D

소프트웨어

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

1.1.0-01(D) / 2019.03

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, 줄은 LF 합니다 .

터미널 소프트웨어 (Tera Term)

Telnet 연결에서 작동 프로그램을 실행하는 등 컨트롤러를 조작합니다 .

FTP 클라이언트 소프트웨어 (FFFTP)

PC 와 컨트롤러 간 파일 전송을 합니다 .

웹 브라우저 (Google Chrome)

교육은 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」를참조하십시오 .

전체의 흐름에서 찾기

5 페이지

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

" 동작 모델 " 에서 찾기

64 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 .

" 전체의 흐름 " 에서 찾기

원하는 프로그램 블록을 선택「1. 로봇 프로그램 작성」

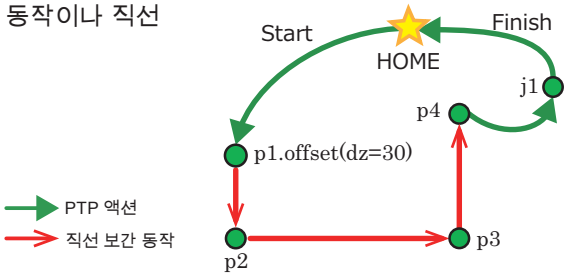
" 동작 모델 " 에서 찾기

원하는 동작 모델을 선택 "2. 샘플 프로그램 "

모델 1 : 기본 동작

85 페이지

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



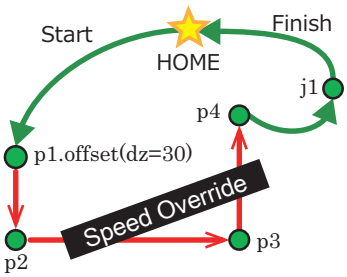
모델 2 : 오버라이드

86 페이지

기본 동작과 같은 동작을 MotionParam () 에서 설정한 동작 속도에 제한을 겁니다 .

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

본 동작 프로그램의 동작 확인을 하는 경우 등에 사용합니다 .



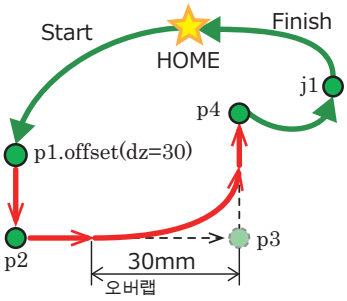
모델 3 : 오버랩

87 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)

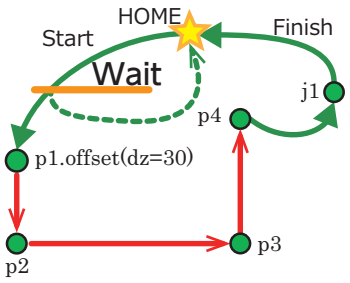


모델 4 : I/O 입력 대기

88 페이지

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

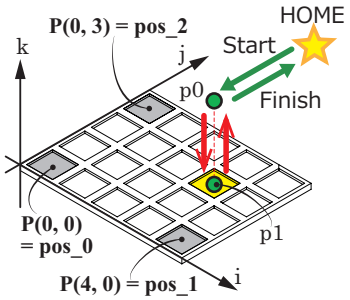
I/O 입력을 사용하면 미리 컨트롤러에 등록된 로봇 프로그램을 I/O 에서 시작할 수 있습니다 .



모델 5 : 팔레트 기능

89 페이지

워크 반송 팔레트 셀의 개수와 네 모서리의 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 한글로 취급 설정

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

프로그램 예

```
#!/usr/bin/python        ——— 인터프리터 지정
# –*– coding: utf–8 –*–  ——— 문자 코드의 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 모듈을 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 좌표계의 정의

월드 좌표계를 사용할 때 설정합니다 .

```
    # 좌표계의 정의
    _BASE = Base( )
```

로봇과의 연결 시작과 초기화

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

I / O 입출력 기능의 초기화

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

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 좌표계의 정의
        _BASE = Base( )
    # 로봇과 연결 시작 초기화
        rb.open( True )
    # I/O 입출력 기능의 초기화 (I / O 미사용시 생략 가능 )
        IOinit( rb )
    # 속도오버라이드 50%
        rb.override( 50 )
```

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( )

접합 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( –50, –250, 350, 90, 0, 180 )
        p2 = Position( –300, –250, 350, 90, 0, 180 )
        p3 = Position( –50, –250, 350, 90, 0, 180 )

        j1 = Joint( 10, 30, 10, 0, 5, 30 )
```

교시 포인트는 Position 형 또는 Joint 형으로 설정합니다 .

인수의 생략

인수는 지정하려는 매개 변수까지 입력합니다 .

(예)

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

rz 이후의 매개 변수는 초기값으로 설정됩니다 .

교시 포인트를 조정

replace( )

월드 좌표계 객체를 대체합니다 .
(자신을 업데이트)

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( –50, –250, 350, 90, 0, 180 )
p1.replace( x=100, rz=–50 )
```

실행결과
[100, –250, 350, –50, 0, 180]

shift( )

월드 좌표계 객체를 이동합니다 .
(자신을 업데이트)

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( –50, –250, 350, 90, 0, 180 )
p1.shift( dx=10 )
```

실행결과
[–40, –250, 350, 90, 0, 180)]

offset( )

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

인수	의미	변수 형태	단위
dx, dy, dz	위치의 오프셋 량 (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( –50, –250, 350, 90, 0, 180 )
p2 = p1.offset( dx=10 )
```

실행결과
p1 = [–50, –250, 350, 90, 0, 180]
p2 = [–40, –250, 350, 90, 0, 180]

파일에 저장된 교시 데이터를 이용

Teachdata( ) 교시 데이터를 로드 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	도구 ID 취득 플래그	bool	-
base	기반 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 사용한 팔레트를 정의
pal = Pallet()
pal.init_3( pos_0, pos_1, pos_2, 5, 4 )
```

init_4( ) 팔레트를 정의합니다 . (4 점 교시)

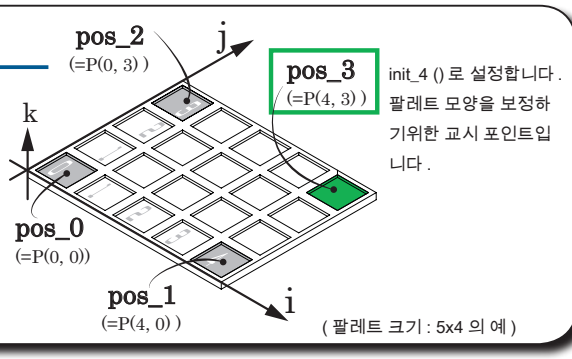
인수	의미	변수 형태	단위
pos_0	[Position] 팔레트에 교시 포인트 (원점) 필 수	float	-
pos_i	[Position] 팔레트에 교시 포인트 (i 방향) 필 수	float	-
pos_j	[Position] 팔레트에 교시 포인트 (j 방향) 필 수	float	-
pos_ij	[Position] 팔레트에 교시 포인트 필 수	float	-
ni	팔레트의 i 방향으로 줄 지어있는 셀의 개수 필 수	integer	-
nj	팔레트의 j 방향으로 줄 지어있는 셀의 개수 필 수	integer	-

```
# 4 점 교시 데이터를 사용한 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 .



