재 위치를 표시합니다 .
월드 좌표 , 조인트 좌표 , 엔코더 펄스의 3 종류의 정보가 화면에 표시됩니다 .

Version 버전 정보

시스템의 각 소프트웨어의 버전을 표시합니다 .



OSS License OSS 라이센스 정보

본 제품에서 사용하고 있는 오픈소스 소프트웨어의 라이센스 정보를 표시합니다 .

2. 교시 (Teaching) 순서

교시 (Teaching) 의 흐름

로봇을 교시용 PC 와 연결하여 , 동작량과 속도의 설정이 완료되면 교시를 합니다 .

Hand Homing Direct Move 등에서 동작을 하고 , 로봇의 동작 좌표를 교시하시기 바랍니다 .

교시 포인트가 확정되면 좌표 데이터를 교시 데이터로 저장해 주시기 바랍니다 .

「Hand Homing」에 의한 원점 좌표로의 이동

P.25

Hand Homing 원점 복귀를 합니다 .

「Direct Move」에 의한 로봇의 동작

P.26

Direct Move 설정한 위치로 매니퓰레이터를 이동합니다 .

좌표 데이터의 저장

P.29

매니퓰레이터의 현재 좌표값을 교시 데이터로 저장합니다 .

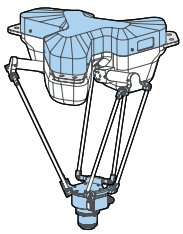
이동 불가 지점부터로의 복구

P.30

매니퓰레이터를 조작 불가능한 위치에서 탈출시킵니다 .

보충 : 원점 자세에서 직교 좌표계의 Jog 동작과 Direct Move 는 할 수 없습니다 .

원점 자세



각 축의 값
J1 = 0 deg
J2 = 0 deg
J3 = 0 deg
J4 = 0 deg

원점 자세는 매니퓰레이터 구조 상 , 월드 좌표계의 조작이 불가능한 자세입니다 .
조인트 좌표계의 Jog 동작으로 J1 ~ J4 를 별도로 조작하거나 , 조인트 좌표계의 Direct Move 로 로봇을 탈출시킵니다 .

P.30 6. 이동 불가 지점에서 복구

「Direct Move」에 의한 로봇의 동작 | 좌표 데이터의 저장 | 이동 불가 지점에서 복구

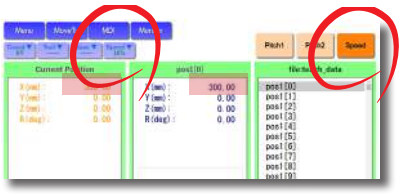
「Hand Homing」에 의한 원점 좌표로의 이동

Hand Homing : 원점 복귀를 합니다 .

순서 1 동작 속도를 설정합니다 .

Teach Main 화면에서 Speed 를 클릭합니다 .

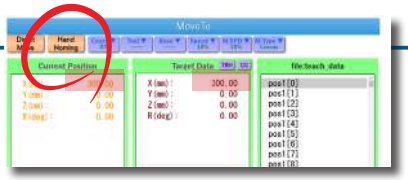
Speed ▾ 10% 로 되어 있는지 확인합니다 .
동작 속도는 10% 정도를 권장합니다 . 동작 속도를 빠르게 하는 경우에는 안전을 충분히 확인한 후 작동하시기 바랍니다 .



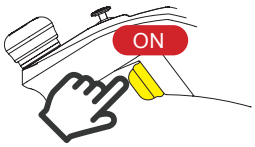
Speed ▾ 의 100% 는 가장 긴 거리를 이동하는 축의 속도를 기준으로 하며 , 사양 최고 속도의 3% 입니다 .

J1 : 54.0 mm/s J2 : 54.0 mm/s R : 54.0 mm/s R_Z : 30.0 deg/s

순서 2 Hand Homing 을 클릭합니다 .



서보 ON 을 합니다 .



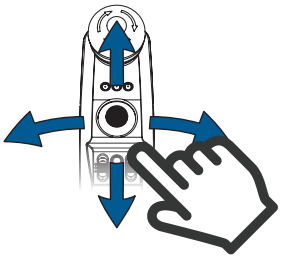
순서 3 이동을 시작합니다 .

JOG 스틱의 경우

컨트롤 레버를 기울이면 이동이 시작됩니다 .

컨트롤 레버를 임의의 방향으로 기울입니다 .
기울이고 있는 동안에만 이동합니다 .

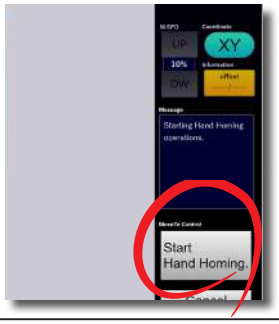
【중지】: 컨트롤 레버에서 손을 뗍니다 .
【완료】: 팝업 화면으로 완료를 알려 줍니다 .



조작 패널의 경우

Start Hand Homing 버튼을 누르면 이동이 시작됩니다 .

【중지】: Cancel 버튼을 누릅니다 .
【완료】: 팝업 화면으로 완료를 알려 줍니다 .

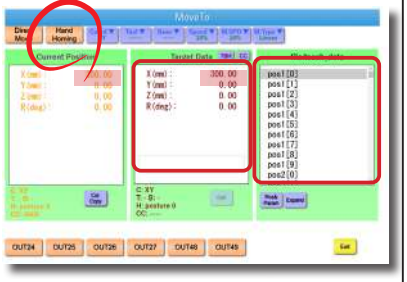


2 교시 (Teaching) 순서

「Direct Move」에 의한 로봇의 동작

Direct Move : 설정한 위치를 향해 매니퓰레이터를 동작시킵니다 .

순서 1 목표 위치의 선택 혹은 입력을 합니다 .



・ 교시 데이터를 등록한 경우 :

➡ file:teach_data 중에서 목표 위치를 선택합니다 .

・ 교시 데이터를 수정하는 경우 :

또는

・ 새로운 목표 위치를 설정하는 경우 :

➡ Target Data 의 대상 조인트를 선택합니다 .

PC 의 키보드로 입력

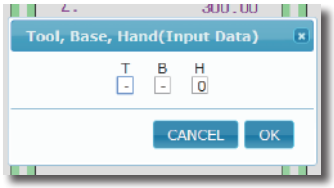
PC 의 키보드로 Enter 키를 눌러 수치를 직접 입력합니다 .

텐키 패널로 입력

Edit 버튼을 클릭하여 텐키 패널로 수치를 입력합니다 .

・ Tool 오프셋 , Base 오프셋 , 자세를 설정하는 경우 :

➡ TBH 을 클릭하여 파라메터를 선택합니다 .



T : Tool 오프셋

말단 Tool 에 따라 선택합니다 .
설정값 : "–"(설정 해제), "1" ~ "7"

B : Base 오프셋

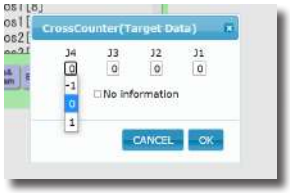
매니퓰레이터의 설치 상태에 따라 선택합니다 .
설정값 : "–"(설정 해제), "1" ~ "3"

H : 자세

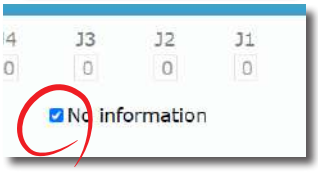
매니퓰레이터의 자세에 따라 선택합니다 .
설정값 : "0"

・ 크로스오버 카운터 (*1) 를 설정하는 경우 :

➡ CC 를 클릭하여 각 조인트의 각도 범위에 따라 아래의 값을 설정합니다 .



설정값	각 조인트의 각도
1	+ 방향으로 180°이상 회전
0	180° ~ 180°
F (*2)	- 방향으로 180°이상 회전



크로스오버 카운터 정보를 사용하지 않는 경우

크로스오버 카운터 정보를 사용하지 않는 경우에는「☑ No Information 」에 체크하여 주시기 바랍니다 .

*1) 크로스오버 카운터는 Position 형 위치데이터를 고유한 Joint 형 각도 데이터로 변환하기 위한 설정입니다 .
Position 형 데이터의 multiturn 파라메터에 설정되어 있습니다 .
상세한 내용은「D 소프트웨어 2 로봇 라이브러리」를 참조하여 주시기 바랍니다 .

*2) 교시 화면에서는 "F", 설정할 때는 "-1" 로 표시됩니다 .

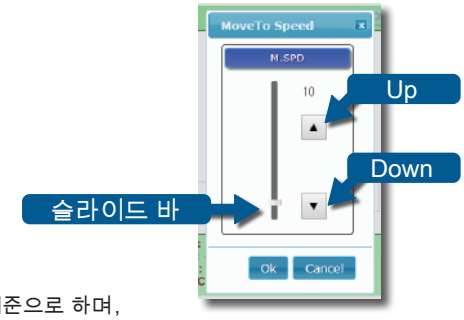
2 교시 (Teaching) 순서

「Hand Homing」에 의한 원점 좌표로의 이동 | 좌표 데이터의 저장 | 이동 불가 지점에서 복구
「Direct Move」에 의한 로봇의 동작 (상세)

순서 2 좌표계와 동작 방법 및 속도를 선택합니다 .

- 좌표계 :
  선택한 교시 데이터와 조작하려는 좌표계를 일치시킵니다 .
  - Coord XY 월드 좌표계
  - Coord Joint 조인트 좌표계

- 동작 방법 :
  월드 좌표계 Coord XY 에서 2 점 간의 이동 모드를 선택합니다 .
  - M.Type Linear : 직선 보간 동작
  - M.Type PTP : PTP 동작
  - M.Type PTP(C2) : PTP 동작 (C2= 크로스오버 카운터 정보 이용)



- 동작 속도 :
  슬라이드 바 또는 ▲ ▼ 버튼으로 동작 속도 (%) 를 조정합니다 .

  이 동작 속도의 100% 는 , 가장 긴 거리를 이동하는 축의 속도를 기준으로 하며,
  사양 최고 속도의 3% 입니다 .

  M.Type Linear 80 mm/s

  M.Type PTP J1 : 54.0 deg/s J2 : 54.0 deg/s
  J3 : 54.0 deg/s R_Z : 30.0 deg/s

다음 페이지로

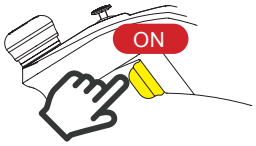
포인트!

MoveTo 화면에서 설정한 좌표값과 오프셋 정보 등은 동작 확인을 위한 임시 설정입니다 .
교시 데이터에는 반영되지 않습니다 .

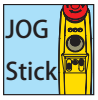
2 교시 (Teaching) 순서

순서 3 Direct Move 를 클릭합니다 .

서보 ON 을 합니다 .



순서 4 이동을 시작합니다 .

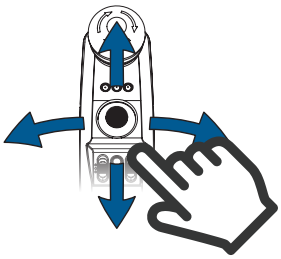


JOG 스틱의 경우

컨트롤 레버를 기울이면 이동이 시작됩니다 .

컨트롤 레버를 임의의 방향으로 기울입니다 .
기울이고 있는 동안에만 이동합니다 .

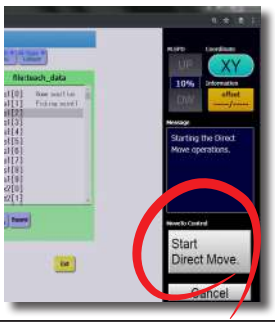
【중지】: 컨트롤 레버에서 손을 뗍니다 .
【완료】: 팝업 화면으로 완료를 알려 줍니다 .



조작 패널의 경우

Start Direct Move 버튼을 누르면 이동이 시작됩니다 .

【중지】: Cancel 버튼을 누릅니다 .
【완료】: 팝업 화면으로 완료를 알려 줍니다 .



2 교시 (Teaching) 순서

「Hand Homing」에 의한 원점 좌표로의 이동 | 「Direct Move」에 의한 로봇의 동작 | 이동 불가 지점에서 복구

좌표 데이터의 저장

현재 위치 화면 의 좌표값을 교시 데이터로 저장합니다 .

순서 1 바꾸려는 대상 교시 데이터를 선택합니다 .

Move To 화면에 있는 경우에는 Exit 를 클릭하여
Teach Main 화면으로 돌아갑니다 .

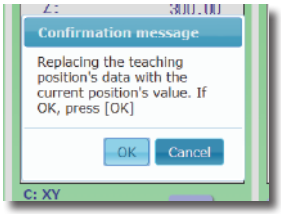
예 : pos1[0] 에 현재 위치 화면 의 좌표를 저장합니다 .

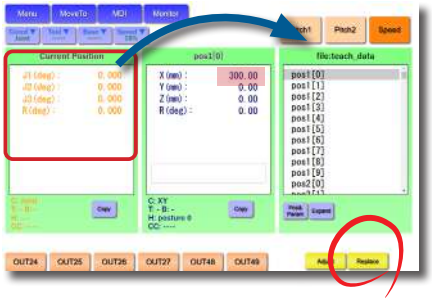
Teach Main 화면

순서 2 Replace 를 클릭합니다 .

현재 위치 화면 의 위치데이터를 pos1[0] 에 저장합니다 .

팝업 화면이 나타납니다 .





OK 를 클릭하여 저장합니다 .
저장이 완료되면 , 팝업 화면이 사라집니다 .

이동 불가 지점에서 복구

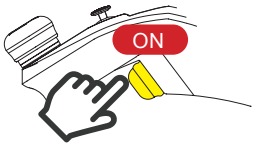
매니퓰레이터를 조작 불가능한 위치에서 복구시킵니다 .

방법 1 JOG 스틱 사용

조인트 좌표계 Coord Joint 에서 J1 ~ J4 를 별도로 JOG 스틱으로 조작하여 , 매니퓰레이터를 조작 가능한 위치 (직교 좌표계의 이동 불가 지점이 아닌 자세) 로 이동시킵니다 .

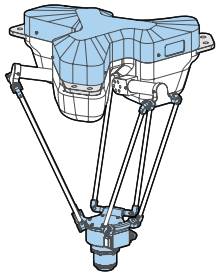
순서 1 JOG 스틱의 슬라이드 스위치 L 로 조인트를 선택합니다 .

순서 2 서보 ON 을 합니다 .



순서 3 컨트롤 레버를 기울여 매니퓰레이터를 조작하여 , 매니퓰레이터를 조작 가능한 위치로 이동시킵니다 .

조작이 가능한 위치의 예



자세의 이미지

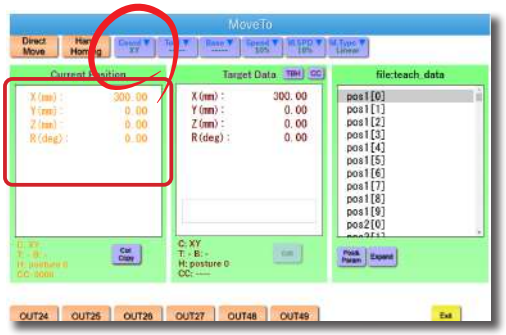
J1 ~ J4 을 대략 왼쪽 그림과 비슷한 자세가 되도록 합니다 .

현재 위치 화면에서 각 조인트의 각도도 확인하면서 구동시키기 바랍니다 .

예) 각 조인트의 값

J1 : 0 deg
J2 : 0 deg
J3 : 0 deg
J4 : 0 deg

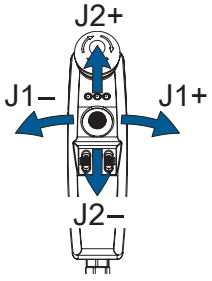
Move To 화면



조인트의 조작 방법

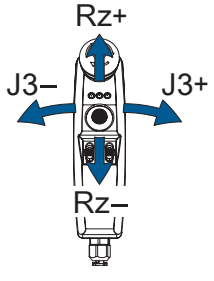
조인트 J1, J2

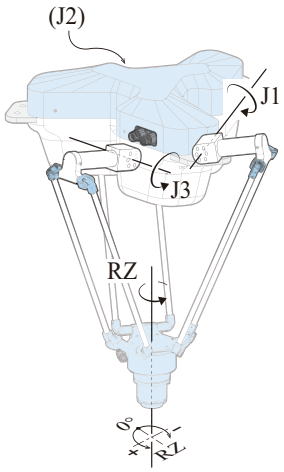
스위치 L 상



조인트 J3, Rz

스위치 L 중

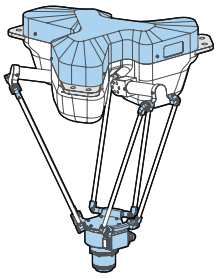




방법 2 조인트 좌표계의 교시 포인트에 Direct Move 로 이동

미리 조인트 좌표계 교시 포인트 (예 : Joint1 [0])에 월드 좌표계에서 이동 가능한 위치를 저장해 놓습니다 . 교시 중 월드 좌표계의 조작 불가능한 지점에 있다 하더라도 , 조인트 좌표계로 전환하고 나서 Direct Move 로 이동하면 쉽게 복구할 수 있습니다 .
복구 이후 , 월드 좌표계로 전환하여 교시를 재개할 수 있습니다 .

예 : Joint1 [0] 에 이동 가능한 지점을 등록합니다 .



자세의 이미지

각 조인트 값의 예시

J1 : 0 deg
J2 : 0 deg
J3 : 0 deg
J4 : 0 deg

Move To 화면

포인트 !

월드 좌표계의 이동 불가 지점을 회피할 때 , 조인트 좌표계를 활용합니다 .

월드 좌표계
동작 범위 내더라도 자세에 따라 이동 불가 지점이 있습니다 .
조인트 좌표계
동작 범위 내에서는 자유롭게 이동할 수 있습니다 .

월드 좌표계 Coord XY 로 동작 중에 매니퓰레이터의 이동 불가 지점에 도달하는 경우 , Direct Move 은 사용할 수 없게 됩니다 . 조인트 좌표계 Coord Joint 로 전환하여 Direct Move 를 사용하거나 , JOG 스틱으로 J1 ~ J4 를 별도로 조작하여 이동 불가 지점으로부터 이동시켜 주시기 바랍니다 .

3. 교시 (Teaching) 데이터 전송

1. 컨트롤러에서 PC 로 전송

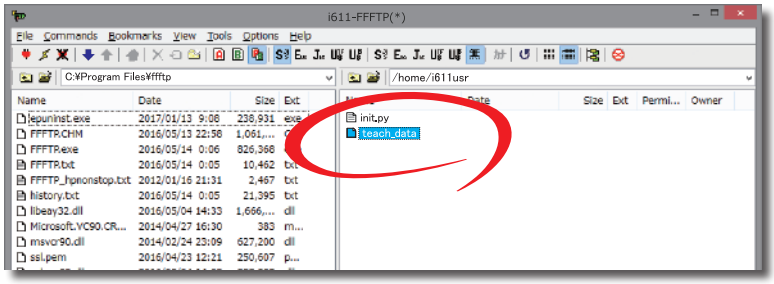
교시 데이터는 PC 에 백업하는 것을 권장합니다 .

교시 데이터는 컨트롤러에 저장되어 있습니다 .
컨트롤러를 교체하는 경우 , 로봇의 교시 데이터나 동작 프로그램을 이식할 수 있습니다 .

순서 1 「FFFTP」를 실행합니다 .

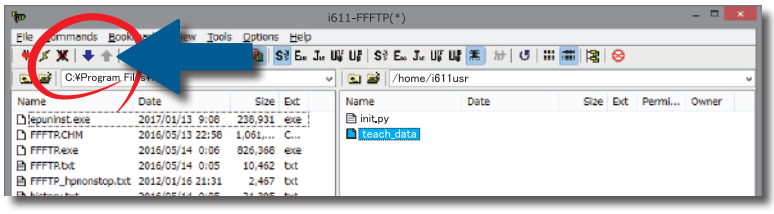
교시 데이터의 저장 위치 및 파일명

저장 경로	/home/i611usr
파일명	teach_data



순서 2 PC 로 파일을 전송합니다 .

다운로드 버튼을 클릭합니다 .



3 교시 (Teaching) 데이터 전송

2. PC 에서 컨트롤러로 전송

	교시 데이터나 동작 프로그램을 다른 컨트롤러에 이식하는 경우에는 반드시 동작 확인을 해 주시기 바랍니다 .	
	교시 데이터를 컨트롤러에 전송하는 경우에는 지정된 경로에 저장해 주시기 바랍니다 .	

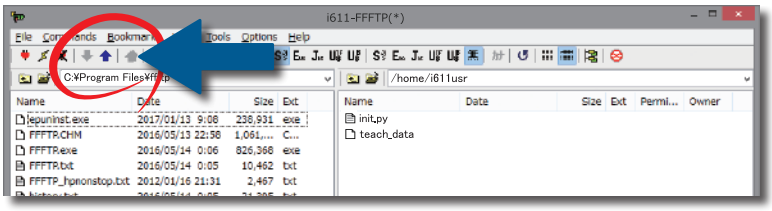
순서 1 「FFFTP」를 실행합니다 .

교시 데이터의 저장 위치

저장 경로	/home/i611usr

순서 2 컨트롤러로 파일을 전송합니다 .

업로드 버튼을 클릭합니다 .



MEMO

C 교시 (Teaching)

5 좌표계

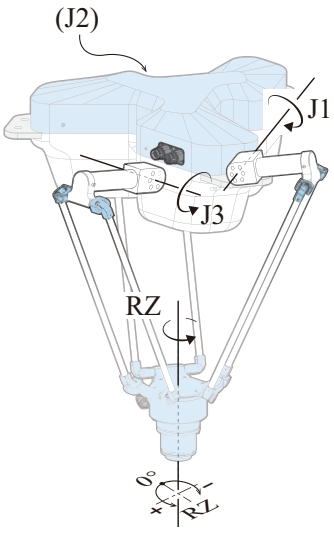
1. 좌표계 체계

본 제품에서 정의하고 있는 좌표계는 조인트 좌표계 , 월드 좌표계 , 베이스 좌표계 , Tool 좌표계 , 유저 좌표계의 5 개입니다 .
각각의 정의를 이해하신 후 , 사용 목적에 맞는 좌표계 (*) 를 선택하시기 바랍니다 .

*)「교시 모드」는 조인트 좌표계와 월드 좌표계에 대응하고 있습니다 . 베이스 좌표계와 Tool 좌표계는 각각 월드 좌표계에 오프셋을 설정하여 사용합니다 .

「좌표계」를 결정할 때 , 매니퓰레이터의「이동 불가 지점」을 고려하시기 바랍니다 .

조인트 좌표계



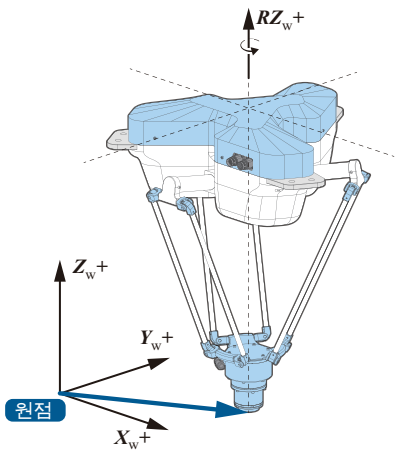
각 조인트에 개별적으로 각도를 설정하여 매니퓰레이터의 자세를 결정합니다 .
각 축은 독립적으로 동작합니다 .

교시 모드로 눈으로 확인하며 위치를 결정할 때나 , 이동 불가 지점으로부터 복귀할 때 등에 사용합니다 .

조인트 좌표계에서는 각 축의 개별 동작을 통해 매니퓰레이터를 움직이기 때문에 ,「자세」(Posture 파라메터) 는 사용하지 않습니다 .

☞ p. 4

월드 좌표계



매니퓰레이터를 설치한 공간의 임의의 점을 원점으로 하여 , Tool center point 의 위치와 방향을 나타내는 직교 좌표계입니다 .

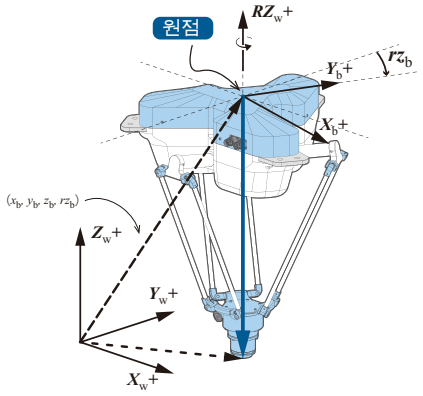
여러 매니퓰레이터를 배치하는 시스템에서 기준이 되는 절대적인 좌표계입니다 .

초기 설정에서 월드 좌표계와 베이스 좌표계는 동일하게 설정되어 있습니다 .

☞ p. 5

1 좌표계 체계

베이스 좌표계

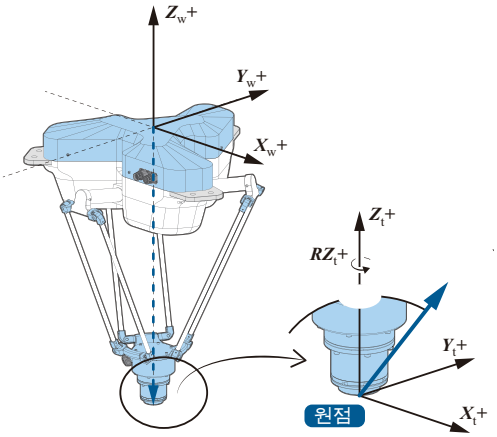


프레임 마운트 바닥면의 중심을 원점으로 하여 , Tool center point 의 위치와 방향을 나타내는 직교 좌표계입니다 .

초기 설정에서 월드 좌표계와 베이스 좌표계는 동일하게 설정되어 있습니다 .

☞ p. 6

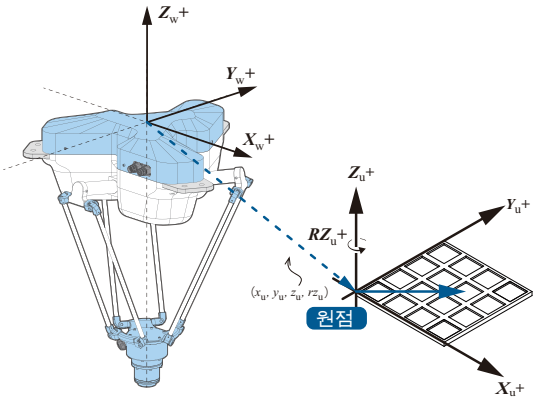
Tool 좌표계



Tool center point 의 중심을 원점으로 하는 직교 좌표계입니다 .

☞ p. 7

유저 좌표계



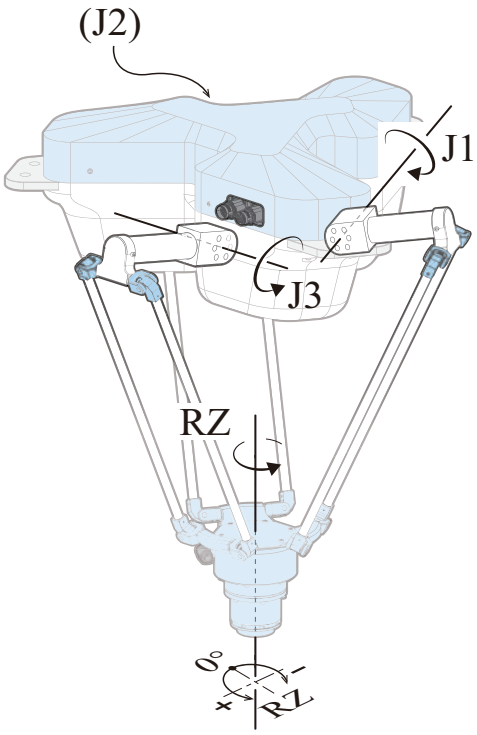
베이스 좌표계를 응용하여 , 임의로 오프셋을 적용한 위치를 원점으로 한 직교 좌표계입니다 .
팔레트 작업 등에서 사용합니다 .

☞ p. 8

2. 조인트 좌표계 (기준 좌표)

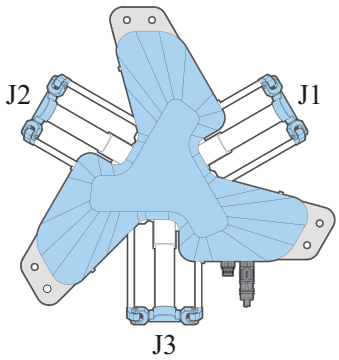
1. 정의

조인트 좌표계는 각 조인트 (J1, J2, J3, J4) 의 각도를 나타내는 좌표계입니다 .
조인트마다의 동작이 가능하며 , 개별적으로 각도를 제어합니다 .
「자세」(Posture 파라메터) 를 사용하지 않습니다 .



왼쪽 그림의 자세 (원점자세)

축	각도
J1	0°
J2	0°
J3	0°
Rz	0°



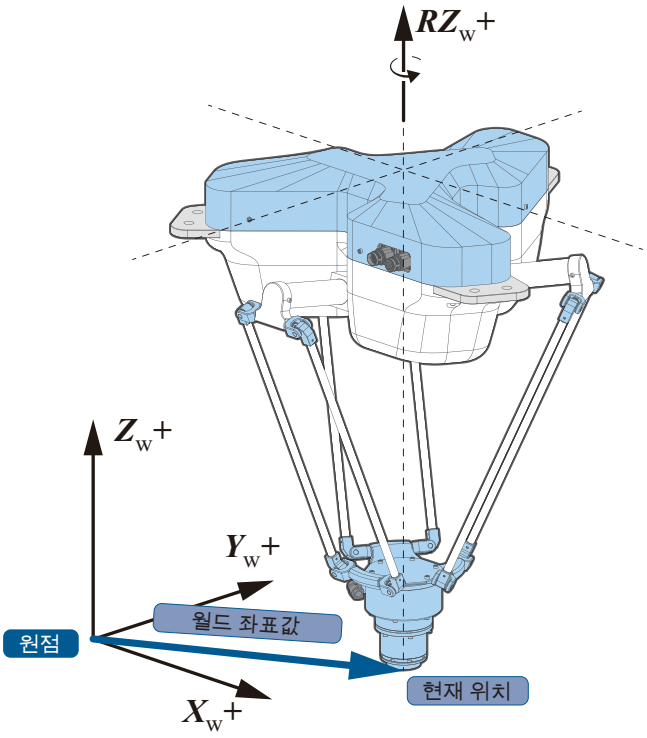
기능	사양		
기준 좌표계	조인트 좌표계		
현재 위치 정보	각 축의 각도 (J1, J2, J3, J4)		
교시 데이터	200 포인트 교시 포인트 10 개를 1 개의 패턴으로 하여 , 최대 20 개의 패턴 사용 가능 (Joint 형 교시 포인트 : joint1[0] ~ [9] … joint20[0] ~ [9])		
JOG 동작	가능		
Direct Move	Joint 형의 교시 데이터를 통해 이동 명령		
Hand Homing	가능		
2 점 간 이동	동작	로봇 프로그램	교시
	PTP	move( ) reljntmove()	Move To
	Line (직선 보간)	line( )	
	Optline (최적 직선 보간)	optline( )	불가

3. 월드 좌표계 (기준 좌표)

1. 정의

월드 좌표계는 매니퓰레이터를 설치한 공간의 임의의 점을 원점으로 하여 , Tool center point 의 위치와 방향을 나타내는 직교 좌표계입니다 .

초기 설정에서 월드 좌표계와 베이스 좌표계는 동일하게 설정되어 있습니다 .



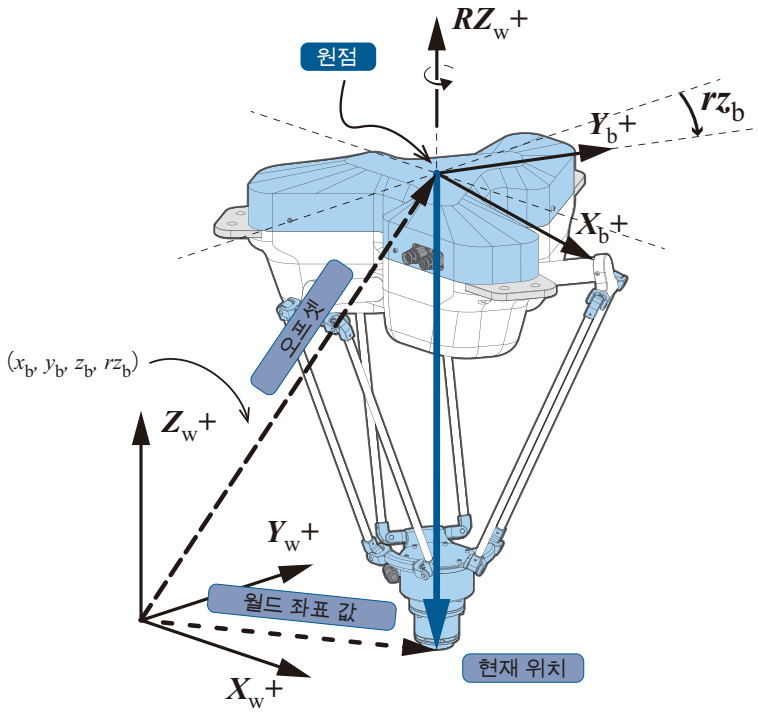
기능	사양		
기준 좌표계	월드 좌표계		
현재 위치 정보	월드 좌표계의 원점에서 본 Tool 말단의 Tool center point 면의 좌표 값 (x, y, z, rz)		
교시 데이터	200 포인트 교시 포인트 10 개를 1 개의 패턴으로 하여 , 최대 20 개의 패턴 사용 가능 (Position 형 교시 포인트 : pos1[0] ~ [9] … pos20[0] ~ [9])		
JOG 동작	가능		
Direct Move	월드 좌표계에서 직교 좌표계의 교시 데이터를 통한 이동 명령		
Hand Homing	불가		
2 점 간 이동	동작	로봇 프로그램	교시
	PTP	move ( )	Move To
	Line (직선 보간)	line ( ) relline( )	
	Optline (최적 직선 보간)	optline( )	불가

4. 베이스 좌표계

1. 정의

베이스 좌표계는 프레임 마운트 바닥면의 중심을 원점으로 한 직교 좌표계입니다 . 월드 좌표계로부터 매니퓰레이터의 설치 상태에 맞추어 베이스 오프셋을 설정합니다 .

베이스 오프셋은 (xb, yb, zb, rzb) 으로 설정합니다 .
초기 설정의 오프셋은 (xb, yb, zb, rzb)=(0, 0, 0, 0) 으로 , 베이스 좌표계의 원점은 월드 좌표계의 원점과 동일합니다 .



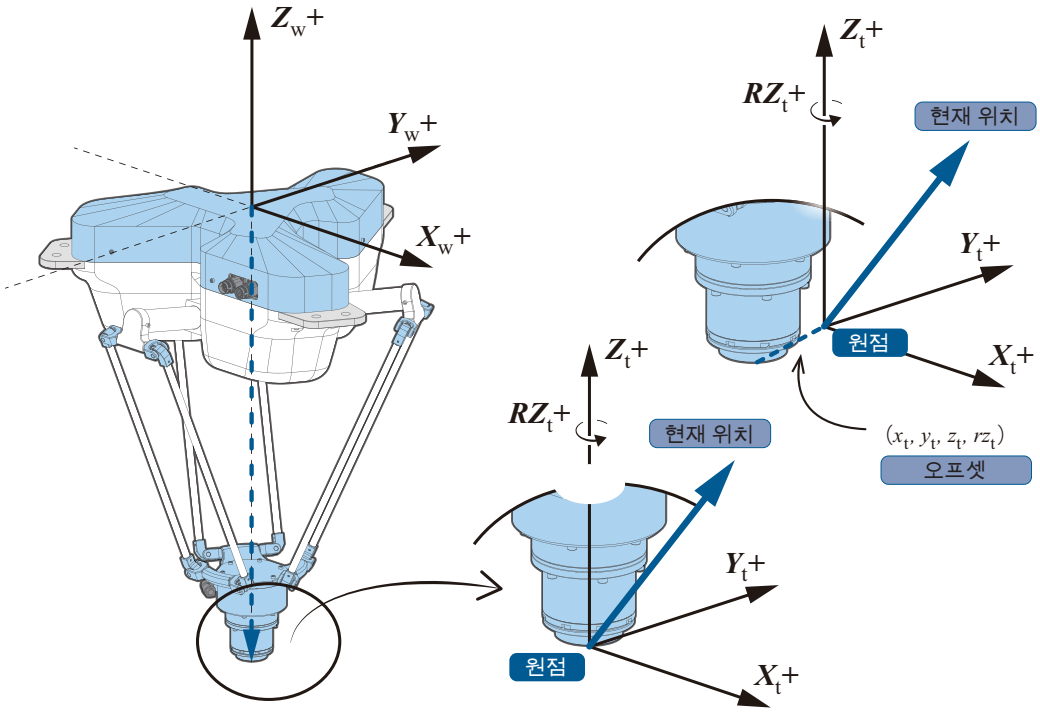
기능	사양		
기준 좌표계	월드 좌표계		
오프셋	3 개까지 설정 가능 (초기 설정의 오프셋은 (x_b, y_b, z_b, rz_b) = (0, 0, 0, 0) 입니다 .)		
현재 위치 정보	월드 좌표 값으로부터 베이스 오프셋 값을 제외한 , Tool 말단의 Tool center point 면의 좌표 (x, y, z, rz,)		
교시 데이터	200 포인트 교시 포인트 10 개를 1 개의 패턴으로 하여 , 최대 20 개의 패턴 사용 가능 (Positon 형 교시 포인트 : pos1[0] ~ [9] … pos20[0] ~ [9])		
JOG 동작	가능		
Direct Move	베이스 좌표계에서 직교 좌표계의 교시 데이터를 통한 이동 명령		
Hand Homing	불가		
2 점 간 이동	동작	로봇 프로그램	교시
	PTP	move ( )	Move To
	Line (직선 보간)	line ( ) relline( )	
	Optline (최적 직선 보간)	optline( )	불가

5. Tool 좌표계

1. 정의

Tool 좌표계는 Tool center point 의 중심을 원점으로 한 좌표계입니다 . Tool 말단을 기준으로 합니다 .
설치하는 Tool 에 따라 오프셋 (xt, yt, zt, rzt) 을 설정합니다 .

Tool 좌표계는 월드 (베이스) 좌표계와 방향이 다르므로 주의하시기 바랍니다 .

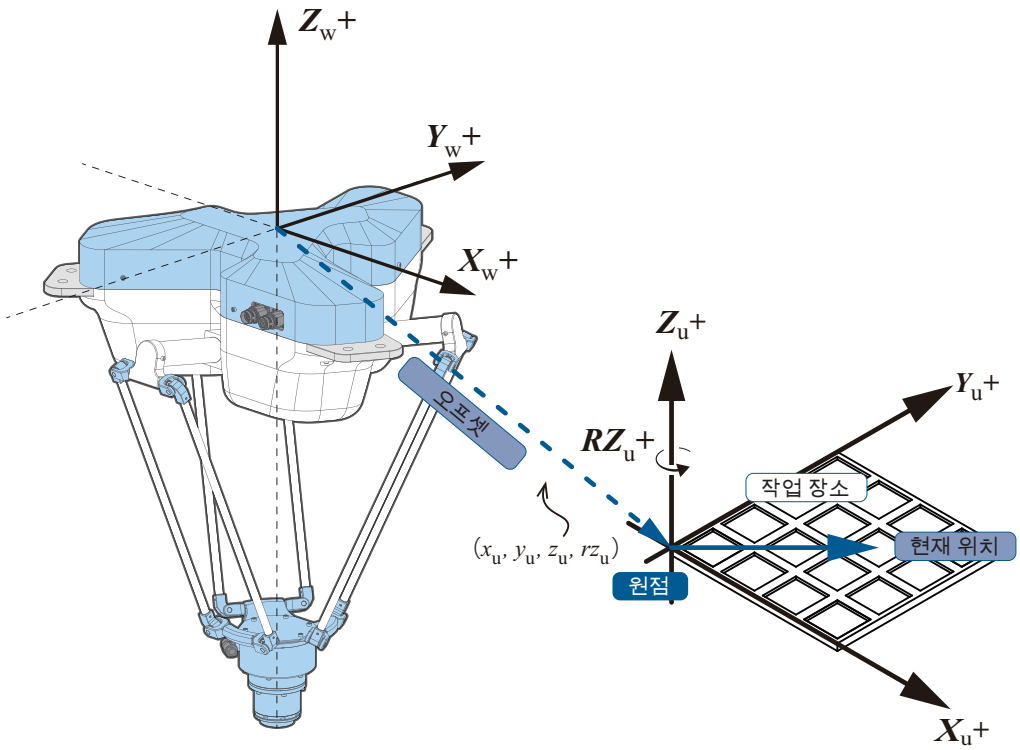


기능	사양		
기준 좌표계	월드 좌표계		
오프셋	8 개까지 설정 가능		
현재 위치 정보	Tool 좌표계의 원점에서 본 좌표 값 (x, y, z, rz)		
교시 데이터	200 포인트 교시 포인트 10 개를 1 개의 패턴으로 하여 , 최대 20 개의 패턴 사용 가능 Position 형 교시 포인트 : pos1[0] ~ [9] … pos20[0] ~ [9])		
JOG 동작	가능		
Direct Move	Tool 좌표계에서 직교 좌표계의 교시 데이터를 통한 이동 명령		
Hand Homing	불가		
2 점 간 이동	동작	로봇 프로그램	교시
	PTP	move ( )	Move To
	Line (직선 보간)	line ( ) relline( )	
	Optline (최적 직선 보간)	optline( )	불가

6. 유저 좌표계

1. 정의

유저 좌표계는 작업 장소에 맞추어 사용자가 작업하기 편하도록 정의한 좌표계입니다 . 매니퓰레이터를 팔레트 등을 따라서 동작시킬 수 있습니다 .

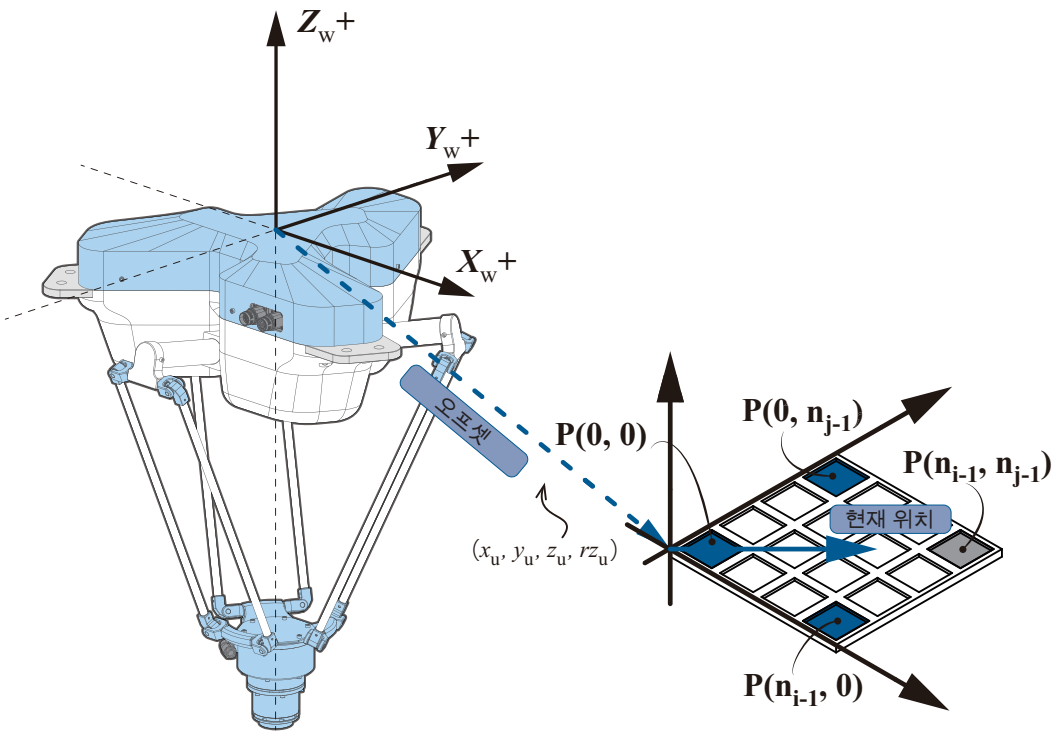


기능	사양		
기준 좌표계	월드 좌표계		
오프셋	3 개까지 설정 가능		
현재 위치 정보	베이스 좌표 값으로부터 유저 오프셋 값을 제외하고 , 유저 좌표계의 원점에서 본 좌표 (x, y, z, rz)		
교시 데이터	200 포인트 교시 포인트 10 개를 1 개의 패턴으로 하여 , 최대 20 개의 패턴 사용 가능 (Position 형 교시 포인트 : pos1[0] ~ [9] … pos20[0] ~ [9])		
JOG 동작	가능		
Direct Move	유저 좌표계에서 직교 좌표계의 교시 데이터를 통한 이동 명령		
Hand Homing	불가		
2 점 간 이동	동작	로봇 프로그램	교시
	PTP	move ( )	Move To
	Line (직선 보간)	line ( ) relline( )	
	Optline (최적 직선 보간)	optline( )	불가

2. 팔레트 연산 기능

n_i 행× n_j 열에 의해 완성된 사각형 팔레트에 대해서 , 네 모퉁이의 유저 좌표 값 (x, y, z, rz) 과 팔레트의 칸의 개수를 지정함으로써 , 각 팔레트 칸의 좌표 (x, y, z, rz) 를 자동 산출합니다 .