팔레트 기능을 이용하기

get_pos( ) 셀의 위치를 가져옵니다 .

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

adjust( ) 팔레트의 셀 위치를 보정합니다 .

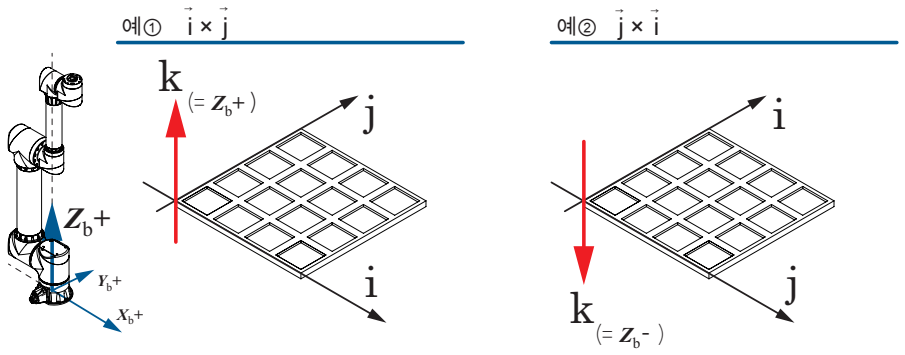
인수	의미	변수 형태	단위
i	팔레트에서 셀의 위치를 지정하는 인덱스 (i 방향) 필 수	integer	-
j	팔레트에서 셀의 위치를 지정하는 인덱스 (j 방향) 필 수	integer	-
di	i 방향셀의 위치의 오프셋 량 필 수	integer	mm
dj	j 방향셀의 위치의 오프셋 량 필 수	integer	mm

포인트!

팔레트의 수직 방향

교시 포인트의 배치에 의해 팔레트 수직 방향 (k 방향) 의 양의 방향이 바뀝니다 .

예 ① i × j : 팔레트 평면에 상승 수직 인 벡터 z + 입니다 .
예 ② j × i : 팔레트 평면에 아래쪽 수직 인 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 (직선보간동작)) 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 : 2	integer	–
passm	경로동작 초기값 : 2	integer	–
overlap	오버랩동작 초기값 : 0.0	float	mm
zone	위치 결정 완료 범위 초기값 : 100	integer	pulse
pose_speed	속도 (자세 보간동작) 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 )
```

5. 로봇 동작의 정의

로봇을 이동

home( ) 모든 축을 Joint 좌표 0deg 로 이동합니다 .

move( ) PTP 동작을 일정한 속도로합니다 . (*)

line( ) 직선 보간 동작을 일정한 속도로합니다 . (*)

optline( ) 직선 보간 동작을 최적의 속도로 변속하면서 움직입니다 .

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

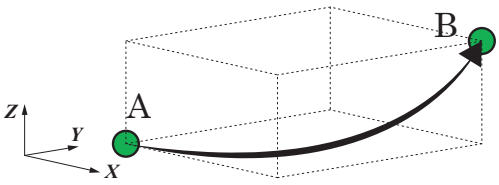
```
## 5. 로봇 동작 설정 ###################################
    # 각 축 좌표 [0, 0, 0, 0, 0, 0] 로 이동
    rb.home()
    # 이동
    rb.move( p1.offset(dz=30) )      ← PTP 동작으로 p1 대해 dz = 30 오프셋 좌표로 이동
    rb.line( p2, p3, p4 )            ← Line 동작에서 p2, p3, p4 와로 이동
    rb.move( j1 )
    # 각 축 좌표 [0, 0, 0, 0, 0,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

포인트 !

PTP 동작과 직선 보간 동작과 최적 직선 보간 동작

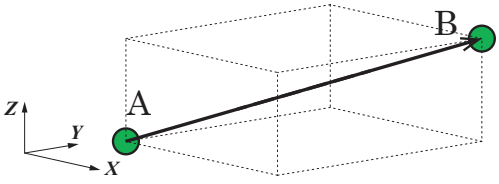
PTP 동작 (move())

모든 관절이 목표 좌표를 향해 일정한 속도 각도에서 동작합니다 . 부드러운 곡선을 그리며 이동하는 동작



직선보간동작 (line())

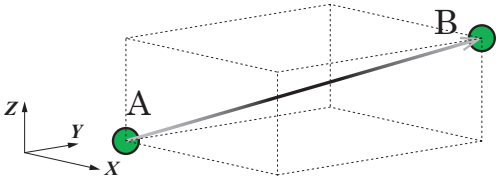
X-Y-Z 축 동기 제어하면서 목적지까지의 궤적이 직선이되도록 일정한 속도로 이동하는 동작



최적 직선 보간 동작 (optline ())

X-Y-Z 축 동기 제어하면서 목적지까지의 궤적이 직선이되도록 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, 137.0, 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

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 · 연속 된 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 않는	string	-

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

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

포인트 !

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 : 목록
    if wait( 8, '1', 10 )[0] ==  1:
    if wait( 9, '1', 10 )[1] ==  '1':
    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 설정 한 I / O 포트에 입력 신호를 켭니다 .

(신호의 상승 에지에서 로봇 프로그램의 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 .

위험

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 (비 치명적 또는 치명적) 의 발생 상태입니다 .
2 개의 오류 상태를 1 개의 제어선으로 확인하는 경우에 사용합니다 .

주 의

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

포인트 !

보충

· 로봇 프로그램이 오류 종료한 경우 :
. . . . . 오류를 재설정 할 때까지 프로그램을 재시작할 수 없습니다 .
· " 프로그램 동작 중 ' 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	목록 형식의 각 축 각도 필 수

```
#Joint 좌표값
    j10=Joint( 0, 30, 60, 0, 90, 90 )

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

Position2Joint( )

Position 좌표값에서 Joint 좌표값으로 변환합니다 .

인수	의미
Position 형	목록 형식의 위치 정보 필 수

```
#Position 형 좌표값
    p10=Position( –50, –250, 350, 90, 0, 180 )

    #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 이어져 있는 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 미리 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 미리 동작 OFF (rb.asyncm (2) 에서도 가능 )
...
rb.close()
```

오버랩 실행중인 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 예외의 발생을 설정합니다 .

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

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

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

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

# 예 4 : 일시 정지 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 경우는 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 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 정지시에 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 정지 시에 발생하는 예외

2 프로그래밍 가이드

6. 종료

로봇 프로그램 종료

close( ) 로봇과의 연결을 종료합니다 .

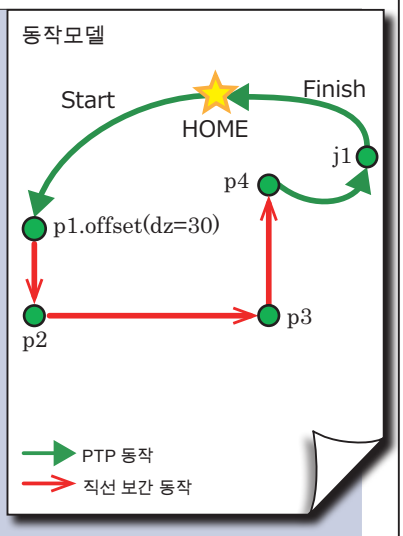
```
# 종료
    rb.close()
```

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

2. 샘플 프로그램

모델 1 기본 동작

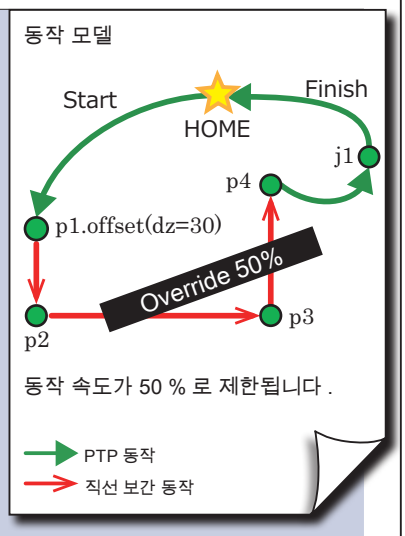
```
#!/usr/bin/python
# –*– coding: utf–8 –*–

## 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()
        # 좌표계의 정의
        _BASE = Base()
        # 로봇과 연결 시작 초기화
        rb.open()
        # I/O 입출력 기능의 초기화 (I/O 미사용시 생략 가능 )
        IOinit( rb )

    ## 3. 교시 포인트 설정 ###################
        p1 = Position( 95, –280, 425, –120, 84, –28 )
        p2 = Position( 95, –280, 240, 154, 80, –114 )
        p3 = Position( 300, –280, 240, 159, 86, –156 )
        p4 = p3.copy()
        p4.shift( dz=40 )
        j1  = Joint( 230, –1, –92, 90, 5, 89 )

    ## 4. 동작 조건 설정 ###################################
        #MotionParam 생성자에서 동작 조건 설정
        m = MotionParam( jnt_speed=10, lin_speed=70 )   ← PTP 동작 속도 10 %, 직선 보간 동작 속도 70mm/s
        #MotionParam 형으로 동작 조건 설정
        rb.motionparam( m )

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

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

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



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 .

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()
    # 좌표계의 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능의 초기화 (I/O 미사용시 생략 가능 )
        IOinit( rb )
    # 오버라이드
        rb.override( 50 )        ——— 오버라이드를 50 % 로 설정

    ## 3. 교시 포인트 설정 ###################
        p1 = Position( 95, –280, 425, –120, 84, –28 )
        p2 = Position( 95, –280, 240, 154, 80, –114 )
        p3 = Position( 300, –280, 240, 159, 86, –156 )
        p4 = p3.offset( dz=40 )
        j1  = Joint( 230, –1, –92, 90, 5, 89 )

    ## 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 동작을 정의 ##############################
        # 작업 시작
        rb.home()                ——— Home 위치로 이동
        rb.move( p1.offset(dz=30) )  ——— 교시 포인트 p1 에서 Z 축 오프셋 30mm 에 PTP 동작
        rb.line( p2, p3, p4 )    ——— 교시 포인트 p2, p3, p4 의 순서로 직선 보간 동작
        rb.move( j1 )            ——— 교시 포인트 j1 에 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 오류
        rb = i611Robot()
    # 좌표계의 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능의 초기화 (I / O 미사용시 생략 가능 )
        IOinit( rb )

    ## 3. 교시 포인트 설정 ###################
        p1 = Position( 95, –280, 425, –120, 84, –28 )
        p2 = Position( 95, –280, 240, 154, 80, –114 )
        p3 = Position( 300, –280, 240, 159, 86, –156 )
        p4 = p3.offset( dz=40 )
        j1  = Joint( 230, –1, –92, 90, 5, 89 )

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

        ## 5. 로봇 동작을 정의 ##############################
        rb.home()
        rb.move( p1 )
        rb.line( p2 )
        # 프로그램 예측 동작 ON
        rb.asyncm( 1 )
        rb.line( p3, p4 )
        rb.join()
        # 프로그램 예측 동작 OFF
        rb.asyncm( 2 )
        # 각 축 좌표 [0, 0, 0, 0, 0, 0]
        rb.home()

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

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

오버랩을 30mm 로 설정

Home 위치로 이동

교시 포인트 p1 에 PTP 동작

교시 포인트 p2 에 직선 보간 동작

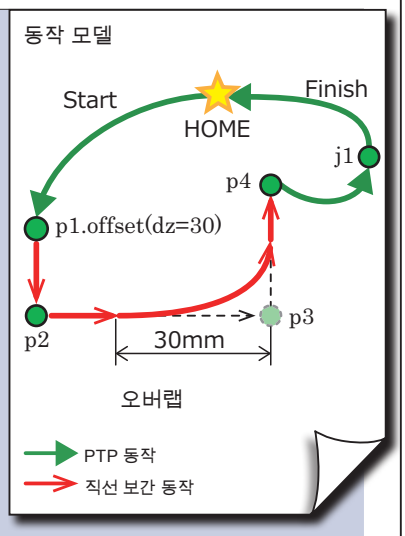
프로그램 예측 동작 ON (오버랩 동작의 실행 준비)

오버랩 동작 p3, p4 로 이동

오버랩 작동 중지 위치로 예측한 프로그램의 완료 대기

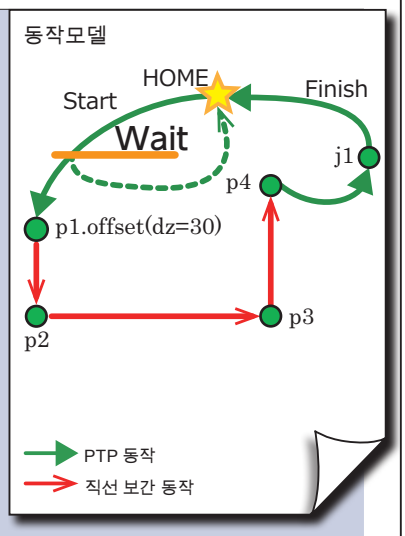
프로그램 예측 동작 OFF

Home 위치로 이동



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
        rb = i611Robot()
    # 좌표계의 정의
        _BASE = Base()
    # 로봇과 연결 시작 초기화
        rb.open()
    # I/O 입출력 기능의 초기화
        IOinit( rb )

    ## 3. 교시 포인트 설정 ######################
        p1 = Position( 95, –280, 425, –120, 84, –28 )
        p2 = Position( 95, –280, 240, 154, 80, –114 )
        p3 = Position( 300, –280, 240, 159, 86, –156 )
        p4 = p3.offset( dz=40 )
        j1  = Joint( 230, –1, –92, 90, 5, 89 )