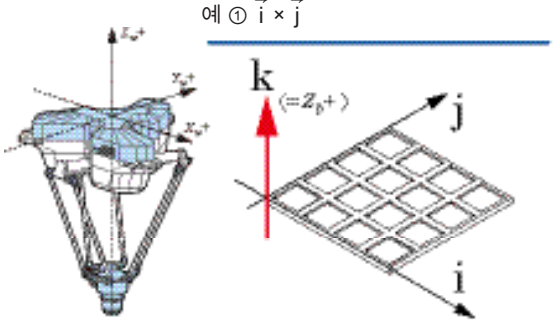
매니퓰레이터는 팔레트 좌표 P(i, j) (0<=i<= n_{i-1}、0<=j<=n_{j-1}) 로 동작합니다 .
최소한 P(0, 0)、P(n_{i-1}, 0)、P(0, n_{j-1}) 의 3 점을 교시합니다 . 4 번째 교시 포인트 P(n_{i-1}, n_{j-1}) 는 사용하는 팔레트의 형상 오차를 보정하기 위해 사용합니다 .



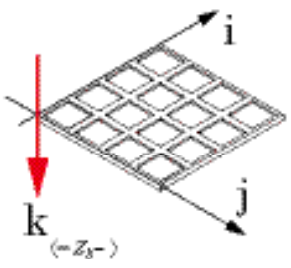
포인트 !

교시 포인트의 순서에 따라 팔레트 수직 방향의 정방향이 바뀝니다 .
예① $\vec{i} \times \vec{j}$ 의 경우 : 평면 (팔레트면) 에 대해 수직인 벡터 $+\vec{z}$가 됩니다 .
예② $\vec{j} \times \vec{i}$ 의 경우 : 예①과 역방향의 $-\vec{z}$가 됩니다 .

예 ① $\vec{i} \times \vec{j}$



예 ② $\vec{j} \times \vec{i}$



MEMO

D

소프트웨어

1. 프로그래밍 가이드
2. 로봇 라이브러리
3. 메모리 맵
4. 프로그램 실행 단계

MEMO

D 소프트웨어

1 프로그래밍 가이드

1. PC 와 동작 환경

주 의

자동 운전을 하기 전에 충분히 테스트를 수행합니다 .
먼저 저속에서 로봇을 동작시켜 일련의 움직임이 안전하게 작동하는지 확인하고 천천히 운전 속도를 올리고 동작 확인을 하십시오 .

로봇 동작 프로그램은 Python 언어로 작성합니다 .

1. PC

본 제품을 사용하기 위해서는 다음의 장비가 필요합니다 . 이 설명서와 안전 설명서를 참조하여 시스템을 구성하십시오 . 권장 사양과 다른 동작 환경에서 소프트웨어가 작동하지 않을 수 있습니다 .

스펙		
개인용 컴퓨터 (PC)	OS	WindowsR 10 (32bit / 64bit) WindowsR 8/8.1 (32bit / 64bit) WindowsR 7 (32bit / 64bit)
	언어	한국어 , 영어 , 일본어
	CPU	1GHz 이상의 32bit 또는 64bit 프로세서
	메모리	1 기가 바이트 (GB) RAM (32 bit) 또는 2 GB RAM (64 bit)
	하드 디스크 용량	512MB 이상의 공간 필요
	통신 기능	유선 LAN 포트 (권장) USB 포트 (*) (유선 LAN 포트가 없는 경우
디스플레이	해상도	1366 × 768 픽셀 이상
	색상	24 비트컬러 (TrueColor) 이상

*) USB Ethernet 어댑터가 별도로 필요합니다 . (추천 제품 : 버팔로 사의 LUA3-U2-ATX)

2. 필수 소프트웨어

Python

Python2.7 을 준수하고 있습니다 .
Python 언어 및 사양은 일반 참고서나 전문 서적을 참고하십시오 .

텍스트 편집기

Python 로봇 프로그램은 텍스트 편집기에서 작성합니다 . (추천 : VSCode)
문자 코드는 UTF-8, 줄바꿈은 LF 입니다 .

터미널 소프트웨어 (Tera Term)

Telnet 연결로 동작 프로그램을 실행하여 컨트롤러를 제어합니다 .

FTP 클라이언트 소프트웨어 (FFFTP)

PC 와 컨트롤러 간 파일 전송을 합니다 .

웹 브라우저 (Google Chrome)

교시는 웹브라우저 (Google Chrome) 에서 실시합니다 .
교시할 PC 에 설치하십시오 .
(Google Chrome : 61 이상)

2. 프로그래밍 가이드

이 장에서는 모듈이나 메소드 , 함수의 대표적인 사용 예를 간략히 설명하고 있습니다 . " 전체의 흐름 " 또는 " 동작 모델 " 에서 선택하십시오 .

모듈이나 방법의 자세한 내용은 「 2 로봇 라이브러리」를 참조하십시오 .

전체의 흐름에서 찾기

5 페이지

동작 프로그램의 전체를 설명합니다 .
「초기 설정」,「교시 포인트 설정」,「동작 조건 설정」,「동작의 정의」,「종료」각 단계를 자세히 설명하고 있습니다 .

" 동작 모델 " 에서 찾기

6 페이지

실용적인 동작 모델에서 목적에 맞는 운영 프로그램을 추천합니다 .
「기본 동작」에서「팔레트 기능」을 사용하는 동작 프로그램을 기재하고 있습니다 .

Python 로봇 프로그램은 대소문자를 구분합니다 .

2 프로그래밍 가이드

" 전체의 흐름 " 에서 찾기

원하는 프로그램 블록을 선택「1. 로봇 프로그램 작성」

로봇 프로그램의 전체

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ################
    .....
    .....

## 2. 초기 설정② ################
    .....

    .....

## 3. 교시 포인트 설정 ###
    .....

    .....

## 4. 동작 조건 설정 ##############
    .....

    .....

## 5. 로봇 동작의 정의 ###########
    .....

    .....

## 6. 종료 ####################
    .....
```

프로그램 블록

1. 초기 설정① 8 페이지

Python 인터프리터의 경로 지정 및 문자 코드를 지정합니다 . 로봇 라이브러리를 사용하는 모듈을 가져옵니다 .

2. 초기 설정② 9 페이지

객체를 생성합니다 .
오버라이드 기능을 이용하여 로봇의 동작 속도를 제한하는 경우 여기에서 정의합니다 .

3. 교시 포인트 설정 10 페이지

교시 포인트 설정과 조정을 합니다 .
저장되어 있는 교시 데이터를 읽어 사용할 수 있습니다 .
팔레트 기능을 이용하려면 여기에서 정의합니다 .

4. 동작 조건 설정 15 페이지

로봇의 동작 속도와 가속 시간 등을 설정합니다 .
동작 조건은 선택 사항이지만 생략하면 기본값입니다 .

5. 로봇 동작의 정의 16 페이지

로봇의 동작을 설정합니다 .
오버랩 기능과 I/O 입력 신호에 연동한 동작 , 사용중인 좌표계 전환 등이 있습니다 .

6. 종료 26 페이지

로봇 프로그램을 종료합니다 .

2 프로그래밍 가이드

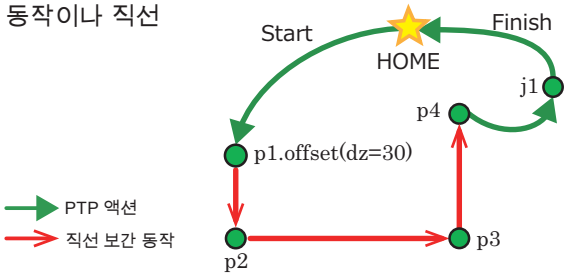
" 동작 모델 " 에서 찾기

원하는 동작 모델을 선택 "2. 샘플 프로그램 "

모델 1 : 기본 동작

27 페이지

교시 포인트를 설정하고 지정된 지점을 향해 PTP 동작이나 직선 보간 동작을 합니다 .



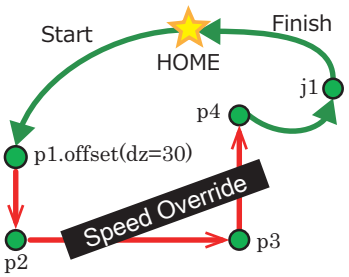
모델 2 : 오버라이드

28 페이지

위의 기본 동작에서 MotionParam () 으로 설정한 동작 속도에 제한을 겁니다 .

오버라이드 (override) 에서 설정한 비율 (%) 로 동작 속도를 제한합니다 .

본 동작 프로그램의 동작 점검을 하는 경우 등에 사용합니다 .



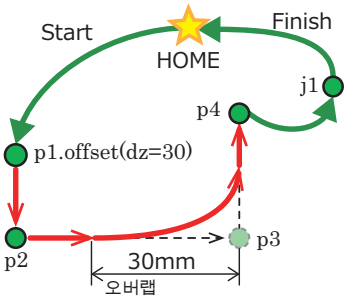
모델 3 : 오버랩

29 페이지

목표 교시 포인트에 접근한 시점에서 다음 움직임을 겹치는 동작을 합니다 .

장애물을 피하기 위해 마련한 경유점 등으로 로봇 동작의 완료를 기다리지 않고 다음 작업을 수행하여 로봇을 움직일 수 있습니다 .

오버랩 양은 임의로 설정할 수 있습니다 .
(오른쪽의 예에서는 30mm 로 설정)

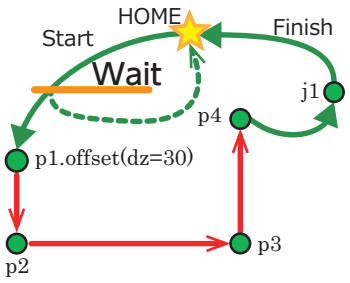


모델 4 : I/O 입력 대기

30 페이지

외부 장치에서 컨트롤러에 입력하는 I/O 신호에 의해 로봇 동작을 제어합니다 .

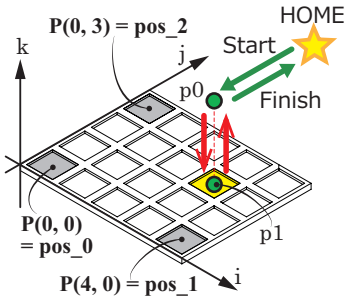
I/O 입력을 사용하면 미리 컨트롤러에 등록된 로봇 프로그램을 I/O 로 시작할 수 있습니다 .



모델 5 : 팔레트 기능

31 페이지

작업 이송 팔레트 셀의 개수와 네 점의 좌표를 정의하면 컨트롤러는 각 셀의 좌표를 자동 계산합니다 .
산출한 팔레트 좌표 (i, j) 를 교시 포인트로 설정합니다 .



2 프로그래밍 가이드

1. 로봇 프로그램 작성

1. 초기 설정①

모듈 가져 오기

Python 인터프리터의 경로 지정과 한글 취급 설정

문자 코드를 지정하지 않고 전각 문자를 사용하면 오류가 발생할 수 있습니다 .

프로그램 예

```
#!/usr/bin/python        ← 인터프리터 지정
# –*– coding: utf–8 –*–  ← 문자 코드 지정
```

모듈 가져오기

각종 모듈 (표준 라이브러리 , 로보틱스 라이브러리 , 고객이 만든 모듈) 을 가져와 로봇을 제어하는 데 필요한 명령을 사용할 수 있습니다 .

모듈	기능
i611_MCS	로봇 제어에 필요한 기본 기능을 사용
teachdata	교시 데이터를 사용
i611_extend	확장 기능을 사용 (팔레트 기능)
rbsys	관리 프로그램을 사용
i611_common	i611Robot 클래스의 메소드에 예외 처리 (*)
i611_io	I/O 신호를 제어
i611shm	공유 메모리에 액세스

```
## 1 . 초기 설정 ①  모듈 가져오기 ######################
    from i611_MCS import *
    from teachdata import *
    from i611_extend import *
    from rbsys import *
    from i611_common import *
    from i611_io import *
    from i611shm import *
```

*) Exception 클래스는 i611_MCS 모듈에서 가져와서 사용할 수 있습니다 .
i611_MCS 모듈에서 from i611_common import * 를 로드하고 있습니다 .

2. 초기 설정②

객체를 생성

로봇 생성자

로봇 객체를 생성합니다 .

```
# i611 로봇 생성자
    rb = i611Robot( )
```

월드 좌표계 정의

월드 좌표계를 사용할 때 설정합니다 .

```
    # 월드 좌표계 정의
    _BASE = Base( )
```

로봇과의 연결 시작 , 초기화

```
    # 로봇과 연결 시작 초기화
    rb.open( True )
```

I/O 입출력 기능을 초기화

```
    # I/O 입출력 기능을 초기화 (I/O 미사용시 생략 가능 )
    IOinit( )
```

오버라이드

PTP 동작 , Joint 동작에 동작 속도의 비율 (%) 을 설정합니다 .

```
# 속도 오버라이드 50%
    rb.override( 50 )
```

```
## 2 . 초기 설정② : 동작 조건 설정 ####################################
    # i611 로봇 생성자
        rb = i611Robot( )
    # 월드 좌표계 정의
        _BASE = Base( )
    # 로봇과 연결 시작 , 초기화
        rb.open( True )
    # I/O 입출력 기능을 초기화 (I/O 미사용시 생략 가능 )
        IOinit( )
    # 속도 오버라이드 50%
        rb.override( 50 )
```

2 프로그래밍 가이드

3. 교시 포인트 설정

교시 포인트 정의

Position( )

월드 좌표계의 교시 포인트를 정의하는 인스턴스를 생성합니다 .

인수	의미	변수 형태	단위
x, y, z	위치 (직교 좌표계)	float	mm
rz, ry, rx	자세 (Z-Y-X 계 오일러 각도)	float	deg
parent	월드 좌표계를 사용하는 설정	float	-
posture	자세	integer	-
multiturn	크로스 오버 카운터 정보	long	-

Joint( )

조인트 좌표계의 교시 포인트를 정의하는 인스턴스를 생성합니다 .

인수	의미	변수 형태	단위
j1, j2, j3, j4, j5, j6	Joint 형의 각 축 데이터 (초기값 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0])	float	deg

```
## 3 . 교시 포인트 설정 ###################################
        p1 = Position( 195, -100, -50, -120, posture=1 )
        p2 = Position( 115, -100, -70, 154 )
        p3 = Position( 130, -90, -100, 159 )
        j1 = Joint( 95, –30, –40, 90, 0, 0 )
```

교시 포인트는 Position 형 또는 Joint 형으로 설정합니다 .

스카라 로봇의 경우 , 처음 임의의 지점으로 이동할 때 , posture 값을 0 혹은 1 로 지정해줘야 합니다 .

좌표 표기법

6 축 수직 다관절 로봇 , 스카라 로봇 , 델타 로봇은 같은 라이브러리를 사용합니다 .

스카라 로봇 , 델타 로봇과 같이 구동축이 6 축 미만인 경우 , 해당 축 혹은 해당 좌표까지만 사용하고 그 뒤의 값들은 무시됩니다 . 4 축인 경우 , ry, rx 값들을 사용하지 않으므로 , 아래와 같이 2 가지 방법 모두 가능하며 , 본 설명서에서는 첫 번째 방법을 기준으로 합니다 .

1) Position = (195, –100, -50, -120, 0, 30)

2) Position = (195, –100, -50, -120)

인수의 생략

인수는 지정하려는 매개 변수까지 입력합니다 .

(예)

rz 부터 인수를 생략하고 p = Position (x, y, z) 로 한 경우

rz 부터의 매개 변수는 초기값으로 설정됩니다 .

교시 포인트를 조정

replace( )

월드 좌표계 객체를 대체합니다 .
(원본 객체를 갱신)

인수	의미	변수 형태	단위
x, y, z	위치 (월드 좌표계) (초기값 = 0.0)	float	mm
rz, ry, rx	자세 (Z-Y-X 계 오일러 각도) (초기값 = 0.0)	float	deg
parent	월드 좌표계를 사용하는 설정	float	-

```
# p1 값을 바꾸고 자신을 업데이트
p1 = Position( 195, -100, -50, -120, posture=1 )
p1.replace( x=100, rz=50 )
```

실행결과
[295, –100, –50, –70]

shift( )

월드 좌표계 객체를 이동합니다 .
(원본 객체를 갱신)

인수	의미	변수 형태	단위
dx, dy, dz	위치 (월드 좌표계)	float	mm
drz, dry, drx	자세 (Z-Y-X 계 오일러 각도)	float	deg

```
# p1 값을 이동하고 자신을 업데이트
p1 = Position( 195, -100, -50, -120, posture=1 )
p1.shift( dx=10 )
```

실행결과
205, –100, –50, -120)]

offset( )

Position 좌표값에 오프셋 좌표값을 추가합니다 .
(원본 객체를 유지하면서 새로운 객체를 생성)

인수	의미	변수 형태	단위
dx, dy, dz	위치의 오프셋량 (직교 좌표계)	float	mm
drz, dry, drx	자세의 오프셋 (Z-Y-X 계 오일러 각도)	float	deg

```
# p1 을 유지하면서 오프셋 p2 를 생성합니다
p1 = Position( 195, -100, -50, -120, posture=1 )
p2 = p1.offset( dx=10 )
```

실행결과
p1 = [195, –100, –50, -120]
p2 = [205, –100, –50, -120]

파일에 저장된 교시 데이터를 이용

Teachdata( ) 교시 데이터를 읽어 Teachdata 클래스의 인스턴스를 생성합니다 .

인수	의미	변수 형태	단위
fname	교시 데이터의 파일 이름	string	-

get_position( ) 교시 데이터 Position 좌표값을 가져옵니다 .

인수	의미	변수 형태	단위
key	Position 좌표 키 이름 필 수	string	-
index	Position 좌표의 인덱스 필 수	integer	-
tool	tool ID 취득 플래그	bool	-
base	base ID 취득 플래그	bool	-
comment	comment 의 취득 플래그	bool	-

get_joint( ) 교시 데이터 Joint 좌표값을 가져옵니다 .

인수	의미	변수 형태	단위
key	Joint 좌표 키 이름 필 수	string	-
index	Joint 좌표의 인덱스 필 수	integer	-
comment	comment 의 취득 플래그	bool	-

```
# 교시 데이터 파일 읽기
data = Teachdata( "teach_data" )
# 교시 포인트 읽기
p1 = data.get_position( "pos1", 0 )   "pos1" 인덱스 [0] 의 포지션 형 데이터를 로드
j1 = data.get_joint( "joint1", 0 )    "joint1" 인덱스 [0] 의 조인트 형 데이터를 로드
```

get_param( ) 교시 데이터의 매개 변수를 가져옵니다 .

인수	의미	변수 형태	단위
key	매개 변수의 키 이름 필 수	string	-
index	파라미터의 인덱스 필 수	integer	-
axis	매개 변수의 축 번호 필 수	integer	-
comment	매개 변수의 comment 취득 플래그	bool	-

팔레트 기능을 이용하기

init_3( ) 팔레트를 정의합니다 . (3 점 교시)

인수	의미	변수 형태	단위
pos_0	[Position] 팔레트에 교시 포인트 (원점) 필 수	float	-
pos_i	[Position] 팔레트에 교시 포인트 (i 방향) 필 수	float	-
pos_j	[Position] 팔레트에 교시 포인트 (j 방향) 필 수	float	-
ni	팔레트의 i 방향으로 줄 지어있는 셀의 개수 필 수	integer	-
nj	팔레트의 j 방향으로 줄 지어있는 셀의 개수 필 수	integer	-

```
# 3 점 교시 데이터를 사용하여 팔레트를 정의
pal = Pallet()
pal.init_3( pos_0, pos_1, pos_2, 5, 4 )
```

init_4( ) 팔레트를 정의합니다 . (4 점 교시)

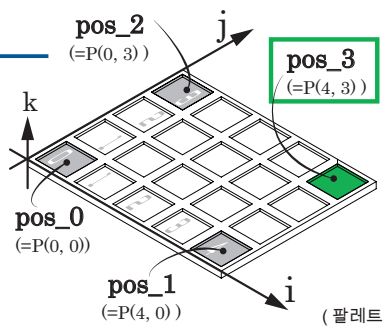
인수	의미	변수 형태	단위
pos_0	[Position] 팔레트에 교시 포인트 (원점) 필 수	float	-
pos_i	[Position] 팔레트에 교시 포인트 (i 방향) 필 수	float	-
pos_j	[Position] 팔레트에 교시 포인트 (j 방향) 필 수	float	-
pos_ij	[Position] 팔레트에 교시 포인트 필 수	float	-
ni	팔레트의 i 방향으로 줄 지어있는 셀의 개수 필 수	integer	-
nj	팔레트의 j 방향으로 줄 지어있는 셀의 개수 필 수	integer	-

```
# 4 점 교시 데이터를 사용하여 팔레트를 정의
pal = Pallet()
pal.init_4( pos_0, pos_1, pos_2, pos_3, 5, 4 )
```

포인트 !

init_3 () 와 init_4 () 의 차이

init_4 () 는 4 점째의 교시 포인트를 설정하여 사용하는 팔레트의 형상 오차를 보정합니다 .
init_3 () 에 비해 정밀하게 로봇을 움직일 수 있습니다 .



init_4 () 로 설정합니다 . 팔레트 모양을 보정하기 위한 교시 포인트입니다 .

(팔레트 크기 : 5x4 의 예)

팔레트 기능을 이용하기

get_pos( ) 셀 위치를 가져옵니다 .

인수	의미	변수 형태	단위
i	팔레트에서 셀 위치를 지정하는 인덱스 (i 방향) 필 수	integer	-
j	팔레트에서 셀 위치를 지정하는 인덱스 (j 방향) 필 수	integer	-
dk	수직 방향의 오프셋 (생략하면 기본값 : 0)	integer	mm

adjust( ) 팔레트의 셀 위치를 보정합니다 .

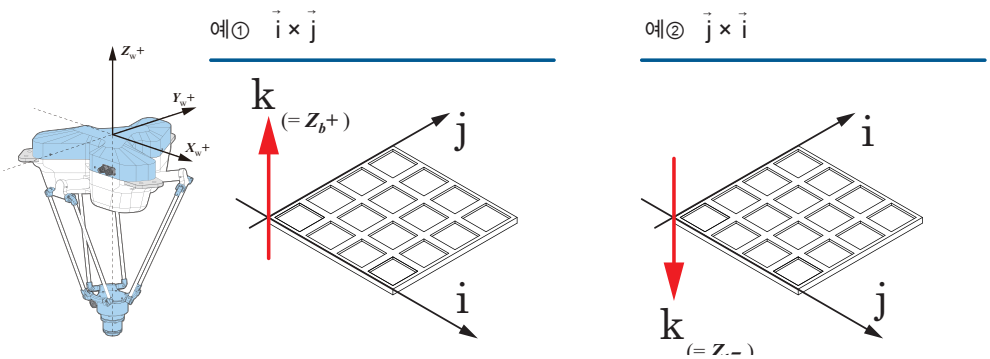
인수	의미	변수 형태	단위
i	팔레트에서 셀 위치를 지정하는 인덱스 (i 방향) 필 수	integer	-
j	팔레트에서 셀 위치를 지정하는 인덱스 (j 방향) 필 수	integer	-
di	i 방향 셀 위치의 오프셋량 필 수	integer	mm
dj	j 방향 셀 위치의 오프셋량 필 수	integer	mm

포인트 !

팔레트의 수직 방향

교시 포인트의 배치에 따라 팔레트 수직 방향 (+k 방향) 이 바뀝니다 .

예 ① i × j : 팔레트 평면에 상승 수직인 벡터 z + 입니다 .
예 ② j × i : 팔레트 평면에 아래쪽 수직인 벡터 z - 입니다 .



4. 동작 조건 설정

로봇의 동작 파라미터를 설정하기

MotionParam( ) 로봇의 동작 파라미터 클래스의 인스턴스를 만듭니다 .

motionparam( ) 동작 파라미터를 설정합니다 .

인수	의미	변수 형태	단위
lin_speed	속도 (Line 동작 (직선 보간 동작)) 초기값 : 5.0	float	mm/s
jnt_speed	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작) 초기값 : 5.0	float	%
acctime	가속 시간 초기값 : 0.4	float	s
dacctime	감속 시간 초기값 : 0.4	float	s
posture	자세 초기값 : 0	integer	–
passm	경로 동작 초기값 : 2	integer	–
overlap	오버랩 동작 초기값 : 0.0	float	mm
zone	위치 결정 완료 범위 초기값 : 100	integer	pulse
pose_speed	속도 (자세 보간 동작) 초기값 : 20	float	%
ik_solver_option	회전방향 초기값 : 0x11111111	long	–

인수를 생략하면 기본값이 설정됩니다 .

```
## 4 . 동작 조건 설정 ##################################
    #MotionParam 생성자에서 동작 조건 설정
    m = MotionParam( jnt_speed=10, lin_speed=70 )
    #MotionParam 형태로 동작 조건 설정
    rb.motionparam( m )
```

2 프로그래밍 가이드

5. 로봇 동작의 정의

로봇을 이동

home( ) 모든 축을 Joint 좌표 0deg 로 이동합니다 .

move( ) PTP 동작을 일정한 속도로 이동합니다 . (*)

line( ) 직선 보간 동작을 일정한 속도로 이동합니다 . (*)

optline( ) 직선 보간 동작을 최적의 속도로 변속하면서 움직입니다 .

*) 메소드 실행 직후에는 motionparam 메소드에서 설정한 동작 파라미터로 동작합니다 .
이 메소드의 인수 안에서 동작 파라미터가 주어진 경우 , 이후의 동작을 변경합니다 .

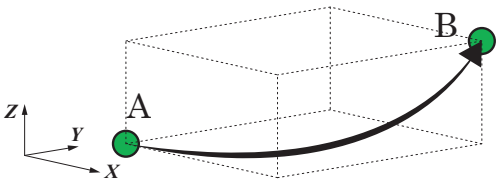
```
## 5. 로봇 동작 설정 ###################################
    # 각 축 좌표 [0, 0, 0, 0] 로 이동
    rb.home()
    # 이동
    rb.move( p1.offset(dz=30) )   ← PTP 동작으로 p1 대해 dz = 30 만큼 오프셋한 좌표로 이동
    rb.line( p2, p3, p4 )         ← Line 동작에서 p2, p3, p4 로 이동
    rb.move( j1 )
    # 각 축 좌표 [0, 0, 0, 0] 로 이동
    rb.move( j1 )                 ← PTP 동작으로 j1 로 이동
```

포인트 !

PTP 동작 , 직선 보간 동작과 최적 직선 보간 동작

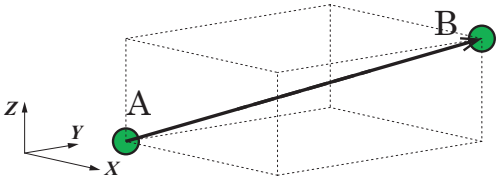
PTP 동작 (move())

모든 관절이 목표 좌표를 향해 일정한 속도와 각도로 동작합니다 . 부드러운 곡선을 그리며 이동하는 동작 .



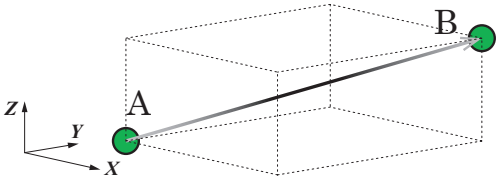
직선 보간 동작 (line())

X-Y-Z 축 동기 제어하면서 목적지까지의 궤적이 직선이 되도록 일정한 속도로 이동하는 동작 .



최적 직선 보간 동작 (optline ())

X-Y-Z 축 동기 제어하면서 목적지까지의 궤적이 직선이 되도록 최적의 속도로 변속하면서 이동하는 동작 .
속도는 % 로 지정합니다 .



도구 오프셋 이용

settool( ) 도구 오프셋을 설정합니다 .

인수	의미	변수 형태	단위
id	도구 번호 필 수 0 : 도구 오프셋 해제 1 - 8 : 도구 오프셋 선택	integer	-
offx	도구 좌표계의 X 축 도구 오프셋	float	mm
offy	도구 좌표계의 Y 축 도구 오프셋	float	mm
offz	도구 좌표계의 Z 축 도구 오프셋	float	mm
offrz	도구 좌표계에서의 Rz 축 주위의 오프셋	float	deg
offry	도구 좌표계에서의 Ry 축 주위의 오프셋	float	deg
offrx	도구 좌표계에서의 Rx 축 주위의 오프셋	float	deg

changetool( ) 도구 오프셋을 선택합니다 .

인수	의미	변수 형태	단위
tid	도구 번호 필 수 0 : 공구 오프셋 해제 1 - 8 : 도구 오프셋 설정	integer	-

```
# 도구 번호 = 1 도구 등록
rb.settool( 1, 0.0, 0.0, –100.0, 0.0, 0.0, 0.0 )

# 도구 #1 로 변경
rb.changetool( 1 )
```

도구 번호의 인수 이름은 changetool() 메소드와 settool() 메소드에서 다릅니다 .

메소드	도구 번호의 인수 이름
changetool()	tid
settool()	id

2 프로그래밍 가이드

I/O 입력과 출력

din( )

I/O 를 입력합니다 .

인수	의미	변수 형태	단위
*adr	입력 포트 필 수 · 1 개의 입력 포트를 지정하는 경우 adr : 입력 포트 번호 · 연속된 여러 입력 포트를 동시에 판독하는 경우 adr [0] : 입력 포트 번호 (시작) adr [1] : 입력 포트 번호 (종료)	string	-

```
# 예 1 : 포트 15 을 지정
if din ( 15 ) == '1':

# 예 2 : 포트 8 포트 10 을 지정
if din ( 8, 10 )[0] == '1':  # 포트 10 을 지정하는 경우
   …
elif din( 8, 10 )[1] == '1': # 포트 9 을 지정하는 경우
…
elif din( 8, 10 )[2] == '1': # 포트 8 을 지정하는 경우
```

dout( )

I/O 를 출력합니다 .

인수	의미	변수 형태	단위
adr	출력 포트 번호 주소 시작 번호 필 수 (설정 범위 : 16 ~ 31)	integer	-
data	I/ O 에서 출력하는 데이터 필 수 문자열의 비트 필드로 설정합니다 . '1'= ON '0'= OFF (초기값) '*'= 변화시키지 않음	string	-

```
# 시작 주소와 출력 데이터의 ON / OFF 를 지정
dout( 16, '11111' )
```

포트 번호에 대한 자세한 정보는 " 메모리 맵 " 을 참조하십시오 .

비트 필드에서 포트 설정

dout (), dlyput (), shotOut (), wait () 메소드의 data 부분은 비트 필드 형식의 문자열입니다 .

예) 출력 포트 16 - 31 의 설정

포트 31 · · · · · · · 포트 16

dout(16, "10001010****1111")

포트 16 에서 설정

포트 31 을 1 로 설정

포트 16 을 1 로 설정

포트 20 에서 23 을 변화시키지 않는다 .

I/O 입력 대기

wait( )

지정한 I/O 입력 패턴이 될 때까지 기다립니다 .

인수	의미	변수 형태	단위
adr	입력 포트 시작 번호 필 수	integer	-
data	입력 대기할 데이터를 지정 필 수 "1" = ON "0" = OFF	string	-
tm	제한 시간 필 수	float, integer	s

```
# 예 1 : 리스트
    # 8 번 포트에 ON 값이 들어올 때까지 10 초간 대기하였을 때 , 조건을 만족하였을 경우
    if wait( 8, '1', 10 )[0] ==  1:
    # 9 번 포트에 ON 값이 들어올 때 까지 10 초간 대기하였을 때 , 입력이 ON 인 경우
    if wait( 9, '1', 10 )[1] ==  '1':
    # 9 번 포트에 ON 값이 들어올 때 까지 10 초간 대기하였을 때 , 10 초 이상 경과하였을 경우
    if wait( 9, '1', 10 )[2] > 10:

# 예 2 : 키워드
    if wait( adr=1, data='1', tm=10 ) == 1:
```

⚠ 주 의

init.py 에 미리 설정되어있는 포트는 변경하지 마십시오 .

(오동작)

I/O 입력에 따라 로봇 프로그램을 시작

컨트롤러 I/O 입력에 따라 미리 등록된 로봇 프로그램을 시작할 수 있습니다 .

단계 1 로봇 프로그램 및 설정 스크립트를 만들고 컨트롤러로 전송합니다 .

설정 스크립트의 파일 이름과 위치

파일 이름	init.py
저장 위치	/home/i611usr

파일 이름과 위치는 변경하지 마십시오 .

컨트롤러는 설정 스크립트 예제가 포함되어 있습니다 .
PC 에 다운로드하여 원하는 항목을 수정하여 사용하십시오 .

단계 2 컨트롤러의 전원을 재투입하고 시스템을 다시 시작합니다 .

(시스템이 시작할 때 "init.py" 가 실행되고 I/O 시작 조건이 설정됩니다 .)

· Safety 커넥터에 점퍼 커넥터가 연결되어 있는지 확인하십시오 .

· 7 세그먼트 표시기에 rdy 가 나타나면 활성화 스위치를 누릅니다 .

단계 3 설정한 I/O 포트에 입력 신호를 켭니다 .

(신호의 상승 엣지에서 로봇 프로그램이 실행을 시작합니다 .)

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

  from rbsys import Robsys


#This is sample program for initial settings.
  if __name__ == '__main__':

        rbs = RobSys()
        rbs.open()

    #Default assignment
     rbs.assign_din( run=0, stop=1, err_reset=2, pause=3 )
     rbs.assign_dout( running=16, svon=17, emo=18, hw_error=19,
                  sw_error=20, abs_lost=21, in_pause=22, error=23 )
    #rbs.set_robtask( "sample.py" )

    rbs.close()
```

시작하려는 프로그램의 파일 이름
(파일 이름은 임의)

포트	상태 이름	명령
0	run	로봇 프로그램 실행
1	stop	감속 정지
2	err_reset	오류 재설정
3	pause	일시 정지

포트	상태 이름	시스템 상태
16	running	로봇 프로그램 상태
17	svon	서보 상태
18	emo	비상 정지 상태
19	hw_error	시스템 정의 오류 (치명적) 상태
20	sw_error	시스템 정의 오류 상태
21	abs_lost	ABS 소실 상태
22	in_pause	일시 정지 상태
23	error	시스템 오류 상태

명령 및 시스템 상태는 임의의 I/O 포트에 할당할 수 있습니다 .

위 험

출력 신호는 안전상 중요한 목적으로는 사용하지 마십시오 .
소프트웨어만으로 처리하기 때문에 안전 회로에 요구되는 신뢰성을 보장할 수 없습니다 .

포인트 !

init.py 에 정의된 초기 설정 포트

신호 및 포트 번호		의미
입력	run=0	로봇 프로그램 실행
	stop=1	감속 정지
	err_reset=2	오류 재설정
	pause=3	일시 정지
출력	running=16	로봇 프로그램 상태
	svon=17	서보 상태
	emo=18	비상 정지 상태
	hw_error=19	시스템 정의 오류 (치명적) 상태
	sw_error=20	시스템 정의 오류 상태
	abs_lost=21	ABS 소실 상태
	in_pause=22	일시 정지 상태
	error=23	시스템 오류 상태 (*)

*) 시스템 정의 오류 (비치명적 또는 치명적) 의 발생 상태입니다 .
2 개의 오류 상태를 1 개의 제어선으로 확인하는 경우에 사용합니다 .

주 의

활성화 스위치를 누르기 전에 매니퓰레이터의 가동 범위 내에 장애물이 없는지 확인하고 주위의 안전을 확보하십시오 .

포인트 !

보충

· 로봇 프로그램이 오류 종료한 경우 :
. . . . . 오류를 재설정할 때까지 프로그램을 재시작할 수 없습니다 .
· " 프로그램 동작중 ' 과 ' 오류 발생 " 상태를 I/O 에 출력하는 기능 :
. . . . . 로봇 프로그램을 I/O 입력에서 시작했을 때만 유효합니다 .
. . . . . (PC 와 같은 터미널 에뮬레이터 프로그램을 통해 시작하면 동작하지 않습니다 .)

좌표 변환

Joint2Position( )

Joint 좌표값에서 Position 좌표값으로 변환합니다 .

인수	의미
Joint 형	리스트 형식의 각 축 각도 필 수

```
#Joint 좌표값
    j10=Joint( 0, 30, -60, 0, 0, 0 )

#Position 좌표값으로 변환 (j10 → 변환 → p10)
p10=rb.Joint2Position( j10 )
```

Position2Joint( )

Position 좌표값에서 Joint 좌표값으로 변환합니다 .

인수	의미
Position 형	리스트 형식의 위치 정보 필 수

```
#Position 형 좌표값
    p10=Position( –50, –250, –50, 90, 0, 0 )

    #Joint 형 좌표값으로 변환 (p10 → 변환 → j10)
    j10=rb.Position2Joint( p10 )
```

overrap 동작을 이용

asyncm( ) 로봇 프로그램의 예측 동작 구간을 설정합니다 .

인수	의미	변수 형태	단위
sw	1 : 프로그램 미리 동작 ON 2 : 프로그램 미리 동작 OFF (기본값)	integer	-

오버랩 동작을 설정한 구간에서는 목표 교시 포인트에 접근한 시점에서 다음 동작이 이어지는 동작을 합니다 .

장애물 회피 등의 동작을 하기 위해 준비된 경유 지점들로 , 로봇의 동작 완료를 기다리지 않고 다음 작업을 수행하도록 로봇을 움직일 수 있습니다 .

```
rb.line (p10) # 교시 포인트 p10 에 직선 보간 이동
rb.asyncm (sw = 1) # 프로그램 예측 동작 ON (rb.asyncm (1) 에서도 가능 )
rb.line (p20, p21) # 교시 포인트 p20 과 p21 에 순서대로 직선 보간 동작으로 이동

rb.join () # 예측한 로봇 프로그램의 동작이 완료되기를 기다리는 함수

rb.asyncm (sw = 2) # 프로그램 예측 동작 OFF (rb.asyncm (2) 에서도 가능 )
...
rb.close()
```

오버랩 실행중에 I/O 입출력

오버랩 동작을 사용하면 I/O 입출력 기능 등 로봇의 운동에 관계없이 처리하는것들을 포함하여 모든 명령이 예측됩니다 .

로봇 동작에 동기화된 동작을 수행하기 위해서는 i611Robot 클래스의 join () 메소드를 사용하기 바랍니다 .

로봇 일시 정지

set_behavior( ) 일시 정지 동작 (행동) 을 설정합니다 .

인수	의미	변수 형태	단위
only_hook	user_hook( ) 메소드에서만 일시 정지 True : 유효 False : 무효 (초기값)	bool	-
servo_off	일시 정지 시에 서보를 OFF 로 설정 True : 유효 False : 무효 (초기값)	bool	-
restore_position	일시 정지 후 재개시에 위치를 일시 정지 전으로 돌아가기 True : 유효 False : 무효 (초기값)	bool	-
no_pause	작업 중단 시에만 일시 정지 True : 유효 (시스템 버전 R0.5.0 와 호환) False : 무효 (초기값)	bool	-

```
# 일시 정지 후 다시 시작하면 자세를 일시 정지 전으로 복귀
rb.set_behavior( only_hook=False, servo_off=False, restore_position=True, no_pause=True )
```

enable_interrupt( ) 감속 정지와 비상 정지의 예외 발생을 설정합니다 .

인수	의미	변수 형태	단위
eid	이벤트 ID 필 수 0 : 동작중 감속 정지 입력시 예외 발생 1 : 동작중 비상 정지 입력시 예외 발생 2 : 일시 정지중 감속 정지 입력시 예외 발생 3 : 일시 정지중 비상 정지 입력시 예외 발생 예외 발생을 비활성화한 경우 , 로봇 프로그램을 정상적으로 종료합니다 .	integer	-
enable	예외 발생 필 수 True : 유효 False : 해제	bool	-

```
# 예 1 : 동작중 감속 정지 입력시 예외 발생을 활성화하려면
rb.enable_interrupt( 0, True )

# 예 2 : 동작중 비상 정지 입력시 예외 발생을 활성화하려면
rb.enable_interrupt( 1, True )

# 예 3 : 일시 정지중 감속 정지 입력시 예외 발생을 해제하려면
rb.enable_interrupt( 2, False )

# 예 4 : 일시 정지중 비상 정지 입력시 예외 발생을 해제하려면
rb.enable_interrupt( 3, False )
```

user_hook( ) 로봇 프로그램을 일시 정지시킵니다 .

일시 정지시키는 위치에 사용하십시오 .
set_behavior (only_hook = True) 을 지정하여 user_hook () 에서만 일시 정지할 수 있습니다 . 지정하지 않는 경우에는 로봇 프로그램의 메소드를 일시 정지할 수 있습니다 .

```
...
rb.user_hook()      # 이 위치에서 프로그램을 일시 정지
...
```

cause_user_error( ) 사용자 정의 오류를 발생시킵니다 .

인수	의미	변수 형태	단위
code	오류 ID 필 수 설정 범위 : 1 – 99	integer	-
critical	True : 사용자 정의 오류 치명적 발생 False : 사용자 정의 오류 발생 (초기값)	bool	-

```
# 사용자 정의 오류 ( 오류 ID : 19) 을 발생시키는 경우
rb.cause_user_error (19, False)

# 사용자 정의 오류 치명적 ( 오류 ID : 01) 을 발생시키는 경우
rb.cause_user_error (01, True)
```

release_stopevent( ) 발생중인 예외 이벤트를 재설정합니다 .

예외 처리의 맨 위에 하십시오 .
예외는 재설정할 때까지 반복합니다 .

```
try:
    …       # 동작
except Robot_stop:
    rb.release_stopevent()
    …       # 대피 동작 등
```

i611Robot () 클래스의 메소드에 대한 예외 처리

Robot_emo( )	비상 정지시 발생하는 예외 (복귀는 할 수 없습니다)
Robot_error( )	오류시 발생하는 예외
Robot_fatalerror( )	치명적인 오류시 발생하는 예외 (복귀는 할 수 없습니다)
Robot_poweroff( )	전원 차단시 발생하는 예외 (복귀는 할 수 없습니다)
Robot_stop( )	감속 정지시 발생하는 예외

2 프로그래밍 가이드

6. 종료

로봇 프로그램 종료

close( ) 로봇과의 연결을 종료합니다 .

```
# 종료
    rb.close()
```

프로그램을 종료할 때와 컨트롤러의 전원을 차단할 때는 , 사용하는 모든 클래스의 close () 메소드를 반드시 실행하십시오 .