    ## 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 동작을 정의 ##############################
    # 작업 시작
        rb.home()                          ——— Home 위치로 이동
    # I/O 입력 대기
        if wait( 5, "1", 120 )[0] == 1:    ——— I / O 입력 포트 번호 5 가 1 이 될 때까지 대기 시간 120 초로 설정
                                                ( 시간 전에 입력 신호 )
            rb.move( p1.offset(dz=30) )    ——— 교시 포인트 p1 에서 Z 축 오프셋 30mm 에 PTP 동작
            rb.line( p2, p3, p4 )          ——— 교시 포인트 p2, p3, p4 의 순서로 직선 보간 동작
            rb.move( j1 )                  ——— 교시 포인트 j1 에 PTP 동작
    # 입력 시간 초과시
        else:                              ——— 초과된 경우
            rb.home()                      ——— Home 위치로 이동
                                                ( 이 예제에서는 Home 위치에서 이동을 안합니다 )
    ## 6. 종료 ##########################################
    # 로봇과의 연결을 종료
        rb.close()

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



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
        rb = i611Robot()
        rb.open()
    # 좌표계의 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 동작을 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 모서리 셀의 위치 설정

팔레트 위치 크기의 정의

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

PTP 동작 속도 30 %

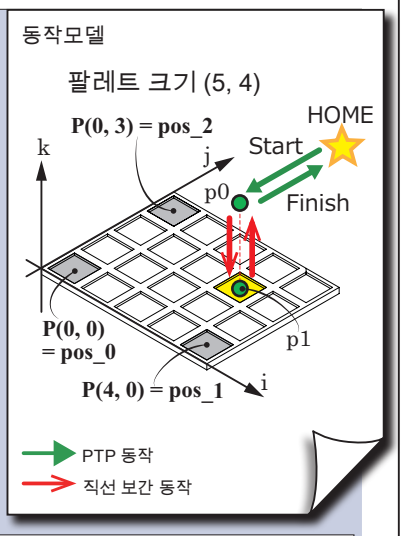
Home 위치로 이동 작업 시작

p0 (팔레트 위치 (3,2) 바로 30mm) 에 PTP 동작으로 이동

p1 (팔레트 위치 (3,2)) 에 직선 보간 동작으로 이동

p0 (팔레트 위치 (3,2) 바로 30mm) 에 PTP 동작으로 이동

Home 위치에 PTP 동작으로 작동



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

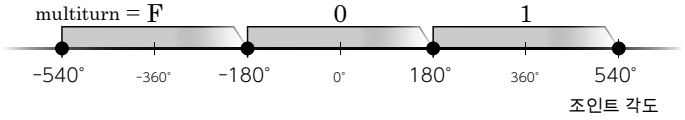
형 (아이콘)	의미
리스트 List	[. . .] 에서 요소를 늘어 놓은 것입니다 .
튜플 Tuple	(. . .) 에서 요소를 늘어 놓은 것입니다 . 튜플 요소를 변경할 수 없습니다 .
딕셔너리 Dict.	{. . .} 는 " 딕셔너리 " 라는 키 (key) 와 값 (value) 을 콜론 (:) 으로 구분합니다 . (예 , key : value)
가변 인수 Vari. No.	리스트를 확장합니다 . 인수앞에「* (별표)」를 1 개 넣습니다。받은 인수는 지정된 순서대로 튜플에 저장됩니다 .
키워드 인수 Keyword	딕셔너리를 확장합니다 . 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) 초기값 : 0.0
	parent [-] float	월드 좌표계를 사용하는 설정 (*1) 초기값 : _BASE
	posture [-] integer	자세 (*2) 초기값 : –1
	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 . J1 에서 J6 의 각 관절의 각도 범위에 따라 다음 값을 설정합니다 . ・ FLAG 부분 : 크로스오버 카운터 정보의 유무 FF : 없음 (초기값) 00 : 있음 (상기 이외는 무효) ・ J1 - J6 부분 : 크로스오버 카운터 값

설정값	각관절의각도
1	180° ~ 540°
0	–180° ~
F (*4)	–540° ~

크로스오버 카운터 값과 조인트 각도의 관계



*1) 설정값은 "_BASE" 입니다 .
*2) 암의 위치 , 방향 , 각도에 따라 8 가지 자세가 있습니다 .

(C 교시 5 좌표계와 자세)

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

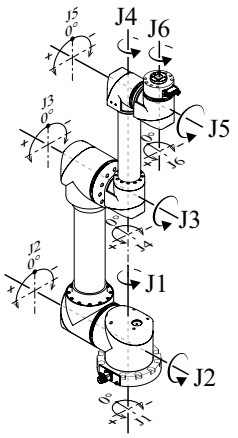
(C 교시 4 교시)

multiturn 매개 변수에 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 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MotionParam [MotionParam]	여러 변수로 구성된 로봇의 동작 매개 변수	
	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 : 2
	passm [-] integer	경로 동작 초기값 : 2
	overlap [mm] float	overlap 동작 초기값 : 0.0
	zone [pluse] 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 속도의 상한을 설정합니다 .
	ik_solver_option [-] long	회전 방향 초기값 : $0 \times \frac{1}{(\mathrm{Rsv.})}\frac{1}{J6}\frac{1}{J5}\frac{1}{J4}\frac{1}{J3}\frac{1}{J2}\frac{1}{J1}$ J1 - J6 의 설정값 0 : 최단 경로 multiturn 매개 변수의 정보를 사용하여 회전합니다 . 1 : multiturn 매개 변수의 정보를 사용 2 : + 방향으로 회전 3 : ー방향으로 회전 + 방향、ー방향은 오른쪽 그림을 참조



회전 방향의 정의

2. 모듈

1. 모듈 목록

로봇 제어 모듈

모듈	클래스	요약
i611_MCS	Base P. 23 ~	월드좌표계를 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 매개 변수를 취급
	i611Robot P. 47 ~	로봇의 동작을 취급

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 (▭ 의 메소드) 는 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

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 (▭ 의 메소드) 는 생성자입니다 .

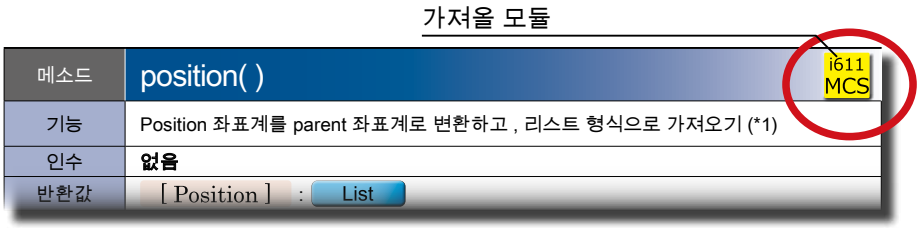
Exception 클래스를 상속한 클래스				클래스
Robot_emo( ) p. 109	Robot_error( ) p. 110	Robot_fatalerror( ) p. 110	Robot_poweroff( ) p. 111	Exception
Robot_stop( ) p. 112				

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 매개 변수를 설정하십시오 .
	`m = MotionParam(jnt_speed=10,lin_speed=70,overlap=30)`
	(생략된 매개 변수는 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 매개 변수를 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 . 
	`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 . 
	`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 수있게하십시오 . (☞ 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 . 
	`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 매개 변수를 설정하십시오 .