2 프로그래밍 가이드

2. 샘플 프로그램

모델 1 기본 동작

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ######################################
  # 라이브러리 가져오기
  from i611_MCS import *
  from i611_extend import *
  from i611_io import *

  def main():
    ## 2. 초기 설정② ###################################
    # i611 로봇 생성자
        rb = i611Robot()
        # 월드 좌표계 정의
        _BASE = Base()
        # 로봇과 연결 시작 초기화
        rb.open()
        # I/O 입출력 기능 초기화 (I/O 미사용시 생략 가능 )
        IOinit()

    ## 3. 교시 포인트 설정 ###################
        p1 = Position( 195, -100, -50, -120, posture=1 )
        p2 = Position( 115, -100, -70, 154 )
        p3 = Position( 130, -90, -100, 159 )
        p4 = p3.copy()
        p4.shift( dz=40 )
        j1 = Joint( 95, -30, -40, 90 )

    ## 4. 동작 조건 설정 ###################################
        #MotionParam 생성자에서 동작 조건 설정
        m = MotionParam( jnt_speed=10, lin_speed=70 )
        #MotionParam 형으로 동작 조건 설정
        rb.motionparam( m )

    ## 5. 로봇 동작 정의 ##############################
        # 작업 시작
        rb.home()
        rb.move( p1.offset(dz=30) )
        rb.line( p2, p3, p4 )
        rb.move( j1 )
        rb.home()

    ## 6. 종료 ########################################
        # 로봇과의 연결을 종료
        rb.close()

if __name__ == '__main__':
        main()
```

#servo on 위해 posture(0 혹은 1) 를 입력

PTP 동작 속도 10 %, 직선 보간 동작 속도 70mm/s

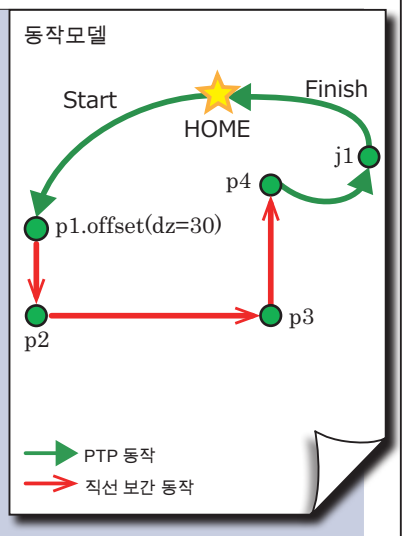
Home 위치로 이동

교시 포인트 p1 에서 Z 축 오프셋 30mm 로 PTP 동작

교시 포인트 p2, p3, p4 의 순서로 직선 보간 동작

교시 포인트 j1 으로 PTP 동작

Home 위치로 이동

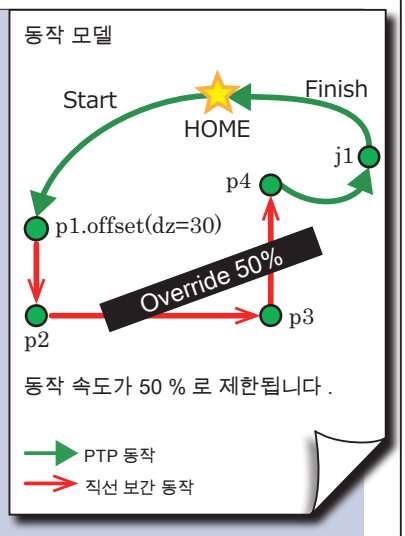


Python 언어에서는 들여쓰기로 문단을 구분합니다 . PDF 를 그대로 복사하면 들여쓰기가 복사되지 않으므로 , Spacebar 키 4 회 혹은 Tab 키 1 회로 예제와 동일하게 들여쓰기하십시오 . Tab 키는 Text editer 에 따라 의도대로 동작하지 않을 수 있습니다 .

2 프로그래밍 가이드

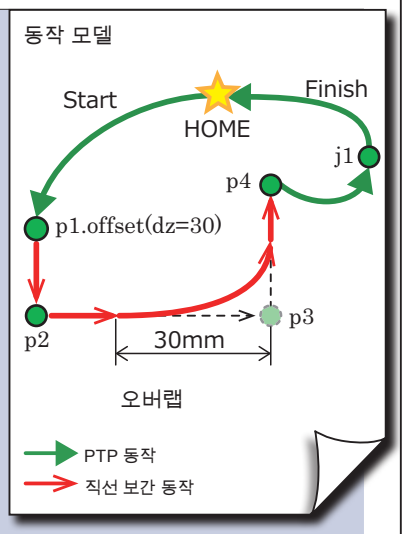
모델 2 오버라이드

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ######################################
# 라이브러리 가져오기
  from i611_MCS import *
  from i611_extend import *
  from i611_io import *

  def main():
    ## 2. 초기 설정② ###################################
    # i611 로봇 생성자
    rb = i611Robot()
    # 월드 좌표계 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능 초기화 (I/O 미사용시 생략 가능 )
        IOinit()
    # 오버라이드
        rb.override( 50 )          ← 오버라이드를 50 % 로 설정

    ## 3. 교시 포인트 설정 ##################
        p1 = Position( 195, -100, -50, -120, posture=1 )   ← #servo on 위해 posture(0 혹은 1) 를 입력
        p2 = Position( 115, -100, -70, 154 )
        p3 = Position( 130, -90, -100, 159 )
        p4 = p3.offset( dz=40 )
        j1 = Joint( 95, -30, -40, 90 )

    ## 4. 동작 조건 설정 ###################################
        #MotionParam 생성자에서 동작 조건 설정
        m = MotionParam( jnt_speed=10, lin_speed=70 )
        #MotionParam 형으로 동작 조건 설정
        rb.motionparam( m )

    ## 5. 로봇 동작 정의 ##################################
        # 작업 시작
        rb.home()                  ← Home 위치로 이동
        rb.move( p1.offset(dz=30) )  ← 교시 포인트 p1 에서 Z 축 오프셋 30mm 로 PTP 동작
        rb.line( p2, p3, p4 )      ← 교시 포인트 p2, p3, p4 의 순서로 직선 보간 동작
        rb.move( j1 )              ← 교시 포인트 j1 으로 PTP 동작
        rb.home()                  ← Home 위치로 이동

    ## 6. 종료 ########################################
        # 로봇과의 연결을 종료
        rb.close()

  if __name__ == '__main__':
        main()
```



모델 3 오버랩

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ######################################
  # 라이브러리 가져오기
  from i611_MCS import *
  from i611_extend import *
  from i611_io import *

  def main():
    ## 2. 초기 설정② ####################################
    # i611 로봇 생성자
        rb = i611Robot()
    # 월드 좌표계 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능 초기화 (I/O 미사용시 생략 가능 )
        IOinit()

    ## 3. 교시 포인트 설정 ###################
        p1 = Position( 195, -100, -50, -120, posture=1 )   ← #servo on 위해 posture(0 혹은 1) 를 입력
        p2 = Position( 115, -100, -70, 154 )
        p3 = Position( 130, -90, -100, 159 )
        p4 = p3.offset( dz=40 )
        j1 = Joint( 95, -30, -40, 90 )

    ## 4. 동작 조건 설정 ####################################
        #MotionParam 생성자에서 동작 조건 설정
        m = MotionParam( jnt_speed=10, lin_speed=70, overlap = 30 )
        #MotionParam 형으로 동작 조건 설정                  ← 오버랩을 30mm 로 설정
        rb.motionparam( m )

        ## 5. 로봇 동작 정의 ##############################
        rb.home()          ← Home 위치로 이동
        rb.move( p1 )      ← 교시 포인트 p1 으로 PTP 동작
        rb.line( p2 )      ← 교시 포인트 p2 로 직선 보간 동작
        # 프로그램 예측 동작 ON
        rb.asyncm( 1 )     ← 프로그램 예측 동작 ON ( 오버랩 동작의 실행 준비 )
        rb.line( p3, p4 )  ← 오버랩 동작 p3, p4 로 이동
        rb.join()          ← 오버랩 작동 중지 위치 . 예측한 프로그램의 완료 대기
        # 프로그램 예측 동작 OFF
        rb.asyncm( 2 )     ← 프로그램 예측 동작 OFF
        # 각 축 좌표 [0, 0, 0, 0]
        rb.home()          ← Home 위치로 이동

    ## 6. 종료 ##########################################
        # 로봇과의 연결을 종료
        rb.close()

  if __name__ == '__main__':
        main()
```



2 프로그래밍 가이드

모델 4 I/O 입력 대기

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ######################################
# 라이브러리 가져오기
  from i611_MCS import *
  from i611_extend import *
  from i611_io import *

  def main():
    ## 2. 초기 설정② ####################################
    # i611 로봇 생성자
        rb = i611Robot()
    # 월드 좌표계 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능 초기화
        IOinit()

    ## 3. 교시 포인트 설정 ######################
        p1 = Position( 195, -100, -50, -120, posture=1 )
        p2 = Position( 115, -100, -70, 154 )
        p3 = Position( 130, -90, -100, 159 )
        p4 = p3.offset( dz=40 )
        j1 = Joint( 95, -30, -40, 90 )

    ## 4. 동작 조건 설정 ####################################
        #MotionParam 생성자에서 동작 조건 설정
        m = MotionParam( jnt_speed=10, lin_speed=70 )
        #MotionParam 형으로 동작 조건 설정
        rb.motionparam( m )

    ## 5. 로봇 동작 정의 ##############################
    # 작업 시작
        rb.home()
    # I/O 입력 대기
        if wait( 5, "1", 10 )[0] == 1:
            rb.move( p1.offset(dz=30) )
            rb.line( p2, p3, p4 )
            rb.move( j1 )
    # 입력 시간 초과시
        else:
            rb.home()
    ## 6. 종료 ########################################
    # 로봇과의 연결을 종료
        rb.close()

if __name__ == '__main__':
    main()
```

#servo on 위해 posture(0 혹은 1) 를 입력

Home 위치로 이동

I/O 입력 포트 번호 5 가 1 이 될 때까지 대기 시간 10 초로 설정
대기 시간이 끝나기 전에 입력 신호가 들어오면

교시 포인트 p1 에서 Z 축 오프셋 30mm 로 PTP 동작

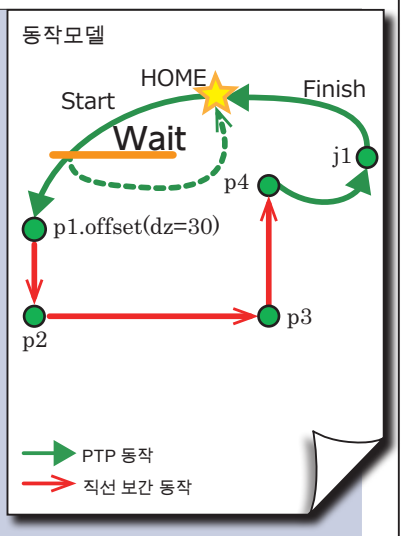
교시 포인트 p2, p3, p4 의 순서로 직선 보간 동작

교시 포인트 j1 으로 PTP 동작

초과된 경우

Home 위치로 이동

(이 예제에서는 Home 위치에서 이동을 안합니다)



2 프로그래밍 가이드

모델 5 팔레트 기능

```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 1. 초기 설정① ######################################
# 라이브러리 가져오기
  from i611_MCS import *
  from i611_extend import *

  def main():
    ### 2. 초기 설정② ###################################
    # i611 로봇 생성자
        rb = i611Robot()
        rb.open()
    # 월드 좌표계 정의
        _BASE = Base()

    ## 3. 교시 포인트 설정 ###########################
        # 교시 데이터 파일 읽기
        data = Teachdata( "teach_data" )
        pos_0 = data.get_position( "pos1", 0 )
        pos_1 = data.get_position( "pos2", 0 )
        pos_2 = data.get_position( "pos3", 0 )
        # 팔레트 정의
        pal = Pallet()
        pal.init_3( pos_0, pos_1, pos_2, 5, 4 )
        # 팔레트 조정
        pal.adjust( 3, 2, 0.4, –0.3 )
        # 작업 위치 검색
        p0 = pal.get_pos( 3, 2, 30 )
        p1 = pal.get_pos( 3, 2 )

    ## 4. 동작 조건 설정 ###################################
        #MotionParam 형으로 동작 조건 설정
        rb.motionparam( jnt_speed=30 )

    ## 5. 로봇 동작 정의 ##############################
    # 작업 시작
        rb.home()
        rb.move( p0 )
        rb.line( p1 )
        rb.line( p0 )
        rb.home()

    ## 6. 종료 ###########################################
    # 로봇과의 연결을 종료
        rb.close()

  if __name__ == '__main__':
        main()
```

미리 pos1 [0], pos2 [0], pos3 [0] 에 교시된 교시 데이터

팔레트의 셀 위치를 산출하기 위해 3 점으로 셀 위치 설정

팔레트 위치 크기 정의

(i, j) = (3, 2) 의 위치를 조정
(i 방향에 + 0.4mm, j 방향에 -0.3mm)

PTP 동작 속도 30 %

Home 위치로 이동

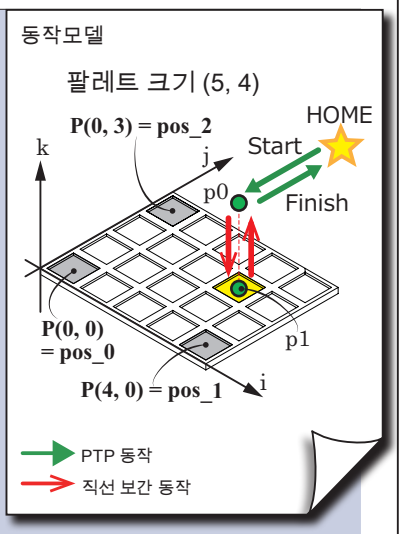
p0 (팔레트 위치 (3,2) 위 30mm) 로 PTP 동작

p1 (팔레트 위치 (3,2)) 로 직선 보간 동작

p0 (팔레트 위치 (3,2) 위 30mm) 로 PTP 동작

Home 위치로 PTP 동작

동작모델



2 프로그래밍 가이드

MEMO

D 소프트웨어

2 로봇 라이브러리

1. 데이터형

변수형의 종류

형 (아이콘)	의미
string string	문자열형 작은 따옴표 (') 또는 큰 따옴표 (") 로 묶습니다.
float float	부동소수점형
integer integer	정수형
long long	긴 정수형 (long)
bool bool	논리형 True / False

데이터형의 종류 1

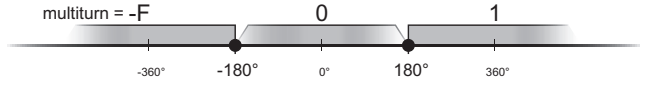
형 (아이콘)	의미
리스트 List	[. . .] 에서 요소를 늘어 놓은 것입니다.
튜플 Tuple	(. . .) 에서 요소를 늘어 놓은 것입니다. 튜플 요소를 변경할 수 없습니다.
딕셔너리 Dict.	{. . .} 는 " 딕셔너리 " 라는 키 (key) 와 값 (value) 을 콜론 (:) 으로 구분합니다. (예 , key : value)
가변 인수 Vari. No.	리스트를 확장합니다. 인수앞에「* (별표)」를 1 개 넣습니다. 받은 인수는 지정된 순서대로 튜플에 저장됩니다.
키워드 인수 Keyword	딕셔너리를 확장합니다. 인수 앞에「* (별표)」를 2 개 넣습니다 . 인수를 받은 시점에서 " 딕셔너리 " 가 되기 위한 순서는 무시됩니다.

데이터형의종류 2

형 (아이콘)	내용	
Position [Position]	월드 좌표계로 표시한 위치 좌표	
	[x, y, z, rz, ry, rx, parent, posture, multiturn] : List	
	x, y, z [mm] float	위치 (직교좌표계) 초기값 : 0.0
	rz,ry,rx [deg] float	자세 (Z-Y-X 계 오일러 각) (각 매니퓰레이터의 형태에 따라 존재하지 않는 값은 무시됩니다.) 초기값 : 0.0
	parent [-] float	월드 좌표계를 사용하는 설정 (*1) 초기값 : _BASE
	posture [-] integer	자세 (*2) 초기값 : 0
	multiturn [-] long	크로스오버 카운터 정보 (*3) 초기값 : 0 x FF 0 0 0 0 0 0 (FLAG J6 J5 J4 J3 J2 J1) (크로스오버 카운터 정보 없음) 크로스오버 카운터는 Position 형 위치 데이터를 고유하게 Joint 형 각도 데이터로 변환하기 위한 설정입니다. 예를 들어 관절의 각도가 –200°인지 160°인지를 판별합니다. 크로스오버 카운터를 설정하면 , 교시에 확인된 조작의 동작을 재현할 수 있습니다. 각 관절의 각도가 ± 180°를 초과할 때 값이 업데이트됩니다. 각 관절의 각도 범위에 따라 다음 값을 설정합니다. 관절의 개수에 따라 해당 값이 적용됩니다 . 예를 들어 , 4 축 로봇에서 J5, J6 값은 무시됩니다. ・ FLAG 부분 : 크로스오버 카운터 정보의 유무 FF : 없음 (초기값) 00 : 있음 (상기 이외는 무효) ・ J1 - J6 부분 : 크로스오버 카운터 값

설정값	각 관절의 각도
1	+ 방향으로 180°이상 회전
0	180° ~ 180°
F (*4)	- 방향으로 180°이상 회전

크로스오버 카운터 값과 조인트 각도의 관계



*1) 설정값은 "_BASE" 입니다.
*2) 암의 위치 , 방향 , 각도에 따라 여러 자세가 있습니다.

(C 교시 5 좌표계와 자세)

*3) 크로스오버 카운터 정보를 "CC" 로 약칭하는 경우가 있습니다.
*4) 교시 화면에서 "-1" 로 표시됩니다.

(C 교시 4 교시)

multiturn 매개변수에 관련 API

i611Robot 클래스의 use_mt( ) 메소드의 설정은 다음 API 의 동작을 바꿉니다 . multiturn 매개 변수는 i611Robot 클래스의 move () 와 Position2Joint () 메소드에서 사용합니다 .
이 API 는 multiturn 정보의 유무에 관계없이 사용할 수 있습니다 .

클래스	API	기능	p.
Position	has_mt()	크로스오버 카운터 정보를 확인합니다 .	30
	Position() (생성자)	월드 좌표계의 교시 포인트를 정의하는 인스턴스를 생성합니다 .	28
	pos2dist()	Position 좌표값을 딕셔너리 형으로 가져옵니다 .	31
	pos2list()	Position 좌표값을 리스트 형으로 가져옵니다 .	31
	position()	Position 좌표값을 Parent 좌표계로 변환하고 , 리스트 형으로 가져옵니다	31
	replace()	Position 좌표값을 치환합니다 . (자신을 갱신합니다 .)	32
MotionParam	ik_solver_option (변수)	회전 방향	41
i611Robot	getpos()	매니퓰레이터의 현재 위치를 Position 형으로 얻습니다 .	60
	Joint2Position()	Joint 좌표값에서 Position 좌표값으로 변환합니다 .	63
	move()	PTP 로 움직입니다 .	66
	Position2Joint()	Position 좌표값에서 Joint 좌표값으로 변환합니다 .	70
	use_mt()	크로스 오버 카운터의 활성화 / 비활성화를 설정합니다 .	80

형 (아이콘)	내용	
Joint [Joint]	각 관절의 각도	
	[j1, j2, j3, j4, j5, j6,] : List	
	j1, j2, j3, j4, j5, j6 [deg] float	Joint 형의 각 축 데이터 (스카라 로봇에서 J5, J6 값은 무시됩니다.) 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MotionParam [MotionParam]	여러 변수로 구성된 로봇의 동작 매개변수	
	lin_speed [mm/s] float	속도 (직선 보간 동작) 초기값 : 5.0
	jnt_speed [%] float	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작) 초기값 : 5.0
	acctime [s] float	가속 시간 초기값 : 0.4
	dacctime [s] float	감속 시간 초기값 : 0.4
	posture [-] integer	자세 초기값 : 0
	passm [-] integer	경로 동작 초기값 : 2
	overlap [mm] float	overlap 동작 초기값 : 0.0
	zone [pulse] integer	위치 결정 완료 범위 초기값 : 100
	pose_speed [%] float	속도 (자세 보간 동작) 초기값 : 20 매니퓰레이터 끝이 방향을 바꾸면서 작동할 때 , 끝 오일러 각도의 동작 속도 상한을 설정합니다 .
	ik_solver_option [-] long	회전 방향 초기값 : $0\ x\ \underset{(Rsv.)}{1}\ \underset{J6}{1}\ \underset{J5}{1}\ \underset{J4}{1}\ \underset{J3}{1}\ \underset{J2}{1}\ \underset{J1}{1}$ J1 - J6 의 설정값 0 : 최단 경로 multiturn 매개변수의 정보를 사용하지 않는 회전입니다 . 1 : multiturn 매개변수의 정보를 사용 2 : + 방향으로 회전 3 : ー방향으로 회전 + 방향 , ー방향은 좌표계 항목을 참조 * 관절의 개수에 따라 해당 값이 적용됩니다 . 예를 들어 , 4 축 로봇에서 J5, J6 값은 무시됩니다

2. 모듈

1. 모듈 목록

로봇 제어 모듈

모듈	클래스	요약
i611_MCS	Base P. 23 ~	월드 좌표계를 규정 (Position 클래스、Coordinate 클래스에서 이용할 더미 클래스)
	Coordinate P. 24 ~	월드 좌표계 개체를 취급
	Position P. 28 ~	월드 좌표계의 Position 좌표값을 취급
	Joint P. 33 ~	조인트 좌표계의 Joint 좌표값을 취급
	MotionParam P. 37 ~	로봇의 동작 매개변수를 취급
	i611Robot P. 47 ~	로봇의 동작을 취급

2 모듈

메소드					클래스
Base( ) p. 23					Base
b2g( ) p. 25	clear( ) p. 25	Coordinate( ) p. 24	copy( ) p. 25	g2b( ) p. 26	Coordinate
inv( ) p. 26	replace( ) p. 27	shift( ) p. 27			
clear( ) p. 29	copy( ) p. 29	has_mt( ) p. 30	offset( ) p. 30	pos2dict( ) p. 31	Position
pos2list( ) p. 31	Position( ) p. 28	position( ) p. 31	replace( ) p. 32	shift( ) p. 32	
clear( ) p. 34	copy( ) p. 34	jnt2dict( ) p. 34	jnt2list( ) p. 35	Joint( ) p. 33	Joint
offset( ) p. 35	replace( ) p. 36	shift( ) p. 36			
clear( ) p. 43	confdefault( ) p. 43	copy( ) p. 44	MotionParam( ) p. 42	motionparam( ) p. 45	Motion Param
mp2dict( ) p. 46	mp2list( ) p. 46				
abort( ) p. 50	adjust_mt( ) p. 50	asyncm( ) p. 51	cause_user_error( ) p. 52	changetool( ) p. 52	i611Robot
check_ready( ) p. 53	close( ) p. 54	disable_mdo( ) p. 54	enable_interrupt( ) p. 55	enable_mdo( ) p. 56	
exit( ) p. 57	get_hw_info( ) p. 57	get_system_port( ) p. 58	get_system_status( ) p. 59	getjnt( ) p. 59	
getmotionparam( ) p. 60	getpos( ) p. 60	home( ) p. 60	i611Robot( ) p. 49	is_open( ) p. 61	
is_pause( ) p. 61	join( ) p. 63	Joint2Position( ) p. 63	line( ) p. 64	MCS_version( ) p. 65	
motionparam( ) p. 65	move( ) p. 66	open( ) p. 67	optline( ) p. 68	override( ) p. 69	
pause( ) p. 69	Position2Joint( ) p. 70	release_stopevent( ) p. 70	reljntmove( ) p. 71	relline( ) p. 72	
restart( ) p. 73	set_behavior( ) p. 74	set_mdo( ) p. 75	settool( ) p. 76	sleep( ) p. 77	
stop( ) p. 78	svoff( ) p. 78	svstat( ) p. 79	toolmove( ) p. 79	use_mt( ) p. 80	
user_hook( ) p. 80	version( ) p. 81				

음영 처리된 (▭) 메소드는 생성자입니다 .

로봇 제어 모듈

모듈	클래스	요약
teachdata 	Teachdata P. 83 ~	교시 데이터 관리
i611_extend 	Pallet P. 93 ~	팔레트 기능을 처리
rbsys 	RobSys P. 98 ~	시스템 관리자 (*) 를 이용

*) 시스템 관리자는 로봇 프로그램의 상태 관리, 교시 상태 제어, 오류 처리를 하는 컨트롤러의 내부 프로그램입니다.

예외 처리 모듈

모듈	클래스	요약
i611_common 	Exception P. 108 ~	i611Robot 클래스 메소드의 예외 처리

함수 모듈

모듈	클래스	요약
i611_io 	(없음) P. 113 ~	함수 모듈 I/O 제어
i611shm 	(없음) P. 119 ~	함수 모듈 공유 메모리에 접근

2 모듈

메소드					클래스
check_format( ) p. 84	close( ) p. 85	flush( ) p. 85	get_coordinate( ) p. 86	get_joint( ) p. 86	Teachdata
get_param( ) p. 87	get_position( ) p. 88	get_tool( ) p. 89	is_open( ) p. 89	open( ) p. 90	
set_joint( ) p. 90	set_param( ) p. 91	set_position( ) p. 92	Teachdata( ) p. 84		
adjust( ) p. 94	get_pos( ) p. 95	init_3( ) p. 90	init_4( ) p. 97	Pallet( ) p. 93	Pallet
assign_din( ) p. 99	assign_dout( ) p. 100	clear_robtask( ) p. 101	close( ) p. 101	cmd_pause( ) p. 102	RobSys
cmd_reset( ) p. 102	cmd_run( ) p. 103	cmd_stop( ) p. 103	get_robtask( ) p. 104	open( ) p. 104	
req_mcmd( ) p. 105	RobSys( ) p. 98	set_robtask( ) p. 106	version( ) p. 107		

음영 처리된 ( ) 메소드는 생성자입니다 .

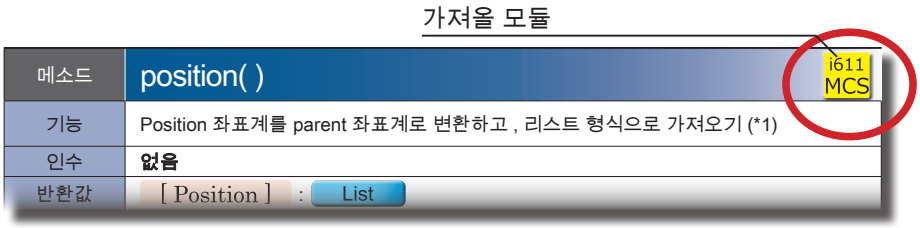
Exception 클래스를 상속한 클래스				클래스
Robot_emo( ) p. 109	Robot_error( ) p. 110	Robot_fatalerror( ) p. 110	Robot_poweroff( ) p. 111	Exception
Robot_stop( ) p. 112				

함수				모듈
din( ) p. 114	dlyOut( ) p. 115	dout( ) p. 115	IOinit( ) p. 116	i611_io
shotOut( ) p. 117	wait( ) p. 118			
shm_read( ) p. 119	shm_write( ) p. 120			i611shm

2. 모듈과 클래스 , 함수를 이용하기 위해

STEP1 모듈 가져오기

로봇 라이브러리를 사용하기 위해서는 사용하는 모듈을 미리 가져오십시오 .
메소드 안에는 미리 가져올 필요가 있는 여러 모듈이 있습니다 .
가져올 모듈은 각 메소드에 아이콘으로 표시하고 있습니다 .



모듈의 표시 및 가져오기

모듈	가져오는 프로그램 예시	포함되어 있는 클래스
i611_MCS i611 MCS	from i611_MCS import *	Base
		Coordinate
		Position
		Joint
		MotionParam
		i611Robot
teachdata Teach data	from teachdata import *	Teachdata
i611_extend i611 Ext.	from i611_extend import *	Pallet
rbsys rbsys	from rbsys import *	RobSys
i611_common i611 COM.	from i611_common import *	Exception
i611_io i611 IO	from i611_io import *	(없음)
i611shm i611 shm	from i611shm import *	(없음)

STEP2 클래스 , 함수를 이용하기 위한 준비

Base, Coordinate, Position, Joint, MotionParam, i611Robot 클래스의 메소드 사용하기

클래스	단계
Base	1. 모듈을 가져오십시오 . i611 MCS `from i611_MCS import *`
Coordinate Position	1. 모듈을 가져오십시오 . i611 MCS 2. 더미 클래스를 정의하십시오 . `_BASE = Base()`
Joint	1. 모듈을 가져오십시오 . i611 MCS

MotionParam	1. 모듈을 가져오십시오 . i611 MCS
	2. MotionParam 인스턴스를 생성합니다
	`m = MotionParam()`
	필요에 따라 로봇 동작 매개변수를 설정하십시오 .
	`m = MotionParam(jnt_speed=10,lin_speed=70,overlap=30)`
	(생략된 매개변수는 초기값으로 설정됩니다 .)

i611Robot	1. 모듈을 가져오십시오 . i611 MCS
	2. i611Robot 인스턴스를 생성하십시오 .
	`rb = i611Robot()`
	3. open() 메소드로 실행할 로봇과 연결하십시오
	`rb.open()`
	로봇의 동작을 수반하는 메소드 (*) 는 로봇의 MotionParam 에서 동작 매개변수를 설정하십시오 .

*) 로봇 동작을 수반하는 메소드

disable_mdo( )	enable_mdo( )	getjnt( )	getmotionparam( )	getpos( )
home( )	join( )	Joint2Position( )	line( )	motionparam( )
move( )	optline( )	override( )	Position2Joint( )	reljntmove( )
relline( )	set_mdo( )	toolmove( )		

STEP2 클래스 , 함수를 이용하기 위한 준비

Teachdata 클래스의 메소드를 이용하기

클래스	단계
Teachdata	1. 모듈을 가져오십시오 . Teach data from teachdata import * 2. Teachdata 인스턴스를 생성하십시오 . td = Teachdata( ) 3. Teachdata 클래스를 open() 메소드를 실행하십시오 . td.open(readonly = False)

Pallet 클래스의 메소드를 이용하기

클래스	단계
Pallet	1 . 모듈을 가져오십시오 . i611 MCS i611 Ext. from i611_MCS import * from i611_extend import * 2 . i611Robot 클래스 인스턴스를 생성하십시오 . rb = i611Robot( ) 3 . i611Robot 클래스의 open() 메소드를 실행하십시오 . rb.open() 4. Pallet 클래스의 인스턴스를 생성하십시오 . pal = Pallet()

STEP2 클래스 , 함수를 이용하기 위한 준비

RobSys 클래스의 메소드를 이용하기

클래스	단계

RobSys	1. 모듈을 가져오십시오 . rbsys
	`from rbsys import *`
	2. RobSys 인스턴스를 생성하십시오 .
	`rbs = RobSys()`
	3. RobSys 클래스 open() 메소드를 실행하십시오 .
	`rbs.open()`
	4. I/O 설정하는 메소드 (*) 인 i611_io 모듈을 가져와서、
	I/O 기능을 사용할 수있게 하십시오 . (☞ p.14) i611 IO

*) I/O 설정하는 메소드

assign_din( ), assign_dout( )

Exception 클래스의 메소드를 이용하기

클래스	단계
Exception	1. 모듈을 가져오십시오 .(*1) i611 COM. `from i611_common import *` 2. Exception 클래스를 상속한 클래스를 사용합니다 . 자세한 내용은 각 클래스의 사용 방법을 참조하십시오 .

*1) 가져올 모듈

i611_MCS 모듈에서 i611_common import * 를 가져오고 있습니다 .
Exception 클래스는 i611_MCS 모듈을 가져와도 사용할 수 있습니다 .

STEP2 클래스 , 함수를 이용하기 위한 준비

i611_io 모듈의 함수를 이용하기

모듈	단계

i611_io	1. 모듈을 가져오십시오 . i611 IO
	`from i611_io import *`
	2. I/O 의 초기화를 해주세요 .
	`IOinit()`
	3. i611Robot 인스턴스를 생성하십시오 .
	`rb = i611Robot()`
	4. i611Robot 의 open() 메소드를 실행하십시오 .
	`rb.open()`
	5. 로봇의 동작을 수반하는 메소드 (*) 는 로봇의 MotionParam 에서 작동 매개변수를 설정하십시오 .

*) 로봇의 동작을 수반하는 메소드 (i611Robot 클래스)

disable_mdo( )	enable_mdo( )	getjnt( )	getmotionparam( )	getpos( )
home( )	join( )	Joint2Position( )	line( )	motionparam( )
move( )	optline( )	override( )	Position2Joint( )	reljntmove( )
relline( )	set_mdo( )	toolmove( )		

i611_shm 모듈의 함수를 이용하기

모듈	단계
i611_shm	1. 모듈을 가져오십시오 . i611 shm `from i611_shm import *`

포인트 !

Python 의 "time" 모듈

"time" 는 시간 관련 정보 함수의 라이브러리입니다 .
시스템의 현재 시각을 확인 또는 시간 형식의 데이터를 생성 등을 수행합니다 . 시작은 1970 년 1 월 1 일 0 시 0 분 0 초 입니다 . 다음 모듈을 가져올 때 사용할 수 있습니다 .

```
import time
```

3. 메소드 목록

	메소드· 함수	기능	모듈 | 클래스	p.
A	abort( )	로봇 프로그램을 중단	i611 MCS i611Robot	50
	adjust( )	팔레트의 셀 위치를 보정	i611 Ext. Pallet	94
	adjust_mt( )	Position 형 좌표값을 문자열로 변환할 때의 CC 값을 보정	i611 MCS i611Robot	50
	assign_din( )	Position 형 좌표값을 문자열로 변환할 때의 CC 값을 보정	rbsys RobSys	99
	assign_dout( )	실제 I/O 및 메모리 I/O 출력 포트에 기능을 할당	rbsys RobSys	100
	asyncm( )	로봇 프로그램의 예측 동작 구간을 설정	i611 MCS i611Robot	51
B	b2g( )	Base 좌표계에서 월드 좌표계로 변환	i611 MCS Coordinate	25
	Base( )	Base 클래스 생성자	i611 MCS Base	23
C	cause_user_error( )	사용자 정의의 오류 발생	i611 MCS i611Robot	52
	changetool( )	도구 오프셋을 선택	i611 MCS i611Robot	52
	check_format( )	교시 데이터 파일의 포맷 버전을 생성	Teach data Teachdata	84
	check_ready( )	로봇이 자동 운전할 수 있는지 확인	i611 MCS i611Robot	53
	clear( )	월드 좌표계 개체 초기화	i611 MCS Coordinate	25
	clear( )	Position 좌표값 초기화	i611 MCS Position	29
	clear( )	Joint 좌표값 초기화	i611 MCS Joint	34
	clear( )	작동 매개변수 초기화	i611 MCS MotionParam	43
	clear_robtask( )	로봇 프로그램의 등록을 해제	rbsys RobSys	101
	close( )	로봇과의 연결을 종료	i611 MCS i611Robot	54
	close( )	교시 데이터 파일 종료	Teach data Teachdata	85
	close( )	시스템 관리자와의 연결을 종료	rbsys RobSys	101
	cmd_pause( )	동작 명령 : 일시 중지	rbsys RobSys	102
	cmd_reset( )	동작 명령 : 오류 재설정	rbsys RobSys	102
	cmd_run( )	동작 명령 : 로봇 프로그램을 실행	rbsys RobSys	103

상자 안에 있는 (▭) 메소드는 생성자입니다 .

모듈 아이콘

i611 MCS i611_MCS | Teach data teachdata | i611 Ext. i611_extend | rbsys rbsys | i611 COM i611_common | i611 IO i611_io | i611 shm i611shm

3 메소드 목록

메소드· 함수	기능	모듈 | 클래스	p.
cmd_stop( )	동작 명령 : 감속 정지	rbsys RobSys	103
confdefault( )	작동 매개변수의 초기값를 설정	i611 MCS MotionParam	43
Coordinate( )	Coordinate 클래스의 생성자	i611 MCS Coordinate	24
copy( )	월드 좌표계 개체 복사	i611 MCS Coordinate	25
copy( )	Position 좌표값 복사	i611 MCS Position	29
copy( )	Joint 좌표값 복사	i611 MCS Joint	34
copy( )	작동 매개변수 복사	i611 MCS MotionParam	44
D din( )	함수 I/O 입력	i611 IO (없음)	114
disable_mdo( )	MDO 동작을 무효로 함	i611 MCS i611Robot	54
dlyOut( )	함수 지정 시간 경과 후 I/O 출력	i611 IO (없음)	115
dout( )	함수 I/O 출력	i611 IO (없음)	115
E enable_interrupt( )	감속 정지와 비상 정지의 예외의 발생을 설정	i611 MCS i611Robot	55
enable_mdo( )	MDO 동작을 활성화	i611 MCS i611Robot	56
exit( )	로봇 프로그램을 강제 종료	i611 MCS i611Robot	57
F flush( )	업데이트된 교시 데이터를 파일로 내보내기	Teach data Teachdata	85
G g2b( )	월드 좌표계에서 Base 좌표계로 변환	i611 MCS Coordinate	26
get_coordinate( )	교시 데이터의 베이스 오프셋 값을 확인	Teach data Teachdata	86
get_hw_info( )	모델명과 시리얼 번호 확인	i611 MCS i611Robot	57
get_joint( )	교시 데이터의 Joint 좌표값 확인	Teach data Teachdata	86
get_param( )	교시 데이터의 매개변수 확인	Teach data Teachdata	87
get_pos( )	셀의 위치 획득	i611 Ext. Pallet	95

상자 안에 있는 (▭) 메소드는 생성자입니다 .

3 메소드 목록

	메소드· 함수	기능	모듈 | 클래스	p.
	get_position( )	교시 데이터의 Position 좌표값을 확인	Teachdata	88
	get_robtask( )	로봇 프로그램의 상태를 확인	RobSys	104
	get_system_port( )	시스템 포트의 상태를 확인	i611Robot	58
	get_system_status( )	시스템 상태와 오류 ID 를 확인	i611Robot	59
	get_tool( )	교시 데이터의 도구 오프셋을 확인	Teachdata	89
	getjnt( )	매니퓰레이터의 현재 위치를 Joint 형으로 확인	i611Robot	59
	getmotionparam( )	현재의 동작 매개변수를 확인	i611Robot	60
	getpos( )	매니퓰레이터의 현재위치를 Position 형으로 확인	i611Robot	60
H	has_mt( )	크로스오버 카운터 정보 확인	Position	30
	home( )	모든 축을 Joint 좌표 0deg 로 이동	i611Robot	60
I	i611Robot( )	i611Robot 클래스의 생성자	i611Robot	49
	init_3( )	팔레트 정의 (3 점 교시)	Pallet	96
	init_4( )	팔레트 정의 (4 점 교시)	Pallet	97
	inv( )	Coordinate 클래스의 객체를 생성하는 역변환을 수행	Coordinate	26
	IOinit( )	함수 I/O 초기화	(없음)	116
	is_open( )	i611Robot 오픈 상태 확인	i611Robot	61
	is_open( )	교시 데이터의 오픈 상태를 확인	Teachdata	8
	is_pause( )	로봇 프로그램의 일시 정지 상태를 확인	i611Robot	60
J	jnt2dict( )	Joint 좌표값을 딕셔너리 형식으로 확인	Joint	34
	jnt2list( )	Joint 좌표값을 딕셔너리 형식으로 확인	Joint	35
	join( )	예측된 로봇 프로그램의 동작 완료를 대기	i611Robot	63
	Joint( )	Joint 클래스의 생성자	Joint	33
	Joint2Position( )	Joint 좌표값에서 Position 좌표값으로 변환	i611Robot	63
L	line( )	직선 보간 동작을 수행	i611Robot	64
M	MCS_version( )	로봇 라이브러리의 버전을 확인	i611Robot	65

모듈 아이콘

i611_MCS | teachdata | i611_extend | rbsys | i611_common | i611_io | i611shm

3 메소드 목록

	메소드· 함수	기능	모듈 | 클래스	p.
	MotionParam( )	MotionParam 클래스의 생성자	i611 MCS MotionParam	42
	motionparam( )	작동 매개변수를 설정	i611 MCS MotionParam	45
	motionparam( )	작동 매개변수를 설정	i611 MCS i611Robot	65
	move( )	PTP 동작을 수행	i611 MCS i611Robot	66
	mp2dict( )	작동 매개변수를 딕셔너리 형식으로 가져오기	i611 MCS MotionParam	46
	mp2list( )	작동 매개변수를 리스트 형식으로 가져오기	i611 MCS MotionParam	46
O	offset( )	Position 좌표값에 오프셋 좌표값을 추가 (자신을 유지하면서 새로운 객체를 생성)	i611 MCS Position	30
	offset( )	Joint 좌표값에 오프셋 좌표값을 추가 (자신을 유지하면서 새로운 객체를 생성)	i611 MCS Joint	35
	open( )	로봇과의 연결을 시작 (초기화)	i611 MCS i611Robot	67
	open( )	교시 데이터 파일을 오픈	Teach data Teachdata	90
	open( )	시스템 관리자와 통신을 시작	rbsys RobSys	104
	optline( )	직선 보간 동작을 최적의 속도로 변속하면서 수행	i611 MCS i611Robot	68
	override( )	override(*) 를 수행	i611 MCS i611Robot	69
P	Pallet( )	Pallet 클래스의 생성자	i611 Ext. Pallet	93
	pause( )	로봇 동작을 일시 중지	i611 MCS i611Robot	69
	pos2dict( )	Position 좌표값을 딕셔너리 형식으로 가져오기	i611 MCS Position	31
	pos2list( )	Position 좌표값을 리스트 형식으로 가져오기	i611 MCS Position	31
	Position( )	Position 클래스의 생성자	i611 MCS Position	28
	position( )	Position 좌표값을 parent 좌표계로 변환하고 리스트 형식으로 가져오기	i611 MCS Position	31
	Position2Joint( )	Position 좌표값에서 Joint 좌표값으로 변환	i611 MCS i611Robot	70
R	release_stopevent( )	발생중인 예외 이벤트를 재설정	i611 MCS i611Robot	70
	reljntmove( )	Joint 좌표계에서 상대 이동	i611 MCS i611Robot	71
	relline( )	직교 좌표계에서 상대 직선 보간 동작을 수행	i611 MCS i611Robot	72

상자 안에 있는 (▭) 메소드는 생성자입니다 .

*) override 는 속도 설정을 덮어씁니다 .

3 메소드 목록

메소드· 함수	기능	모듈 | 클래스	p.
replace( )	월드 좌표계 개체를 대체 (자신을 업데이트)	i611 MCS Coordinate	27
replace( )	Position 좌표값을 대체 (자신을 업데이트)	i611 MCS Position	32
replace( )	Joint 좌표값을 대체 (자신을 업데이트)	i611 MCS Joint	36
req_mcmd( )	시스템 상태 및 동작 명령 상태를 확인	rbsys RobSys	105
restart( )	일시 정지에서 재시작 신호를 발생	i611 MCS i611Robot	73
RobSys( )	RobSys 클래스의 생성자	rbsys RobSys	98
S set_behavior( )	일시 정지 동작 (행동) 을 설정	i611 MCS i611Robot	74
set_joint( )	교시 데이터 Joint 좌표값을 업데이트	Teach data Teachdata	90
set_mdo( )	MDO 동작 (*) 의 설정	i611 MCS i611Robot	75
set_param( )	교시 데이터의 매개변수를 업데이트	Teach data Teachdata	91
set_position( )	교시 데이터의 좌표값을 업데이트	Teach data Teachdata	92
set_robtask( )	로봇 프로그램을 등록	rbsys RobSys	106
settool( )	도구 오프셋을 설정	i611 MCS i611Robot	76
shift( )	월드 좌표계 개체를 이동 (자신을 업데이트)	i611 MCS Coordinate	27
shift( )	Position 좌표값를 이동 (자신을 업데이트)	i611 MCS Position	32
shift( )	Joint 좌표값를 이동 (자신을 업데이트)	i611 MCS Joint	36
shm_read( )	함수 공유 메모리를 읽기	i611 shm (없음)	119
shm_write( )	함수 공유 메모리에 기록	i611 shm (없음)	120
shotOut( )	함수 지정 시간 만 I/O 출력	i611 IO (없음)	117
sleep( )	지정된 시간 동안 , 처리를 일시 중지	i611 MCS i611Robot	77
stop( )	로봇을 감속 정지	i611 MCS i611Robot	78

*) MDO 동작 : 동작 중에 지정된 조건에서 I/O 출력을 변경하는 기능입니다 .

모듈 아이콘

i611 MCS i611_MCS | Teach data teachdata | i611 Ext. i611_extend | rbsys rbsys | i611 COM. i611_common | i611 IO i611_io | i611 shm i611shm

3 메소드 목록

	메소드· 함수	기능	모듈	클래스	p.
	svoff( )	서보를 OFF 로 설정	i611 MCS	i611Robot	78
	svstat( )	서보 상태를 확인	i611 MCS	i611Robot	79
T	Teachdata( )	Teachdata 클래스의 생성자	Teach data	Teachdata	84
	toolmove( )	도구 좌표계에서 상대 동작을 수행	i611 MCS	i611Robot	79
U	use_mt( )	크로스오버 카운터의 활성화 / 비활성화를 설정	i611 MCS	i611Robot	80
	user_hook( )	로봇 프로그램을 일시 중지	i611 MCS	i611Robot	80
V	version( )	시스템 버전을 확인	i611 MCS	i611Robot	81
	version( )	시스템 관리자의 버전을 확인	rbsys	RobSys	107
W	wait( )	함수 지정한 I/O 입력 패턴이 될 때까지 대기	i611 IO	(없음)	118

상자 안에 있는 (▭) 메소드는 생성자입니다 .

4. 로봇 라이브러리

프로그램을 종료하거나 컨트롤러의 전원을 차단할 때는 , 사용하고 있는 모든 클래스의 close() 메소드를 반드시 실행시켜 주십시오 .

로봇 프로그램은「대문자」 /「소문자」를 구분합니다 ..

「예외 (Exception)」에 관하여

로봇 프로그램의 작성이 잘못된 경우에는「예외」가 발생합니다 .
예외가 발생하면 에러 코드를 확인한 후 , 적절한 조치를 취해주십시오 .

프로그램 실행 결과 확인

각 메소드에는 사용 예시를 기재하였습니다 . 메소드의 실행 결과를 print 문으로 확인할 수 있습니다 .

사용 예

```
# 교시 데이터를 불러오기
    data=Teachdata( './teach_data' )
#Key : Joint1, index 번호 : 0 의 데이터를 획득한다 .
    jnt_0=data.get_Joint( 'Joint1', 0 )
    print jnt_0.jnt2list()
```

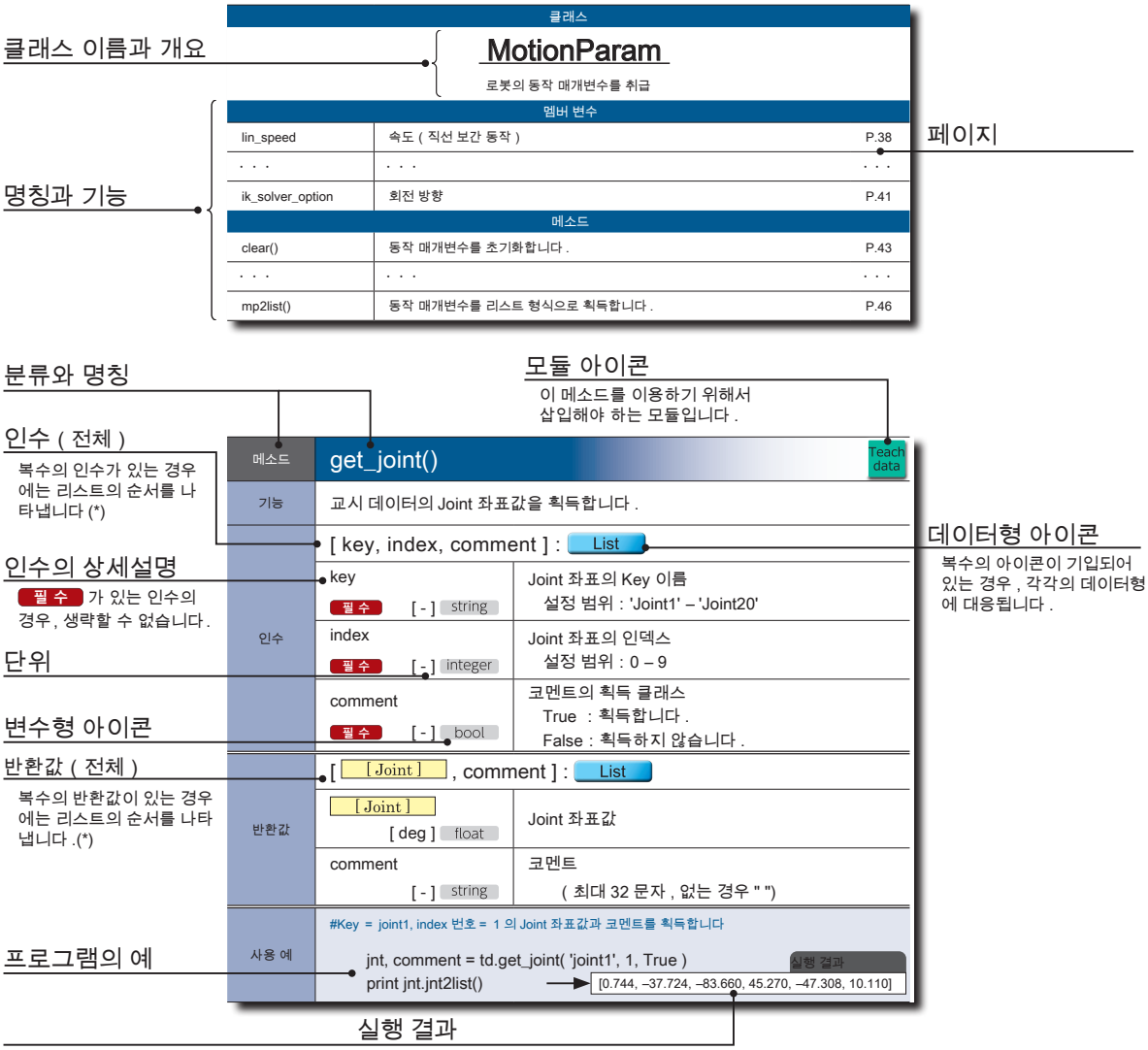
실행 결과 (본 예시는 Joint 형의 데이터를 표시)

```
# 실행 결과 확인
[0.744, –37.724, –83.660, 45.270, 0.0, 0.0]
```

(이후의 설명에서는「print…」를 생략 , 실행 결과만을 표기합니다 .)