*) 로봇의 동작시키는 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 .  `from i611_shm import *`

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( )	기반 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( )	월드좌표계 개체 초기화	i611 MCS Coordinate	25
	clear( )	Position 좌표값 초기화	i611 MCS Position	29
	clear( )	Joint 좌표값 초기화	i611 MCS Joint	34
	clear( )	작동 매개 변수 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 (▭ 의 메소드) 는 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

메소드· 함수	기능	모듈 | 클래스	p.
cmd_stop( )	동작 명령 : 감속 정지	rbsys RobSys	103
confdefault( )	작동 매개 변수의 초기값를 설정	i611 MCS MotionParam	43
Coordinate( )	Coordinate 클래스의 생성자	i611 MCS Coordinate	24
copy( )	월드좌표계 개체 복사	i611 MCS Coordinate	25
copy( )	Position 좌표값 복사	i611 MCS Position	29
copy( )	Joint 좌표값 복사	i611 MCS Joint	34
copy( )	작동 매개 변수 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( )	월드좌표계에서 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 매개 변수 확인	Teach data Teachdata	87
get_pos( )	셀의 위치 획득	i611 Ext. Pallet	95

상자 안에 있는 (▭ 의 메소드) 는 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 매개 변수를 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 매개 변수를 설정	i611 MCS MotionParam	45
	motionparam( )	작동 매개 변수를 설정	i611 MCS i611Robot	65
	move( )	PTP 동작을 수행	i611 MCS i611Robot	66
	mp2dict( )	작동 매개 변수를 딕셔너리 형식으로 가져오기	i611 MCS MotionParam	46
	mp2list( )	작동 매개 변수를 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 (▭ 의 메소드) 는 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 매개 변수를 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( )	월드좌표계 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 (▭ 의 메소드) 는 생성자입니다 .

4. 로봇 라이브러리

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

포인트 !

로봇 프로그램은「대문자」 / 「소문자」를 구분합니다 ..

포인트 !

「예외 (Exception)」에 관하여

로봇 프로그램의 작성이 잘못된 경우 등은「예외」가 발생합니다 .
예외가 발생한 경우는 에러 코드를 확인한 후 , 적절한 조치를 취해주십시오 .

포인트 !

프로그램 실행 결과의 확인

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

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)

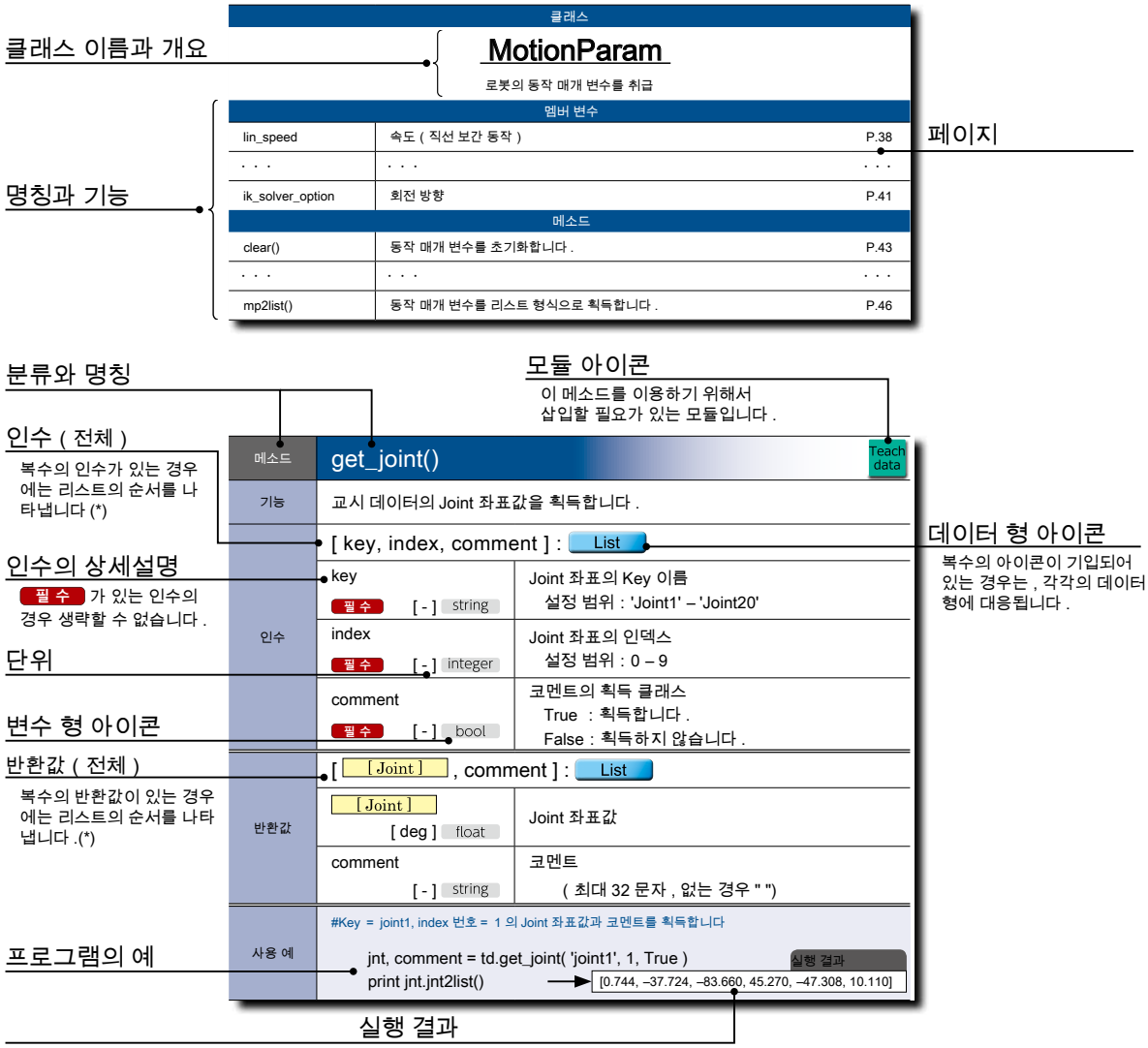
```
# 실행 결과 확인
[0.744, –37.724, –83.660, 45.270, –47.308, 10.110]
```

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

4 로봇 라이브러리

포인트!

로봇 라이브러리를 보는 법



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

필수 아이콘

- 필 수 : 생략 불가능한 인수

데이터 형 아이콘

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

변수 형 아이콘

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

단위

- [mm]
- [deg]
- [s]
- [–] (단위 없음)

데이터 형 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	자신 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 . (자신을 업데이트)	P.27
shift()	월드 좌표계 객체를 이동합니다 . (자신을 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	자신에게 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, ry=2, rx=3, parent=_BASE )
```

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

예 3 : 키워드 (6 개)

```
CO3=Coordinate( x=1, y=2, z=3, rz=1, ry=2, rx=3 )
```

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

예 4 : 키워드 (1 개), 지정하지 않은 값은 초기값이됩니다 .

```
CO4=Coordinate( x=10 )
```

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

메소드	b2g() i611 MCS
기능	Base 좌표계에서 월드 좌표계로 변환합니다 .
인수	[x, y, z, rz, ry, rx] : List (좌표 변환을하기 위해 모든 인수가 필요합니다 .)
	x, y, z 필 수 [mm] float — 단위 (기본 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 (월드 좌표계)
	RZ, RY, RX [deg] float — 자세 (Z-Y-X 계 오일러 각도)

사용 예	# 리스트에서 모든 인수를 지정합니다 .
	CO1=Coordinate()
	CO1.b2g(1, 2, 3, 4, 5, 6) → 실행 결과 [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]

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, 5, 6,_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	# 모든 Coordinate 객체 CO1 복사합니다 .
	#CO1 을 선언합니다 .
	CO1=Coordinate(x=1, y=2, z=3, rz=4, ry=5, rx=6, parent=_BASE)
	
	#CO1 을 새로운 Coordinate 객체 CO1C 에 복사합니다 .
	CO1C=CO1.copy()
	실행 결과
	CO1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >]
	↓
	copy()
	↓새로운 포인트 개체 : CO1C
	CO1C : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >]

메소드	g2b()	i611 MCS
기능	월드 좌표계에서 Baseq 좌표계로 변환합니다 .	
인수	[X, Y, Z, RZ, RY, RX] : List (좌표 변환을하기 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, 5, 6) → 실행 결과 [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]	

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, 5, 6) → [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >]
	CO2 = CO1.inv()	

메소드	replace() i611 MCS
기능	월드 좌표계 객체를 대체합니다 . (자신을 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	자신에게 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, rx=6) → [1.0, 0.0, 0.0, 0.0, 0.0, 6.0, < ... >]

메소드	shift() i611 MCS
기능	월드 좌표계 객체를 이동합니다 . (자신을 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	자신에게 참조 (Coordinate 객체)

사용 예	# 예 1 : 리스트 (2 개)
	CO1 = Coordinate()
	CO1.replace(1, 2, 3, 4, 5, 6) → 실행 결과 [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	CO1.shift(80, 70) → [81.0, 72.0, 3.0, 4.0, 5.0, 6.0]
	# 예 2 : 키워드 (1 개)
	CO2 = Coordinate()
	CO2.replace(1, 2, 3, 4, 5, 6) → [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	CO2.shift(dx=80) → [81.0, 2.0, 3.0, 4.0, 5.0, 6.0]

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 . (자신을 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 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE, –1, 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, < ... >, –1, 0xFF000000] (실행 결과)
	# 예 2 : 키워드
	P2=Position(x=1, y=2, z=3, rz=1, ry=2, rx=3, parent=_BASE, posture=1)
	→ [1.0, 2.0, 3.0, 1.0, 2.0, 3.0, < ... >, 1, 0xFF000000]
	# 예 3 : 키워드 (1 개) 를 지정하지 않은 값은 초기값이 됩니다 .
	P3=Position(rx=103) → [0.0, 0.0, 0.0, 0.0, 0.0, 103.0, < ... >, –1, 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, 5, 6)
	
	#P1 를 초기화합니다 .
	P1.clear()
	
	실행 결과
	P1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –1, 0xFF000000]
	↓
	clear()
	↓
	P1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, –1, 0xFF000000]

posture 매개 변수의 초기값

posture 는 3bit 의 값입니다 . 초기값은 "-1" 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, ry=5, rx=6, 0xFF000000)
	
	#P1 을 새로운 Position 객체 P1C 에 복사합니다 .
	P1C=P1.copy()
	
	실행 결과
	P1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –1, 0xFF000000]
	↓
	copy()
	↓새로운 포인트 객체 : P1C
	P1C : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –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		

프로그램에서 Position 클래스를 크로스오버 카운터 정보가 생략된 매개 변수로 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 (직교좌표계)
	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

```
# 어떤 Position 객체 P1 오프셋 좌표값을 추가합니다 .
    #P1 을 선언합니다 .
    P1=Position( x=1, y=2, z=3, rz=4, ry=5, rx=6 )

    #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, 5.0, 6.0, ... ]
↓
offset()
↓
처음 오프셋 :P1
P1    : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, ... ]

새로운 포인트 객체 : P1ofs
P1ofs : [101.0, 2.0, 3.0, -6.0, 5.0, 6.0, ... ]
```

```
    #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, 5.0, 6.0, ... ]
```

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, 5, 6)
	
	#P1 을 딕셔너리 형식으로 출력합니다 .
	P1.pos2dict() → 실행 결과
	{'parent': < ... >, 'rx': 6.0, 'ry': 5.0, 'rz': 4.0, 'y': 2.0, 'x': 1.0, 'z': 3.0, 'posture': –1, 'multiturn': 0xFF000000}

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

사용 예	# 어떤 Position 객체 P1 을 목록 형식으로 출력합니다 .(*2)
	#P1 을 선언합니다 .
	P1=Position(1, 2, 3, 4, 5, 6)
	
	#P1 을 리스트 형식으로 출력합니다 .
	P1.pos2list() → 실행 결과
	[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –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, 5, 6)
	
	#P1 을 리스트 형식으로 출력합니다 .
	P1.position() → 실행 결과
	[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –1, 0xFF000000]

* 1)i611Robot.use_mt (True) 을 설정하는 경우는 반환 값에 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) (자신을 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 : 리스트에 지정하는 경우
    P1.replace( 1, 2, 3, 4, 5, 6 )

    # 예2 : 2 개의 가변 인수에서 지정하는 장소
    P2.replace( 7, 8 )

    # 예3 : 2 개의 키워드로 지정하는 경우
    P3.replace( x=1, rx=6 )
```

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, 5.0, 6.0, < ... >, –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, 0.0, 0.0, 6.0, < ... >, –1, 0xFF000000]
```

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

사용 예

```
# 모든 Position 객체 P1 을 이동합니다 .(*2)
    #P1 을 선언합니다 .
    P1=Position( 1, 2, 3, 4, 5, 6 )