포인트!

로봇 라이브러리를 보는 법



*) [Position] [Joint] [MotionParam] 가 기재되어 있는 경우가 있습니다 . 각 데이터형의 상세한 내용을 참조해주세요 . (☞ P.3)

필수 아이콘
- 필 수 : 생략 불가능한 인수

데이터형 아이콘
- List : 리스트
- Tuple : 튜플
- Dict. : 딕셔너리
- Vari. No. : 가변 인수
- Keyword : 키워드

변수형 아이콘
- long : long 형
- string : String 형
- integer : Integer 형
- float : float 형
- bool : bool 형

단위
- [mm]
- [deg]
- [s]
- [–] (단위 없음)

데이터형 아이콘
- [Position] : Position 형
  [x, y, z, rz, ry, rx, parent, posture, multiturn]
- [Joint] : Joint 형
  [j1, j2, j3, j4, j5, j6,]
- [MotionParam] : MotionParam 형
  [lin_speed, jnt_speed, acctime, dacctime, posture,passm, overlap, zone, pose_speed, ik_solver_option]

1. 모듈 : i611_MCS

클래스	
Base	
월드 좌표계를 규정합니다 .(*)	
멤버 변수	
—	—
메소드	
—	—

(*) Position 클래스 , Coordinate 클래스에서 사용할 더미 클래스입니다 .

생성자	Base() i611 MCS
기능	월드 좌표계의 Position 클래스 , Coordinate 클래스에서 이용할 더미 클래스의 인스턴스를 만듭니다 .
인수	없음
반환값	자기 참조 (Base 클래스 객체)
사용 예	_BASE=Base()

클래스	

Coordinate

월드 좌표계 객체를 다룹니다 .

멤버 변수	
x, y, z	위치 (절대 좌표)
rz, ry, rx	자세 (Z-Y-X 계 오일러 각도)
parent	월드 좌표계를 사용하는 설정합니다 .

메소드		
b2g()	Base 좌표계에서 월드 좌표계로 변환합니다 .	P.25
clear()	월드 좌표계 개체를 초기화합니다 .	P.25
copy()	월드 좌표계를 복사합니다 .	P.25
g2b()	월드 좌표계에서 Base 좌표계로 변환합니다 .	P.26
inv()	역변환을 수행할 Coordinate 클래스의 객체를 생성합니다 .	P.26
replace()	월드 좌표계 객체를 대체합니다 . (자가 업데이트)	P.27
shift()	월드 좌표계 객체를 이동합니다 . (자가 업데이트)	P.27

생성자	Coordinate() i611 MCS
기능	월드 좌표계에서 정의된 Coordinate 클래스의 인스턴스를 생성합니다 .
인수	[x, y, z, rz, ry, rx, parent] : List Keyword 숫자를 생략하면 기본값이 설정됩니다 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]
반환값	자기 참조 (Coordinate 객체)

사용 예

```
# 예 1 : 인수를 생략합니다 . ( 초기값 설정 )
    CO1=Coordinate()
```

실행 결과 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]

```
# 예 2 : 키워드 ( 모두 )
    CO2=Coordinate( x=1, y=2, z=3, rz=1, parent=_BASE )
```

[1.0, 2.0, 3.0, 1.0, 0.0, 0.0, < ... >]

```
# 예 3 : 키워드 (6 개 )
    CO3=Coordinate( x=1, y=2, z=3, rz=1 )
```

[1.0, 2.0, 3.0, 1.0, 0.0, 0.0, < ... >]

```
# 예 4 : 키워드 (1 개 ), 지정하지 않은 값은 초기값이 됩니다 .
    CO4=Coordinate( x=10 )
```

[10.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]

i611 MCS

실행 결과
[1.0, 2.0, 3.0, 4.0, 0.0, 0.0]

메소드	clear() i611 MCS
기능	월드 좌표계의 객체를 초기화합니다 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]
인수	없음
반환값	없음

사용 예	# 월드 좌표를 초기화합니다 .
	CO1=Coordinate(1, 2, 3, 4, 0, 0,_BASE)
	CO1.clear() → 실행 결과 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]

메소드	copy() i611 MCS
기능	월드 좌표계를 복사합니다 .
인수	없음
반환값	복사한 새로운 좌표계 개체 (Coordinate 객체)

사용 예	# 임의의 Coordinate 객체 CO1 복사합니다 .
	#CO1 을 선언합니다 .
	CO1=Coordinate(x=1, y=2, z=3, rz=4, parent=_BASE)
	
	#CO1 을 새로운 Coordinate 객체 CO1C 에 복사합니다 .
	CO1C=CO1.copy()
	실행 결과
	CO1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >]
	↓
	copy()
	↓새로운 포인트 개체 : CO1C
	CO1C : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >]

메소드	g2b()	i611 MCS
기능	월드 좌표계에서 Base 좌표계로 변환합니다 .	
인수	[X, Y, Z, RZ, RY, RX] : List (좌표 변환을 하기 위해 모든 인수가 필요합니다 .)	
	X, Y, Z 필 수 [mm] float	단위 (월드 좌표계)
	RZ, RY, RX 필 수 [deg] float	자세 (Z-Y-X 계 오일러 각도)
반환값	[x, y, z, rz, ry, rx] : List	
	x, y, z [mm] float	단위 (기본 좌표계)
	rz, ry, rx [deg] float	자세 (Z-Y-X 계 오일러 각도)

사용 예	# 리스트에서 모든 인수를 지정합니다 .
	CO1=Coordinate()
	CO1.g2b(1, 2, 3, 4) → 실행 결과 [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]	

메소드	inv()	i611 MCS
기능	역변환을 수행할 Coordinate 클래스의 객체를 생성합니다 .	
인수	없음	
반환값	새로 생성된 Coordinate 객체	

사용 예	# 미리 임의의 좌표계의 교시 포인트를 사용할 수 있어야 합니다 .
	#CO1 를 선언합니다 .
	CO1 = Coordinate() → 실행 결과 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]
	# 인수로 지정하는 값으로 업데이트합니다 .
	CO1.replace(1, 2, 3, 4) → [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >]
	
	CO2 = CO1.inv()	

메소드	replace() i611 MCS
기능	월드 좌표계 객체를 대체합니다 . (자가 업데이트)
인수	[x, y, z, rz, ry, rx, parent] : List Keyword
	x, y, z [mm] float — 단위 (월드 좌표계)
	rz, ry, rx [deg] float — 자세 (Z-Y-X 계 오일러 각도)
	parent [-] float — 월드 좌표계를 사용하는 설정
	인수를 생략하면 기본값이 설정됩니다 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]
반환값	자기 참조 (Coordinate 객체)

사용 예	# 예 1 : 리스트 (2 개) 지정하지 않은 값은 초기값이 됩니다 .
	CO2=Coordinate()
	CO2.replace(7, 8) → 실행 결과 [7.0, 8.0, 0.0, 0.0, 0.0, 0.0, < ... >]
	# 예 2 : 키워드 (2 개) 지정하지 않은 값은 초기값이 됩니다 .
	CO3=Coordinate()
	CO3.replace(x=1, rz=4) → [1.0, 0.0, 0.0, 4.0, 0.0, 0.0, < ... >]

메소드	shift() i611 MCS
기능	월드 좌표계 객체를 이동합니다 . (자가 업데이트)
인수	[dx, dy, dz, drz, dry, drx] : List Keyword
	dx, dy, dz [mm] float — 단위 (월드 좌표계)
	drz, dry, drx [deg] float — 자세 (Z-Y-X 계 오일러 각도)
	인수를 생략하면 이동하지 않습니다 .
반환값	자기 참조 (Coordinate 객체)

사용 예	# 예 1 : 리스트 (2 개)
	CO1 = Coordinate()
	CO1.replace(1, 2, 3, 4) → 실행 결과 [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	CO1.shift(80, 70) → [81.0, 72.0, 3.0, 4.0, 0.0, 0.0]
	# 예 2 : 키워드 (1 개)
	CO2 = Coordinate()
	CO2.replace(1, 2, 3, 4) → [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	CO2.shift(dx=80) → [81.0, 2.0, 3.0, 4.0, 0.0, 0.0]

클래스

Position

월드 좌표계의 Position 좌표값 (*) 를 다룹니다 .

멤버 변수	
—	—

메소드		
clear()	Position 좌표값을 초기화합니다 .	P.29
copy()	Position 좌표값을 복사합니다 .	P.29
has_mt()	크로스오버 카운터 정보를 확인합니다 .	P.30
offset()	Position 좌표값에 오프셋 좌표값을 추가합니다 . (자신을 유지하면서 새로운 객체를 생성합니다 .)	P.30
pos2dict()	Position 좌표값을 딕셔너리 형식으로 가져옵니다 .	P.31
pos2list()	Position 좌표값을 리스트 형식으로 가져옵니다 .	P.31
position()	Position 좌표값을 parent 좌표계로 변환하고 리스트 형식으로 가져옵니다 .	P.31
replace()	Position 좌표값을 대체합니다 . (자가 업데이트)	P.32
shift()	Position 좌표값을 이동합니다 . (자가 업데이트)	P.32

*) 로봇 프로그램에서 취급 좌표입니다 .
교시 좌표뿐만 아니라 교시하지 않는 좌표도 처리할 수 있습니다 .

생성자	Position() i611 MCS
기능	월드 좌표계의 교시 포인트를 정의하는 인스턴스를 생성합니다 .
인수	[Position] [x, y, z, rz, ry, rx, parent, posture, multiturn] : List Keyword 인수를 생략하면 기본값이 설정됩니다 . (스카라 로봇에서 ry, rx 는 무시됩니다 .) 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE, 0, 0xFF000000] ・ 2 번째 이후의 인스턴스 생성시에는 최근 값이 적용됩니다 . ・ clear() 를 호출하여 최근 값이 재설정되고 초기값으로 돌아갑니다 .

사용 예	# 예 1 : 인수를 생략합니다 . (초기값 설정)
	P1=Position() → [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, 0, 0xFF000000] (실행 결과)
	# 예 2 : 키워드
	P2=Position(x=1, y=2, z=3, rz=1, parent=_BASE, posture=0)
	→ [1.0, 2.0, 3.0, 1.0, 0.0, 0.0, < ... >, 0, 0xFF000000]
	# 예 3 : 키워드 (1 개) 를 지정하지 않은 값은 초기값이 됩니다 .
	P3=Position(drz=103) → [0.0, 0.0, 0.0, 103.0, 0.0, 0.0, < ... >, 0, 0xFF000000]

[Position] 자세한 내용은 MotionParam 클래스 (P. 37) 를 참조하십시오 .

메소드	clear() i611 MCS
기능	Position 좌표값을 초기화합니다 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE,–1, 0xFF000000]
인수	없음
반환값	없음

사용 예	# 임의의 Position 객체 P1 를 초기화합니다 .
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4)
	
	#P1 를 초기화합니다 .
	P1.clear()
	
	실행 결과
	P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
	↓
	clear()
	↓
	P1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

posture 매개변수의 초기값

posture 는 3bit 의 값입니다 . 초기값은 "0" 입니다 . 새로 입력한 경우 덮어 씁니다 .
지정하지 않으면 이전의 설정 값을 상속합니다 .

메소드	copy() i611 MCS
기능	Position 좌표값을 복사합니다 .
인수	없음
반환값	복사한 새로운 교시 포인트 개체 (Position 객체)

사용 예	# 임의의 Position 객체 P1 를 복사합니다 .
	#P1 을 선언합니다 .
	P1=Position(x=1, y=2, z=3, rz=4, 0xFF000000)
	
	#P1 을 새로운 Position 객체 P1C 에 복사합니다 .
	P1C=P1.copy()
	
	실행 결과
	P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
	↓
	copy()
	↓새로운 포인트 객체 : P1C
	P1C : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

메소드	has_mt()	i611 MCS
기능	크로스오버 카운터 정보를 확인합니다 .	
인수	없음	
반환값	res0 [-] bool	크로스오버 카운터 정보의 유무 True : 있음 False : 없음

사용 예	# 크로스오버 카운터 정보를 확인합니다 .
	#P1 을 선언합니다 .
	P1=Position(x=1, y=2, z=3, rz=4)
	#P1 의 크로스오버 카운터 정보를 확인합니다 .
	P1.has_mt() → 실행 결과 : False	

프로그램에서 Position 클래스를 크로스오버 카운터 정보가 생략된 매개변수로 생성하면 크로스오버 카운터 정보가 없는 Position 형 데이터가 만들어집니다 .

메소드	offset()	i611 MCS
기능	Position 좌표값에 오프셋 좌표값을 추가합니다 . (자신을 유지하면서 새로운 객체를 생성합니다 .)	
인수	[dx, dy, dz, drz, dry, drx] : List Keyword	
	dx, dy, dz [mm] float	위치의 오프셋 량 (직교 좌표계)
	drz, dry, drx [deg] float	자세의 오프셋 량 (Z-Y-X 계 오일러 각도)
	인수를 생략하면 오프셋은 설정되지 않습니다 .	
반환값	자신이 유지하고 있는 오프셋값을 참조합니다 . (Position 객체)	

사용 예	# 임의의 Position 객체 P1 에 오프셋 좌표값을 추가합니다 .
	#P1 을 선언합니다 .
	P1=Position(x=1, y=2, z=3, rz=4)
	#P1 에서 오프셋된 새 Position 객체 P1ofs 를 생성합니다 .
	P1ofs=P1.offset(dx=100, drz=–10)
		실행 결과
		P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, ...]
		↓
		offset()
		↓
		처음 오프셋 :P1
		P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, ...]
		새로운 포인트 객체 : P1ofs
		P1ofs : [101.0, 2.0, 3.0, –6.0, 0.0, 0.0, ...]

	#P1 에서 오프셋시키고 싶은 값만 리스트에 지정된 경우 새 Position 객체 P1ofs 를 생성합니다 . P1ofs=P1.offset(100, 0, 0, -10)	새로운 포인트 객체 : P1ofs P1ofs : [101.0, 2.0, 3.0, –6.0, 0.0, 0.0, ...]

메소드	pos2dict() i611 MCS
기능	Position 좌표값을 딕셔너리 형식으로 가져옵니다 .(*1)
인수	없음
반환값	[Position] : Dict.

사용 예	# 임의의 Position 객체 P1 을 딕셔너리 형식으로 출력합니다 .(*2)
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4)
	
	#P1 을 딕셔너리 형식으로 출력합니다 .
	P1.pos2dict() → 실행 결과
	{'parent': < ... >, 'rx': 0.0, 'ry': 0.0, 'rz': 4.0, 'y': 2.0, 'x': 1.0, 'z': 3.0, 'posture': 0, 'multiturn': 0xFF000000}

메소드	pos2list() i611 MCS
기능	Position 좌표값을 리스트 형식으로 가져옵니다 .(*1)
인수	없음
반환값	[Position] : List

사용 예	# 임의의 Position 객체 P1 을 목록 형식으로 출력합니다 .(*2)
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4)
	
	#P1 을 리스트 형식으로 출력합니다 .
	P1.pos2list() → 실행 결과
	[1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

메소드	position() i611 MCS
기능	Position 좌표값을 parent 좌표계로 변환하고 리스트 형식으로 가져옵니다 .(*1)
인수	없음
반환값	[Position] : List

사용 예	# 임의의 Position 객체 P1 을 리스트 형식으로 출력합니다 .(*2)
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4)
	
	#P1 을 리스트 형식으로 출력합니다 .
	P1.position() → 실행 결과
	[1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

* 1)i611Robot.use_mt (True) 을 설정하는 경우는 반환값에 multiturn 항목이 추가됩니다 .
i611Robot.use_mt (False) (초기값) 의 경우는 추가되지 않습니다 .
* 2) 이 예제는 크로스오버 카운터 정보를 보유하지 않은 경우입니다 .

메소드	replace()	i611 MCS
기능	Position 좌표값을 대체합니다 .(*1) (자가 업데이트)	
인수	[Position] : List Keyword	
	인수를 생략하면 값이 업데이트되지 않습니다 .	
반환값	자신에 대한 참조 (Position 객체)	

사용 예	# 임의의 Position 객체 P1, P2, P3 를 대체합니다 .(*2)
	#P1, P2, P3 를 선언합니다 .
	P1=Position()
	P2=Position()
	P3=Position()
	
	# 예 1 : 리스트로 지정하는 경우
	P1.replace(1, 2, 3, 4)
	
	# 예2 : 2 개의 가변 인수로 지정하는 경우
	P2.replace(7, 8)
	
	# 예3 : 2 개의 키워드로 지정하는 경우
	P3.replace(x=1, rz=4)
		실행 결과
		P1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		↓ replace()
		P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		P2 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		↓ replace()
		P2 : [7.0, 8.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		P3 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		↓ replace()
		P3 : [1.0, 0.0, 0.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

메소드	shift()	i611 MCS
기능	Position 좌표값을 이동합니다 .(*1) (자가 업데이트)	
인수	[dx, dy, dz, drz, dry, drx] : List Keyword	
	dx, dy, dz [mm] float — 위치의 이동량 (직교 좌표계)	
	drz, dry, drx [deg] float — 자세의 이동량 (Z-Y-X 계 오일러 각도)	
	인수를 생략하면 이동하지 않습니다 .	
반환값	자기 참조 (Position 객체)	

사용 예	# 임의의 Position 객체 P1 을 이동합니다 .(*2)
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4)
	
	#P1 를 옮긴 위치로 업데이트합니다 .
	P1.shift(dx=80)
		실행 결과
		P1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]
		↓
		shift()
		↓
		원래 객체 :P1
		P1 : [81.0, 2.0, 3.0, 4.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

* 1) i611Robot.use_mt (True) 을 설정하는 경우는 반환값에 multiturn 항목이 추가됩니다 .
i611Robot.use_mt (False) (초기값) 의 경우는 추가되지 않습니다 .
* 2) 이 예제는 크로스오버 카운터 정보를 적용하지 않은 경우입니다 .

클래스

Joint

Joint 좌표계의 Joint 좌표값의 각도 데이터를 처리합니다 (*)

멤버 변수	
j1, j2, j3, j4, j5, j6	Joint 각도

메소드		
clear()	Joint 좌표값을 초기화합니다 .	P.34
copy()	Joint 좌표값을 복사합니다 .	P.34
jnt2dict()	Joint 좌표값을 딕셔너리 형식으로 가져옵니다 .	P.34
jnt2list()	Joint 좌표값을 리스트 형식으로 가져옵니다 .	P.35
offset()	Joint 좌표값에 오프셋 좌표값을 추가합니다 . (자신을 유지하면서 새로운 객체를 생성합니다 .)	P.35
replace()	Joint 좌표값을 대체합니다 . (자가 업데이트합니다 .)	P.36
shift()	Joint 좌표값을 이동합니다 . (자가 업데이트합니다 .)	P.36

*) Joint 좌표값
프로그램에서 취급 좌표입니다 .
교시 좌표뿐만 아니라 교시하지 않는 좌표도 처리할 수 있습니다 .

생성자	Joint() i611 MCS
기능	Joint 좌표계의 교시 포인트를 정의하는 인스턴스를 만듭니다 .
인수	[Joint] [j1, j2, j3, j4, j5, j6] : List Keyword 인수를 생략하면 기본값이 설정됩니다 . (스카라 로봇에서 j5, j6 는 무시됩니다 .) 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
반환값	자기 참조 (Joint 개체)

사용 예	# 예 1 : 인수를 생략합니다 . (초기값 설정합니다 .)
	J1=Joint() → 실행 결과 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
	# 예 2 : 리스트
	J2=Joint(1, 1, 1, 1) → [1.0, 1.0, 1.0, 1.0, 0.0, 0.0]
	# 예 3 : 키워드 (6 개)
	J3=Joint(j1=1, j2=2, j3=3, j4=4) → [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	# 예 4 : 키워드 (1 개). 지정하지 않은 값은 초기값이됩니다 .
	J4=Joint(j4 = 4) → [0.0, 0.0, 0.0, 4.0, 0.0, 0.0]

메소드	clear()	i611 MCS
기능	Joint 좌표값을 초기화합니다 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
인수	없음	
반환값	없음	

사용 예	# 임의의 Joint 개체 J1 를 초기화합니다 .
	#J1 을 선언합니다 .
	J1=Joint(1, 2, 3, 4)
	
	#J1 을 초기화합니다 .
	J1.clear()
	
	실행 결과
	J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	↓
	clear()
	↓
	J1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	

메소드	copy()	i611 MCS
기능	Joint 좌표값을 복사합니다 .	
인수	없음	
반환값	복사한 새로운 Joint 좌표 객체 (Joint 객체)	

사용 예	# 임의의 Joint 개체 J1 를 복사합니다 .
	#J1 을 선언합니다 .
	J1=Joint(j1=1, j2=2, j3=3, j4=4)
	
	#J1 을 새로운 Joint 개체 J1C 에 복사합니다 .
	J1C=J1.copy()
	
	실행 결과
	J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	↓
	copy()
	↓새로운 포인트 객체 : J1C
	J1C : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]	

메소드	jnt2dict()	i611 MCS
기능	Joint 좌표값을 딕셔너리 형식으로 가져옵니다 .	
인수	없음	
반환값	[Joint] : Dict.	

사용 예	# 임의의 Joint 개체 J1 을 딕셔너리 형식으로 출력합니다 .
	#J1 을 선언합니다 .
	J1=Joint(1, 2, 3, 4)
	
	#J1 을 딕셔너리 형식으로 출력합니다 .
	J1.jnt2dict()
	
	실행 결과
	{'j4': 4.0, 'j5': 0.0, 'j6': 0.0, 'j1': 1.0, 'j2': 2.0, 'j3': 3.0}	

메소드	jnt2list() i611 MCS
기능	Joint 좌표값을 리스트 형식으로 가져옵니다 .
인수	없음
반환값	[Joint] : List

사용 예	# 임의의 Joint 개체 J1 을 목록 형식으로 출력합니다 .
	#J1 을 선언합니다 .
	J1=Joint(1, 2, 3, 4)
	
	#J1 을 리스트 형식으로 출력합니다 .
	J1.jnt2list() → 실행 결과 [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]

메소드	offset() i611 MCS
기능	Joint 좌표값에 오프셋 좌표값을 추가합니다 . (자신을 유지하면서 새로운 객체를 생성합니다 .)
인수	[dj1, dj2, dj3, dj4, dj5, dj6] : List Keyword dj1, dj2, dj3, dj4, dj5, dj6 [deg] float : 관절 각도 오프셋 량 (초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]) 인수를 생략하면 오프셋은 설정되지 않습니다 .
반환값	새로운 각 축의 각도 객체 (Joint 객체)

사용 예	# 임의의 Joint 개체 J1 오프셋 좌표값을 추가합니다 .
	#J1 을 선언합니다 .
	J1=Joint(1, 2, 3, 4)
	
	#J1 에서 오프셋된 새로운 Joint 개체 J1ofs 를 생성합니다 .
	J1ofs=J1.offset(dj1=80, dj4=–30)
	#J1 에서 오프셋하고 싶은 값을 숫자만 지정하는 경우
	# 리스트 형식으로 dj1, dj2 을 설정하는 방법
	J2ofs=J1.offset(80, –30)
	
	실행 결과
	J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	↓
	offset()
	↓
	원래 객체 :J1
	J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
	
	새로운 포인트 객체 : : J1ofs
	J1ofs : [81.0, 2.0, 3.0, -26.0, 5.0, 0.0]
	
	새로운 포인트 객체 : J2ofs
	J2ofs : [81.0, -28.0, 3.0, 4.0, 0.0, 0.0]

메소드	replace() i611 MCS
기능	Joint 좌표값을 대체합니다 . (자가 업데이트)
인수	[Joint] : List Keyword 인수를 생략하면 값이 업데이트되지 않습니다 .
반환값	자기 참조 (Joint 객체)

사용 예

```
# 임의의 Joint 개체 J1, J2, J3 를 대체합니다 .
    #J1, J2, J3 을 선언합니다 .
    J1=Joint()
    J2=Joint()
    J3=Joint()

    # 예 1 : 리스트에 지정하는 경우
    J1.replace( 1, 2, 3, 4, 0, 0 )

    # 예2 : 2 개의 가변 인수로 지정하는 경우
    J2.replace( 7, 8 )

    # 예3 : 2 개의 키워드로 지정하는 경우
    J3.replace( j1=1, j4=4 )
```

실행 결과

```
J1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
↓ replace()
J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]

J2 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
↓ replace()
J2 : [7.0, 8.0, 0.0, 0.0, 0.0, 0.0]

J3 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
↓ replace()
J3 : [1.0, 0.0, 0.0, 4.0, 0.0, 0.0]
```

메소드	shift() i611 MCS
기능	Joint 좌표값을 이동합니다 (자가 업데이트)
인수	[dj1, dj2, dj3, dj4, dj5, dj6] : List Keyword dj1, dj2, dj3, dj4, dj5, dj6 [deg] float — 관절 각도의 이동량 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] 인수를 생략하면 이동량은 설정되지 않습니다 .
반환값	자기 참조 (Joint 개체)

사용 예

```
# 임의의 Joint 개체 J1 을 이동합니다 .
    #J1 을 선언합니다 .
    J1.replace( 1, 2, 3, 4 )

    #J1 을 옮겨 자가 업데이트합니다 .
    J1.shift( dj1=80 )
```

실행 결과

```
J1 : [1.0, 2.0, 3.0, 4.0, 0.0, 0.0]
↓
shift()
↓
원래 객체 : J1
J1 : [81.0, 2.0, 3.0, 4.0, 0.0, 0.0]
```

클래스

MotionParam

로봇의 동작 매개변수를 취급합니다 .

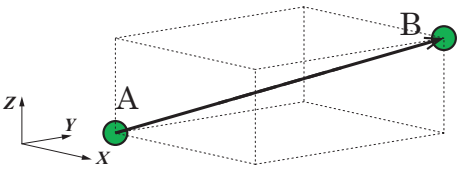
멤버 변수		
lin_speed	속도 (Line 동작 (직선 보간 동작))	P.38
jnt_speed	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작)	P.38
acctime	가속 시간	P.38
dacctime	감속 시간	P.39
posture	자세	P.39
passm	Pass 동작	P.39
overlap	오버랩 거리	P.40
zone	위치 결정 완료 범위	P.40
pose_speed	속도 (자세 보간 동작)(*)	P.41
ik_solver_option	회전 방향	P.41
메소드		
clear()	동작 매개변수를 초기화합니다 .	P.43
confdefault()	동작 매개변수의 초기값을 설정합니다 .	P.43
copy()	동작 매개변수를 복사합니다 .	P.44
motionparam()	동작 매개변수를 설정합니다	P.45
mp2dict()	동작 매개변수를 딕셔너리 형식으로 가져옵니다 .	P.46
mp2list()	동작 매개변수를 리스트 형식으로 가져옵니다 .	P.46

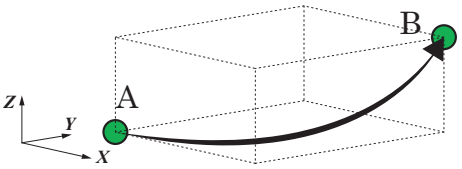
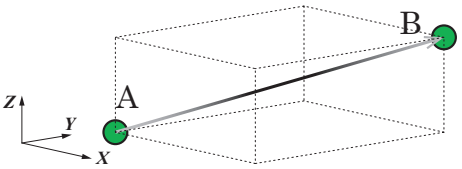
*) 매니퓰레이터 선단이 방향을 바꾸면서 작동할 때 , 선단의 오일러 각도 동작 속도의 상한을 설정합니다 .

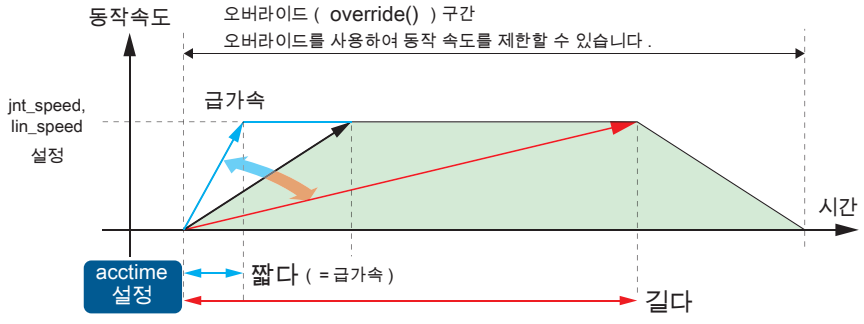
무리한 급가속 / 급감속을 설정하면 로봇이 진동하는 원인이 됩니다 .

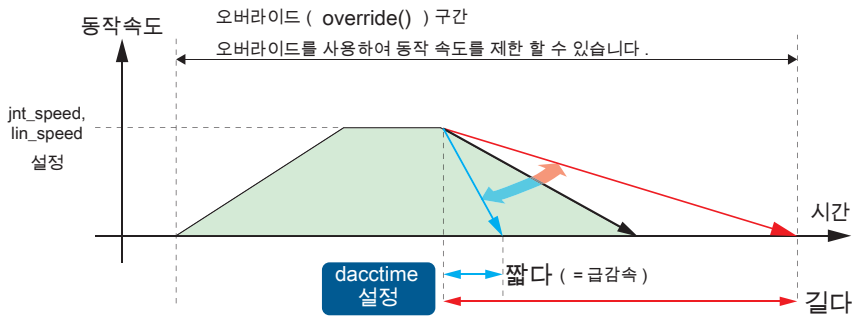
매개변수의 조정은 처음에는 완만하게 가속 · 감속을 해서 로봇에 진동이 발생하지 않는 것을 확인하면서 해야합니다 .

MotionParam 형의 멤버 변수

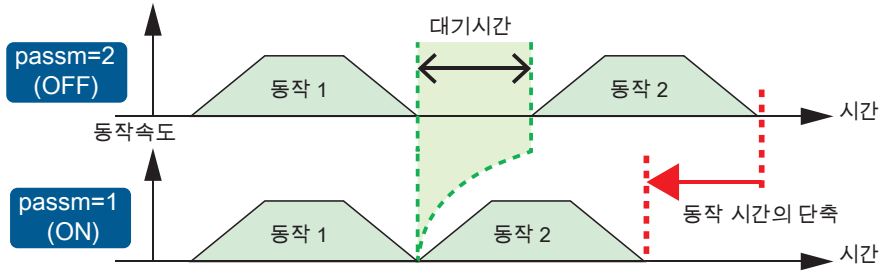
멤버 변수	lin_speed		
기능	속도 (Line 동작 (직선 보간 동작))	초기값 단위· 형	5.0 [mm/s] float
보충	X-Y-Z 축을 동기 제어하면서 목적지까지의 궤적이 직선이 되도록 일정한 속도로 이동하는 동작		

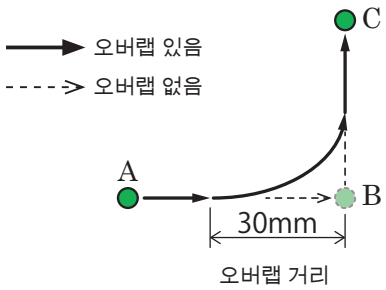
멤버 변수	jnt_speed		
기능	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작)	초기값 단위· 형	5.0 [%] float
보충	모든 관절이 목표 좌표를 향해 등속도 , 각도로 동작 . 부드러운 곡선을 그리며 이동하는 동작 . 최적 직선 보간 동작도 jnt_speed 에서 속도를 설정합니다 . 변속하기 위해 최고 속도에 대한 % 로 설정합니다 .		PTP 동작 , Joint 동작  최적 직선 보간 동작 

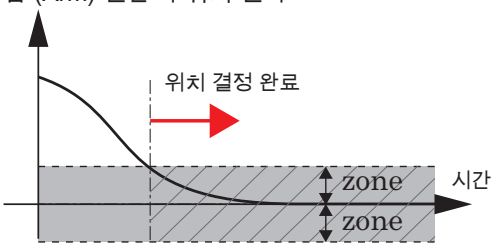
멤버 변수	acctime		
기능	가속 시간	초기값 단위· 형	0.4 [s] float
보충	lin_speed, jnt_speed 에서 설정한 속도에 도달하는 시간을 설정합니다 . 		

멤버 변수	dacctime		
기능	감속 시간	초기값 단위· 형	0.4 [s] float
보충	lin_speed, jnt_speed 에서 설정한 속도에서 목표 좌표에 감속 정지할 때까지의 시간 .  		

멤버 변수	posture		
기능	자세	초기값 단위· 형	0 [-] integer
보충	스카라 로봇은 자세가 존재하지 않습니다 .		

멤버 변수	passm		
기능	Pass 동작	초기값 단위· 형	2 [-] integer
보충	설정값 : 1=ON , 2=OFF passm 동작 매개변수를 ON 하면 동작 사이의 대기 시간을 생략하고 전체 동작 시간을 단축할 수 있습니다 .  asyncm (1) 오버랩 (예측 읽기) 를 사용하면 passm 동작 매개변수의 설정에 관계없이 항상 ON 으로 같은 동작을 합니다 .		

멤버 변수	overlap		
기능	오버랩 거리	초기값 단위· 형	0.0 [mm] float
보충	목표 지점 (B) 에 접근한 시점에서 다음 동작이 겹쳐져 있는 동작을 시작합니다 . 장애물을 피하기 등의 동작을 하기 위해 준비한 경유 지점 (B) 에서 동작을 정지시키지 않고 로봇을 부드럽게 움직일 수 있습니다 .	 예 : 오버랩 =30mm	

멤버 변수	zone		
기능	위치 결정 완료 범위	초기값 단위· 형	100 [pulse] integer
보충	로봇 팔 끝이 목표 지점에 접근하고 위치 결정 완료 판정을 하는 엔코더 펄스 범위를 설정합니다 .	암 (Arm) 선단의 위치 편차 	

멤버 변수	pose_speed		
기능	속도 (자세 보간 동작)	초기값 단위· 형	20 [%] float
보충	설정값 : 100% = 매니퓰레이터 최고 속도 사양 매니퓰레이터 선단이 방향을 바꾸면서 작동할 때 , 끝 오일러 각도의 동작 속도의 상한을 설정합니다 .		

멤버 변수	ik_solver_option		
기능	회전 방향	초기값 단위· 형	0x11111111 [-] long
보충	Position 형으로 지정된 좌표에 동작이나 Position 형에서 Joint 형식으로 변환할 때의 각 축의 회전 방향을 지정합니다 . $0x\underset{(\mathrm{Rsv.})}{1}\ 1\ \underset{J6}{1}\ \underset{J5}{1}\ \underset{J4}{1}\ \underset{J3}{1}\ \underset{J2}{1}\ \underset{J1}{1}$ J1 - J6 설정 0 : 지름길 의 정보를 사용하지 않는 회전입니다 . 1 : multiturn 매개변수의 정보를 사용합니다 . 2 : + 방향으로 회전합니다 . 3 : - 방향으로 회전합니다 . + 방향 , - 방향은 좌표계 항목을 참조 * 관절의 개수에 따라 해당 값이 적용됩니다 . 예를 들어 , 4 축 로봇에서 J5, J6 값은 무시됩니다		

생성자	MotionParam() i611 MCS
기능	로봇의 동작 매개변수 클래스의 인스턴스를 생성한다 .
인수	[MotionParam] [lin_speed, jnt_speed, acctime, dacctime, posture, passm, overlap, zone, pose_speed, ik_solver_option] : List Keyword 인수를 생략하면 기본값이 설정됩니다 . 초기값 : [5.0, 5.0, 0.4, 0.4, 0, 2, 0.0, 100, 20, 0x11111111]
반환값	자기 참조 (MotionParam 개체)

사용 예	# 예 1 : 인수를 생략합니다 . (초기값을 설정)
	m=MotionParam()
	
	실행 결과
	초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	↓
	MotionParam
	↓
	동작 매개변수
	m : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	
	# 예 2 : 인수에 지정된 동작 매개변수를 설정합니다 .
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	
	초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	↓
	MotionParam (lin_speed, jnt_speed, overlap)
	인수 지정
	↓
	동작 매개변수
	m : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]

메소드	clear() i611 MCS
기능	동작 매개변수를 초기화합니다 . 초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
인수	없음
반환값	없음

사용 예	# 임의의 MotionParam 객체 m 을 초기화합니다 .
	#m 을 선언합니다 .
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	#m 을 초기화합니다 .
	m.clear()
	실행 결과
	m : [70.0, 1.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
	↓
	clear()
	↓
	m : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]

메소드	confdefault() i611 MCS
기능	동작 매개변수의 초기값을 변경합니다 .
인수	[MotionParam] : Keyword 인수를 생략하면 기본값이 설정됩니다 . 초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
반환값	없음
사용 예	m.clear() m.confdefault(lin_speed=70, overlap=30) m.clear() 실행 결과 변경 전 초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111] ↓ confdefault ↓ 변경 후 초기값 : [70.0, 5.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111] lin_speed overlap

동작 매개변수의 초기값이 설정되는 타이밍

confdefault () 메소드에서 설정을 변경한 초기값은 그 다음에 MotionParam 클래스의 인스턴스를 생성 , 복사 , 삭제될 때 반영됩니다 .

메소드	copy() i611 MCS
기능	동작 매개변수를 복사합니다 .
인수	[MotionParam] : List Keyword 인수를 지정하면 현재 설정되어 있는 동작 매개변수를 변경하여 복사합니다 . 인수를 생략하면 현재 설정되어 있는 동작 매개변수의 값이 복사됩니다 .
반환값	복사한 새로운 동작 매개변수 (MotionParam 객체)

사용 예

#MotionParam 에서 동작 매개변수를 미리 설정해야합니다 .

```
m=MotionParam( jnt_speed=10, lin_speed=70, overlap=30 )
```

예 1 : 인수를 생략합니다 . (현재 설정되어있는 동작 매개변수)

```
mcopy = m.copy()
```

실행 결과

```
초기값          : [ 5.0,  5.0, 0.4, 0.4, 2, 2,  0.0, 100, 20.0, 0x11111111]
  ↓
MotionParam     (lin_speed ↓, jnt_speed ↓, overlap ↓)
  ↓
동작 매개변수
  m             : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
  ↓
copy            ( 인수 생략 )
  ↓
복사한 동작 매개변수
mcopy           : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
```

예 2 : 인수에 지정된 동작 매개변수를 변경하여 복사합니다 .

```
mcopy = m.copy( jnt_speed=15 )
```

```
동작 매개변수
    m           : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
    ↓
copy            (jnt_speed ↓)
인수 지정
    ↓
복사한 동작 매개변수
  mcopy         : [70.0, 15.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
```

메소드	motionparam() i611 MCS
기능	동작 매개변수를 설정합니다 .
인수	[MotionParam] : List Keyword 인수를 생략하면 기본값이 설정됩니다 . 초기값 : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
반환값	자기 참조 (MotionParam 개체)

사용 예

#MotionParam 에서 동작 매개변수를 미리 설정해야 합니다 .

```
m=MotionParam()
```

예 1 : 인수를 생략합니다 . (초기값을 설정)

```
mm=m.motionparam()
```

실행 결과

```
m  : [ 5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
↓
motionparam  ( 인수 생략 )
↓
mm : [ 5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
```

예 2 : 동작 매개변수를 변경합니다 .

```
mm=m.motionparam( lin_speed=70, jnt_speed=10, overlap=30 )
```

```
m          :  [ 5.0,  5.0, 0.4, 0.4, 2, 2,  0.0, 100, 20.0, 0x11111111]
↓
motionparam          lin_speed   jnt_speed        overlap
인수 설정
↓
mm         :  [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
```

메소드	mp2dict() i611 MCS
기능	동작 매개변수를 딕셔너리 형식으로 획득합니다 .
인수	없음
반환값	[MotionParam] : Dict.

사용 예	# 예 : MotionParam 에서 동작 매개변수를 미리 설정하십시오 .
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	
	modict = m.mp2dict()
	실행 결과
	{'lin_speed': 70.0, 'zone': 100, 'acctime': 0.400, 'pose_speed': 20.0,
	'dacctime': 0.400, 'overlap': 30.0, 'passm': 2, 'jnt_speed': 10.0, 'posture': 0}
	# 직접 값을 얻습니다 .
	lin_speed = m.mp2dict()['lin_speed'] → lin_speed=70.0

메소드	mp2list() i611 MCS
기능	동작 매개변수를 리스트 형식으로 획득합니다 .
인수	없음
반환값	[MotionParam] : List

사용 예	# 예 : MotionParam 에서 동작 매개변수를 미리 설정하십시오 .
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	
	실행 결과
	molist=m.mp2list() → [70.0, 10.0, 0.400, 0.400, 2, 2, 30.0, 100, 20.0]
	
	# 지정 수량만 획득합니다 .
	molist=m.mp2list()[0:2] → print molist
	[70.0, 10.0]
	
	# 직접 값을 획득합니다 .
	lin_speed=m.mp2list()[0] → print lin_speed
	lin_speed=70.0

4 로봇 라이브러리

(이어서 i611Robot 메소드)

메소드		
release_stopevent()	발생 중인 예외 이벤트를 재설정합니다 .	P.70
reljntmove()	Joint 좌표계에서 상대 동작을 합니다 .	P.71
relline()	직교 좌표계에서 상대 직선 보간 동작을 합니다 .	P.72
restart	일시 정지에서의 재개신호를 발행합니다 .	P.73
set_behavior()	일시 정지때의 동작 (행동) 을 설정합니다 .	P.74
set_mdo()	MDO 동작을 설정합니다 .	P.75
settool()	Tool 오프셋을 설정합니다 .	P.76
sleep()	지정된 시간 동안 일시 정지합니다 .	P.77
stop()	로봇을 감속 정지합니다 .	P.78
svoff()	서보를 OFF 로 설정합니다 .	P.78
svstat()	서보 상태를 얻습니다 .	P.79
toolmove()	Tool 좌표계에서 상대 동작을 합니다 .	P.79
use_mt()	크로스오버 카운터의 활성화 / 비활성화를 설정합니다 .	P.80
user_hook()	로봇 프로그램을 일시 정지합니다 .	P.80
version()	시스템 버전을 얻습니다 .	P.81

생성자	i611Robot() i611 MCS
기능	i611 클래스의 인스턴스를 만듭니다 .
인수	[host, port] : List Keyword
	host [-] string — 대상 IP 주소 초기값 : '127.0.0.1'
	port [-] integer — 연결 포트 번호 초기값 : 12345
	인수를 생략하면 기본값이 설정됩니다 .
반환값	자신의 클래스 객체를 반환합니다 .

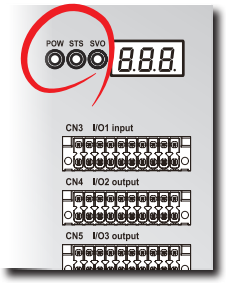
사용 예	# 예 1 : 인수를 생략합니다 .(초기값을 설정합니다 .)
	rb=i611Robot()
	
	# 예 2 : 리스트
	rb=i611Robot('127.0.0.1', 12345)
	
	# 예 3 : 키워드 인수 (모두 지정)
	rb= i611Robot(host='127.1.1.1', port=10000)
	
	# 예 4 : 키워드 인수 (host 만 지정)
	rb= i611Robot(host='127.1.1.1')
	
	# 예 5 : 키워드 인수 (port 만 지정)
	rb= i611Robot(port=3000)

포인트!

i611Robot 클래스 제한

서보 OFF 상태에서 i611Robot 클래스를 사용할 수 없습니다 .
(시동시 상태 · 비상 정지 · 주 전원 OFF · 시스템 에러시 포함)

서보 ON 의 상태는 컨트롤러의 LED (SVO) 도 표시됩니다 .



i611Robot 인스턴스는 하나의 프로그램에서 1 번만 생성할 수 있습니다 .