    #P1 를 옮긴 위치로 업데이트합니다 .
    P1.shift( dx=80 )
```

실행 결과

```
P1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –1, 0xFF000000]
↓
shift()
↓
원래 객체 :P1
P1 : [81.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, –1, 0xFF000000]
```

* 1) i611Robot.use_mt (True) 을 설정하는 경우는 반환 값에 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 . (자신을 업데이트합니다 .)	P.36
shift()	Joint 좌표값을 이동합니다 . (자신을 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 . 초기값 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
반환값	자신에게 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, 1) → [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
	# 예 3 : 키워드 (6 개)
	J3=Joint(j1=1, j2=2, j3=3, j4=4, j5=5, j6=6) → [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	# 예 4 : 키워드 (1 개) 지정하지 않은 값은 초기값이됩니다 .
	J4=Joint(j6 = 6) → [0.0, 0.0, 0.0, 0.0, 0.0, 6.0]

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, 5, 6)
	
	#J1 을 초기화합니다 .
	J1.clear()
	
	실행 결과
	J1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	↓
	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, j5=5, j6=6)
	
	#J1 을 새로운 Joint 개체 J1C 에 복사합니다 .
	J1C=J1.copy()
	
	실행 결과
	J1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	↓
	copy()
	↓새로운 포인트 객체 : J1C
	J1C : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]	

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, 5, 6)
	
	#J1 을 딕셔너리 형식으로 출력합니다 .
	J1.jnt2dict()
	
	실행 결과
	{'j4': 4.0, 'j5': 5.0, 'j6': 6.0, '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, 5, 6)
	
	#J1 을 리스트 형식으로 출력합니다 .
	J1.jnt2list() → 실행 결과 [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]

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	# 어떤 Joint 개체 J1 오프셋 좌표값을 추가합니다 .
	#J1 을 선언합니다 .
	J1=Joint(1, 2, 3, 4, 5, 6)
	
	#J1 에서 오프셋 된 새로운 Joint 개체 J1ofs 를 생성합니다 .
	J1ofs=J1.offset(dj1=80, dj6=–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, 5.0, 6.0]
	↓
	offset()
	↓
	원래 객체 :J1
	J1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
	
	새로운 포인트 객체 : : J1ofs
	J1ofs : [81.0, 2.0, 3.0, 4.0, 5.0, –24.0]
	
	새로운 포인트 객체 : J2ofs
	J2ofs : [81.0, -28.0, 3.0, 4.0, 5.0, 6.0]

메소드	replace() i611 MCS
기능	Joint 좌표값을 대체합니다 . (자신을 업데이트합니다 .)
인수	[Joint] : List Keyword 인수를 생략하면 값이 업데이트되지 않습니다 .
반환값	자신에게 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, 5, 6)
	# 예2 : 2 개의 가변 인수로 지정하는 경우
	J2.replace(7, 8)
	# 예3 : 2 개의 키워드로 지정하는 경우
	J3.replace(j1=1, j6=6)

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, 5.0, 6.0]

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, 0.0, 0.0, 6.0]

메소드	shift() i611 MCS
기능	Joint 좌표값을 이동합니다 (자신을 업데이트합니다 .)
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	자신에게 참조 (Joint 개체)

사용 예	# 임의의 Joint 개체 J1 을 이동합니다 .
	#J1 을 선언합니다 .
	J1.replace(1, 2, 3, 4, 5, 6)
	#J1 을 옮겨 자신을 업데이트합니다 .
	J1.shift(dj1=80)

실행 결과

J1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
↓
shift()
↓
원래 객체 : J1
J1 : [81.0, 2.0, 3.0, 4.0, 5.0, 6.0]

클래스

MotionParam

로봇의 동작 매개 변수를 취급합니다 .

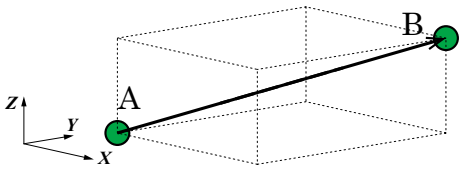
멤버 변수		
lin_speed	속도 (Line 동작 (직선 보간 동작))	P.38
jnt_speed	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작)	P.38
acctime	가속 시간	P.38
dacctime	감속 시간	P.39
posture	자세	P.39
passm	Pass 동작	P.39