프로그램 안에서 루프를 할 때는 i611Robot 클래스의 인스턴스는 루프 전에 생성하십시오 .

메소드	abort()	i611 MCS
기능	로봇 프로그램을 중단합니다 .(*)	
인수	없음	
반환값	없음	
사용 예	rb.abort()	

*) 로봇이 동작 중인 경우만 활성화합니다 .

메소드	adjust_mt()	i611 MCS
기능	Position 형 좌표값을 문자열로 변환할 때의 크로스오버 카운터 값을 보정합니다 .	
인수	원본으로 사용하는 Position 좌표로 변환한 문자열 [pos, str_x, str_y, str_z, str_rz, str_ry, str_rx] : List	
	pos 필 수	[Position] : 원본으로 사용하는 Position 좌표
	str_x 필 수 [-] string	x 좌표의 문자열
	str_y 필 수 [-] string	y 좌표의 문자열
	str_z 필 수 [-] string	z 좌표의 문자열
	str_rz 필 수 [-] string	rz 좌표의 문자열
	str_ry 필 수 [-] string	ry 좌표의 문자열
	str_rx 필 수 [-] string	rx 좌표의 문자열
반환값	보정한 크로스오버 카운터 값	
사용 예	rb = i611Robot() rb.open() pos = rb.getpos() pos_value = pos.position() pos_str = [str (round(x,2)) for x in pos_value[0:6]] new_mt = rb.adjust_mt(pos, pos_str[0], pos_str[1], pos_str[2], pos_str[3], pos_str[4], pos_str[5]) pos_str += [str (pos_value[7]), "0x%06X" % new_mt] print "Position String:%s" % pos_str	

메소드	asyncm() i611 MCS	
기능	로봇 프로그램의 예측 동작 구간을 설정합니다 .	
인수	sw [-] integer	1 : 프로그램 예측 동작 ON 2 : 프로그램 예측 동작 OFF (초기값)
반환값	성공한 경우 : True [-] bool	
	실패한 경우 : 예외가 발생합니다 .	

사용 예	rb.line(p10) # 교시 포인트 p10 에 직선 보간 운동합니다 .
	rb.asyncm(sw=1) # 프로그램 예측 동작 ON (rb.asyncm (1) 에서도 가능)
	rb.line(p20,p21) # 교시 포인트 p20 과 p21 에 순서대로 직선 보간 동작으로 이동합니다 .
	
	rb.join() # 예측된 로봇 프로그램 동작의 완료를 기다립니다 .
	
	rb.asyncm(sw=2) # 프로그램 예측 동작 OFF (rb.asyncm (2) 에서도 가능)
	
	...
	rb.close()	

메소드	cause_user_error()	i611 MCS
기능	사용자 정의 에러를 발생시킵니다 .	
인수	[code, critical] : List Keyword	
	code 필 수 [-] integer	에러 ID 설정 범위 : 1 – 99
	critical [-] bool	True : 사용자 정의 에러 치명적 발생 False : 사용자 정의 에러 발생 (초기값)
반환값	없음	

사용 예	# 사용자 정의 에러 (에러 ID : 19) 을 발생시키는 경우
	rb.cause_user_error(19, False)
	# 사용자 정의 에러 - 치명적 (에러 ID : 01) 을 발생시키는 경우
	rb.cause_user_error(01, True)	

사용자 정의 에러에 대해

사용자가 에러 ID (임의 생략 불가) 를 사용하여 cause_user_error () 메소드를 실행하면 에러 상태가 로봇 프로그램은 종료합니다 .

사용자 정의 에러	에러 리셋 방법	7 세그먼트 LED 표시
치명적	전원의 재투입	
에러	' 에러 리셋 신호 '	

메소드	changetool()	i611 MCS
기능	Tool 오프셋을 선택합니다 .	
인수	tid 필 수 [-] integer	Tool 번호 0 : Tool 오프셋을 해제합니다 . 1 – 8 : Tool 오프셋을 선택합니다 .
반환값	성공한 경우 : True [-] bool	
	실패한 경우 : 예외가 발생합니다 .	
사용 예	# 1 . 미리 Tool 오프셋을 설정 해둡니다 . rb.settool(1, 0.0, 0.0, –100.0, 0.0, 0.0, 0.0) # Tool No.1 을 설정합니다 . # 2 . Tool 오프셋을 선택합니다 . # 예 1 : 수치 지정 rb.changetool(1) # Tool No.1 을 선택 # 예 2 : 키워드 rb.changetool(tid=1) # Tool No.1 을 선택	

메소드	check_ready()	i611 MCS
기능	로봇이 자동 운전할 수 있는지를 확인합니다 .	
인수	없음	
반환값	res [-] integer	자동 운전할 수 있습니다 . 0 : 로봇 프로그램 실행 자동 운전은 할 수 없습니다 . 로봇의 동작을 수반하지 않는 프로그램을 실행할 수 있습니다 . 1 : 비상 정지 중입니다 . 2 : 서보 OFF 입니다 . 3 : 자동 모드가 아닙니다 . (JOG 스틱 연결 중 등) 4 : 조작 권한이 잡히지 않습니다 . 5 : 기타 에러 발생 중입니다 .
사용 예	res = i611Robot.check_ready() if res != 0: … # 메시지 표시와 외부 출력 통지 등	

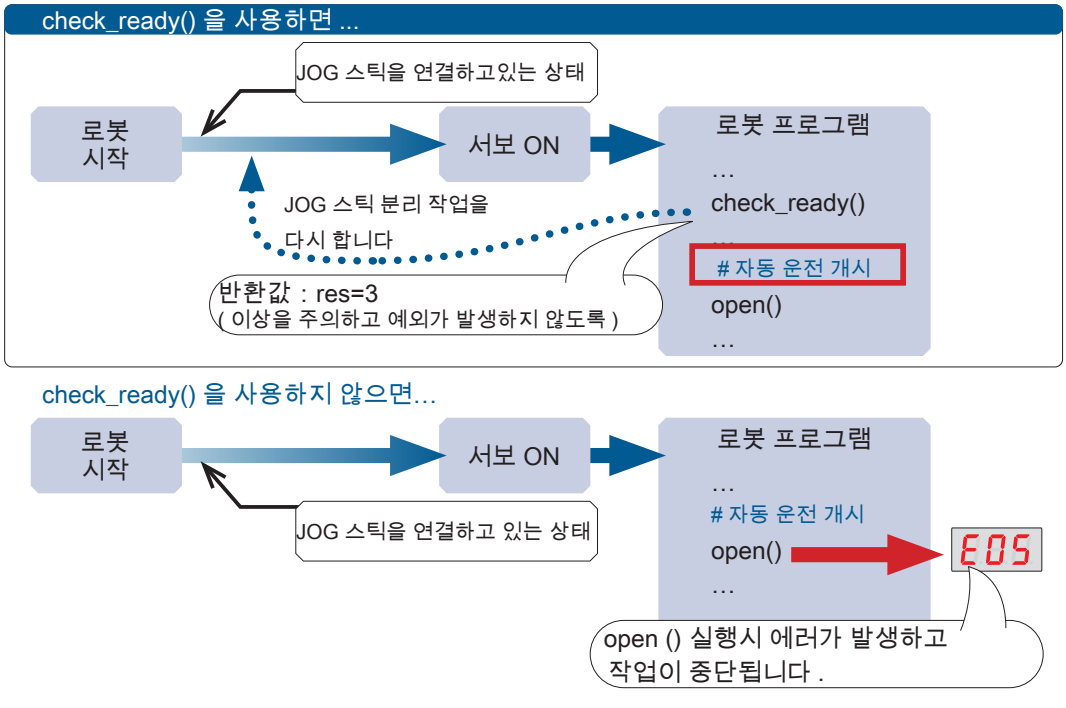
포인트!

check_ready() 메소드의 사용법

i611Robot 클래스의 open () 메소드를 수행 할 수 있는지를 딕셔너리에 확인하는 메소드입니다 .

반환값이 '0' 이 아닌 경우 로봇은 동작하지 않습니다 .
i611Robot 클래스의 생성자 또는 open () 실행시 예외가 발생합니다 .
이 메소드에서 상태를 확인하면 예외 발생을 방지할 수 있습니다 .

예) 컨트롤러에 JOG 스틱을 연결한 채로 자동 운전 프로그램을 실행하면 ...



메소드	close()	i611 MCS
기능	로봇과의 연결을 종료합니다 .	
인수	없음	
반환값	성공한 경우 : True [-] bool (성공시에만 돌아갑니다 .)	
사용 예	rb.close()	

exit () 와 close () 에 대해

exit () 처리는 close () 처리도 이루어집니다 .

메소드	disable_mdo()	i611 MCS
기능	MDO 동작을 비활성화합니다 .	
인수	bitfield 필 수 [-] integer	MDO 관리 번호 (*1) 비활성화하는 MOD 관리 번호에 해당하는 bit 를 세웁니다 . 설정 범위 : 0 – 255
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .	

사용 예	# 미리 MDO 동작 설정을 해둡니다 (*2)
	rb.set_mdo(1, 23, 0, 1, 30)
	rb.set_mdo(8, 23, 1, 2, 10)
	rb.enable_mdo(129) # MDO 관리 번호 1, 8 활성화
	
	#MDO 동작을 비활성화합니다 .
	# 예 1 : 수치 지정
	rb.disable_mdo(129) # MDO 관리 번호 1, 8 비활성화
	# 예 2 : 키워드
	rb.disable_mdo(bitfield=129) # MDO 관리 번호 1 , 8 비활성화	

* 1) 비트 필드의 설정은 56 페이지의「비트 필드 의한 MDO 관리 번호 설정」을 참조하십시오 .
* 2) set_mdo () 의 자세한 내용은 75 페이지를 , enable_mdo () 은 56 페이지를 참조하십시오 .

메소드	enable_interrupt() i611 MCS	
기능	감속 정지와 비상 정지의 예외 발생을 설정합니다 .	
인수	[eid, enable] : List	
	eid 필 수 [-] integer	이벤트 ID 0 : 동작 중 감속 정지 입력시의 예외 발생 1 : 동작 중 비상 정지 입력시의 예외 발생 2 : 일시 정지 중 감속 정지 입력시의 예외 발생 3 : 일시 정지 중 비상 정지 입력시의 예외 발생 예외 발생을 비활성화한 경우 , 로봇 프로그램을 정상적으로 종료합니다 .
	enable 필 수 [-] bool	예외 발생 True : 활성화 False : 비활성화
반환값	res0 [-] bool	True : 성공 False : 실패

사용 예	# 예 1 : 동작 중의 감속 정지 입력시의 예외 발생을 활성화합니다 .
	rb.enable_interrupt(0, True)
	
	# 예 2 : 동작 중의 비상 정지 입력시의 예외 발생을 활성화합니다 .
	rb.enable_interrupt(1, True)
	
	# 예 3 : 일시 정지 중의 감속 정지 입력시의 예외 발생을 비활성화합니다 .
	rb.enable_interrupt(2, False)
	
	# 예 4 : 일시 정지 중 비상 정지 입력시의 예외 발생을 비활성화합니다 .
	rb.enable_interrupt(3, False)	

메소드	enable_mdo()	i611 MCS
기능	MDO 동작 (*1) 을 활성화합니다 .	
인수	bitfield 필 수 [-] integer	MDO 관리 번호 (*2) 활성화 MOD 관리 번호에 해당하는 bit 를 세웁니다 . 설정 범위 : 0 – 255
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .	

사용 예	# 미리 MDO 동작 설정을 해둡니다 (*3)
	rb.set_mdo(1, 23, 0, 1, 30)
	rb.set_mdo(8, 23, 1, 2, 10)
	
	#MDO 동작을 활성화합니다
	# 예 1 : 수치 지정
	rb.enable_mdo(129) # MDO 관리 번호 1, 8 을 활성화합니다 .
	# 예 2 : 키워드
	rb.enable_mdo(bitfield=129) # MDO 관리 번호 1, 8 을 활성화합니다 .	

* 1) MDO 동작은 동작에 지정된 조건에서 I / O 출력을 단락하거나 개방하는 기능입니다 .
* 2) 비트 필드의 설정은 56 페이지의「비트 필드에 의한 MDO 관리 번호 설정」을 참조하십시오 .
* 3) set_mdo () 의 자세한 내용은 75 페이지를 참조하십시오 .

포인트 !

비트 필드에 의한 MDO 관리 번호 설정

enable_mdo () 는 각 mdo 동작을 한 번에 ON / OFF 를 전환할 수 있습니다 .

예) 관리 번호 8 과 1 을 활성화합니다 .

```
rb.enable_mdo(bitfield=129)
```

수치지정 8 . . . 수치지정 1
1000 0001
관리 번호 1 을 활성화합니다 .
관리 번호 8 을 활성화합니다 .

포인트 !

enable_mdo () 에서 한 번에 설정 가능한 수

enable_mdo() 메소드에서 활성화할 수있는 MDO 의 개수는 총 4 개입니다 . set_mdo () 메소드에서 설정한 MDO 에서 선택하십시오 .

enable_mdo() 을 여러 번에 나누어 실행해도 5 개 이상을 동시에 활성화할 수 없습니다 .

메소드	exit()	i611 MCS
기능	로봇 프로그램을 강제종료합니다 .	
인수	res [-] integer	0 : 정상종료 0 기타 : 이상종료
반환값	없음	
사용 예	rb.exit(0)	

포인트 !

exit() 메소드에 대해

로봇 프로그램을 성공적으로 완료하면 이 메소드는 필요없습니다 .

- exit() 처리는 close () 처리도 이루어집니다 .
- exit() 메소드는 표준 Python 라이브러리 sys 모듈의 sys.exit () 와 동일합니다 .

인수에 0 이외를 지정하여 로봇 프로그램을 종료한 경우

컨트롤러는 시스템정의에러가 E 14 되고 , 로봇 프로그램은 비정상 종료합니다 . 이 에러에서 복구 I/O 에 'reset' 신호를 입력합니다 .

포트 할당은 「 3 메모리 맵 」 을 참조하십시오 .

메소드	get_hw_info()	i611 MCS
기능	모델명과 시리얼 번호를 얻습니다 .	
인수	없음	
반환값	[model name, serial number] : List	
	model name [-] string	모델명
	serial number [-] string	일련 번호
사용 예	model, serial = i611Robot.get_hw_info()	

이 메소드는 스태틱 메소드입니다 . i611Robot 클래스의 인스턴스 정의를 하지 않고도 호출할 수 있습니다

메소드	get_system_port()	i611 MCS
기능	시스템 포트의 상태를 획득합니다 .	
인수	없음	
반환값	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error, (rsv.)] : List	
	running [-] integer	로봇 프로그램 상태 1 : 실행 중 (컨트롤러 LED 지시자 (STS) 에 표시) 0 : 정지 중
	svon [-] integer	서보 상태 1 : 서보 ON 중 0 : 서보 OFF 중
	emo [-] integer	비상 정지 상태 1 : 비상 정지 중 0 : 없음
	hw_error [-] integer	시스템 정의 에러 (치명적) 상태 1 : 에러발생 중 0 : 없음
	sw_error [-] integer	시스템 정의 에러 상태 1 : 에러 발생 중 0 : 없음
	abs_lost [-] integer	ABS 소실 상태 1 : ABS 소실 중 0 : 없음
	in_pause [-] integer	일시 정지 상태 1 : 일시 정지 중 0 : 없음
	error [-] integer	시스템 에러 상태 (*) 1 : 시스템 에러 발생 중 0 : 없음
	(rsv.) [-] integer	(예약)

사용 예

```
# 개별 시스템의 상태를 확인합니다 .
    running = rb.get_system_port()[0]      # 로봇 프로그램 상태
    svon = rb.get_system_port()[1]         # 서보 상태
    emo = rb.get_system_port()[2]          # 비상 정지 상태
    hw_error = rb.get_system_port()[3]     # 시스템 정의 에러 ( 치명적 ) 상태
    sw_error = rb.get_system_port()[4]     # 시스템 정의 에러 상태
    abs_lost = rb.get_system_port()[5]     # ABS 소실 상태
    in_pause = rb.get_system_port()[6]     # 일시 정지 상태
    error = rb.get_system_port()[7]        # 시스템 에러 상태
```

메소드	get_system_status()	i611 MCS
기능	시스템 상태와 에러 ID 를 획득합니다 .	
인수	없음	
반환값	[status, err_id] : List	
	status [-] integer	시스템 상태 [컨트롤러의 표기] 1 : 시동 중 [ini] 2 : 대기 상태 [rdy] 3 : ABS 소실 상태 [inc] 4 : 교시 중 [tch] 6 : 로봇 프로그램 실행 중 [run] 10 : 시스템 정의 에러 발생 중 [E88] 11 : 시스템 정의 에러 (치명적) 발생 중 [c88] 12 : 유저 정의 에러 발생 중 [u88] 13 : 유저 정의 에러 (치명적) 발생 중 [r88] (88 : 에러 ID)
	err_id [-] integer	에러 ID 에러 ID 는 에러 발생시 7 세그먼트 LED 표시 장치에 나타나는 값입니다 (정상 시 0)
사용 예	# 스태틱 메소드로 호출 status, err_id = i611Robot.get_system_status()	

이 메소드는 스태틱 메소드입니다 . i611 Robot 클래스의 인스턴스 정의를 하지 않아도 호출가능합니다 .

메소드	getjnt()	i611 MCS
기능	매니퓰레이터의 현재 위치를 Joint 형으로 얻습니다 .	
인수	없음	
반환값	성공했을 경우 : [Joint] : List	
	실패했을 경우 : 예외가 발생합니다 .	
사용 예	rb.home() pos01=rb.getjnt()	

메소드	getmotionparam()	i611 MCS
기능	현재의 동작 매개변수를 얻습니다 .	
인수	없음	
반환값	성공한 경우 : [MotionParam] : List MotionParam 클래스로 설정된 요소를 참조할 수 있습니다 . 실패한 경우 : 예외가 발생합니다 .	

사용 예	#MotionParam 의 인스턴스를 참조합니다 .
	t_lin_speed=rb.getmotionparam().lin_speed
	t_lin_overlap=rb.getmotionparam().overlap	

MotionParam 형식에 대한 자세한 내용은 P.37~ 를 참조하십시오 ..

메소드	getpos()	i611 MCS
기능	매니퓰레이터의 현재 위치를 Position 형으로 얻습니다 .	
인수	없음	
반환값	성공한 경우 : [Position] : List 실패한 경우 : 예외가 발생합니다 .	
사용 예	rb.home() pos01=rb.getpos()	

메소드	home()	i611 MCS
기능	모든 축의 Joint 값이 0 deg 이 되도록 이동합니다 .	
인수	없음	
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .	
사용 예	rb.home()	

메소드	is_open()	i611 MCS
기능	i611 Robot 의 오픈 상태를 확인한다 .	
인수	없음	
반환값	res0 [-] bool	True : 오픈 중 False : 오픈되지 않음
사용 예	if not rb.is_open(): ... # 오픈되어 있지 않은 경우 실행할 명령을 기술합니다 .	

메소드	is_pause()	i611 MCS
기능	로봇 프로그램의 일시 정지 중인 상태를 확인합니다 .	
인수	없음	
반환값	res0 [-] bool	True : 일시 정지 중 False : 일시 정지 중이 아닙니다 .

사용 예 (*)

```
## 별도 스레드에서 일시 정지 , 재기동의 상태를 감시합니다 .
def thread_fnc(rb):
    while not thread_end:
# 상태를 확인
        pause_st = rb.is_pause()
        print 'This status is {}.'.format(pause_st)
        print "th:wait stop",din(DIN_STOP)
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            rb.pause()
        if din(DIN_RESTART) == "1":
            rb.restart()

# 예 ) 로봇 프로그램 샘플
    try:
        while True:
          #· · line(),move() 등의 동작 프로그램 설명· ·
          # user hook 를 이용해 일시 정지
            rb.user_hook()
                #· · line(),move() 등의 동작 프로그램 설명· ·
    except Robot_emo:                    # 비상 정지 스위치 누름 이벤트 핸들러
      #· · · ·
    except Robot_stop:                   # 감속 정지 입력 감지 이벤트 핸들러
      #· · · ·
    finally:
        rb.close()
```

*) 다음 페이지의 사용 예를 참고하여 주십시오 .

포인트!

is_pause() 메소드의 사용 예의 보충

is_pause() 메소드를 사용하기 위해서 필요한 준비

- 별도 스레드의 인스턴스를 생성하여 주십시오 .
  예 : threadTest=threading.Thread(target=thread_fnc,args=[rb])

- 별도 스레드의 Daemon 을 설정하여 주십시오 .
  예 : threadTest.setDaemon(True)

- 별도 스레드를 시작하여 주십시오 .
  예 : threadTest.start()

- 일시 정지 등의 동작을 설정하여 주십시오 . (정지 위치는 set_behavior() 의 no_pause 플래그의 값에 따라 변경됩니다 .)
  예 : rb.set_behavior(only_hook=False,servo_off=False,restore_position=True,no_pause=False)

- 동작 중에 감속 정지 , 비상 정지의 예외 인터럽트 처리 활성화를 설정합니다 .
  (기본값은 전부 비활성화 (False) 입니다 .)
  예 : rb.enable_interrupt(0,True) # 동작 중에 감속 정지 입력 시의 예외 발생을 활성화
  rb.enable_interrupt(1,True) # 동작 중에 비상 정지 입력 시의 예외 발생을 활성화

- 동작 매개변수를 설정합니다 .
  예 : Cnt0 = MotionParam(lin_speed=70, jnt_speed=10,acctime=0.4,dacctime=0.4,overlap=100.0)
  rb.motionparam(Cnt0)

- I/O 를 정의합니다 .
  예 : DIN_STOP = 7 # 감속 정지
  DIN_PAUSE = 8 # 일시 정지
  DIN_RESTART = 9 # 재기동 .

메소드	join() i611 MCS
기능	예측된 로봇 프로그램의 동작 완료를 기다립니다 .
인수	없음
반환값	실패한 경우 : 예외가 발생합니다 .(실패할 경우에만 발생합니다 .)

사용 예	rb.line(P10) # 교시 포인트 P10 으로 직선 보간 운동
	rb.asyncm(sw=1) # 프로그램 예측 동작 ON (개시)
	rb.line(P20) # 교시 포인트 P20 으로 직선 보간 운동합니다 .
	rb.line(P21) # 교시 포인트 P21 으로 직선 보간 운동합니다 .
	rb.join() # 예측된 로봇 프로그램의 동작을 완료를 기다립니다 .
	
	rb.asyncm(sw=2) # 프로그램 예측 동작 OFF (종료)
	...
	rb.close()

포인트!

asyncm() 와 join() 메소드의 사용법

asyncm (sw=2) 을 실행하기 전에 , join() 메소드의 실행하는 것으로 예측된 동작 명령의 실행을 완료할 때까지 대기합니다 .

메소드	Joint2Position() i611 MCS
기능	Joint 좌표값을 Position 좌표값으로 변환합니다 .
인수	[Joint] : List 필 수 (인수는 생략할 수 없습니다 .)
반환값	성공한 경우 : [Position] : List 실패한 경우 : 예외가 발생합니다 .

사용 예	#Joint 좌표값
	j10=Joint(0, 30, 60, 0, 90, 90)
	#Position 좌표값 변환 (j10 →변환→ p10)
	p10=rb.Joint2Position(j10)
	실행 결과
	J10 : [0, 30, -60, 0, 90, 90]
	↓Joint2Position()
	P10 : [373.20499999999998, 100.0, -60.0, 30.000012857270395, 0.0, 0.0, < ... >, 0]

메소드	line()	i611 MCS
기능	직선 보간 동작 실행합니다 .	
인수	[Position] [Joint] [MotionParam] : List 1 개 이상의 Position 형 , Joint 형의 위치 좌표와 , MotionParam 형의 동작 매개변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 , 이후의 동작은 변경된 동작 매개변수로 실행됩니다 .	
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .	

사용 예	# 예 1
	# 위치 좌표 p10 으로 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 으로 주어진 조건에 따릅니다 .
	rb.line(p10)
	
	# 예 2
	# 위치 좌표 p10 으로 향한 뒤 , p20 으로 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 의 설정에 따릅니다 .
	rb.line(p10, p20)	

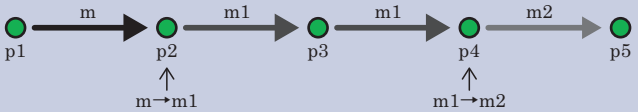
포인트 !

line() 과 move() 에 관하여

미리 Positon 형 또는 , Joint 형의 위치 좌표를 정의합니다 .

예 : p1=Position(–50, –250, 350, 90, 0, 180)

```
# 예 : motionparam() 메소드로 설정한 동작 매개변수를 변경하면서 p1 에서 p5 으로 이동합니다 #####
m=MotionParam()
m.motionparam()
rb.line(p1, p2)                          동작 매개변수 m 으로 p1 에서 p2 으로 이동
m1=m.motionparam( lin_speed=6, jnt_speed=20 )   동작 매개변수 m 을 m1 으로 변경
rb.line(p3, p4)                          동작 매개변수 m1 으로 p2 에서 p3 을 경유하여 p4 으로 이동
m2=m.motionparam( lin_speed=7, jnt_speed=40 )   동작 매개변수 m1 을 m2 으로 변경
rb.line(p5) # move()                     동작 매개변수 m2 으로 p4 에서 p5 로 이동
```



i611 MCS

실행 결과

```
[0, 3, 2, 4]
```

메소드	motionparam() i611 MCS
기능	동작 매개변수를 설정합니다 .
인수	[MotionParam] : Keyword List
	인수를 생략할 시 기본값으로 설정됩니다 .
반환값	성공한 경우 : True [-] bool
	실패한 경우 : 예외가 발생합니다 .

사용 예	# 예 1 : MotionParam 형의 인스턴스로 설정합니다
	m=MotionParam()
	rb.motionparam(m)
	
	# 예 2 : MotionParam 형의 멤버 변수의 키워드로 설정합니다
	rb.motionparam(posture=0, passm=1, overlap=4.8, zone=20, pose_speed=5.0)

MotionParam 형의 상세 설명은 P. 37 ~ 을 참고하여 주십시오 .

메소드	move() i611 MCS
기능	PTP 이동을 합니다
인수	[Position] 또는 [Joint] 또는 [MotionParam] : List 1 개 이상의 Position 형 , Joint 형의 위치 좌표와 MotionParam 형의 동작 매개변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 이후의 동작은 변경된 동작 매개변수로 실행됩니다 .
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .
사용 예	

```
# 예 1
#   위치 좌표 p10 으로 PTP 이동을 합니다
#   동작 조건은 , MotionParam 으로 주어진 조건에 따릅니다 .
    rb.move( p10 )

# 예 2
#   위치 좌표 p10 으로 이동한 뒤 , p20 으로 PTP 이동을 합니다 .
#   동작 조건은 , MotionParam 으로 주어진 조건에 따릅니다 .
    rb.move( p10, p20 )

# 예 3
#   크로스오버 카운터 정보를 사용
    rb.use_mt(True)
    …
    rb.move( p10 )
    …
    rb.close()
```

포인트!

크로스오버 카운터 정보를 사용할 경우

move() 메소드를 호출하기 전에 반드시 , use_mt(True) 를 호출해주십시오 .
use_mt(False) 를 호출 또는 use_mt() 를 동작 중에 한번도 호출하지 않은 경우 , 크로스오버 카운터의 정보를 이용하지 않는 동작입니다 .
크로스오버 카운터에 대한 자세한 내용은 P.3, use_mt() 메소드에 대한 자세한 내용은 P. 80 을 참고하여 주십시오 .
크로스오버 카운터 ('ik_solver_option') 의 설정은 move() 메소드와 Position2Joint() 메소드에서 사용됩니다 .

메소드	open() i611 MCS
기능	로봇과의 연결을 시작합니다 . (초기화합니다 .)
인수	permission [-] bool : 인수는 True 뿐 입니다 . True : 조작 권한을 획득합니다 .(초기값) 인수를 생략할 경우 초기값으로 설정됩니다 .
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .
사용 예	rb.open(True)

포인트!

조작 권한과 open() 메소드에 관하여

조작 권한을 획득 가능한 프로세스는 시스템 전체에서 1 개의 프로세스뿐입니다 .
조작 권한을 획득하지 않고 사용하는 경우엔 , 복수의 프로세스를 생성할 수는 있지만 , 조작권한이 요구되는 메소드를 실행할 시 예외가 발생합니다 ..

open() 의 실행 횟수는 프로그램 중 1 회 뿐입니다 . 한번 close() 를 실행한 후 , 2 번째 open() 을 실행할 경우 예외가 발생합니다 .
반복문을 실행할 경우 , open() 은 반복문의 앞에 작성하여 주십시오 .

메소드	optline() i611 MCS
기능	직선 보간 운동을 최적의 속도로 변속하며 실행합니다
인수	[Position] [Joint] [MotionParam] : List 1 개 이상의 Position 형 , Joint 형의 위치 좌표와 , MotionParam 형의 동작 매개변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 , 이후의 동작은 변경된 동작 매개변수로 실행됩니다 .
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .

사용 예	# 예 1
	# 위치 좌표 p10 으로 최적 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 에서 주어진 조건에 따릅니다 .
	rb.optline(p10)
	
	# 예 2
	# 위치 좌표 p10 으로 이동한 뒤 , p20 으로 최적 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 에서 주어진 조건에 따릅니다 .
	rb.optline(p10, p20)

	optline( ) 의 속도는 , lin_speed (직선 보간 운동) 와 달리 jnt_speed (PTP 동작· Joint 동작· 최적 직선 보간 운동) 에서 설정합니다 .	

메소드	override()	i611 MCS
기능	오버라이드를 실행합니다 .	
인수	ovr 필 수 [%] integer or float	motionparam() 에서 설정한 로봇의 구동 속도에 배율을 적용하여 속도를 조정합니다 . (생략 불가) 설정 범위 : 0 ~ 200
반환값	실패한 경우 : 예외가 발생합니다 .(실패한 경우에만 반환합니다 .)	

사용 예	# 오버라이드를 50%로 설정한다 .
	rb.override(50)	

메소드	pause()	i611 MCS
기능	로봇의 동작을 일시 정지합니다 .(*)	
인수	없음	
반환값	없음	

사용 예

```
## 별도 스레드에서 일시 정지 상태를 감시합니다 .
def thread_fnc(rb):
    while not thread_end:
        pause_st = rb.is_pause()
        print 'This status is {}.'.format(pause_st)
        print "th:wait stop",din(DIN_STOP)
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            # 일시 정지합니다 .
            rb.pause()
        if din(DIN_RESTART) == "1":
            rb.restart()

# 예 ) 로봇 프로그램 샘플
    try:
        while True:
            #· · line(),move() 등의 동작 프로그램 설명· ·
            # user hook 으로 일시 정지
            rb.user_hook()
                #· · line(),move() 등의 동작 프로그램 설명· ·
    except Robot_emo:                    # 비상 정지 SW 누름 이벤트 핸들러
      #· · · ·
    except Robot_stop:                   # 감속 정지 입력감지 이벤트 핸들러
      #· · · ·
    finally:
        rb.close()
```

메소드	Position2Joint() i611 MCS
기능	Position 좌표값을 Joint 좌표값으로 변경합니다 .
인수	[Position] : List 필 수 (인수는 생략할 수 없습니다 .)
반환값	성공한 경우 : [Joint] : List 실패한 경우 : 예외가 발생합니다 .

사용 예	#Position 형 좌표값
	p10=Position(200, 0, –80, 90, 0, 0)
	#Joint 형 좌표값으로 변경 (p10 → 변경 → j10)
	j10=rb.Position2Joint(p10)
	실행 결과
	p10 : [200, 00, –80, 90, 0, 0]
	↓Position2Joint()
	j10 : [-60, 120, -80.0, 90.0, 0.0, 0.0]

메소드	release_stopevent() i611 MCS
기능	발생 중의 예외 이벤트를 리셋합니다 .(*)
인수	없음
반환값	없음
사용 예	try: … # 동작 except Robot_stop: rb.release_stopevent() … # 대피 동작 등

*)· 예외 처리의 가장 앞에 작성하여 주십시오 .
· 리셋하기 전까지 예외가 반복하여 발생합니다 .

메소드	reljntmove() i611 MCS	
기능	Joint 좌표계 대한 상대동작을 합니다	
인수	[dj1, dj2, dj3, dj4, dj5, dj6] : Keyword	
	dj1 [deg] float	J1 축의 이동량
	dj2 [deg] float	J2 축의 이동량
	dj3 [deg] float	J3 축의 이동량
	dj4 [deg] float	J4 축의 이동량
	dj5 [deg] float	J5 축의 이동량
	dj6 [deg] float	J6 축의 이동량
반환값	없음	

사용 예	# Joint 형의 좌표값을 준비합니다 .
	J1 = Joint(45, 45, –45, –45, 0, 0)
	# 동작 매개변수를 설정합니다 .
	m=MotionParam(jnt_speed=10, lin_speed=70, overlap=30)
	rb.motionparam(m)
	...
	rb.move(J1)
	
	#Joint 좌표계에서 J1 을 35 도 만큼 오프셋시킨 위치로 이동합니다 .
	rb.reljntmove(dj1=35)	

메소드	relline()	i611 MCS
기능	직교 좌표계에서 상대 직선 보간 운동을 한다	
인수	[dx, dy, dz, drz, dry, drx] : Keyword	
	dx [mm] float	X 축 방향으로 오프셋량
	dy [mm] float	Y 축 방향으로 오프셋량
	dz [mm] float	Z 축 방향으로 오프셋량
	drz [deg] float	Rz 축을 중심으로 오프셋량
	dry [deg] float	Ry 축을 중심으로 오프셋량
	drx [deg] float	Rx 축을 중심으로 오프셋량
반환값	없음	

사용 예	#Position 형의 좌표값을 준비합니다 .(*)
	P10 = Position(95, –280, -40, 154, 0, 0, posture = 0)
	# 동작 매개변수를 설정합니다 .
	m=MotionParam(jnt_speed=10, lin_speed=70, overlap=30)
	rb.motionparam(m)
	...
	
	rb.move(P10)
	# 직교 좌표계에서 X 축 방향으로 15mm 오프셋한 위치로 이동합니다 .
	rb.relline(dx=15)	

*) 예의 교시 포인트는 단순화시킨 것입니다 . 실제 교시을 통해 획득한 교시 데이터를 이용해 주십시오 .

메소드	restart() i611 MCS
기능	일시 정지 상태로부터 재기동 신호를 보냅니다 .
인수	없음
반환값	없음 실제로 재기동하기 전에 처리가 되돌아옵니다 . 메인 스레드 이외의 별도 스레드에서 호출해 주십시오 .
사용 예	

```
## 별도 스레드에서 재기동 시키기 .
def thread_fnc(rb):
    while not thread_end:
        pause_st = rb.is_pause()
        print 'This status is {}.'.format(pause_st)
        print "th:wait stop",din(DIN_STOP)
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            rb.pause()
        if din(DIN_RESTART) == "1":
                #   재기동 시키기
            rb.restart()

# 예 ) 로봇 프로그램 샘플
    try:
        while True:
          #・・line(), move() 등의 동작 프로그램 기재・・
          # user hook 를 이용하여 일시 정지
            rb.user_hook()
                #・・line(), move() 등의 동작 프로그램 기재・・
    except Robot_emo:                    # 비상정지 SW 누름 이벤트 핸들러
      #・・・・
    except Robot_stop:                   # 감속 정지 입력 감지 이벤트 핸들러
      #・・・・
    finally:
        rb.close()
```

메소드	set_behavior()	i611 MCS
기능	일시 정지의 방법을 설정합니다 .	
인수	[only_hook, servo_off, restore_position, no_pause] : List Keyword	
	only_hook [-] bool	user_hook() 으로만 일시 정지를 가능하도록 합니다 . True : 활성화 False : 비활성화 (기본값)
	servo_off [-] bool	일시 정지 시에 서보를 OFF 상태로 전환한다 . True : 활성화 False : 비활성화 (기본값)
	restore_position [-] bool	일시 정지 후 재기동 시(*1) 위치를 일시 정지 전으로 되돌립니다 .(*2) True : 활성화 False : 비활성화 (기본값)
	no_pause [-] bool	일시 정지를 동작의 구분 (*3) 만으로 실행 True : 활성화 (시스템 버전 R0.5.0 와 호환) False : 비활성화 *4) (기본값)
	인수를 생략할 시 기본값으로 설정됩니다 .	
반환값	없음	

사용 예	# 일시 정지 후 재기동 시에 , 위치를 일시 정지 전으로 되돌립니다 .
	rb.set_behavior(only_hook=False, servo_off=False, restore_position=True, no_pause=True)	

*1) 동작의 재개는 , 서보를 ON 시킨 후 실행 (run) 하여 주십시오 .

*2) 일시 정지 후 다시 서보를 ON 시키며 위치가 틀어지게된 경우에도 , 일시 정지 전의 위치로 돌아가서 동작을 재개할 수 있습니다 .
동작 중에 일시 정지한 경우는 , 이 설정과 관계없이 원위치로 돌아가서 재기동시켜 주십시오 .

*3) 동작의 구분은 , i611Robot 클래스 메소드가 호출되었을 때 동작을 완료한 직후를 말합니다 .
일시 정지를 원하는 경우에는 동작과 관련된 메소드를 정기적으로 호출하거나 user_hook() 메소드를 삽입하여 주십시오 .

*4) 동작의 구분 혹은 동작 중에 일시 정지합니다 .

메소드	set_mdo()	i611 MCS
기능	MDO 동작을 설정합니다 .	
인수	[mdoid, portno, value, kind, distance] : List Keyword	
	mdoid 필 수 [-] integer	MDO 관리 번호 설정 범위 : 1 ~ 8
	portno 필 수 [-] integer	포트 출력 번호 설정 범위 : 0 ~ 12,287
	value 필 수 [-] integer	I/O 출력 0 : LOW 1 : HIGH
	kind 필 수 [-] integer	조건 1 : 시작점에서 일정 범위 떨어져 있음 2 : 끝점에서 일정 범위 내로 접근
	distance 필 수 [mm] float	거리 설정 범위 : 0.0 ~
반환값	성공한 경우 : True [-] bool	
	실패한 경우 : 예외가 발생합니다 .	
사용 예	# 예 1 : 수치 지정 rb.set_mdo(1, 23, 0, 1, 30) # MDO 관리 번호 1 에 설정 rb.set_mdo(8, 23, 1, 2, 10) # MDO 관리 번호 8 에 설정 # 예 2 : 키워드 rb.set_mdo(mdoid=1, portno=23, value=0, kind=1, distance=30) # MDO 관리 번호 1 에 설정 rb.set_mdo(mdoid=8, portno=23, value=1, kind=2, distance=10) # MDO 관리 번호 8 에 설정	

포인트 !

MDO 동작

MDO : Middle Digital Out

동작 중에 지정된 조건에서 I/O 출력을 LOW/HIGH 로 전환하는 기능입니다 .

예) 출력을 LOW 로 합니다 .
value=0
kind=1
distance=30

예) 출력을 HIGH 로 합니다 .
value=1
kind=2
distance=10

메소드	settool()	i611 MCS
기능	Tool 오프셋을 설정합니다 .	
인수	[id, offx, offy, offz, offrz, offry, offrx] : List Keyword	
	id 필 수 [-] integer	Tool 번호 0 : Tool 오프셋을 해제합니다 . 1 – 8 : Tool 오프셋을 선택합니다 .
	offx [mm] float	Tool 좌표계에서 X 축의 Tool 오프셋량
	offy [mm] float	Tool 좌표계에서 Y 축의 Tool 오프셋량
	offz [mm] float	Tool 좌표계에서 Z 축의 Tool 오프셋량
	offrz [deg] float	Tool 좌표계에서 Rz 축을 중심으로 회전하는 Tool 오프셋량
	offry [deg] float	Tool 좌표계에서 Ry 축을 중심으로 회전하는 Tool 오프셋량
	offrx [deg] float	Tool 좌표계에서 Rx 축을 중심으로 회전하는 Tool 오프셋량
	초기값 : [0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
반환값	성공한 경우 : True [-] bool	
	실패한 경우 : 예외가 발생합니다 .	

사용 예	# 1 . Tool 오프셋 (Tool No.1) 을 설정합니다 .
	# 예 1 : 수치 지정
	rb.settool(1, 0.0, 0.0, –100.0, 0.0, 0.0, 0.0)
	# 예 2 : 키워드
	rb.settool(id=1, offx=0.0, offy=0.0, offz=–20.0, offrz=0.0, offry=0.0, offrx=0.0)
	
	# 2 . Tool 오프셋에서 Tool No.1 을 선택합니다 .
	# 예 1 : 수치 지정
	rb.changetool(1)
	# 예 2 : 키워드
	rb.changetool(tid=1)	

Tool 번호의 인수명은 changetool() 메소드와 settool() 메소드에서 다르게 사용됩니다 .

메소드	Tool 번호의 인수명
changetool()	tid
settool()	id

메소드	sleep() i611 MCS
기능	처리를 일시 정지합니다 . 일시 정지하는 시간을 인수로 설정합니다 .
인수	sec 필 수 [s] integer — 일시 정지하는 시간
반환값	없음
사용 예	#Exception 을 검출할 시 , 정상 종료 try. # 5 초 동안 처리를 일시 정지합니다 . rb.sleep(sec=5) ... except Robot_emo # 비상 정지 SW 누름 이벤트 핸들러 (복귀 불능) # 필요한 에러 처리 [ex) 종료 처리] 를 기재합니다 .

*) Robot_emo() 클래스를 활성화하기 위해 , 미리 enable_interrupt() 을 기술하여 주십시오 .
(☞ enable_interrupt()…P. 55, Robot_emo()…P. 109)
예) enable_interrupt(1,True) # 동작 중 비상 정지 입력 시의 예외 발생을 활성화한다 .

sleep() 메소드와 Python 의 sleep 함수

Python 의 sleep 함수는 , 일시 정지 중에 비상 정지 스위치가 눌러지더라도 비상 정지의 예외 처리를 발생시킬 수 없습니다 . 로봇 라이브러리의 sleep() 메소드를 사용하면 sleep 중에도 로봇 관련 예외를 발생시킬 수 있게 됩니다 .

메소드	stop() i611 MCS
기능	로봇을 감속 정지합니다 .(*)
인수	없음
반환값	없음

사용 예

```
## 별도 스레드에서 감속 정지를 명령합니다 .
def thread_fnc(rb):
    while not thread_end:
        pause_st = rb.is_pause()
        print 'This status is {}.'.format(pause_st)
        print "th:wait stop",din(DIN_STOP)
    #   감속 정지
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            rb.pause()
        if din(DIN_RESTART) == "1":
            rb.restart()

# 예 ) 로봇 프로그램 샘플
    try:
        while True:
          # line(), move() 등의 동작 프로그램 기재
          # user hook 를 이용하여 일시 정지
            rb.user_hook()
              # line(), move() 등의 동작 프로그램 기재
    except Robot_emo:                    # 비상정지 SW 누름 이벤트 핸들러
      #· · · ·
    except Robot_stop:                   # 감속 정지 입력 감지 이벤트 핸들러
      #· · · ·
    finally:
        rb.close()
```

*) 자동 운전 중 이외의 상태에서는 다음 메소드를 사용하여 정지시킵니다
정지시킬 수 있는 메소드 :
abort() 는 로봇 동작 중에만 정지가 가능하며 , 그 외에의 경우에선 정지하지 않습니다 . (다음 프로그램 실행으로 넘어갑니다)

관련 메소드
재기동 확인메소드 : is_pause()
정지 위치 설정메소드 : set_behavior()

메소드	svoff() i611 MCS
기능	서보를 OFF 로 설정합니다 .
인수	없음
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .
사용 예	rb.svoff()

메소드	svstat()	i611 MCS
기능	서보 상태를 얻습니다 .	
인수	없음	
반환값	state [-] integer	성공 시에만 반환값이 있습니다 . 1 : 서보 ON 0 : 서보 OFF –1 : 비상정지 중
사용 예	if rb.svstat() == 1: # 서보 ON ... elif rb.svstat() == 0: # 서보 OFF ... elif rb.svstat() == –1: # 비상 정지 중	

메소드	toolmove()	i611 MCS
기능	Tool 좌표계를 기준으로 상대 동작을 합니다 .	
인수	[dx, dy, dz, drz, dry, drx] : List Keyword	
	dx [mm] float	Tool 좌표계에서 X 축 방향으로의 이동량
	dy [mm] float	Tool 좌표계에서 Y 축 방향으로의 이동량
	dz [mm] float	Tool 좌표계에서 Z 축 방향으로의 이동량
	drz [deg] float	Tool 좌표계에서 Rz 를 중심으로 회전량
	dry [deg] float	Tool 좌표계에서 Ry 를 중심으로 회전량
	drx [deg] float	Tool 좌표계에서 Rx 를 중심으로 회전량
	기본값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
반환값	성공한 경우 : True [-] bool	
	실패한 경우 : 예외가 발생합니다 .	

사용 예	# 예 : Positon 형 [dx, dy, dz, drz, dry, drx] 의 교시 데이터를 리스트로 정의합니다 .
	p10=Position(95, –280, –40, 154, 0, 0, posture = 0)
	
	# 예 : 좌표위치 p10 으로 이동 후 , Tool 좌표계를 기준으로 dx=15mm 만큼 이동합니다 .
	...
	rb.move(p10)
	rb.toolmove(dx=15)
	...
	rb.close()	

메소드	use_mt()	i611 MCS
기능	크로스오버 카운터의 활성화 / 비활성화를 설정합니다 .	
인수	mt [-] bool	True : 활성화 False : 비활성화 (기본값 : 시스템 버전 R 0.5.0 호환)
반환값	없음	

사용 예	# 크로스오버 카운터 정보를 사용합니다 .
	rb.use_mt(True)	

크로스오버 카운터에 관한 메소드

크로스오버 카운터를 활성화시킬 경우 아래의 API 의 구동이 바뀌게 됩니다

[생성자]

Position()

[메소드]

Position 클래스 : replace(),pos2list(),pos2dict(),position(),motionparam()

i611Robot 클래스 : getpos(),Joint2Position(),Position2Joint(),move()

메소드	user_hook()	i611 MCS
기능	로봇 프로그램을 일시 정지합니다 .	
인수	없음	
반환값	없음	
사용 예	... rb.user_hook() # 이 위치에서 프로그램을 일시 정지합니다	

user_hook() 메소드에 관하여

Robsys 클래스의 로봇 제어 명령 이외의 처리에 일시 정지하고 싶은 장소에서 , 이 메소드를 배치합니다 .
로봇 프로그램 상의 특정 부분에서만 일시 정지를 할 경우에는 , set_behavior() 에서「user_hook 으로만 일시 정지를 가능하도록 합니다 .」를 금지한 상태로 , 로봇 프로그램을 일시 정지하고 싶은 위치에 user_hook() 를 배치합니다 .

메소드	version()	i611 MCS
기능	시스템 버전을 얻습니다 .	
인수	없음	
반환값	[res0, major, minor, patch, build, date, option] : List	
	res0 [-] bool	True : 성공 False: 실패
	major [-] integer	Major Version
	minor [-] integer	Minor Version
	patch [-] integer	Patch Version
	build [-] integer	Build Version
	date [-] string	Build 한 날짜
	option [-] string	Option
사용 예	rb.version() → [True, 0, 6, 9, 7, u'04:37:07 Nov 28 2017', u'SIM No-spiio ']	

MEMO

2. 모듈 : teachdata

클래스	
Teachdata	
교시 데이터 관리	
멤버 변수	
—	—

메소드		
check_format()	교시 데이터 파일의 포맷 버전을 가져옵니다 .	P.84
close()	교시 데이터 파일을 닫습니다 .	P.85
flush()	업데이트된 교시 데이터을 파일로 내보냅니다 .	P.85
get_coordinate()	교시 데이터의 Base 오프셋을 가져옵니다 .	P.86
get_joint()	교시 데이터의 Joint 좌표값을 가져옵니다 .	P.86
get_param()	교시 데이터의 매개 변수를 가져옵니다 .	P.87
get_position()	교시 데이터의 Position 좌표값을 가져옵니다 .	P.88
get_tool()	교시 데이터의 Tool 오프셋을 가져옵니다 .	P.89
is_open()	교시 데이터 파일의 오픈 상태를 확인합니다 .	P.89
open()	교시 데이터 파일을 오픈합니다 .	P.90
set_joint()	교시 데이터의 Joint 좌표값를 업데이트합니다 .	P.90
set_param()	교시 데이터의 매개 변수를 업데이트합니다 .	P.91
set_position()	교시 데이터의 Position 좌표값 업데이트합니다 .	P.92

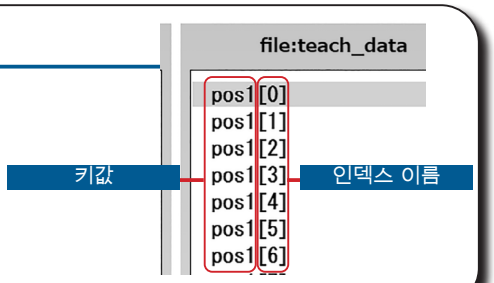
생성자	Teachdata() Teach data
기능	교시 데이터를 로드하여 Teachdata 클래스의 인스턴스를 작성합니다 .
인수	fname [-] string — 교시 데이터의 파일 이름 초기값 : '/home/i611usr/teach_data'
반환값	자기 참조 (Teachdata 클래스 객체)

사용 예	# 교시 데이터 파일를 지정한 경우
	td=Teachdata(fname = './home/i611usr/teach_data')
	
	# 인수를 생략한 경우
	td = Teachdata()

포인트!

「키」와「인덱스」

키 , 인덱스 , 교시 화면에 표시되는 것으로 각각 대응하고 있습니다 .



메소드	check_format() Teach data
기능	교시 데이터 파일의 포맷 버전을 가져옵니다 .
인수	fname 필 수 [-] string — 교시 데이터 파일의 전체 경로
반환값	ver [-] string — " 버전 문자열 "
사용 예	ver = Teachdata.check_format("/home/i611usr/teach_data") 파일 이름

스태틱 메소드에 대한 Teachdata 클래스의 인스턴스는 필요하지 않습니다 .

메소드	close() Teach data
기능	교시 데이터 파일을 닫습니다 .
인수	없음
반환값	없음

사용 예	# 교시 데이터 파일의 종료합니다 .
	td.close()

프로그램 종료시에는 반드시 close () 메소드를 실행하십시오 .

교시 데이터를 Read/Write 모드에서 여는 경우 , 업데이트 데이터를 실제 파일에 내보낸 후 배타적 (독점) 처리를 해제합니다 .

메소드	flush() Teach data
기능	업데이트된 교시 데이터를 파일로 내보냅니다 .
인수	없음
반환값	없음
사용 예	# 교시 데이터 파일에 대한 업데이트를 합니다 . .. td.flush() ... td.close()

· flush () 메소드는 내부적으로 모든 티치 데이터 (.json) 를 내 보냅니다 .
업데이트마다가 아니라 , 업데이트가 일정량 쌓였을 때 수행할 것을 권장합니다 .

· 데이터를 업데이트하는 경우 close () 할 때도 내부에서 실행하고 있습니다 .

메소드	get_coordinate()	Teach data
기능	교시 데이터의 Base offset 을 가져옵니다 .	
인수	index 필 수 [-] integer	Base ID 설정 범위 : 0 – 3 0 : Base 인스턴스를 반환 1 – 3 : Base 좌표계의 Coordinate 인스턴스를 반환
반환값	baseoffset	Base offset 의 인스턴스 인수에 지정한 ID 에 따라 해당 인스턴스가 반환됩니다 . index=0 : Base index=1, 2, 3 : 해당 Base 오프셋의 Coordinate 인스턴스

사용 예	# index 번호 : 1, Base 좌표의 오프셋값을 교시 데이터 파일에서 가져옵니다 .
	baseoffset = td.get_coordinate(1)
	실행 결과 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]	

메소드	get_joint()	Teach data
기능	교시 데이터의 Joint 좌표값을 가져옵니다 .	
인수	[key, index, comment] : List	
	key 필 수 [-] string	Joint 좌표의 키값 설정 범위 : 'Joint1' – 'Joint20'
	index 필 수 [-] integer	Joint 좌표의 인덱스 설정 범위 : 0 – 9
	comment [-] bool	Comment 의 획득 플래그 True : 획득 False : 획득하지 않습니다 .
반환값	[[Joint] , comment] : List	
	[Joint] [deg] float	Joint 좌표값
	comment [-] string	Comment (최대 32 문자)
사용 예	#key = joint1, index 번호 = 1 의 Joint 좌표값과 Comment 를 가져옵니다 . jnt, comment = td.get_joint('joint1', 1, True) print jnt.jnt2list() 실행 결과 [0.744, –37.724, –83.660, 45.270, 0.0, 0.0]	

지정된 key 와 index 의 데이터가 존재하지 않을 때는 예외 (Robot_error) 가 발생합니다 .

메소드	get_param() Teach data	
기능	교시 데이터의 매개 변수를 가져옵니다 .	
인수	[key, index, axis, comment] : List	
	key 필 수 [-] string	매개 변수의 키값 설정 범위 : 'param1' – 'param4'
	index 필 수 [-] integer	매개 변수의 인덱스 설정 범위 : 0 – 9
	axis 필 수 [-] integer	매개 변수의 축 번호 설정 범위 : 1 – 8
	comment [-] bool	매개 변수의 Comment 의 획득 플래그 True : 획득 False : 획득하지 않습니다 . (초기값)
반환값	param [-] string	매개 변수 문자열 (최대 32 문자 / 교시 화면에서 입력한 값입니다 .)

사용 예	#key : "param2", index 번호 : 1, 2 번째 교시 데이터 파일을 획득
	param = td.get_param("param2", 1, 2)	

지정된 key, index 와 axis 의 데이터가 존재하지 않을 때는 예외가 발생합니다 .

메소드	get_position() Teach data	
기능	교시 데이터의 Position 좌표값을 가져옵니다 .	
인수	[key, index, tool, base, comment] : List	
	key 필 수 [-] string	Position 좌표의 키값 설정 범위 : 'pos1' – 'pos20'
	index 필 수 [-] integer	Position 좌표의 인덱스 설정 범위 : 0 – 9
	tool [-] bool	Tool ID 의 획득 플래그 True : 획득 False : 획득하지 않습니다 . (초기값)
	base [-] bool	Base ID 의 획득 플래그 True : 획득 False : 획득하지 않습니다 . (초기값)
	comment [-] bool	Comment 의 획득 플래그 True : 획득 False : 획득하지 않습니다 . (초기값)
반환값	[pos, toolid, baseid, comment] : List	
	pos [mm] float	Position 좌표값 ([Position] 객체)
	toolid [-] integer	Tool ID 반환값 : 0 – 8 (0 은 tool 없음)
	baseid [-] integer	Base ID 반환값 : 0 – 3 (0 은 tool 없음)
	comment [-] string	Comment (최대 32 문자 , 없는 경우 ' ')
사용 예	#key = pos1, index 번호 = 1 의 데이터를 가져옵니다 .。 #(Tool ID(True), Base ID(True), Comment (True) 를 가져옵니다 .) pos, toolid, baseid, comment = td.get_position('pos1', 1, True, True, True)	

실행 결과 예

```
pos     : [21.0, 259.94, -50.61, 53.890, 0.0, 0,0, < ... >, 0.0]
toolid  : [3]
baseid  : [0]
comment : [test]
```

지정된 key 와 index 의 데이터가 존재하지 않을 때는 예외가 발생합니다

메소드	get_tool() Teach data	
기능	교시 데이터의 Tool 오프셋을 가져옵니다 .	
인수	index 필 수 [-] integer	Tool ID 설정 범위 : 0 – 8 (0 으로 하면 반환값은 [0, 0, 0, 0, 0, 0] 이 됩니다 .)
반환값	[dx, dy, dz, drz, dry, drx] List	
	dx, dy, dz [mm] float	Tool 오프셋값 위치 (월드 좌표계)
	drz, dry, drx [deg] float	Tool 오프셋값 각도 (Z-Y-X 계 오일러 각)

사용 예	# index 번호 : 1, tool 오프셋 값을 교시 데이터 파일에서 가져옵니다 .
	tooloffset=td.get_tool(1)
	...
	실행 결과
	[1, u'0.00', u'0.00', u'0.00', u'0.00', u'0.00', u'0.00']	

지정된 key 와 index 의 데이터가 존재하지 않을 때는 예외 (Robot_error) 가 발생합니다 .

메소드	is_open() Teach data	
기능	교시 데이터의 오픈 상태를 확인합니다 .	
인수	없음	
반환값	res [-] integer	0 : 오픈하지 않음 1 : ReadOnly 모드 (읽기 전용) 2 : Read/Write 모드 (읽기 / 쓰기)

사용 예	# 교시 데이터 파일의 오픈 상태를 확인합니다 .
	td = Teachdata()
	td.open(readonly=False)
	if td.is_open() == 2:
	return False
	...	

메소드	open()	i611 IO / Teach data
기능	교시 데이터 파일를 오픈합니다 .	
인수	readonly [-] bool	True : ReadOnly 모드 (읽기 전용) 에서 오픈 (초기값) False : Read/Write 모드 (읽기 / 쓰기) 에서 오픈
반환값	없음	

사용 예	#Read only 모드에서 오픈합니다 .
	td = Teachdata()
	td.open(readonly=Ture)
	
	#Read/Write 모드에서 오픈합니다 .
	td = Teachdata()
	td.open(readonly=False)	

・조작모드가「교시」의 경우 열 수 없습니다 .

・Read/Write 모드에서 오픈하면 다른 프로세스에서 Read / Write 모드에서는 열 수 없습니다 .

・교시 데이터 파일의 버전이 미확인 버전 (R1.0.0 이상) 인 경우 , 예외가 발생합니다 .
・교시 데이터 파일 이전 버전의 경우는 읽을 수 있지만 오류 표시가 나옵니다 .
(교시 데이터 파일을 변환할 것을 권장합니다 .)

메소드	set_joint()	i611 IO / Teach data
기능	교시 데이터의 Joint 좌표값을 업데이트합니다 .	
인수	[key, index, jnt, comment] : List	
	key 필 수 [-] string	키 Joint 의 이름 : joint1 – joint20
	index 필 수 [-] integer	인덱스 설정 범위 : 0 – 9
	jnt 필 수 [-] float	업데이트 Joint 좌표값 ([Joint] 객체)
	comment [-] string	Comment 문자열 (최대 32 문자)
반환값	없음	

사용 예	#Joint 형의 키값 : "joint1", index 번호 : 2, Joint 형 좌표 : jnt, Comment : "home" 를 업데이트합니다 .
	jnt = Joint(0, 0, 0, 0, 0, 0)
	td.set_joint("joint1" , 2, jnt, "home")	

・이미 존재하는 데이터에만 사용할 수 있습니다 .
・지정된 key 와 index 가 없는 경우는 예외가 발생합니다 .
・Read / Write 모드가 아닌 경우 예외가 발생합니다 .
・본 메소드 호출 후에 flush () 를 실행하면 파일을 업데이트합니다 .

메소드	set_param() Teach data	
기능	교시 데이터의 매개 변수를 갱신합니다 .	
인수	[key, index, axis, paramstr comment] : List	
	key 필 수 [-] string	키 Param 의 이름 : param1 ~ param4
	index 필 수 [-] integer	인덱스 설정 범위 : 0 – 9
	axis 필 수 [-] integer	축 설정 범위 : 1 – 8
	paramstr 필 수 [-] string	매개 변수 문자열 (최대 32 문자)
	comment [-] string	Comment 문자열 (최대 32 문자)
반환값	없음	

사용 예	# 키 : param2, index 번호 : 3, 축 번호 : 1, 매개 변수 문자열 "1.00" 를 업데이트
	td.set_param("param2", 3, 1, "1.00")	

- 이미 존재하는 데이터에 대해서만 지정할 수 있습니다 .
  지정된 key, index 와 axis 의 데이터가 존재하지 않는 경우는 예외가 발생합니다 .
- Read / Write 모드가 아닌 경우는 예외가 발생합니다 .
- 본 메소드 호출 후에 flush () 를 실행하면 파일을 업데이트합니다 .

메소드	set_position() Teach data	
기능	교시 데이터의 Position 좌표값를 업데이트합니다 .	
인수	[key, index, pos, tooloffset, baseoffset, comment] : List	
	key 필 수 [-] string	키 Position 의 이름 : pos1 ~ pos20
	index 필 수 [-] integer	인덱스 설정 범위 : 0 – 9
	pos 필 수 [-] float	업데이트 Position 좌표값 ([Position] 객체)
	tooloffset [-] integer	이 Position 좌표값을 결정하는 데 사용한 Tool 오프셋의 ID 설정 범위 : 0 – 8 초기값 : 0 (Tool 오프셋을 사용하지 않습니다 .)
	baseoffset [-] integer	이 Position 좌표값를 결정할 때 사용하는 Base 오프셋의 ID 설정 범위 : 0 – 3 초기값 : 0 (Base 오프셋을 사용하지 않습니다 .)
	comment [-] string	Comment 문자열 (최대 32 문자)
반환값	없음	

사용 예	# Position 형의 키값 : pos2, index 번호 : 2, Tool ID : 1 , Base 오프셋 : 사용하지 않음 , Comment " work" 를 업데이트
	pos = Position(95, –280, 425, –120, 0, 0)
	td.set_position("pos2", 2, pos, 1, 0, "work")	

· 이미 존재하는 데이터에 대해서만 지정할 수 있습니다 .
지정된 key 와 index 가없는 경우는 예외가 발생합니다 .
· Read / Write 모드가 아닌 경우는 예외가 발생합니다 .
본 메소드 호출 후에 flush () 를 실행하면 파일을 업데이트합니다 .

3. 모듈 : i611_extend

클래스		
Pallet		
팔레트 기능을 수행합니다 .		
멤버 변수		
-	-	
메소드		
adjust()	셀의 위치를 보정합니다 .	P.94
get_pos()	셀의 위치를 가져옵니다 .	P.95
init_3()	팔레트 정의 (3 점 교시)	P.96
init_4()	팔레트 정의 (4 점 교시)	P.97

생성자	Pallet()	i611 Ext.
기능	팔레트 기능 Pallet 클래스의 인스턴스를 만듭니다 .	
인수	없음	
반환값	자신의 클래스 객체를 반환	

실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 .

메소드	adjust()	i611 MCS / i611 Ext.
기능	셀의 위치를 보정합니다 .	
인수	[i, j, di, dj] : List	
	i 필 수 [-] integer	팔레트의 셀의 위치를 지정하는 인덱스 (i 방향)
	j 필 수 [-] integer	팔레트의 셀의 위치를 지정하는 인덱스 (j 방향)
	di 필 수 [mm] integer	i 방향의 셀의 위치의 오프셋값
	dj 필 수 [mm] integer	j 방향의 셀의 위치의 오프셋값
반환값	없음	

사용 예	# 예 : 팔레트의 4 가지의 모서리의 좌표를 정의하고 팔레트를 정의합니다 . (*)
	pos_0=Position(–100, –100, –100, posture=0)
	pos_1=Position(–170, –100, –100)
	pos_2=Position(–100, –180, –100)
	pos_3=Position(–170, –180, –100)
	pal=Pallet()
	pal.init_4(pos_0, pos_1, pos_2, pos_3, 8, 7)
	
	# 팔레트의 오프셋값을 설정합니다 .
	pal.adjust(4, 4, 10, 10)
	rb.close()	

*) 실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 .

메소드	get_pos()	i611 MCS / i611 Ext.
기능	셀의 위치를 가져옵니다 .	
인수	[i, j, dk] : List	
	i 필 수 [-] integer	팔레트의 셀의 위치를 지정하는 인덱스 (i 방향)
	j 필 수 [-] integer	팔레트의 셀의 위치를 지정하는 인덱스 (j 방향)
	dk [mm] integer	수직방향의 오프셋값 초기값 : 0
	오프셋값을 설정한 경우 팔레트 좌표에서 셀의 k 방향으로 오프셋 좌표를 가져옵니다 . 인수 dk 을 생략한 경우는 초기값이 설정됩니다 .	
반환값	[Position] 팔레트의 위치 (i, j) 의 셀의 좌표	

사용 예

```
# 예 : 팔레트의 4 가지의 모서리의 좌표와 팔레트를 정의합니다 . (*)
    pos_0=Position( –250, –250, –100, posture = 0 )
    pos_1=Position( –170, –250, –100 )
    pos_2=Position( –250, –180, –100 )
    pos_3=Position( –170, –180, –100 )
    pal=Pallet()
    pal.init_4( pos_0, pos_1, pos_2, pos_3, 8, 7 )
    pal.adjust( 4, 4, 10, 10 )

# 오프셋을 포함한 팔레트의 지정된 인덱스의 좌표를 가져옵니다 .
    p00=pal.get_pos( 0, 0, 10 )

# get_pos() 에서 얻은 좌표로 이동합니다 .
    rb.move( p00 )
    rb.close()
```

*) 실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 ..

포인트 !

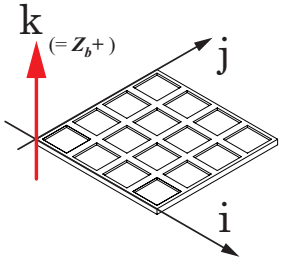
팔레트의 수직 방향

교시 포인트의 배치에 의해 팔레트 수직 방향 (k 방향) 의 양의 방향이 바뀝니다 .

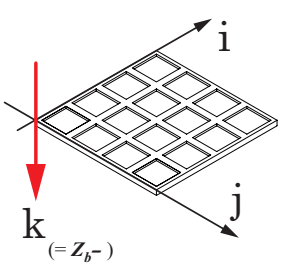
예① $\vec{i} \times \vec{j}$: 평면 (팔레트면) 에 수직인 벡터 z + 입니다 .

예② $\vec{j} \times \vec{i}$: 예①과 반대 방향의 z - 입니다 .

예 ① $\vec{i} \times \vec{j}$



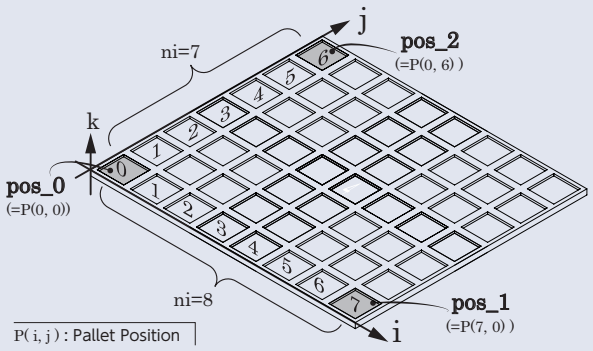
예 ② $\vec{j} \times \vec{i}$



메소드	init_3()	i611 MCS / i611 Ext.
기능	팔레트를 정의합니다 . (3 점 교시)	
인수	[pos_0, pos_i, pos_j, ni, nj] : List	
	pos_0 필 수 [-] float	[Position] 팔레트의 교시 포인트 (원점)
	pos_i 필 수 [-] float	[Position] 팔레트의 교시 포인트 (i 방향)
	pos_j 필 수 [-] float	[Position] 팔레트의 교시 포인트 (j 방향)
	ni 필 수 [-] integer	팔레트의 i 방향에 줄 지어있는 셀 개수
	nj 필 수 [-] integer	팔레트의 j 방향에 줄 지어있는 셀 개수
반환값	res [-] bool	성공했을 때만 돌아갑니다 True : 성공

사용 예

```
# 예 : 팔레트의 세 꼭지점의 좌표를 정의합니다 (*)
    pos_0=Position( –250, –250, –100, posture=0 )
    pos_1=Position( –170, –250, –100 )
    pos_2=Position( –250, –180, –100 )

#3 점 교시 데이터를 사용한 팔레트를 정의합니다 .
    pal=Pallet()
    pal.init_3( pos_0, pos_1, pos_2, 8, 7 )

    rb.close()
```

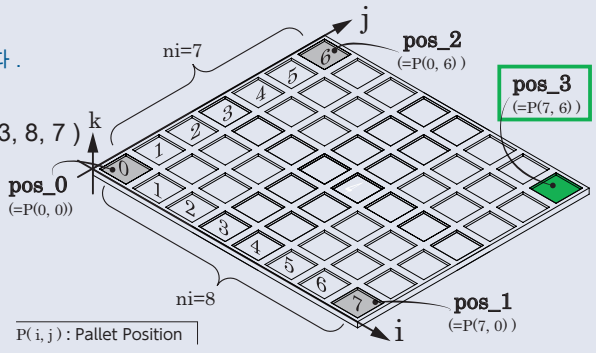


*) 실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 .

메소드	init_4()	i611 MCS / i611 Ext.
기능	팔레트를 정의합니다 . (4 점 교시)	
인수	[pos_0, pos_i, pos_j, pos_ij, ni, nj] : List	
	pos_0 필 수 [-] float	[Position] 팔레트의 교시 포인트 (원점)
	pos_i 필 수 [-] float	[Position] 팔레트의 교시 포인트 (i 방향)
	pos_j 필 수 [-] float	[Position] 팔레트의 교시 포인트 (j 방향)
	pos_ij 필 수 [-] float	[Position] 팔레트의 교시 포인트
	ni 필 수 [-] integer	팔레트의 i 방향에 줄지어 있는 셀 개수
	nj 필 수 [-] integer	팔레트의 j 방향에 줄지어 있는 셀 개수
반환값	res [-] bool	성공했을 때만 돌아갑니다 True : 성공

사용 예

```
# 예 : 팔레트의 네 꼭지점의 좌표를 정의합니다 (*)
    pos_0=Position( –250, –250, –100, posture=0 )
    pos_1=Position( –170, –250, –100 )
    pos_2=Position( –250, –180, –100 )
    pos_3=Position( –170, –180, –100 )

#4 점 교시 데이터를 사용한 팔레트를 정의합니다 .
    pal=Pallet()
    pal.init_4( pos_0, pos_1, pos_2, pos_3, 8, 7 )

    rb.close()
```



*) 실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 .

4. 모듈 : rbsys

클래스	
RobSys	
시스템 매니저 제어	
멤버 변수	
-	-

메소드		
assign_din()	물리적 I/O 와 메모리 I/O 의 입력 포트에 기능을 할당합니다 .	P.99
assign_dout()	물리적 I/O 와 메모리 I/O 의 출력 포트에 기능을 할당합니다 .	P.100
clear_robtask()	로봇 프로그램의 등록을 해제합니다 .	P.101
close()	시스템 매니저와의 접속을 종료합니다 .	P.101
cmd_pause()	동작 명령 : 일시 정지	P.102
cmd_reset()	동작 명령 : 에러 리셋	P.102
cmd_run()	동작 명령 : 로봇 프로그램 실행	P.103
cmd_stop()	동작 명령 : 감속 정지	P.103
get_robtask()	로봇 프로그램의 상태를 얻습니다 .	P.104
open()	시스템 매니저와의 통신을 시작합니다 .	P.104
req_mcmd()	시스템 상태 및 명령 상태를 얻습니다 .	P.105
set_robtask()	로봇 프로그램을 등록합니다 .	P.106
version()	시스템 매니저의 버전 정보를 얻습니다 .	P.107

RobSys 클래스는 설정 스크립트 (init.py) 에서 I/O 제어 및 작업 관리를 위한 관리용 프로그램 인터페이스입니다 .
로봇 프로그램에서는 i611Robot 클래스의 메소드를 이용하고 , RobSys 클래스는 사용하지 마십시오 .

Constructor	RobSys() rbsys	
기능	로봇 시스템의 인스턴스를 작성합니다 .	
인수	host [-] string	연결 대상의 IP 어드레스 지정 초기값 : '127.0.0.1'
	인수를 생략하면 초기값으로 설정됩니다 .	
반환값	자신의 클래스 오브젝트 반환	

사용 예	# 예 1 : 인수 생략 (초기값으로 설정)
	rbs = RobSys()
	
	# 예 2 : 키워드
	rbs = RobSys(host='127.1.1.1')	

메소드	assign_din() i611 IO rbsys
기능	물리적 I/O 와 메모리 I/O 의 입력 포트에 기능을 할당합니다 .
인수	[run, stop, err_reset, pause] : Keyword

run [-] integer	로봇 프로그램 실행 초기값 : –1
stop [-] integer	감속 정지 초기값 : –1
err_reset [-] integer	에러 리셋 초기값 : –1
pause [-] integer	일시 정지 초기값 : –1

설정 범위 :

- 물리적 I/O : 0 – 15
- 메모리 I/O : 32 – 12287

- 할당된 값이 없으면 -1 으로 지정됩니다 .
- 인수를 생략하면 초기값으로 설정됩니다 .
- 동일한 I/O 에 중복해서 할당할 수 없습니다 .
- 할당된 물리적 I/O 는「 3 메모리 맵 」을 참조해주십시오 .
- 설정 범위 0 – 15 는 입력 포트 신호명의 IN1 ~ IN16 (CN3 : I/O 커넥터 1) 에 해당합니다 .
- 입력 신호는 Low → Hi 의 입력 엣지를 검출합니다 .

반환값	
	성공했을 경우 : True [-] bool
	실패했을 경우 : 예외 발생

사용 예

```
#init.py 로 지정합니다 : ( 이하는 권장 설정 )
    rbs.assign_din( run=0, stop=1, err_reset=2, pause=3 )
```

메소드	assign_dout()	i611 IO	rbsys
기능	물리적 I/O 와 메모리 I/O 의 출력 포트에 기능을 할당합니다 .		
인수	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error] : Keyword		
	running [-] integer	로봇 프로그램 상태 초기값 : –1	
	svon [-] integer	서보 상태 초기값 : –1	
	emo [-] integer	비상 정지 상태 초기값 : –1	
	hw_error [-] integer	시스템 정의 에러 (치명적) 상태 초기값 : –1	
	sw_error [-] integer	시스템 정의 에러 상태 초기값 : –1	
	abs_lost [-] integer	ABS 소실 상태 초기값 : –1	
	in_pause [-] integer	일시 정지 상태 초기값 : –1	
	error [-] integer	시스템 에러 상태 (*) 초기값 : –1	
	설정 범위 : · 물리적 I/O : 16 – 31 · 메모리 I/O : 32 – 12287 · 할당된 값이 없으면 -1 으로 지정됩니다 . · 인수를 생략하면 초기값으로 설정됩니다 . · 동일한 I/O 에 중복해서 할당할 수 없습니다 . · 할당된 물리적 I/O 는 「 3 메모리 맵 」 을 참조해주십시오 . · 설정 범위 16 – 31 는 신호명의 O1 ~ O16 (CN4 : I/O 커넥터 2、CN5 : I/O 커넥터 3) 에 해당합니다 . · 출력이 ON 이 되면 , 대응하는 출력 포트가 Hi 가 됩니다 .		
반환값	성공했을 경우 : True [-] bool		
	실패했을 경우 : 예외 발생		

사용 예	# init.py 로 지정합니다 : (이하는 권장 설정)
	rbs.assign_dout(running=16, svon=17, emo=18, hw_error=19, sw_error=20, abs_lost=21, in_pause=22, error=23)		

*) 시스템 정의 에러 (비치명적 또는 치명적) 의 발생 상태를 나타냅니다 .
2개의 에러 상태를 하나의 제어선에서 확인하는 경우에 사용합니다 .

시스템 매니저를 사용하지 않고 로봇 프로그램을 기동했을 경우 (*) 는 running 의 출력은 표시하지 않습니다 .running 을 출력하는 경우에는 시스템 매니저를 통해 다시 시작하여 주세요 .

*) 예를 들면 터미널 소프트웨어로 시작된 경우입니다 .

메소드	clear_robtask()	rbsys
기능	로봇 프로그램의 등록을 해제합니다 .	
인수	없음	
반환값	res0 [-] bool	True : 성공 False : 실패

사용 예	# 성공
	if rbs.clear_robtask()[0] == True:
	
	# 실패
	if rbs.clear_robtask()[0] == False:	

clear_robtask() 메소드는 실행 중인 로봇 프로그램은 정지하지 않습니다 .

메소드	close()	rbsys
기능	시스템 매니저와의 접속을 종료합니다 .	
인수	없음	
반환값	없음	
사용 예	rbs.close()	

메소드	cmd_pause()	rbsys
기능	동작 명령 : 일시 정지	
인수	없음	
반환값	res0 [-] bool	True : 성공 False : 실패
사용 예	rbs.cmd_pause()	

포인트!

cmd_pause() 의 사용 시점

일시 정지는 동작 명령어 내 혹은 user_hook() 메소드를 실행하는 시점에 cmd_pause() 가 실행되면 , 일시 정지할 수 있습니다 .
동작을 재개하려면 cmd_run() 로 실시합니다 .

메소드	cmd_reset()	rbsys
기능	동작 명령 : 에러 리셋	
인수	없음	
반환값	res0 [-] bool	True : 성공 False : 실패
사용 예	rbs.cmd_reset()	

포인트!

cmd_reset() 로 리셋이 가능한 에러

에러의 종류			cmd_reset() 통한 해제 가능 여부
E88	E**	시스템 정의 에러	○
c88	c**	시스템 정의 에러 치명적	×
u88	u**	유저 정의 에러	○
r88	r**	유저 정의 에러 치명적	×

메소드	cmd_run() rbsys
기능	동작 명령 : 로봇 프로그램 실행 (일시 정지 중이라면 , 프로그램 실행이 재개됩니다 .)
인수	fname [-] string : 프레임 이름 인수를 생략하면 , set_robtask() 으로 지정한 로봇 프로그램이 실행됩니다 .
반환값	없음

사용 예	# 예 1 : set_robtask() 으로 지정한 로봇 프로그램을 실행하는 경우 , 인수를 생략합니다 .
	rbs.set_robtask('sample.py')
	rbs.cmd_run()
	
	# 예 2 : 인수로 파일명을 지정하는 경우
	rbs.cmd_run('sample.py')

메소드	cmd_stop() rbsys
기능	동작 명령 : 감속 정지
인수	res0 [-] bool : True : 성공 False : 실패
반환값	없음
사용 예	rbs.cmd_stop()

포인트!

cmd_stop() 과 i611Robot 클래스의 enable_interrupt() 간의 관계

i611Robot 클래스의 enable_interrupt() (감속 정지 인터럽트 설정) 에 의해 , 감속 정지 후의 로봇 프로그램의 동작이 다릅니다 .

감속 정지 인터럽트 enable_interrupt(eid, enable)	감속 정지 후의 로봇 프로그램
유효 enable_interrupt(0, True)	예외 발생
무효 enable_interrupt(0, False)	정상 종료

감속 정지의 인터럽트가 유효한 상태로 , 로봇 프로그램에서 예외로 처리되지 않은 경우 , 로봇프로그램은 이상 종료하게 됩니다 .

메소드	get_robtask() rbsys	
기능	로봇 프로그램의 상태를 획득합니다 .	
인수	없음	
반환값	[res0, res1, res2] : List	
	res0 [-] bool	True : 로봇 프로그램 실행 중 False : 정지 중
	res1 [-] integer	실행 중이거나 마지막으로 실행된 로봇 프로그램의 프로세스 ID
	res2 [-] string	등록된 로봇 프로그램의 파일명
사용 예	# 로봇 프로그램의 실행 상태 획득 rb.get_robtask()[0] # 실행 중이거나 마지막으로 실행된 로봇 프로그램의 프로세스 ID 획득 rb.get_robtask()[1] # 등록된 로봇 프로그램의 파일명 획득 rb.get_robtask()[2]	

메소드	open() rbsys
기능	시스템 매니저와의 통신을 시작합니다 .
인수	없음
반환값	없음
사용 예	rbs.open()

메소드	req_mcmd() rbsys
기능	시스템 상태 및 명령 상태를 획득합니다 .
인수	없음
반환값	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error] : List

running [-] integer	로봇 프로그램 상태 1 : 실행 중 (컨트롤러 LED (STS) 에 표시) 0 : 정지 중
svon [-] integer	서보 상태 1 : 서보 ON 0 : 서보 OFF
emo [-] integer	비상 정지 상태 1 : 비상 정지 중 0 : 없음
hw_error [-] integer	시스템 정의 에러 (치명적) 상태 1 : 에러 발생 중 0 : 없음
sw_error [-] integer	시스템 정의 에러 상태 1 : 에러 발생 중 0 : 없음
abs_lost [-] integer	ABS 소실 상태 1 : ABS 소실 중 0 : 없음
in_pause [-] integer	일시 정지 상태 1 : 일시 정지 중 0 : 없음
error [-] integer	시스템 에러 상태 (*) 1 : 시스템 에러 발생 중 0 : 없음

사용 예

```
# 개별적으로 시스템의 상태를 확인
    running = rbs.req_mcmd()[0]    # 로봇 프로그램 상태
    svon = rbs.req_mcmd()[1]       # 서보 상태
    emo = rbs.req_mcmd()[2]        # 비상 정지 상태
    hw_error = rbs.req_mcmd()[3]   # 시스템 정의 에러 ( 치명적 ) 상태
    sw_error = rbs.req_mcmd()[4]   # 시스템 정의 에러 상태
    abs_lost = rbs.req_mcmd()[5]   # ABS 소실 상태
    in_pause = rbs.req_mcmd()[6]   # 일시 정지 상태
    error = rbs.req_mcmd()[7]      # 시스템 에러 상태
```

*) 시스템 정의 에러 (비치명적 또는 치명적) 의 발생 상태를 나타냅니다 .
2개의 에러 상태를 하나의 제어선에서 확인하는 경우에 사용합니다 .

메소드	set_robtask() rbsys	
기능	로봇 프로그램을 등록합니다 .	
인수	fname 필 수 [-] string	프로그램 파일명
반환값	res0 [-] bool	True : 성공 False : 실패 (지정된 파일이 존재하지 않음)
사용 예	rbs.set_robtask('sample01.py')	

포인트!

set_robtask() 메소드로는 로봇 프로그램을 등록만 가능합니다 . 로봇 프로그램 기동에는 사용할 수 없습니다 .

포인트!

I/O 의 할당과 Python 포트 간의 관계

입력

기능	관련 메소드	Python 포트 번호	
		물리적 I/O (*2)	시스템 I/O
로봇 프로그램 실행	cmd_run()	0	4288
감속 정지	cmd_stop()	1	4289
에러 리셋	cmd_reset()	2	4290
일시 정지	cmd_pause()	3	4291

출력

기능	관련 메소드	Python 포트 번호	
		물리적 I/O (*2)	시스템 I/O
로봇 프로그램 상태	req_mcmd()[0]	16	4160
서보 상태	req_mcmd()[1]	17	4096
비상 정지 상태	req_mcmd()[2]	18	4097
시스템 정의 에러 (치명적) 상태	req_mcmd()[3]	19	4098
시스템 정의 에러 상태	req_mcmd()[4]	20	4161
ABS 소실 상태	req_mcmd()[5]	21	4099
일시 정지 상태	req_mcmd()[6]	22	4162
시스템 에러 상태 (*1)	req_mcmd()[7]	23	4163

*1) 시스템 정의 에러 (비치명적 또는 치명적) 의 발생 상태를 나타냅니다 .
2 개의 에러 상태를 하나의 제어선에서 확인하는 경우에 사용합니다

*2) 이 I/O 의 할당을 권장합니다 . 메모리 I/O 에 대한 상세한 내용은「 3 메모리 맵 」을 참조해 주십시오 .

메소드	version() rbsys	
기능	시스템 매니저의 버전 정보를 취득합니다 .	
인수	없음	
반환값	[res0, major, minor, patch, build] : List	
	res0 [-] bool	True : 성공 False : 실패
	major [-] integer	메이저 버전
	minor [-] integer	마이너 버전
	patch [-] integer	패치 버전
	build [-] integer	빌드 버전
사용 예	version=rbs.version() print version	

5. 모듈 : i611_common

클래스

Exception

i611Robot 클래스에 속한 메소드의 예외 처리를 합니다 .

Exception 클래스를 계승한 클래스		
Robot_emo	비상 정지 시 발생하는 예외 (복귀 불가능)	P.109
Robot_error	에러 시 발생하는 예외	P.110
Robot_fatalerror	치명적 에러 시 발생하는 예외 (복귀 불가능)	P.110
Robot_poweroff	전원 차단 시 발생하는 예외 (복귀 불가능)	P.111
Robot_stop	감속 정지 시 발생하는 예외	P.112

Exception 클래스는 i611_MCS 모듈을 import 하는 것만으로도 사용할 수 있습니다 .
i611_MCS 모듈 내에서 from i611_common import 를 로드합니다 .

클래스	Robot_emo()	i611 COM. i611 MCS
기능	비상 정지 시 발생하는 예외 (복귀 불가능) 비상 정지 시의 이벤트 핸들러입니다 .	
인수	없음	
반환값	없음	

사용 예

```
# 준비 ----------------------------------------------------------------------------------------
## 1 . 초기 설정 ①  모듈 가져오기 ######################
    from i611_MCS import *
## 2 . 초기 설정 ② : 동작 조건 설정 ################################
    # i611 로봇 생성자
        rb = i611Robot()
    # 월드 좌표계 정의
        _BASE = Base()
    # 로봇과 접속 개시 , 초기화
        rb.open( True )
    # 동작 중 비상 정지 입력에 의한 예외 발생 활성화
        rb.enable_interrupt( 1, True )
...
# 예외 처리 ----------------------------------------------------------------------------------
# 예 1 : Exception 검출하여 정상 종료
        try:
        …
        except Robot_emo:   # 비상 정지 시 발생하는 예외
            # 필요한 에러 처리 ( 종료 처리 ) 기술
# 예 2 : Exception 검출하여 에러 발생 종료
        try:
        …
        except Robot_emo:   # 비상 정지 시 발생하는 예외
        rb.exit(1)   # 에러 발생 종료 ──→ E 14 가 발생합니다 ( P. 57 참조 )
```

포인트!

Robot_emo 클래스에 대한 보충 설명

비상 정지 스위치 눌렀을 때의 로봇 프로그램 작동

동작 프로그램에 try: ... except: 코드가 · · ·

- 포함되지 않은 경우 :

  프로그램은 에러 상태로 종료합니다 . 컨트롤러는 E 13 을 표시합니다 .

- 포함된 경우 :

  프로그램은 정상 종료됩니다 .(*)

*) 비상 정지 스위치를 누르면 5 초 이내에 로봇 프로그램이 종료하도록 프로그램을 작성해 주십시오 .
5 초가 넘으면 에러 상태로 종료합니다 . 7seg 표시 : E 17

클래스	Robot_error() i611 COM. i611 MCS
기능	에러 시 발생하는 예외 에러 발생 시의 이벤트 핸들러입니다 .
인수	없음
반환값	없음

사용 예	# 준비 ----------
	## 1 . 초기 설정 ① 모듈 가져오기 ##########
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##########
	# i611 로봇 생성자
	rb = i611Robot()
	# 월드 좌표계 정의
	_BASE = Base()
	# 로봇과 접속 개시 , 초기화
	rb.open(True)
	...
	
	# 예외 처리 ----------
	#Exception 검출하여 정상 종료
	try:
	…
	except Robot_error: # 에러 시 발생하는 예외
	# 필요한 에러 처리 (종료 처리) 기술

클래스	Robot_fatalerror() i611 COM. i611 MCS
기능	치명적 에러 시 발생하는 예외 (복귀 불가능) 치명적 에러 발생 시의 이벤트 핸들러입니다 .
인수	없음
반환값	없음

사용 예	# 준비 ----------
	## 1 . 초기 설정 ① 모듈 가져오기 ##########
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##########
	# i611 로봇 생성자
	rb = i611Robot()
	# 월드 좌표계 정의
	_BASE = Base()
	# 로봇과 접속 개시 , 초기화
	rb.open(True)
	…
	
	# 예외처리 ----------
	#Exception 검출하여 정상 종료
	try:
	…
	except Robot_fatalerror: # 치명적 에러 시 발생하는 예외
	# 필요한 에러 처리 (종료 처리) 기술

클래스	Robot_poweroff() i611 MCS
기능	전원 차단 시 발생하는 예외 (복귀 불가능) 전원 차단 시의 이벤트 핸들러입니다 .(*)
인수	없음
반환값	없음

사용 예	# 준비 ----------
	## 1 . 초기 설정 ① 모듈 가져오기 ##########
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##########
	# i611 로봇 생성자
	rb = i611Robot()
	# 월드 좌표계 정의
	_BASE = Base()
	# 로봇과 접속 개시 , 초기화
	rb.open(True)
	…
	
	# 예외 처리 ----------
	#Exception 검출하여 정상 종료
	try:
	…
	except Robot_poweroff: # 전원 차단 시에 발생하는 예외
	# 필요한 에러 처리 (종료 처리) 기술

*) 컨트롤러에 PoF 가 표시되는 시점에 발생합니다 .
전원을 차단하려면 , 그 때까지 프로세스를 완료해주시기 바랍니다 .

클래스	Robot_stop() i611 MCS
기능	감속 정지 시 발생하는 예외 감속 정지 시의 이벤트 핸들러입니다 .
인수	없음
반환값	없음

사용 예

```
# 준비 ----------------------------------------------------------------------------------------
## 1 . 초기 설정 ①  모듈 가져오기 ######################
    from i611_MCS import *

## 2 . 초기 설정② : 동작 조건 설정 ################################
    # i611 로봇 생성자
        rb = i611Robot()
    # 월드 좌표계 정의
        _BASE = Base()
    # 로봇과 접속 개시 , 초기화
        rb.open( True )
    # 동작 중의「감속 정지」입력 시의 예외 발생 활성화
        rb.enable_interrupt( 0, True )
...

# 예외 처리 ----------------------------------------------------------------------------------
# 예 1 : Exception 검출하여 정상 종료
        try:
        …
        except Robot_stop:   #「감속 정지」검사 이벤트 핸들러
        # 필요한 에러 처리 ( 종료 처리 ) 기술

# 예 2 : Exception 검출하여 에러 종료
        try:
        …
        except Robot_emo:   #「감속 정지」검사 이벤트 핸들러
        rb.exit(1)   # 에러 발생 종료
```

E 14 가 발생합니다 . (P. 57 참조)

포인트!

Robot_stop 클래스에 대한 보충 설명

감속 정지가 발생했을 때의 동작 프로그램의 행동

동작 프로그램에 try: ... except: 코드가・・・

・ 기술되어 있지 않은 경우 :

프로그램은 에러 종료합니다 . 컨트롤러는 E 16 를 표시합니다 .

・ 기술되어 있는 경우 :

프로그램은 정상 종료됩니다 .

6. 모듈 : i611_io

클래스	
(없음)	
I/O 제어	
멤버 변수	
—	—
함수	
din()	I/O 입력 P.114
dlyOut()	지정시간 경과 후 I/O 출력 P.115
dout()	I/O 출력 P.115
IOinit()	I/O 초기화 P.116
shotOut()	지정시간 동안 I/O 출력 P.117
wait()	지정한 I/O 입력 패턴이 될 때까지 대기 P.118

함수	din()	i611 IO
기능	I/O 입력	
인수	[*adr] : List	
	*adr 필 수 [-] string	입력 포트 · 1 개의 입력 포트를 지정하는 경우 adr : 입력 포트 번호 · 여러 개의 입력 포트 범위 지정하는 경우 adr[0] : 입력 포트 번호 (시작) adr[1] : 입력 포트 번호 (끝)
반환값	[*port] : List	
	*port [-] string	지정된 입력 포트의 실행 결과를 '0' 또는 '1' 로 돌려줍니다 입력 포트를 여러 개 지정하는 경우 , 리스트로 받습니다 .
사용 예	# 예 1 : 포트 15 번 지정 if din (15) == '1': # 예 2 : 포트 8 번부터 10 번까지 지정 if din (8, 10)[0] == '1': # 포트 10 번을 지정하는 경우 … elif din(8, 10)[1] == '1': # 포트 9 번을 지정하는 경우 … elif din(8, 10)[2] == '1': # 포트 8 번을 지정하는 경우	

주 의

	init.py 에 미리 설정되어 있는 포트는 사용하지 마십시오	(오동작)

함수	dlyOut()	i611 IO
기능	지정 시간 경과 후 I/O 출력	
인수	[num, dat, tim] : List Keyword	
	num 필 수 [-] integer	지연 출력 포트 번호
	dat 필 수 [-] string	I/O 에서 출력하는 데이터 문자열의 비트 필드로 설정합니다 (☞ 116 페이지의「비트 필드에서 포트 설정 」참고) '1' = ON '0' = OFF (초기값) '*' = 변화 없음
	tim 필 수 [s] integer	지연 시간
반환값	없음	

사용 예	# 예 1 : 리스트 (출력 포트 : 8, 데이터 출력 : ON, 지연 시간 : 10 초)
	dlyOut(8, '1', 10)
	
	# 예 2 : 키워드 (출력 포트 : 1, 데이터 출력 : OFF, 지연 시간 : 10 초)
	dlyOut(num=1 ,dat='0', tim=10)	

함수	dout()	i611 IO
기능	I/O 출력	
인수	[adr, data] : List Keyword	
	adr 필 수 [-] integer	출력 포트 시작 번호 설정 범위 : 16 ~ 31
	data 필 수 [-] string	I/O 에서 출력되는 데이터 문자열의 비트 필드로 설정합니다 (☞ 116 페이지의「비트 필드에서 포트 설정 」참고) '1' = ON '0' = OFF (초기값) '*' = 변화 없음
반환값	없음	

사용 예	# 시작 어드레스와 출력 데이터의 ON/OFF 지정 (20, 19, 18, 17, 16 번 포트 ON)
	dout(16, '11111')	

함수	IOinit()	i611 IO
기능	I/O 초기화 (*)	
인수	[IPaddress, port] : List	
	IPaddress [-] string	IP 주소 초기값 : '127.0.0.1'
	port [-] integer	포트 번호 초기값 : 12345
	인수를 생략하면 기본값으로 설정됩니다	
반환값	없음	

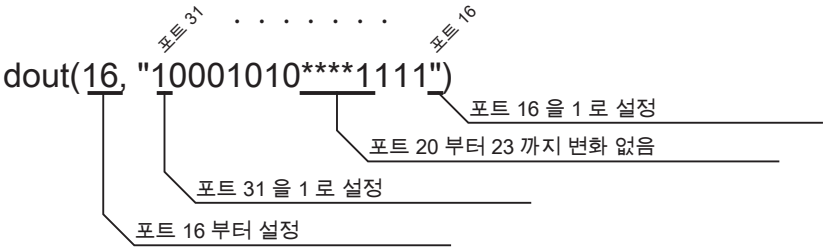
사용 예	# 예 1 : 인수 생략하는 경우 (초기값으로 설정됨)
	IOinit()
	
	# 예 2 : 리스트 (전체)
	IOinit('127.0.0.1', 12345)	

*) IOinit() 는 저장된 내용에 영향을 미치지 않습니다
메모리 I/O 에 액세스하기 위한 인터페이스를 초기화합니다 .

비트 필드에서의 포트 설정

dout(), dlyput(), shotOut(), wait() 메소드의 데이터 부분은 비트 필드 형식의 문자열입니다

예) 출력 포트 16 – 31 의 설정



함수	shotOut() i611 IO
기능	지정 시간 동안 I/O 출력 (tm 에서 설정한 시간이 경과하면 이전의 I/O 출력으로 복귀)
인수	[adr, data, tm] : List Keyword
	adr 필 수 [-] integer 출력 포트 어드레스 번호
	data 필 수 [-] string I/O 에서 출력되는 데이터 문자열의 비트 필드로 설정합니다 (☞ 116 페이지의「비트 필드에서 포트 설정 」참고) '1' = ON '0' = OFF (초기값) '*' = 변화 없음
	tm 필 수 [s] integer 출력 시간
반환값	없음

사용 예	# 예 1 : 리스트 (출력 포트 : 8, 데이터 출력 : ON, 출력 시간 : 10 초)
	shotOut(8, '1', 10)
	
	# 예 2 : 키워드 (출력 포트 : 1, 데이터 출력 : OFF, 출력 시간 : 10 초))
	shotOut(adr=1, data='0', tm=10)

함수	wait()	i611 IO
기능	지정한 I/O 입력 패턴이 될 때까지 대기	
인수	[adr, data, tm] : List Keyword	
	adr 필 수 [-] integer	입력 포트 어드레스 번호
	data 필 수 [-] string	입력 대기 데이터 지정 '1' = ON '0' = OFF (초기값 : 0)
	tm 필 수 [s] float integer	타임 아웃 시간
반환값	[res0, res1, res2] : List	
	res0 [-] integer	결과 1 : 입력 값과 일치 0 : 타임 아웃 –1 : 기타 오류
	res1 [-] string	입력 값
	res2 [s] float integer	조건 성립까지의 경과 시간

사용 예	# 예 1 : 리스트
	# 8 번 포트에 ON 값이 들어올 때까지 10 초간 대기하였을 때 , 조건을 만족하였을 경우
	if wait(8, '1', 10)[0] == 1:
	# 9 번 포트에 ON 값이 들어올 때 까지 10 초간 대기하였을 때 , 입력이 ON 인 경우
	if wait(9, '1', 10)[1] == '1':
	# 9 번 포트에 ON 값이 들어올 때 까지 10 초간 대기하였을 때 , 10 초 이상 경과하였을 경우
	if wait(9, '1', 10)[2] > 10:
	
	# 예 2 : 키워드
	if wait(adr=1, data='1', tm=10) == 1:	

7. 모듈 : i611shm

클래스		
(없음)		
공유 메모리에 액세스합니다 .		
멤버 변수		
-	-	
함수		
shm_read()	공유 메모리를 읽어옵니다 .	P.119
shm_write()	공유 메모리에 기록합니다 .	P.120

함수	shm_read()	i611 shm
기능	공유 메모리를 읽어옵니다 .	
인수	[index, num] : List	
	index 필 수 [-] integer	읽어 올 수 있는 공유 메모리 주소 설정 범위 : 0x0100 – 0x3800 「 3 메모리 맵」을 참조해 주십시오
	num [-] integer	연속으로 읽어 오는 변수의 개수 초기값 : 1
	인수 num 를 생략하면 기본값이 설정됩니다	
반환값	res [-] string	쉼표로 구분된 문자열 num 에 지정된 숫자의 값을 하나의 쉼표로 구분된 문자열로 반환합니다

사용 예	# 현재 J1 의 지령 값 (Joint 좌표)
	val_list = shm_read(0x3050, 6).split(',')
	joint0 = float(val_list[0])	

포인트!

쉼표 구분에 대하여

.split() 의 인수로 구분 기호인 쉼표 (,) 를 사용합니다 . 이를 통해 값을 구분할 수 있습니다

```
val_list = shm_read( 0x3050, 6 ).split( ',' )
```

함수	shm_write()	i611 shm
기능	공유 메모리에 기록하기	
인수	[index, num] : List	
	index [-] integer	기록이 가능한 공유 메모리 주소 설정 범위 : 0x1800 – 0x23F8 「 3 메모리 맵 」을 참조해 주십시오
	num [-] integer	연속하여 읽는 값의 리스트 또는 튜플
반환값	없음	
사용 예	shm_write(0x1800, 10) shm_write(0x1C00, (3.5, 4.3))	

기록 가능한 공유 메모리와 개수

메모리 주소	변수형	개수
0x1800 – 0x1BFC	integer (4byte)	256 개
0x1C00 – 0x23F8	float (8byte)	256 개

D 소프트웨어

3 메모리맵

1. 시작하기

공유 메모리와 메모리 I/O 는 RAM (휘발성 메모리) 입니다 .

전원 OFF 시 모든 메모리 내용은 삭제됩니다 .
시스템 시작 후 모든 값은 0 으로 초기화됩니다 .

시스템 시작시 공유 메모리의 사용자 블록을 제외한 모든 것이 현재의 새로운 정보로 업데이트됩니다 .

포인트 !

메모리 내용 저장

메모리 내용은 파일에 저장하는 것을 권장합니다 .
저장할 경우 , 작성한 후 flush () 또는 close () 를 호출합니다 .
(flush () : 파일 버퍼의 강제 플러시)

2. 공유 메모리

1. 공유 메모리의 구조

데이터 블록	오프셋	바이트	내용
헤더	+0x0000	256	공유 메모리의 헤더
메모리 I/O	+0x0100	1024	메모리 I/O 와 같은 내용
(예약)	+0x0500	768	-
시스템 관리자	+0x0800	4096	시스템 관리자 영역
유저	+0x1800	4096	사용자 영역 (4 바이트 정수 및 float 형)
제어 관리자	+0x2800	4096	기본 프로세스 영역

공유 메모리에 대한 액세스

공유 메모리에 대한 액세스는 로봇 라이브러리 shm_read (), shm_write () 를 사용합니다 .
shm_write () 는 사용자 블록만 사용할 수 있습니다 . 기타 영역은 읽기 전용 영역입니다 .

2. 메모리맵 (공유 메모리)

헤더블록

R : 읽기전용 | R/W : 읽기쓰기겸용

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x0000	(예약)	–	–	–	–	–	–
+0x0008	업데이트 카운터	4	unsigned short	update_counter	"1" : Native 업데이트 중	1ms	R
+0x000C	업데이트 중 플래그	2	unsigned short	now_updating	Native 업데이트 중에 1 이 된다 .	1ms	R
+0x000E	(예약)	–	–	–	–	–	–

메모리 I/O 블록

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x0100	Digital In/Out	4	unsigned int	dio_io	I/O 입력 x16、출력 x16	1ms	R
+0x0104	Hand Digital In/Out	4	unsigned int	dio_handio	암 (Arm) I/O 입력 x8、출력 x4	1ms	R
+0x0108	(예약)	–	–	–	–	–	–
+0x0300	System SI (입력) 0	4	unsigned int	mio_si0	서보 상태 , 비상 정지 등	1ms	R
+0x0304	(예약)	–	–	–	–	–	–
+0x0308	System SI (입력) 2	4	unsigned int	mio_si2	사용자 프로그램 가동 상황 등	1ms	R
+0x030C	(예약)	–	–	–	–	–	–
+0x0318	System SL (출력) 2	4	unsigned int	mio_sl2	프로그램 실행 입력 등	1ms	R
+0x031C	(예약)	–	–	–	–	–	–
+0x0320	User In/Out (입출력)	480	unsigned int	mio_pi0[120]	사용자 영역	수시	R

시스템 관리자 블록

R : 읽기전용 | R/W : 읽기쓰기겸용

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x0800	robtask name	32	char (string)	robtask_name[32]	"set_robtask 에 등록된 사용자 프로그램 이름 "	업데이트 시	R
+0x0820	running program name	32	char (string)	running_name[32]	실행중인 사용자 프로그램 이름	업데이트 시	R
+0x0840	running program pid	4	unsigned int	running_pid	" 실행중인 사용자 프로그램의 pid "	업데이트 시	R
+0x0844	assign_din(run)	2	short	assign_port[0]	입력 할당 포트 (run) 또는 -1	업데이트 시	R
+0x0846	assign_din(stop)	2	short	assign_port[1]	입력 할당 포트 (stop) 또는 -1	업데이트 시	R
+0x0848	assing_din(err_reset)	2	short	assign_port[2]	입력 할당 포트 (err_reset) 또는 -1	업데이트 시	R
+0x084A	assign_din(pause)	2	short	assign_port[3]	입력 할당 포트 (pause) 또는 -1	업데이트 시	R
+0x084C	assign_out(running)	2	short	assign_port[4]	출력 할당 포트 (running) 또는 -1	업데이트 시	R
+0x084E	assign_out(svon)	2	short	assign_port[5]	출력 할당 포트 (svon) 또는 -1	업데이트 시	R
+0x0850	assign_out(emo)	2	short	assign_port[6]	출력 할당 포트 (emo) 또는 -1	업데이트 시	R
+0x0852	assign_out(hw_error)	2	short	assign_port[7]	출력 할당 포트 (hw_error) 또는 -1	업데이트 시	R
+0x0854	assign_out(sw_error)	2	short	assign_port[8]	출력 할당 포트 (sw_error) 또는 -1	업데이트 시	R
+0x0856	assign_out(abs_lost)	2	short	assign_port[9]	출력 할당 포트 (abs_lost) 또는 -1	업데이트 시	R
+0x0858	assign_out(in_pause)	2	short	assign_port[10]	출력 할당 포트 (in_pause) 또는 -1	업데이트 시	R
+0x085A	assign_out(error)	2	short	assign_port[11]	출력 할당 포트 (running) 또는 -1	업데이트 시	R
+0x085C	(예약)	–	–	–	–	–	–

사용자 블록

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x1800	intval0	4	integer	intval0	사용자 변수 (정수)	비주기적	R/W
+0x1804	intval1	4	integer	intval1	사용자 변수 (정수)	비주기적	R/W
+0x1808	intval2 – intval255	1016	integer	intval(n)	사용자 변수 (정수)	비주기적	R/W
+0x1C00	floatval0	8	double	floatval0	사용자 변수 (부동 소숫점)	비주기적	R/W
+0x1C08	floatval1	8	double	floatval1	사용자 변수 (부동 소숫점)	비주기적	R/W
+0x1C10	floatval2 – floatval255	2032	double	floatval(n)	사용자 변수 (부동 소숫점)	비주기적	R/W
+0x2400	(예약)	–	–	–	–	–	–

사용자 블록에 쓰기 가능한 공유 메모리와 개수

메모리 번지	변수 유형	개수
0x1800 – 0x1BFC	integer(*)	256 개
0x1C00 – 0x23F8	float	256 개

*) 4 바이트입니다 .

제어 관리자 블록

컨트롤러 상태 (csts)

R : 읽기전용 | R/W : 읽기쓰기겸용

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x2800	errcode	2	unsigned short	errcode	크리티컬 에러 No.	발생시	R
+0x2802	bTeachMode	2	unsigned short	bTeachMode	교시 중 플래그	업데이트 시	R
+0x2804	bSPILargeFrame	2	unsigned short	bSPILargeFrame	대형 프레임용 SPI 통신 플래그	업데이트 시	R
+0x2806	(예약)	–	–	–	–	–	–

매니퓰레이터의 정보 (rbcfg)

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x2C00	manip_type	36	char (string)	manip_type	매니퓰레이터 정보	업데이트 없음	R
+0x2C24	manip_serial	36	char (string)	manip_serial	매니퓰레이터 시리얼	업데이트 없음	R
+0x2C48	format_version(major)	4	unsigned int	format_version[0]	데이터 구조의 버전	업데이트 없음	R
+0x2C4C	format_version(minor)	4	unsigned int	format_version[1]	데이터 구조의 버전	업데이트 없음	R
+0x2C50	format_version(patch)	4	unsigned int	format_version[2]	데이터 구조의 버전	업데이트 없음	R
+0x2C54	parameter_version(major)	4	unsigned int	parameter_version[0]	데이터 구조의 버전	업데이트 없음	R
+0x2C58	parameter_version(minor)	4	unsigned int	parameter_version[1]	데이터 구조의 버전	업데이트 없음	R
+0x2C5C	parameter_version(patch)	4	unsigned int	parameter_version[2]	데이터 구조의 버전	업데이트 없음	R
+0x2C60	(예약)	–	–	–	–	–	–

로봇의 상태 (rbsts)

R : 읽기전용 | R/W : 읽기쓰기겸용

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x3000	명령값 (직교 좌표) X[mm]	8	double	cmdx	현재 명령값	1ms	R
+0x3008	명령값 (직교 좌표) Y[mm]	8	double	cmdy	현재 명령값	1ms	R
+0x3010	명령값 (직교 좌표) Z[mm]	8	double	cmdz	현재 명령값	1ms	R
+0x3018	명령값 (직교 좌표) Rz[deg]	8	double	cmdrz	현재 명령값	1ms	R
+0x3020	명령값 (직교 좌표) Ry[deg]	8	double	cmdry	현재 명령값	1ms	R
+0x3028	명령값 (직교 좌표) Rx[deg]	8	double	cmdrx	현재 명령값	1ms	R
+0x3030	(예약)	–	–	–	–	–	–
+0x3040	암 (Arm) 자세 (구조 플래그)	4	unsigned int	posture	자세 (0 ~ 7)	1ms	R
+0x3044	(예약)	–	–	–	–	–	–
+0x3048	특정 포인트 정보	4	unsigned int	singular	현재 위치의 특이점 근방 여부	1ms	R
+0x304C	다 회전 정보	4	unsigned int	multiturn	각 축의 다 회전 정보	1ms	R
+0x3050	명령값 (Joint) J1[deg]	8	double	joint[0]	현재 명령값	1ms	R
+0x3058	명령값 (Joint) J2[deg]	8	double	joint[1]	현재 명령값	1ms	R
+0x3060	명령값 (Joint) J3[deg]	8	double	joint[2]	현재 명령값	1ms	R
+0x3068	명령값 (Joint) J4[deg]	8	double	joint[3]	현재 명령값	1ms	R
+0x3070	명령값 (Joint) J5[deg]	8	double	joint[4]	현재 명령값	1ms	R
+0x3078	명령값 (Joint) J6[deg]	8	double	joint[5]	현재 명령값	1ms	R
+0x3080	(예약)	–	–	–	–	–	–
+0x3090	속도	8	double	velocity	현재 명령값	1ms	R
+0x3098	비정상 속도	4	unsigned int	vel_error_axes	속도 오류 발생 축	발생시	R
+0x309C	소프트 리미트	4	unsigned int	softlimit	각 축의 제한 근방 여부	1ms	R
+0x30A0	서보 OFF 직전 위치 J1[rad]	8	double	joint_svon_to_svoff[0]	서보 OFF 직전 위치	발생시	R
+0x30A8	서보 OFF 직전 위치 J2[rad]	8	double	joint_svon_to_svoff[1]	서보 OFF 직전 위치	발생시	R
+0x30B0	서보 OFF 직전 위치 J3[rad]	8	double	joint_svon_to_svoff[2]	서보 OFF 직전 위치	발생시	R
+0x30B8	서보 OFF 직전 위치 J4[rad]	8	double	joint_svon_to_svoff[3]	서보 OFF 직전 위치	발생시	R
+0x30C0	서보 OFF 직전 위치 J5[rad]	8	double	joint_svon_to_svoff[4]	서보 OFF 직전 위치	발생시	R
+0x30C8	서보 OFF 직전 위치 J6[rad]	8	double	joint_svon_to_svoff[5]	서보 OFF 직전 위치	발생시	R
+0x30D0	(예약)	–	–	–	–	–	–
+0x30E0	서보 OFF 직전 위치 저장 플래그	4	unsigned int	b_saved	서보 OFF 직전 위치가 유효한지	발생시	R
+0x30E4	(예약)	–	–	–	–	–	–
+0x37E8	(예약)	–	–	–	–	–	–
+0x37F0	(예약)	–	–	–	–	–	–
+0x37F8	(예약)	–	–	–	–	–	–

3. 메모리 I/O

1. 메모리맵 (물리적 I/O)

R : 읽기전용 | R/W : 읽기쓰기겸용

종별	주소	내용	커넥터 단자 (신호 이름)	Python 포트 번호		사용자 엑세스
컨트롤러의 물리적 DIDO 4 bytes (I/O 커넥터)	0L	디지털 입력 CN3 : I/O 커넥터 1 (입력)	2A (IN1)	0	0x0000	R
			2B (IN2)	1	0x0001	R
			3A (IN3)	2	0x0002	R
			3B (IN4)	3	0x0003	R
			4A (IN5)	4	0x0004	R
			4B (IN6)	5	0x0005	R
			5A (IN7)	6	0x0006	R
			5B (IN8)	7	0x0007	R
			6A (IN9)	8	0x0008	R
			6B (IN10)	9	0x0009	R
			7A (IN11)	10	0x000A	R
			7B (IN12)	11	0x000B	R
			8A (IN13)	12	0x000C	R
			8B (IN14)	13	0x000D	R
			9A (IN15)	14	0x000E	R
			9B (IN16)	15	0x000F	R
	0H	디지털 출력 CN4 : I/O 커넥터 2 (출력)	2A (O1P), 2B (O1N)	16	0x0010	R/W
			3A (O2P), 3B (O2N)	17	0x0011	R/W
			4A (O3P), 4B (O3N)	18	0x0012	R/W
			5A (O4P), 5B (O4N)	19	0x0013	R/W
			6A (O5P), 6B (O5N)	20	0x0014	R/W
			7A (O6P), 7B (O6N)	21	0x0015	R/W
			8A (O7P), 8B (O7N)	22	0x0016	R/W
			9A (O8P), 9A (O8N)	23	0x0017	R/W
		디지털 출력 CN5 : I/O 커넥터 3 (출력)	2A (O9P), 2B (O9N)	24	0x0018	R/W
			3A (O10P), 3B (O10N)	25	0x0019	R/W
			4A (O11P), 4B (O11N)	26	0x001A	R/W
			5A (O12P), 5B (O12N)	27	0x001B	R/W
			6A (O13P), 6B (O13N)	28	0x001C	R/W
			7A (O14P), 7B (O14N)	29	0x001D	R/W
			8A (O15P), 8B (O15N)	30	0x001E	R/W
			9A (O16P), 9A (O16N)	31	0x001F	R/W
암 (Arm)DIDO 8 bytes (암 I/O 커넥터)	1L	디지털 입력	6A (I1)	32	0x0020	R
			6B (I2)	33	0x0021	R
			5A (I3)	34	0x0022	R
			5B (I4)	35	0x0023	R
		(예약)	–	36 – 47	0x0024 – 0x002F	–
	1H	디지털 출력	3A (O1)	48	0x0030	R/W
			3B (O2)	49	0x0031	R/W
		(예약)	–	50 – 63	0x0032 – 0x003F	–
	2	(예약)	–	64 – 95	0x0040 – 0x005F	–
(예약)	3 – 127	–	–	96 – 4095	0x0060 – 0x0FFF	–

2. 메모리맵 (시스템 I/O)

R : 읽기전용 | R/W : 읽기쓰기겸용

종별	주소	내용	Python 포트 번호		사용자 엑세스
System SI 16 bytes	128	서보 상태	4096	0x1000	R
		비상 정지 상태	4097	0x1001	R
		시스템 정의 오류 (치명적) 상태	4098	0x1002	R
		ABS 소실 상태	4099	0x1003	R
		(예약)	4100 – 4103	0x1004 – 0x1007	–
		(예약)	4104 – 4111	0x1008 – 0x100F	–
		(예약)	4112 – 4127	0x1010 – 0x101F	–
	129	(예약)	4128 – 4159	0x1020 – 0x103F	–
	130	로봇 프로그램 상태	4160	0x1040	R/W
		시스템 정의 오류 상태	4161	0x1041	R/W
		일시 정지 상태	4162	0x1042	R/W
		시스템 오류 상태 (*)	4163	0x1043	R/W
		시스템 상태	4164 – 4167	0x1044 – 0x1047	R/W
		오류 코드	4168 – 4175	0x1048 – 0x104F	R/W
		(예약)	4176– 4183	0x1050 – 0x1057	–
		(예약)	4184– 4187	0x1058 – 0x105B	–
		(예약)	4188	0x105C	–
		(예약)	4189 – 4191	0x105D – 0x105F	–
	131	(예약)	4192 – 4223	0x1060 – 0x107F	–
System SL 16 bytes	132	(예약)	4224 – 4255	0x1080 – 0x109F	–
	133	(예약)	4256 – 4287	0x10A0 – 0x10BF	–
	134	로봇 프로그램 실행	4288	0x10C0	R/W
		감속 정지	4289	0x10C1	R/W
		오류 재설정	4290	0x10C2	R/W
		일시 정지	4291	0x10C3	R/W
		(예약)	4292 – 4319	0x10C4 – 0x10DF	–
	135	(예약)	4320 – 4351	0x10E0 – 0x10FF	–
User Input/Output 480 bytes	136–255	사용자용 (*)	4352 – 8191	0x1100 – 0x1FFF	R/W

*) 시스템 정의 오류 (비치명적 또는 치명적) 의 발생 상태입니다 .

메모리 I/O 에 대해

물리적 I/O 및 시스템 I/O 를 묶어서 메모리 I/O 라고 칭합니다 .
IOinit () 함수는 메모리 I/O 에 액세스하기 위한 인터페이스를 초기화할 뿐입니다 .
메모리 내용에 영향을 미치지 않습니다 .

I/O 할당과 Python 포트의 관계

입력

기능	관련 메소드	Python 포트 번호	
		물리적 I/O(*2)	디지털 I/O
로봇 프로그램 실행	cmd_run()	0	4288
감속 정지	cmd_stop()	1	4289
오류 재설정	cmd_reset()	2	4290
일시 정지	cmd_pause()	3	4291

출력

기능	관련 메소드	Python 포트 번호	
		물리적 I/O(*2)	디지털 I/O
로봇 프로그램 상태	req_mcmd()[0]	16	4160
서보 상태	req_mcmd()[1]	17	4096
비상 정지 상태	req_mcmd()[2]	18	4097
시스템 정의 오류 (치명적) 상태	req_mcmd()[3]	19	4098
시스템 정의 오류 상태	req_mcmd()[4]	20	4161
ABS 소실 상태	req_mcmd()[5]	21	4099
일시 정지 상태	req_mcmd()[6]	22	4162
시스템 오류 상태 (*1)	req_mcmd()[7]	23	4163

*1) 시스템 정의 오류 (비치명적 또는 치명적) 의 발생 상태입니다 .
2 개의 오류 상태를 1 개의 제어선으로 확인하는 경우에 사용합니다 .

*2) 권장 설정입니다 .

D 소프트웨어

4 프로그램 실행 단계

1. 전체 흐름

1. 로봇 프로그램의 시작 방법

단계 1 교시 포인트를 준비합니다 .

교시 포인트를 미리 준비해 둡니다 .

(취급설명서「C 교시」도 참조하십시오 .)

단계 2 로봇 프로그램을 작성합니다 .

PC 의 텍스트 편집기에서 로봇 프로그램을 만듭니다 .

단계 1 에서 준비한 티칭 포인트를 로봇 프로그램에 설정합니다 .

(「1 프로그래밍 가이드」,「2 로봇 라이브러리」을 참조하십시오 .)

단계 3 로봇 프로그램을 컨트롤러로 전송합니다 .

FTP 클라이언트 소프트웨어로 PC 와 컨트롤러 간 파일을 전송합니다 .

(연결 방법은 사용 설명서「C 교시」을 참조하십시오 .)

단계 4 로봇 프로그램을 실행합니다 .

시작하는 방법은 두 가지입니다 .

방법 1 「I/O 입력」에서 시작하는 경우

I/O 입력 (하드웨어 신호) 에서 로봇 프로그램을 실행합니다 .
init.py I/O 포트를 설정합니다 .
주로 생산 현장에서 자동 운전 시에 사용합니다 .

(「1 프로그래밍 가이드」,「2 로봇 라이브러리」을 참조하십시오 .)

또는

(init.py 는 컨트롤러에 미리 준비되어 있습니다 .)

방법 2 「터미널 소프트웨어」에서 시작하는 경우

터미널 소프트웨어를 사용하여 컨트롤러에 연결하고 전송한 로봇 프로그램을 실행합니다 .
주로 테스트 및 디버깅에 사용합니다 .

(연결 방법은 취급설명서「C 교시」을 참조하십시오)

1 전체 흐름

⚠ 주 의

⛔	init.py 에는 전원 투입과 동시에 자동 운전을 시작하는 로봇 프로그램을 작성하지 마십시오 .	
❗	로봇을 손상시킬 수 있는 기계와 로봇을 조합하거나 공유할 경우 , 다른 기계의 작업 공간 밖에서 모든 기능과 동작 프로그램을 개별적으로 시험할 것을 권장합니다 .	

로봇 프로그램의 시작 방법과 로그 출력의 관계

로봇 프로그램의 실행 방법의 차이에 따라 로그 데이터의 출력 방법이 다릅니다 .

「I/O 입력」에서 시작하는 경우

표준 출력 오류 출력을 파일에 저장됩니다 .

・ 로봇 프로그램

출력 정보	저장 위치	파일 이름
표준 출력	/opt/i611/log	userprog_out.log
오류 출력		userprog_err.log

・ init.py

출력 정보	저장 위치	파일 이름
표준 출력	/opt/i611/log	sys_out.log
오류 출력		sys_err.log

「터미널 소프트웨어」에서 시작하는 경우

로그 파일에는 저장되지 않습니다 .

오류 정보 등의 모든 출력 데이터는 터미널 소프트웨어의 화면에 표시됩니다 .

・ 로봇 프로그램 (xxx.py)

사용자가 로봇 라이브러리를 이용하여 로봇 동작을 자유롭게 만들 수 있습니다 .

파일 이름은 사용자가 자유롭게 설정할 수 있습니다 . (xxx.py)

・ 시스템 프로그램 (init.py)

로봇 라이브러리 RobSys 클래스를 사용하여 I/O 입력에서 시작 제어를 위한 설정을 설명합니다 .

파일 이름은 변경하지 마십시오 .

2. 실행 방법

「I/O 입력」에서 시작하는 경우

「1 프로그래밍 가이드」,「2 로봇 라이브러리」를 참조하십시오 .

「터미널 소프트웨어」에서 시작하는 경우

컨트롤러와 통신을 시작합니다 .

(화면 예제는 Windows 8.1 입니다)

192.168.0.23 - Tera Term VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
SABRE_SDB login: i611usr
Password:

BusyBox v1.23.2 (2016-09-01  10:04:36 JST) built-in shell
Enter 'help' for a list of built-in commands.

$ pwd
/home/i611usr
$
```

login : i611usr

Password : i611

pwd : 현재 작업중인 디렉토리를 확인합니다 .

프로그램의 실행 및 제어합니다 .

192.168.0.23 - Tera Term VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
$ run xxx.py
$ pause
$ stop
$ reset
$
```

run : Python 프로그램 xxx.py 를 실행합니다 .(동작중 다른 명령 입력 가능)

pause : 실행중인 프로그램을 일시 중지합니다 .(재개 시 run)

stop : 실행중인 프로그램을 중지합니다 .

reset : 초기화합니다 .

192.168.0.23 - Tera Term VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
$ python xxx.py
```

python : Python 프로그램 xxx.py 를실행합니다 .(디버깅 명령어 사용 가능)

Z

자료

1. 블록다이어그램
2. 유지보수
3. 용어집
4. 문제해결

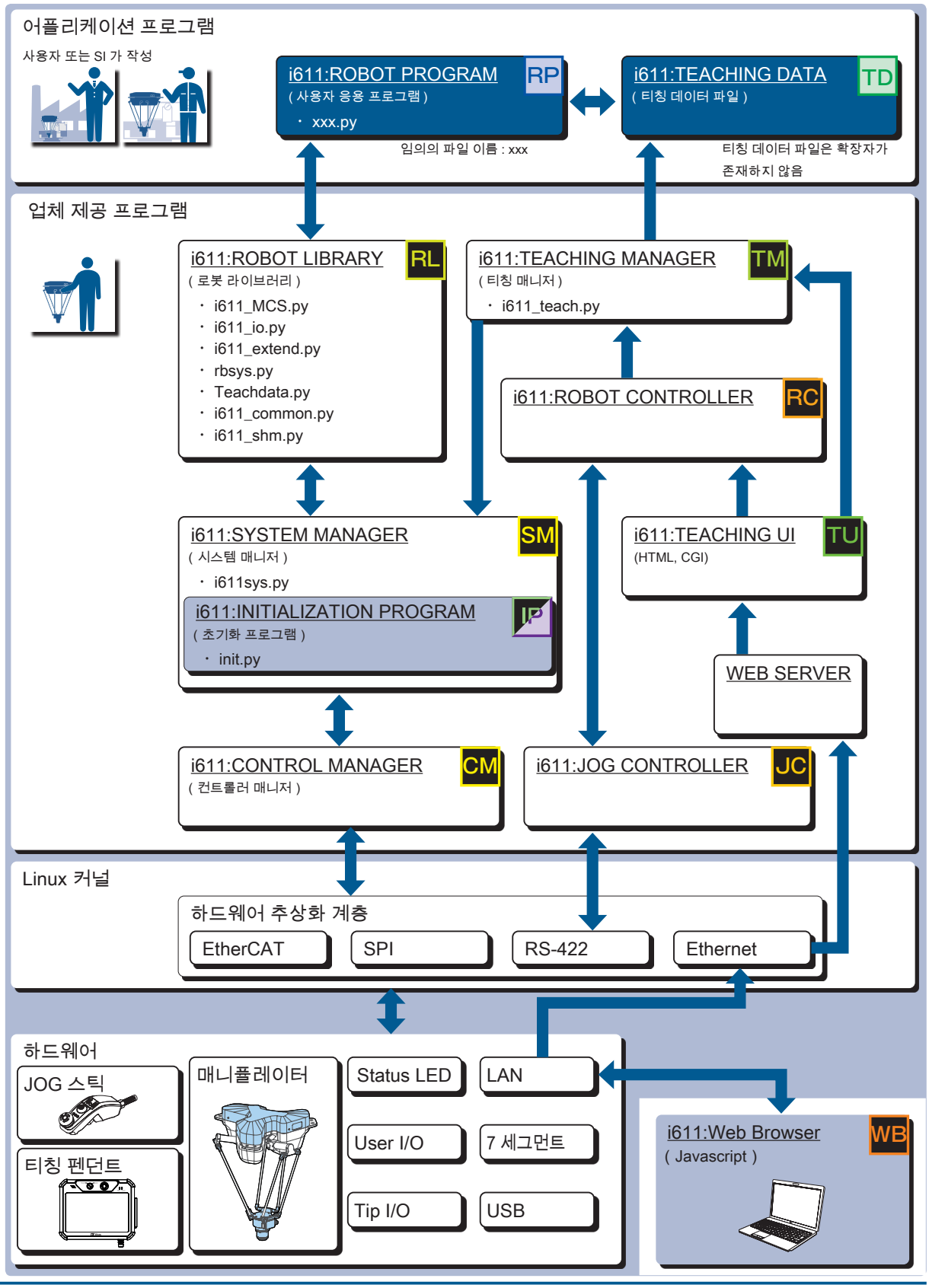
MEMO

Z 자료편

1 블록다이어그램

1. 시스템 블록도

1. 시스템 블록도

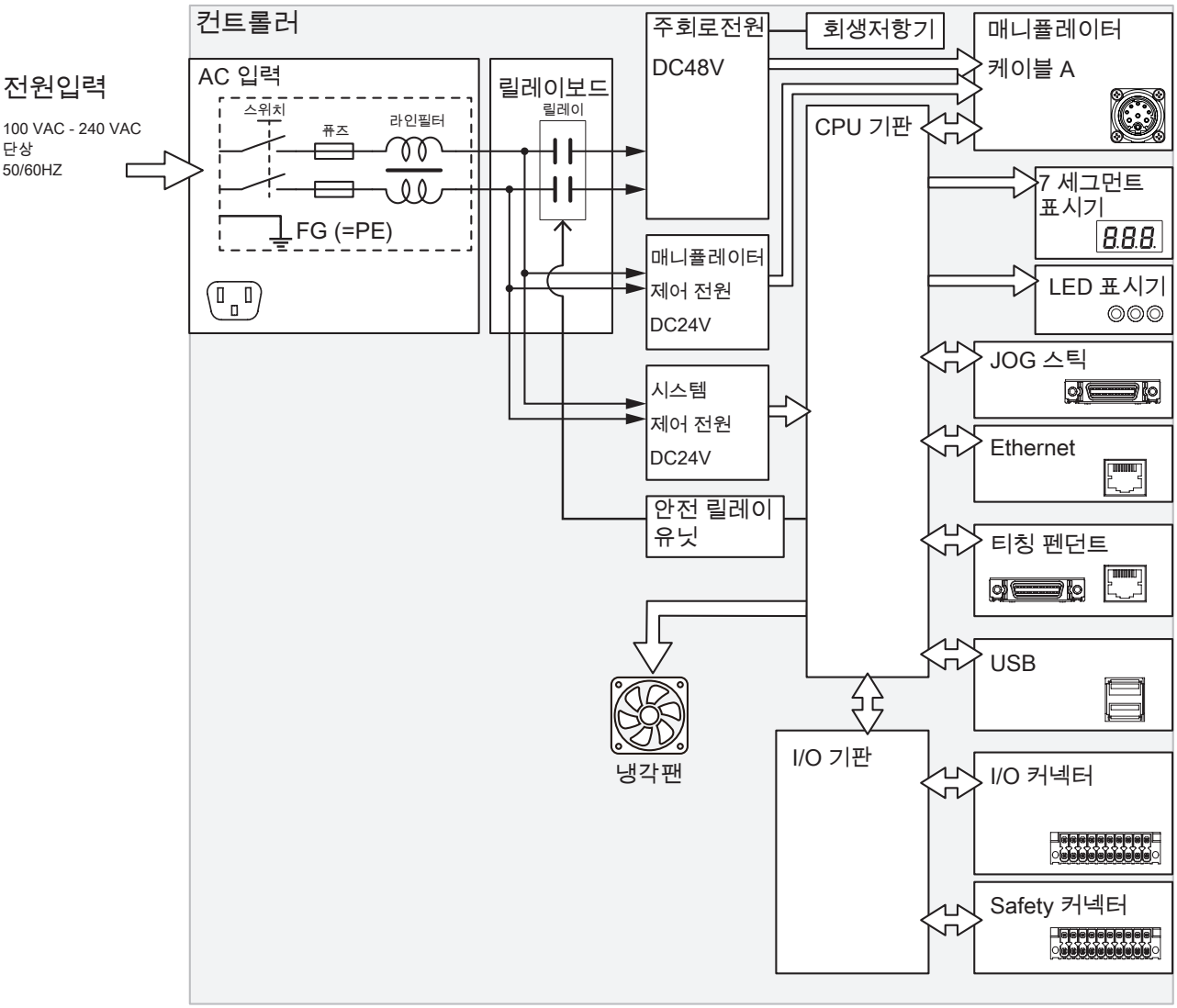


2. 프로그램 목록

프로그램 이름	요약
i611:ROBOT PROGRAM RP	사용자 또는 SI 가 만듭니다 . 매니퓰레이터를 제어하는 로봇 프로그램입니다 .
i611:TEACHING DATA TD	사용자 또는 SI 가 만듭니다 . 티칭으로 설정한 좌표 정보를 저장한 파일입니다 .
i611:INITIALIZATION PROGRAM IP	제조업체가 제공합니다 . 사용자 또는 SI 가 수정가능합니다 . I/O 설정 및 로봇 프로그램의 실행 방법을 설명하는 스크립트 파일입니다 .
i611:SYSTEM MANAGER SM	제조업체가 제공합니다 . 로봇 프로그램의 상태 관리 , 티칭 상태 제어 , 오류 처리를 진행합니다
i611:ROBOT LIBRARY RL	제조업체가 제공합니다 . 로봇 동작의 프로그래밍에 필요한 다양한 모듈을 포함하는 파일입니다 .
i611:CONTROL MANAGER CM	제조업체가 제공합니다 . 시스템 시작 , 중지를 포함한 상태 관리 및 오류 처리를 합니다 . 실시간으로 매니퓰레이터 각 관절의 각도와 자세 (WORLD 좌표계) 의 관계를 구하는 계산 및 가감속 제어를 합니다 . 움직임을 만들어내는 핵심부분입니다 .
i611:TEACHING MANAGER TM	제조업체가 제공합니다 . 매니퓰레이터의 위치 결정을 하기 위한 기능이며 , 파일에 저장된 좌표 그룹을 출력하고 , 출력된 좌표를 사용자 프로그램에서 사용할 수 있도록 합니다
i611:ROBOT CONTROLLER RC	제조업체가 제공합니다 . 사용자 인터페이스 및 JOG 동작을 제어합니다 .
i611:JOG CONTROLLER JC	제조업체가 제공합니다 . JOG 스틱을 제어합니다 . ROBOT CONTROLLER 에서 진동 , LED, 부저를 신호를 받아 처리합니다 .
i611:TEACHING UI TU	제조업체가 제공합니다 . 티칭을 위한 설정이나 로봇을 동작할 브라우저 UI 입니다 .
i611:Web Browser WB	Google 이 제공하는 Chrome Browser 를 사용합니다 . 티칭을 위한 Javascript 를 동작하게 합니다 .

2. 하드웨어 블록 다이어그램

1. 컨트롤러 블록 다이어그램



Z 자료편

2 유지보수

1. 점검

1. 점검 시 유의할 점

⚠ 주 의		
	본 제품을 안전하고 오래 사용하기 위해 점검을 반드시 실시하여 고장을 미연에 방지하고 안전 확보를 하십시오 .	
	플라스틱 부분을 청소할 때에는 석유계 제품을 사용하지 마십시오 .	 (변형· 변색)

다음의 준비를 한 후 검사를 시작하십시오 .

1. 다른 작업자가 움직이는 동시에 안에서 작업을 하지 않도록 컨트롤러와 안전구역 입구에 ' 점검 중 ' 등의 표시를 한다 .
2. 작업자는 작업 전에 컨트롤러를 잠그고 열쇠를 휴대하는 등의 제어의 우선권을 확보한다 .
3. 작업에 필요한 충분한 공간과 조명을 확보한다 .
4. 산업용 로봇 특별 교육을 수료한 사람이 검사하도록 한다 .
5. 당직자를 배치해 전체를 바라볼 수있는 위치에 배치하고 , 즉시 비상 정지할 수있는 준비를 한다 .
6. 서로의 신호 방법을 확인한다 .
7. 점검 기록을 3 년 이상 저장한다 .

점검 기간 및 운전 시간 기준

15h/ 일× 20 일 / 월× 3 개월 = 약 1,000h

일일 점검이나 정기 점검을 반드시 실시하여 이상이 없음을 확인하여 주십시오 .

이상이 있으면 즉시 보수나 필요한 조치를 취하여 , 고장을 미연에 방지하고 안전을 확보해 주십시오 .

가능한 한 가동 범위 밖에서 실시하도록 하고 , 부득이하게 가동 범위 안에서 실시하는 경우에는 사전에 안전 대책을 조치하고 나서 실행해 주십시오 .

시스템 전체의 보수 점검은 시스템을 통합하는 보수 계획에 따라서 실시해 주십시오 .

메가 테스트 (절연저항 측정) 는 실시하지 말아 주십시오 .

2. 일일 점검과 정기 점검

일일 점검 ; 운전 전에 실시할 것

전원 켜기 전 (전원을 투입하기 전에 아래의 점검 항목을 확인하십시오 .)

점검항목 (내용)	이상시의 조치
1. 전원 케이블이 단단하게 연결되어 있습니까 ?	확실하게 연결하십시오 .
2. 매니퓰레이터 케이블은 확실히 안쪽까지 끼워져 잠겨 있습니까 ?	확실하게 연결하십시오 .
3. I/O 커넥터 , Safety 커넥터는 확실하게 연결되어 있습니까 ?	확실하게 연결하십시오 .
4. 매니퓰레이터의 연결 부위는 느슨하지 않습니까 ?	볼트를 확실하게 조여주십시오 .
5. 탑 플랜지 부착 볼트는 느슨하지 않습니까 ?	볼트를 확실하게 조여주십시오 .
6. 매니퓰레이터의 수지부분에 금이 있거나 균열은 없습니까 ?	사용을 중지하고 서비스 센터에 문의하십시오 .
7. 가루나 기름 등의 이물질이 묻어 있지 않습니까 ?	이상이 없는지 확인하고 , 청소해 제거해 주십시오 .
8. 동작 영역 내에 물건은 없습니까 ?	간섭이 없도록 물체를 치워주십시오 .
9. 컨트롤러의 흡기구와 배기구가 먼지로 막혀 있지 않습니까 ?	청소해 제거해 주십시오 .
10. JOG 스틱 (옵션품) 에 금이 있거나 갈라진 곳은 없습니까 ?	금이나 균열이 없는 제품을 사용해 주십시오 .
11. 티칭 펜던트 (옵션품) 에 금이 있거나 갈라진 곳은 없습니까 ?	금이나 균열이 없는 제품을 사용해 주십시오 .
12. 티칭 펜던트의 메인 케이블과 통신 케이블은 확실하게 연결되어 있습니까 ?	확실하게 연결하십시오 .
13 케이블이 파괴되거나 손상되지 않습니까 ?	찢어지거나 흠집이 없는 케이블을 사용해 주십시오 .
14. 케이블은 기름이나 물에 잠기지 않습니까 ?	기름이나 물을 깨끗하게 제어해 주십시오 .
15. 전원 , 전압은 정상입니까 ?	이상이 없는지 확인해 주십시오 .
16. 이상한 냄새는 나지 않습니까 ?	사용을 중지하고 서비스 센터에 문의하십시오 .
17. 컨트롤러 전면의 커넥터에 유 . 수분과 먼지 , 이물질 등이 묻어있습니까 ?	기름이나 물을 깨끗이 제거하십시오 .
18. 사용온도 , 습도는 사용조건의 범위 내에서 동작합니까 ?	사용 환경의 범위 내에서 사용하십시오 .
19. 장비 , 설비의 연결 부분의 이완 , 위치 엇갈림은 없습니까 ?	이상이 없는지 확인하십시오 .
20. 관절부와 말단 장치 등의 가동부에 이물질이 있지 않습니까 ?	이상이 없는지 확인하십시오 .

전원이 켜진 후 (로봇을 주시하면서 전원을 켜주세요 .)

점검 항목 (내용)	이상 시의 조치
21. 전원 투입에 의해 비정상적인 움직임이나 이상한 소리나 이상한 냄새가 나지 않습니까 ?	확실하게 연결하십시오 .

운전 중 (프로그램을 구동하면서)

점검 항목 (내용)	이상 시의 조치
22. 매니퓰레이터의 위치 오차가 발생합니까 ?	베이스 또는 말단 장치의 볼트가 느슨해지지 않았습니까 ? 치구류의 위치가 달라지지 않았습니까 ?
23. 프로그램 동작에 의해 매니퓰레이터에서 비정상적인 동작이나 이상진동 , 이상한 소리나 냄새가 나지 않습니까 ?	서비스 센터에 문의하십시오 .

정기 점검 ; 1 개월에 1 회 일일 점검보다 상세한 점검을 실시

매니퓰레이터

점검 항목 (내용)	이상 시의 조치
1. 매니퓰레이터 각 부분의 볼트가 느슨하지 않습니까 ?	볼트를 확실하게 조여주십시오 .
2. 커넥터의 고정 볼트 또는 연결 단자대의 볼트가 느슨하지 않습니까 ?	볼트를 조여 주십시오 .
3. 관절부 유닛 (감속기) 에서 이상한 소리가 나지 않습니까 ?	서비스 센터에 문의하십시오 .

컨트롤러

점검 항목 (내용)	이상 시의 조치
1. 컨트롤러의 흡배기 필터가 더럽습니까 ?	청소 또는 새로운 부품으로 교체하십시오 .

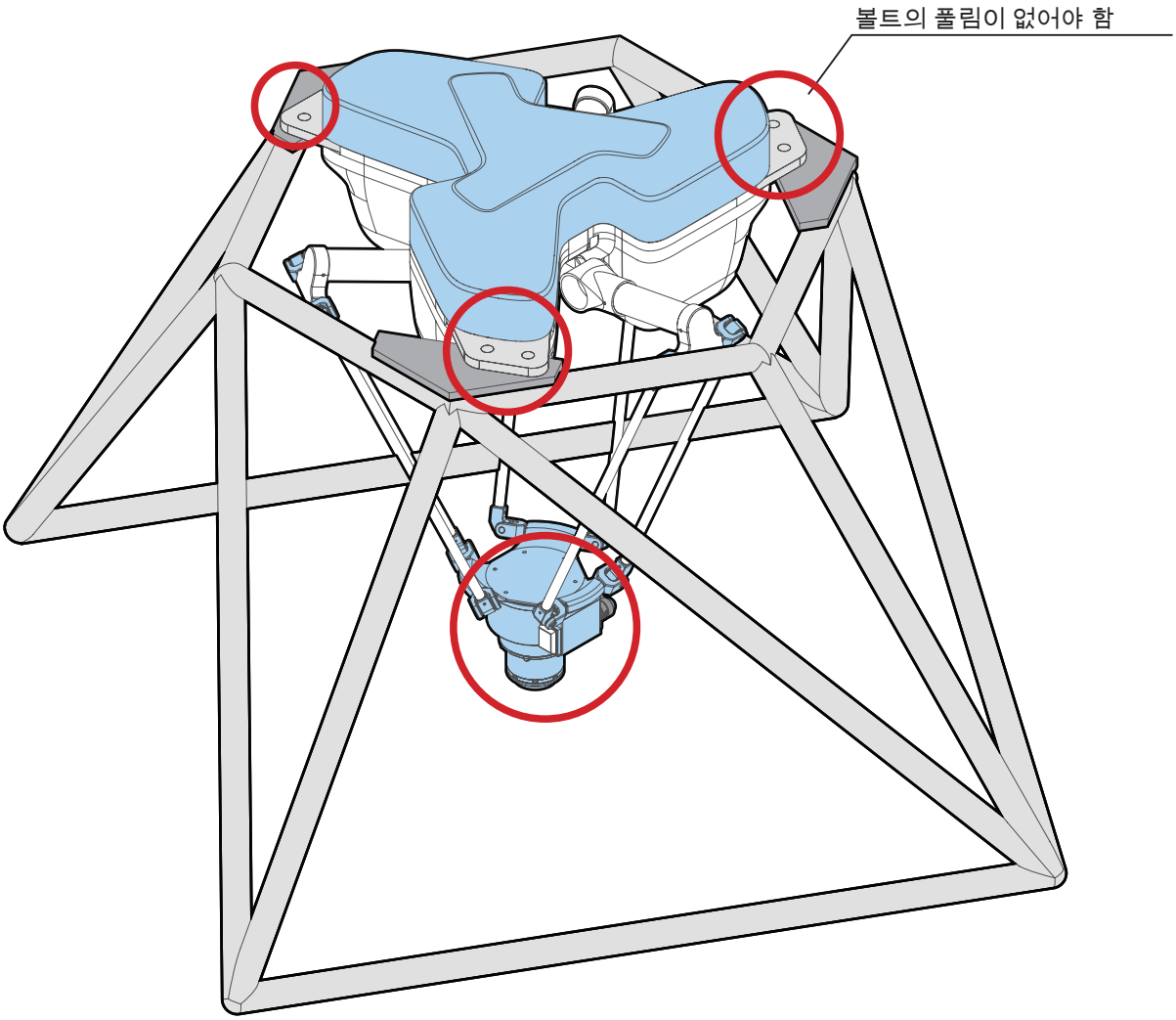
티칭 펜던트 (옵션품)

점검 항목 (내용)	이상 시의 조치
1. 티칭 펜던트의 스피커에서 이상한 소리가 나지 않습니까 ?	서비스 센터에 문의하십시오 .
2. 티칭 펜던트의 필터가 더럽습니까 ?	서비스 센터에 문의하십시오 .

2. 유지보수

1. 매니퓰레이터

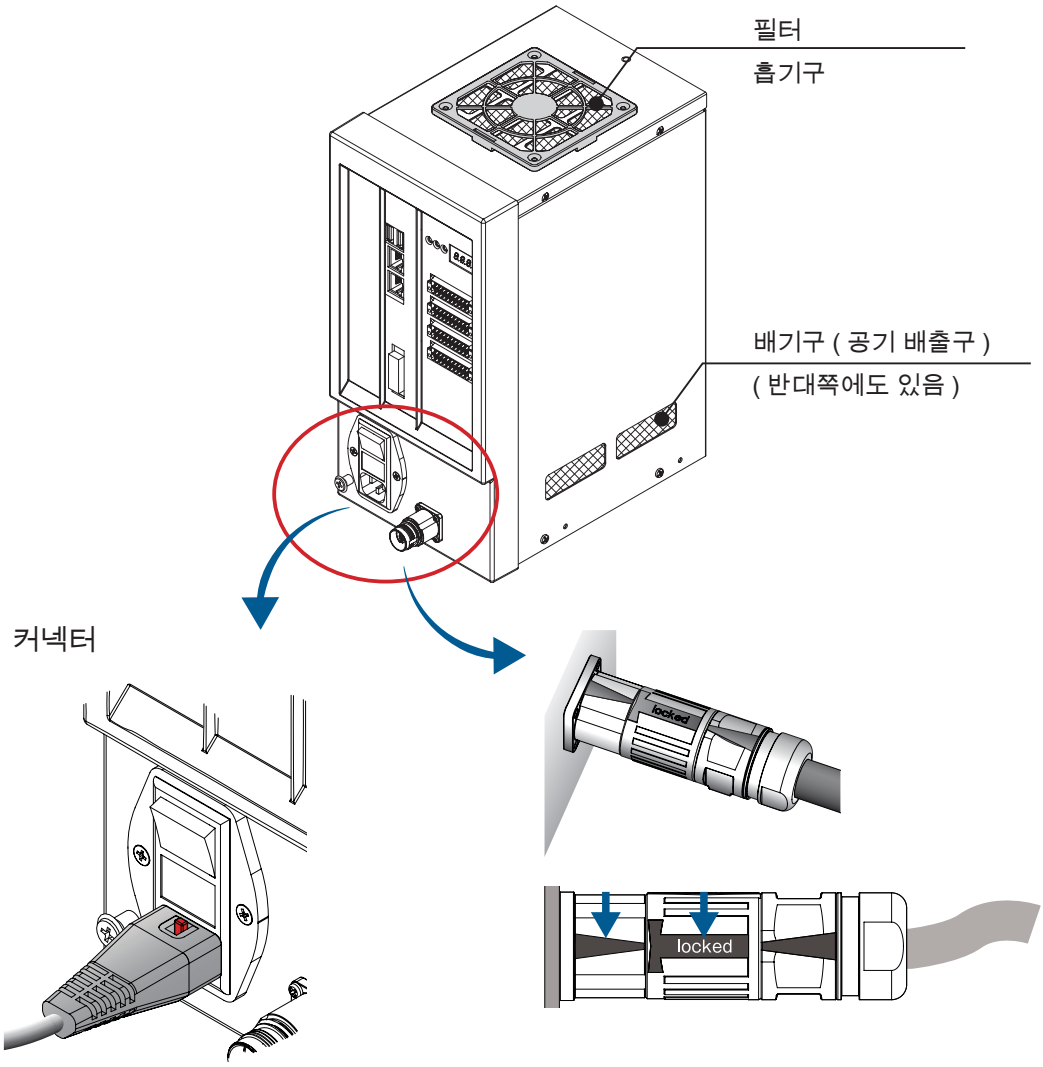
점검항목	내용
【운전 이전】 외관	암이나 관절부 등에 가루나 기름 등의 이물질이 스며있지 않은지 확인하십시오 . 볼트의 풀림이 없는지 확인하심시오 . 인코더 커버가 손상되지 않았는지 확인하십시오 .
【운전 중】 소음 , 위치 어긋남	구동음에 이상이 없는지 확인하십시오 . 위치 오차가 발생하지 않는것을 확인하십시오 .



○ 볼트 체결 부위

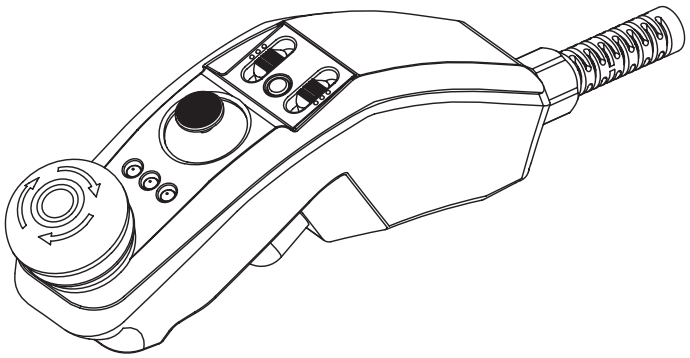
2. 컨트롤러

점검항목	내용
냉각 팬 배기구	먼지 , 이물질에 의해 배기구가 막혀 있지 않은지 확인하십시오 . 배기구는 컨트롤러의 좌우 측면에 있습니다 .
냉각 팬 흡기구	먼지 , 이물질에 의해 공기 흡입구가 막혀 있지 않은지 확인하십시오 . 막힘이나 필터의 손상이 확인 된 경우는 필터를 교환하십시오 .
커넥터	확실하게 체결되어 있는지 아래의 그림과 같은 방법으로 확인하십시오 . 먼지 , 이물질이 없는지 확인하십시오 .



3. JOG 스틱

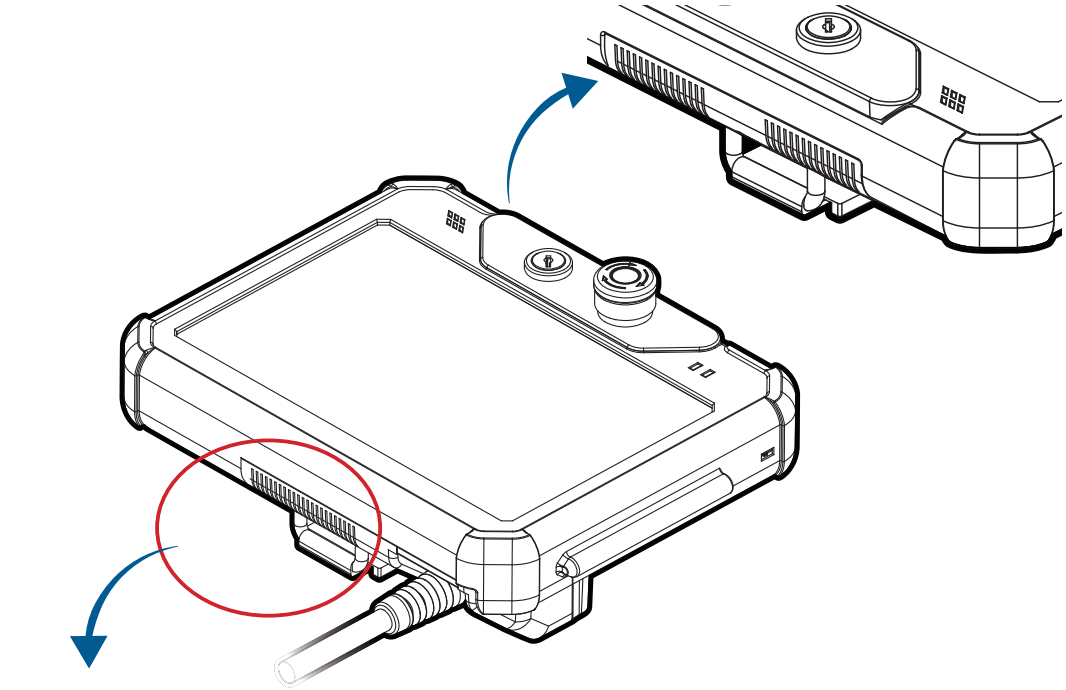
점검항목	내용
외관	금이나 균열이 없는지 확인하십시오 .



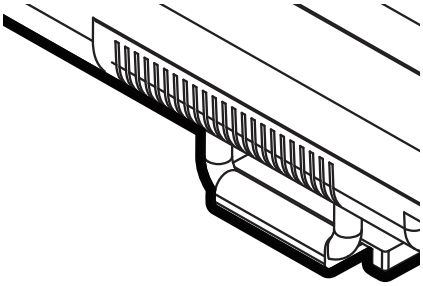
4. 티칭 펜던트

점검항목	내용
외관	금이나 균열이 없는지 확인하십시오 .
흡 · 배기구	먼지 , 이물질에 의해 공기 흡 · 배기구가 막혀 있지 않은지 확인하십시오 . 막힘이나 필터의 손상이 확인 된 경우는 서비스 센터에 문의하십시오 .
커넥터	확실하게 체결되어 있는지 확인하십시오 . 먼지 , 이물질이 없는지 확인하십시오 .

흡 · 배기구 (반대편)



흡 · 배기구



MEMO

Z 자료편

3 용어집

1 용어집

A		
ABS 엔코더 ABS Encoder	앱솔루트 (절대) 엔코더 . 각도 데이터를 외부로 출력할 수 있는 검출기 . 전원을 끄고나서도 위치 정보를 잃지 않는다	
ABS 소실 ABS Lost	조인트 유닛의 엔코더가 절대 위치 정보를 잃은 상태 . 주로 매니퓰레이터가 전원이 꺼진 상태에서 브레이크가 해제되었을 때에 발생한다 . ABS 원점 복귀를 할 필요가 있다 .	
ABS 원점복귀 ABS Zero Return	ABS 소실 상태로부터 복귀하는 작업 . 매니퓰레이터를 영점 마크에 맞춘 원점 자세로 조인트 유닛의 엔코더를 리셋하고 각도 데이터를 재구성한다 .	
API API	응용 프로그래밍 인터페이스 (Application Programming Interface) 의 약자 . 소프트웨어 인터페이스 .	
암 Arm	관절과 관절 사이의 알루미늄 통 부분 . 기종마다 길이가 다르다 .	
비동기 Asynchronous System	운동 동작중 , 다른 작업의 처리를 동시에 수행하는 것을 가능하게 하는 시스템 . 목표점을 예측하고 , 운동 도중에 목표점 전환을 가능하게 한다 .	
B		
베이스 좌표계 Base coordinate System	로봇의 베이스 바닥면에 설정한 좌표계 기본적으로 오프셋이 0 일때 , 월드 좌표계와 일치하고있다	
C		
C.CODE C.CODE	컨트롤러와 매니퓰레이터를 조합하는 코드입니다 . 개체마다 할당합니다 . Connection Code 의 약자 .	
CN1 커넥터 CN1 Connector	매니퓰레이터 케이블을 연결하는 컨트롤러 측의 커넥터 .	
CN2 커넥터 CN2 Connector	JOG 스틱 , 티칭 펜던트 또는 점퍼 커넥터를 연결하는 컨트롤러 측의 커넥터 .	
커맨드 큐 Command queue	파일 전송 , 명령 줄 처리 또는 세션 종료 명령 등 여러 명령의 순서를 지정하는 기능	
컨트롤러 Controller	매니퓰레이터의 복잡한 움직임을 종합적으로 제어하는 장치	
컨트롤 매니져 Control Manager	시스템의 기동 , 정지를 포함한 상태관리 , 에러 핸들링 , 로봇 각 관절의 각도와 로봇이 작업하는 손끝의 위치 자세 (WORLD 좌표계) 와의 관계를 구하는 계산과 가감속 제어를 실시한다 .	
E		
오일러 각 Euler angles	2 개의 직교 좌표계의 위치 관계를 나타낸다 .	

1 용어집

F		
FTP FTP	네트워크에서 파일 전송을 위한 통신 프로토콜 (File Transfer Protocol 의 약자).	
공장 출하시 설정 Factory default settings	공장 초기화된 상태 .	
제 1 암 Frist Arm	조인트에 연결된 암	
H		
원점 Home Position	매니퓰레이터의 모든 좌표값이 0 인 자세 .	
홈 위치 Home Position	월드 좌표계에서 각 좌표 0 의 위치 (0, 0, 0, 0).	
홀드 Hold	감속 정지 , 정지 후 대기 . 로봇 프로그램을 종료하지 않고 다시 사용할 수 있는 상태 .	
I		
I/O 시작 I/O Start	물리적 I/O 또는 메모리 I/O 에 입력된 명령에 의해 로봇 프로그램을 시작하는 조작 또는 기능 .	
초기 설정 프로그램 Initialization Program	초기 설정 프로그램 (init.py)	
초기값 Initial Value	공장 출하시의 설정값	
J		
JOG 스틱 Jog Stick	수동으로 로봇 조작이 가능한 장비 . 교시 때 사용한다 .	
관절 Joint	매니퓰레이터의 가동부 . 모터 , 엔코더 유닛의 일체화 구조 .	
점퍼 커넥터 Jumper Connector	자동 모드용 커넥터 . 부속품 .	
Joint 좌표계 Joint coordinate System	조인트 축으로 설정한 좌표계 . 각 로봇 조인트 축의 각도를 통해 위치와 자세를 정의하는 데 사용하는 좌표계이다 .	

1 용어집

L		
직선 보간 동작 Linear Interpolation	X-Y-Z 축을 동기 제어하면서 합성된 궤적이 직선이 되도록 이동한다 . Line 동작과 같다 .	
Line 동작 Line Motion	X-Y-Z 축을 동기 제어하면서 합성된 궤적이 직선이 되도록 이동한다 . 직선 보간 동작과 같다 .	
M		
매니퓰레이터 Manipulator	" 로봇의 구성 요소 중 물리적인 동작을 수행하는 부분 . 다수의 암과 조인트로 구성되어 있다 ."	
매니퓰레이터 케이블 Manipulator Cable	컨트롤러와 매니퓰레이터를 연결하는 케이블 .	
원점 자세 Mechanical Home Position	매니퓰레이터의 모든 관절을 영점 마크에 맞춘 때의 자세 . ABS 원점 복귀를 할 때의 자세	
메모리 I/O Memory I/O	컨트롤러의 물리적 I/O 및 시스템 I/O 의 총칭 .	
MDO Middle Digital Out	동작 중에 지정된 조건에서 I/O 출력을 LOW/HIGH 로 전환하는 기능 ..	
멀티턴 Multiturn	크로스오버카운터 정보	
O		
직교 좌표계 Orthogonal coordinate system	X-Y-Z 축의 좌표계 . WORLD 좌표계 . 베이스 좌표계 . 유저 좌표계 등 , 모든 직교 좌표계의 총칭 .	
오버라이드 Override	속도설정치에 비율 (%) 을 곱해 설정치를 덮어쓰기 한다 .	
P		
부모 좌표계 Parent Coordinate System	WORLD 좌표계 기준으로 직교 좌표계 형식으로 교시 포인트를 설정하는 데 사용하는 정보 . Position 클래스 , Coordinate 클래스에서 사용한다 .	
물리적 I/O Physical I/O	컨트롤러 Tip I/O 를 연결하는 포트	
PTP 동작 Point To Point Motion	모든 관절이 목표 좌표를 향해 일정한 속도로 부드러운 곡선을 그리며 이동하는 동작	
위치 Position	위치 정보	

1 용어집

R		
재개 Resume	단계 정지 상태 . 홀드 정지 상태에서 다시 작동한다 .	
로봇 Robot	매니퓰레이터 , 컨트롤러를 포함한 총칭 .	
로봇 라이브러리 Robot Library	로봇 동작의 프로그래밍에 필요한 다양한 모듈을 포함하는 파일 .	
로봇 위치 Robot Position	매니퓰레이터의 끝 좌표 (Position 형).	
S		
안전 커넥터 Safety Connector	이상시 로봇의 구동 전원을 차단하고 매니퓰레이터 동작을 정지하기 위한 별도의 외부 보호 장치에 연결하는 인터페이스 커넥터 .	
안전 플러그 Safety Plug	인터락 플러그와 같다 . 안전을 위해 운전 조작 회로를 차단할 수 있는 플러그	
제 2 암 Second Arm	제 1 암과 말단부 사이의 암	
서보 통신 Servo communication	컨트롤러와 매니퓰레이터 관절 사이의 통신 . 동작 명령 및 상태 모니터링 데이터를 송수신하고 있다 .	
감속 정지 Slow down Stop	서보 제어의 의해 감속하면서 정지한다 .	
단계 정지 Step stop	운동 동작 명령을 하나의 실행 단계로 정의하여 , 동작이 완료할 때마다 정지하는 상태 .	
동기 Synchronous System	운동 동작 중 목표 지점에 도달할 때까지 다른 작업을 기다리게 하는 시스템	
시스템 I/O System I/O	시스템 , 프로그램의 포트	
시스템 매니저 System Manager	사용자 로봇 프로그램의 상태 관리 , 교시 상태 제어 , 에러 핸들링의 시스템 제어를 수행한다 .	

1 용어집

T		
작업 좌표계 Task coordinate System	작업에 의해 결정되는 좌표계	
교시 (작업) Teaching	매니퓰레이터를 실제로 움직여 제어 프로그램 동작을 기억시키는 작업 . JOG 스틱과 PC 를 사용하여 작업하는 데 필요한 정보를 설정한다 .	
교시 데이터 Teaching Data	교시 포인트의 데이터 파일	
교시 매니저 Teaching Manager	교시 작업을 제어하는 작업 (i611_teach.py)。	
교시 파라메터 Teaching Parameter	교시 포인트 정보 . 좌표값과 자세값을 포함한다 .	
티칭 펜던트 Teaching Pendant	교시를 위한 각종 설정 모드 변경이 가능한 태블릿 PC	
교시 포인트 Teaching Point	교시로 설정한 매니퓰레이터의 좌표 . 자세가 포함되어 있다 .	
툴 중점 Tool Center Point	툴 접합면을 가진 기계적인 인터페이스 .	
툴 좌표계 Tool Coordinate System	매니퓰레이터의 끝인 tool 을 기준으로 설정한 좌표계	
툴 플랜지 Tool Flange	평평한 툴 접합면을 가진 기계적인 인터페이스 톱 플랜지 , 말단 (끝) 플랜지와 같다 .	
툴 I/O Tool I/O	매니퓰레이터 끝에 장착하는 tool 의 전기적 인터페이스	
U		
유저 로봇 프로그램 User Robot Program	사용자가 만든 로봇 동작 프로그램 . (= 로봇 프로그램)	
W		
워크 Work	작업 대상이 되고 있는 상품 및 부품 .	
월드 좌표계 World coordinate System	지상 또는 작업 바닥에 설정한 좌표계 초기 설정된 오프셋이 0 이기 때문에 베이스 좌표계와 일치한다 .	

Z 자료편

4 문제 해결

1. 오류 로그

1. 오류 로그 구성

로봇은 이상을 감지하면 오류 로그를 저장합니다
오류 발생시 오류 로그를 PC 로 다운로드한 후 서비스 센터에 문의하십시오 .

오류 로그 파일은 .tgz 형식의 압축 파일입니다 .
압축 오류 로그는 다음 파일로 구성되어 있습니다 .

에러 로그 폴더 : /opt/i611/log

파일명	내용
ndstatus	상태 로그
userprog_out.log	사용자 로봇 프로그램 출력 (print 출력)
userprog_err.log	사용자 로봇 프로그램 출력 (예외 출력)
sys_out.log	시스템 관리자 출력 (print 출력)
sys_err.log	시스템 관리자 출력 (예외 출력)

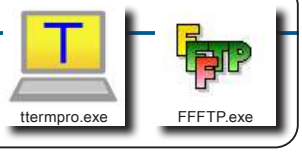
(오류 로그 파일 크기가 200kB 를 초과하면 분할하여 저장됩니다 .)

2. 오류 로그 얻는 방법

단계 1 PC 와의 연결

컨트롤러와 PC 를 연결합니다 .

「C 교시 」를 참고하십시오 .



단계 2 오류 로그의 저장 위치를 선택

USB 메모리를 사용하지 않는 경우

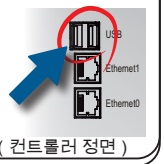
컴퓨터에 다운로드 합니다 .

(오류 로그는 컨트롤러에 저장되어 있습니다 .)

USB 메모리를 사용하는 경우

USB 메모리에 다운로드합니다

USB 메모리를 컨트롤러에
삽입합니다 .



(컨트롤러 정면)

단계 3 터미널 프로그램을 시작하고 , Telnet 연결하기

/ opt / i611 / tools 에있는 get_log.sh 을 실행합니다

```
$ cd /opt/i611/tools
$ ./get_log.sh
```

단계 4 에러로그를 얻기

USB 메모리를 사용하지 않는 경우

컨트롤러 /tmp 에 생성된 에러로그를 FTP 클라이언트에서 PC 로 전송합니다 .

USB 메모리를 사용하는 경우

USB 메모리에 에러 로그가 생성됩니다 .

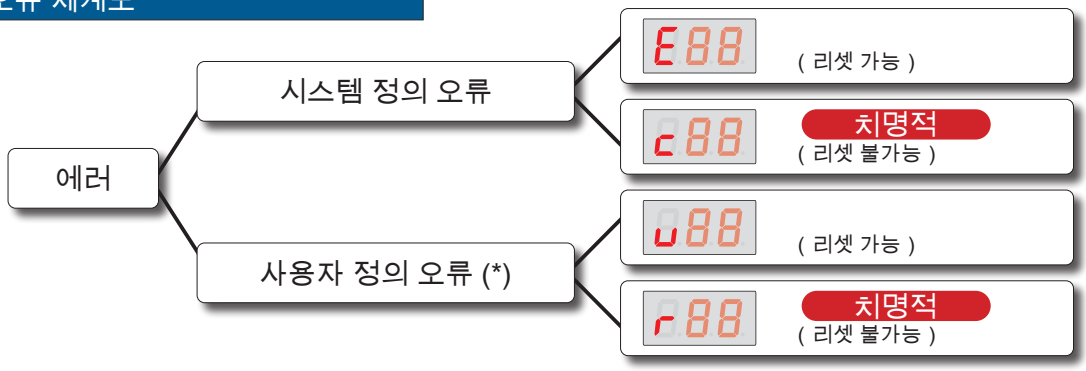
오류 로그 파일 예 : Log_SN16110017A_20170123_102914.tgz

2. 문제 해결

1. 에러의 종류

에러는 4 가지로 분류되고 있습니다 .
에러의 종류나 코드를 확인하세요 . 문제 해결을 참고에 대처하세요 .
에러는 컨트롤러 전면의 7 세그먼트 LED 표시기에 표시됩니다 .

오류 체계도



오류의 종류	오류시 행동지침	
시스템 정의 오류 E88	오류를 재설정할 때까지 로봇은 동작하지 않습니다 .	
	사용자 프로그램	예외가 발생합니다 .
	재설정 방법	오류의 원인을 제거하고 에러 리셋 명령 (cmd_reset ()) 또는 I / O 를 입력 (재설정 후 대기 상태가 됩니다 .)
시스템 정의 오류 치명적 c88	전원을 다시 켤 때까지 로봇은 작동하지 않습니다 . 컨트롤러의 내부 동작은 계속하고 있습니다 .	
	사용자 프로그램	강제 종료합니다 .
	재설정 방법	오류의 원인을 제거하고 전원을 재투입
사용자 정의 오류 (*) u88	사용자 프로그램에서 전용 API 를 호출할 때 발생합니다 . 오류가 재설정 될 때까지 로봇은 작동하지 않습니다 .	
	사용자 프로그램	예외가 발생합니다 .
	재설정 방법	오류의 원인을 제거하고 에러 리셋 명령 (cmd_reset ()) 또는 I / O 를 입력 (재설정 후 대기 상태가됩니다 .)
사용자 정의 오류 치명적 (*) r88	전원을 다시 켤 때까지 로봇은 작동하지 않습니다 . 컨트롤러의 내부 동작은 계속하고 있습니다 .	
	사용자 프로그램	예외가 발생합니다 .
	재설정 방법	오류의 원인을 제거하고 전원을 재투입

*) 오류 코드의 두 자리 숫자번호는 사용자가 정의합니다 . 사용하는 응용 프로그램에 맞게 작성하십시오 .

2 문제 해결

2. 시스템 정의 오류 목록

오류코드		의미
E01	E01	init.py 가 발견되지 않았다 .
E02	E02	init.py 에서 오류가 발생했다 .
E03	E03	로봇 프로그램이 실행되지 않았다 .
E04	E04	로봇 프로그램이 설정되어 있지 않았다 .
E05	E05	로봇 프로그램을 실행할 수 없는 모드로 되어 있었다 .
E06	E06	i611Robot 클래스의 open () 가 실행되기 전에 로봇 동작 API 를 사용했다 .
E07	E07	ABS 원점을 잃어버린 동안 로봇 프로그램이 수행되었다 .
E08	E08	로봇 프로그램이 비정상적으로 종료했다 .
E09	E09	로봇 프로그램이 비상 정지 중에 i611Robot 클래스의 open () 를 실행했다 .
E10	E10	로봇 프로그램이 서보 OFF 중에 i611Robot 클래스의 open () 를 실행했다 .
E11	E11	로봇 프로그램이 조작 권한을 취득하지 않았다 .
E12	E12	로봇 프로그램이 시스템 관리자와 통신할 수 없었다 .
E13	E13	비상 정지의 예외가 없다 .
E14	E14	로봇 프로그램의 exit () 메소드가 비정상적으로 종료했다 .
E15	E15	로봇 프로그램이 예외로 종료했다 .
E16	E16	감속 정지의 예외가 없다 .
E17	E17	시스템 종료 과정을 완료하지 않았다 .
E18	E18	메모리 I / O 에 액세스할 수 없었다 .
E19	E19	i611Robot 클래스의 인스턴스가 하나의 프로세스에서 여러 번 만들어졌다 .
E20	E20	i611Robot 클래스의 open () 이 하나의 프로세스에서 여러 번 열렸다 .
E21	E21	다른 스레드에서 API 부정 호출이 발생했다 .
E40	E40	티칭 과정에서 비정상적으로 종료했다 .
E53	E53	홈 디렉토리 (/ home / i611usr) 폴더의 사용량이 한도를 초과했다 .
E99	E99	기타 오류가 발생했다 .

2 문제 해결

3. 시스템 정의 오류 (치명적) 목록

오류코드		의미
c01	c01	시스템 관리자를 시작하지 못했습니다 .
c02	c02	시스템 관리자가 비정상적으로 종료했다 .
c03	c03	시스템 관리자가 제어 관리자와 통신할 수 없었다 .
c04	c04	JOG 조작 모드 중에 오류가 발생했습니다 .
c05	c05	제어 관리자가 비정상적으로 종료했다 .
c06	c06	컨트롤러의 저장 공간의 여유가 없어졌다 .
c10	c10	(조인트) 회로가 손상되었다 .
c11	c11	(조인트) 과전류가 발생했다 .
c12	c12	(조인트) 브레이크의 결함이 발생했다 . (서보 OFF → ON 시)
c13	c13	(조인트) 과도한 토크가 감지되었다 .
c14	c14	(조인트) 과부하 (열) 가 발생되었다 .
c15	c15	(조인트) 구동 전압이 떨어졌다 .
c16	c16	(조인트) AC 전원 이상이 발생했다 .
c17	c17	(조인트) 서보 통신 이상이 발생했다 .
c18	c18	(조인트) 서보 ON 표시 이상 1 이 발생했다 . (정상 동작이 안 된다 .)
c19	c19	(조인트) 서보 ON 표시 이상 2 가 발생했다 .(Z 검출이 안 된다 .)
c20	c20	(조인트) ABS 손실 : 앱솔루트 엔코더값을 검출할 수 없다 .
c21	c21	(조인트) ABS 소실 : 앱솔루트 엔코더값에 오류가 발생했다 .
c22	c22	(조인트) ABS 소실 : 인크리멘탈 엔코더 값을 검출할 수 없다 .
c23	c23	(조인트) ABS 소실 : 인크리멘탈 엔코더가 손상되었다 .
c24	c24	(조인트) ABS 소실 : 인크리멘탈 엔코더의 배터리 전압이 떨어졌다 .
c25	c25	(조인트) 상태 변경 조건에 오류가 발생했다 .
c26	c26	Tip I / O 에서 이상이 발생했다 .
c28	c28	내부 모니터 처리에서 이상이 발생했다 .
c29	c29	냉각 팬이 정지했다 .

2 문제 해결

시스템 정의 오류 (치명적)

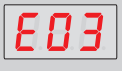
오류코드		의미
c30	c30	회생 저항기 이상 1 이 발생했다 .
c31	c31	주회로 릴레이가 고장했다 .
c32	c32	" 비상 정지 회로 " 에 배선 이상이 감지되었다 .
c33	c33	" 모드 회로 " 에 배선 이상이 감지되었다 .
c34	c34	제어 전원에 이상이 발생했다 .
c35	c35	돌입 방지 저항이 발열 이상이 감지되었다 .
c36	c36	회생 저항기 이상 2 가 발생했다 .
c37	c37	회생 저항기 이상 3 이 발생했다 .
c39	c39	로봇의 통신이 단절했다 .
c40	c40	" 문 회로 " 에 이중화 신호의 불일치가 발생했다 .
c41	c41	" 모드 회로 " 에 이중화 신호의 불일치가 발생했다 .
c42	c42	상태 전이 시간에 따른 슬레이브 오류가 발생했다 .
c43	c43	인터럽트에 의한 통신 오류가 발생했습니다 .
c44	c44	속도 오버의 슬레이브 에러가 발생했다 .
c58	c58	SPI 회로에 이상이 발생했다 .
c59	c59	로봇 정의 파일이 이상이 감지되었다 .
c60	c60	작업 오류가 발생했습니다 .
c89	c89	(조인트) EtherCAT 통신 패킷에 이상이 발생하였다 .
c91	c91	(조인트) 위치 편차 이상 속도 이상이 감지되었다 .
c92	c92	(조인트) 조인트 파라미터 이상이 감지되었다 .
c93	c93	(조인트) 엔코더 통신 이상이 발생했다 .
c94	c94	(조인트) 제어 보드가 과열되었다 .
c95	c95	(조인트) EtherCAT 통신의 동기화 이상이 발생했다 .
c96	c96	(조인트) 제어 동기화에 이상이 발생했다 .
c98	c98	전원이 차단되었다 .
c99	c99	기타 오류가 발생했다 .

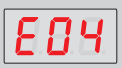
4. 에러 대처법

에러의 종류	에러시 행동지침	
시스템 정의 오류 	오류를 재설정할 때까지 로봇은 동작하지 않습니다 . 다음의 해당 문제 해결을 참고하여 대처하십시오 .	
	사용자 프로그램	예외가 발생합니다 .
	재설정 방법	오류의 원인을 제거하고 에러 리셋 명령 (cmd_reset ()) 또는 I / O 입력 (재설정 후 대기 상태가됩니다 .)
시스템 정의 오류 치명적 	전원이 다시 공급될 때까지 로봇은 동작하지 않습니다 . 컨트롤러의 내부 동작은 계속하고 있기 때문에 외부에서의 입력이 불가능합니다 . 개선되지 않는 경우에는 서비스 센터에 문의하십시오 .	
	사용자 프로그램	강제 종료합니다 .
	재설정 방법	오류의 원인을 제거하고 전원을 재투입

	init.py 가 발견되지 않았다 .	
	원인	컨트롤러 / home / i611usr / 에 init.py 가 없습니다 .
	대처	/ opt / i611 / tools / 에 있는 init.py 를 / home / i611usr / 에 복사합니다 .

	init.py 에서 오류가 발생했다 .	
	원인	init.py 코딩에 잘못이 있습니다 .
	대처	init.py 를 확인하십시오 .

	로봇 프로그램이 실행되지 않았다 .	
	원인	로봇 프로그램이 제대로 지정되어 있지 않습니다 .
	대처	지정한 파일 이름을 확인하십시오 . 예) rbs = RobSys() rbs.open() rbs.set_robtask ('filename.py')

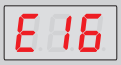
	로봇 프로그램이 설정되어 있지 않았다 .	
	원인	rbs.set_robtask ('filename.py') 에 의한 지정이 빠져 있거나 존재하지 않는 파일이 지정되어 있습니다 .
	대처	rbs.set_robtask ('filename.py') 에서 지정하는 파일 이름을 확인하십시오 .

2 문제 해결

E05	로봇 프로그램을 실행할 수 없는 모드로 되어 있었다 .	
	원인	컨트롤러의 CN2 에 점퍼 커넥터 (부속품) 이 연결되어 있지 않습니다 .
	대처	CN2 에 점퍼 커넥터가 연결되어 있는지 확인하십시오 . CN2 에 JOG 스틱이 연결되어 있는 경우 , 또는 아무것도 연결되어 있지 않으면 로봇 프로그램을 실행할 수 없습니다 .
E06	i611Robot 클래스의 open () 가 실행되기 전에 로봇 동작 API 를 사용했다 .	
	원인	로봇 프로그램 중에 i611Robot 클래스에 대한 설명이 없습니다 . 또는 i611_MCS 모듈을 가져올 수 없습니다 .
	대처	로봇 프로그램이 rb=i611Robot( ) rb.open( ) 을 기술되어 있는지 확인하십시오 .
E07	ABS 원점을 잃어버린 동안 로봇 프로그램이 수행되었다 .	
	원인	ABS 원점 복귀가 이루어지고 있지 않습니다 . 또는 컨트롤러가 꺼져있을 때에 브레이크가 해제되었습니다 .
	대처	조작의 ABS 원점 조정을 진행하십시오 . ・ 설치설명서 「8. ABS 원점 복귀」 ・ 사용설명서「C 티칭 편」
E08	로봇 프로그램이 비정상적으로 종료했다 .	
	원인	로봇 프로그램에 예기치 않은 예외가 발생했습니다 .
	대처	로봇 프로그램의 오류 내용을 확인하여 오류 부분을 수정하십시오 .
E09	로봇 프로그램이 비상 정지 중에 i611Robot 클래스의 open () 를 실행했다 .	
	원인	비상 정지 스위치가 눌러진 상태에서 로봇 프로그램을 시작했습니다 .
	대처	비상 정지 스위치를 해제하십시오 .
E10	로봇 프로그램이 서보 OFF 중에 i611Robot 클래스의 open () 를 실행했다 .	
	원인	서보를 끈 상태에서 i611Robot 클래스의 open () 를 실행했습니다 .
	대처	서보를 켜주십시오 .

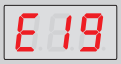
E11	로봇 프로그램이 조작 권한을 취득하지 않았다 .	
	원인	i611Robot 클래스에서 rb=i611Robot( ) rb.open(permission=False) 호출 또는 open () 을 중복 수행했습니다 .
	대처	로봇 프로그램에서 rb=i611Robot( ) rb.open(permission=True) (← rb.open( ) 도 가능합니다) 이 포함되어 있는지 확인하십시오 . 또는 여러 open () 를 호출하지 않았는지 확인하십시오 .
E12	로봇 프로그램이 시스템 관리자와 통신할 수 없었다 .	
	원인	예기치 않은 오류가 발생했습니다 .
	대처	컨트롤러의 전원을 재투입하십시오 . 개선되지 않는 경우에는 서비스 센터에 문의하십시오 .
E13	비상 정지의 예외가 되지 않았다 .	
	원인	try 문의 설명이 없거나 try 문에 except 설명이 부족합니다 .
	대처	로봇 프로그램에서 try: … except Robot_emo: 이 포함되어 있는지 확인하십시오 .
E14	로봇 프로그램의 exit () 메소드가 비정상적으로 종료했다 .	
	원인	로봇 프로그램에서 exit () 메소드의 인수가 0 이 아닌 값을 지정합니다 .
	대처	정상 종료시에는 exit () 메소드의 인수를 0 으로 합니다 . rb=i611Robot() … rb.exit (0)
E15	로봇 프로그램이 예외로 종료했다 .	
	원인	프로그래밍 오류로 인한 예외가 발생합니다 .
	대처	오류 내용을 확인하여 오류 부분의 프로그램을 수정하십시오 .

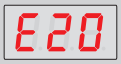
2 문제 해결

	감속 정지의 예외가 없다 .	
	원인	try 문의 설명이 없거나 try 문에 except 설명이 부족합니다 .
	대처	로봇 프로그램에서 try: … except Robot_stop: 이 포함되어 있는지 확인하십시오 .

	시스템 종료 과정을 완료하지 않았다 .	
	원인	비상 정지 시의 인터럽트 처리 (Robot_emo) 가 시간 초과했습니다 .
	대처	비상 정지 인터럽트 처리는 5 초 이내에 완료하고 프로그램을 종료하도록 설정하십시오 .

	메모리 I / O 에 액세스 할 수 없었다 .	
	원인	I / O 커넥터가 제대로 연결되어 있지 않거나 제어 관리자가 비정상적으로 종료했습니다 .
	대처	I / O 커넥터의 연결 상태를 확인하십시오 .

	i611Robot 클래스의 인스턴스가 하나의 프로세스에서 여러 번 만들어졌다 .	
	원인	로봇 프로그램에서 i611Robot 클래스의 인스턴스가 여러 설명되어 있습니다 .
	대처	하나의 프로세스에서 i611Robot 클래스의 인스턴스를 중복해 호출하고 있는지 확인하십시오 .

	i611Robot 클래스의 open () 이 하나의 프로세스에서 여러 번 실행되었다 .	
	원인	로봇 프로그램에서 i611Robot 클래스의 open () 이 여러 설명되어 있습니다 .
	대처	하나의 프로세스에서 i611Robot 클래스의 open() 를 중복해 호출하고 있는지 확인해 주십시오 .

	다른 스레드에서 API 부정 호출이 발생했다 .	
	원인	인스턴스를 생성하지 않았거나 다른 스레드에서 호출이 금지되는 메소드로 호출했습니다 .
	대처	인스턴스를 생성하고 호출합니다 . i611Robot 클래스의 API 로 다른 스레드에서 호출할 수 있는 것은 , abort(), stop(), pause(), restart() 입니다 .

2 문제 해결

E40	티칭 과정에서 비정상적으로 종료했다 .	
	원인	티칭 관리자가 비정상적으로 종료되었습니다 .
	대처	컨트롤러의 전원을 재투입하십시오 . 개선되지 않는 경우에는 서비스 센터에 문의하십시오 .
E53	홈 디렉토리 (/ home / i611usr) 폴더의 사용량이 한도를 초과했다 .	
	원인	컨트롤러의 홈 디렉토리의 용량이 부족합니다 .
	대처	/ home / i611usr 에있는 불필요한 파일을 / home / i611usr / ext 로 이동하는 등 폴더의 용량을 확보하십시오 .
E99	기타 오류가 발생했다 .	
	원인	예기치 않은 오류가 발생했습니다 .
	대처	컨트롤러의 전원을 재투입하십시오 . 개선되지 않는 경우에는 서비스 센터에 문의하십시오 .

MEMO

MEMO

서비스 센터
제우스 : 경기도 화성시 안녕남로 132
e-mail : zero@globalzeus.com