overlap	오버랩 거리	P.40
zone	위치 결정 완료 범위	P.40
pose_speed	속도 (자세 보간 동작)(*)	P.41
ik_solver_option	회전 방향	P.41
메소드		
clear()	동작 매개 변수를 초기화합니다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

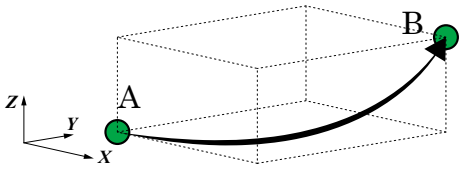
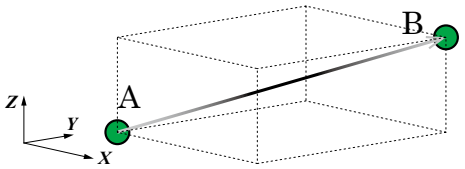
*) 매니퓰레이터 선단이 방향을 바꾸면서 작동할 때 , 선단의 오일러 각도 동작 속도의 상한을 설정합니다 .

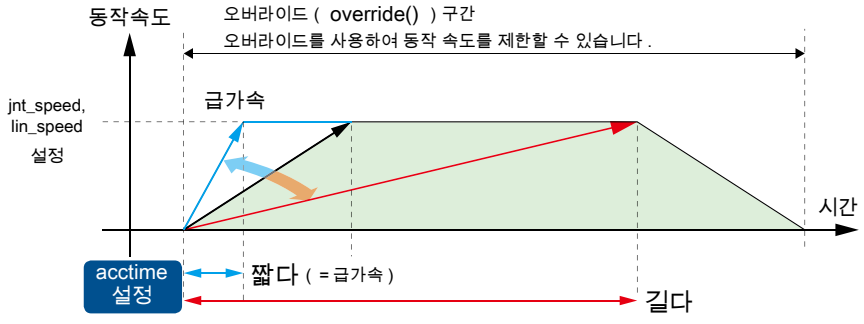
무리한 급가속 / 급감속을 설정하면 로봇이 진동하는 원인이 됩니다 .

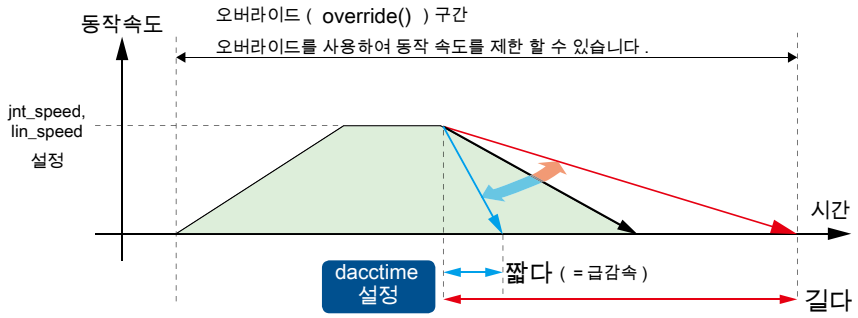
매개 변수의 조정은 처음에는 완만하게 가속 · 감속을 해서 로봇에 진동이 발생하지 않는 것을 확인하면서 해야합니다 .

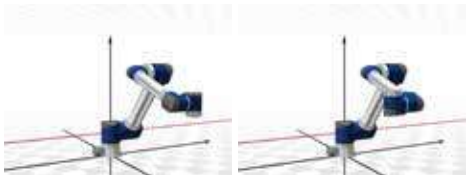
MotionParam 형의 멤버 변수

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

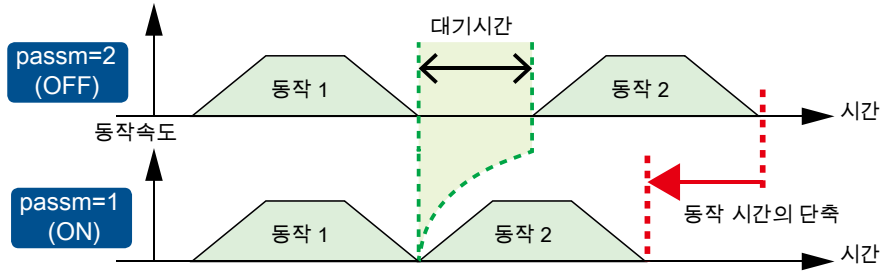
멤버 변수	jnt_speed		
기능	속도 (PTP 동작 , Joint 동작 , 최적 직선 보간 동작)	초기값 단위· 형	5.0 [%] float
보충	모든 관절이 목표 좌표를 향해 등속도 각도로 동작 . 부드러운 곡선을 그리며 이동하는 동작 . 최적 직선 보간 동작도 jnt_speed 에서 속도를 설정합니다 . 변속하기 위해 최고 속도에 대한 % 로 설정합니다 .	PTP 동작 , Joint 동작  최적 직선 보간 동작 	

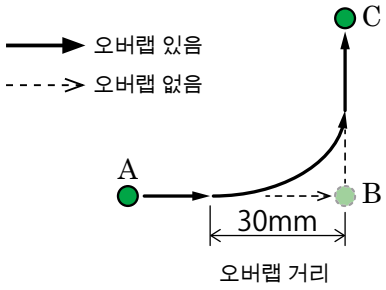
멤버 변수	acctime		
기능	가속 시간	초기값 단위· 형	0.4 [s] float
보충	lin_speed, jnt_speed 에서 설정한 속도에 도달하는 시간을 설정합니다 . 		

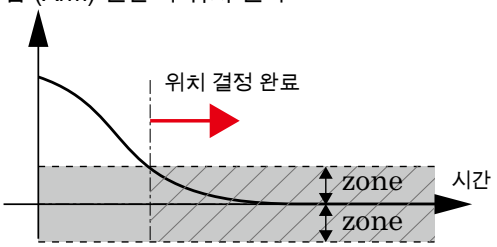
멤버 변수	dacctime		
기능	감속 시간	초기값 단위· 형	0.4 [s] float
보충	lin_speed, jnt_speed 에서 설정한 속도에서 목표 좌표에 감속 정지할 때까지의 시간 .  		

멤버 변수	posture		
기능	자세	초기값 단위· 형	2 [-] integer
보충	설정 범위 : 0 – 7 매니퓰레이터의 자세는 ①암 (Arm) 의 단위 ②관절 각도 로 정의됩니다 .  예 : 초기값「 2 」		

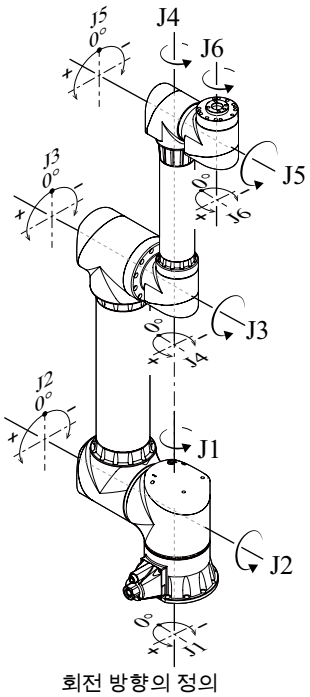
자세에 대한 자세한 내용은 교시편「 5 좌표계와 자세 」를 참조하십시오 .

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

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 판정을하는 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% = 45deg/s 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 . 0 x 1 1 1 1 1 1 1 1 (Rsv.) J6 J5 J4 J3 J2 J1 J1 - J6 설정 0 : 지름길 multiturn 매개 변수의 정보를 사용하지 않는 회전입니다 . 1 : multiturn 매개 변수의 정보를 사용합니다 . 2 : + 방향으로 회전합니다 . 3 : - 방향으로 회전합니다 . + 방향 , - 방향은 오른쪽 그림을 참조		 회전 방향의 정의

생성자	MotionParam() i611 MCS
기능	로봇의 동작 매개 변수 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, 2, 2, 0.0, 100, 20, 0x11111111]
반환값	자신에게 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 매개 변수
	m : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	
	# 예 2 : 인수에 지정된 동작 매개 변수를 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 매개 변수
	m : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]

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

사용 예	# 모든 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 매개 변수의 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

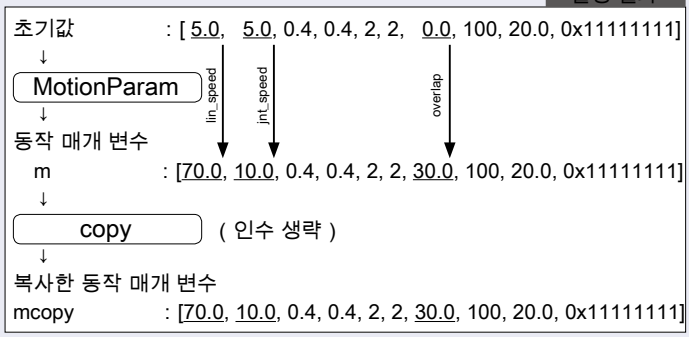
동작 매개 변수의 초기값이 설정되는 타이밍

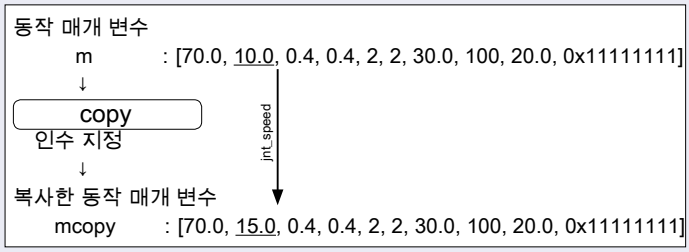
confdefault () 메소드에서 설정을 변경한 초기값은 그 다음에 MotionParam 클래스의 인스턴스를 생성 , 복사 , 삭제될 때 반영됩니다 .

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

사용 예	#MotionParam 에서 동작 매개 변수를 미리 설정해야합니다 .
	m=MotionParam(jnt_speed=10, lin_speed=70, overlap=30)
	
	# 예 1 : 인수를 생략합니다 . (현재 설정되어있는 동작 매개 변수)
	mcopy = m.copy()
	
	# 예 2 : 인수에 지정된 동작 매개 변수를 변경하여 복사합니다 .
	mcopy = m.copy(jnt_speed=15)

실행 결과





메소드	motionparam() i611 MCS
기능	동작 매개 변수를 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	자신에게 참조 (MotionParam 개체)

사용 예

#MotionParam 에서 동작 매개 변수를 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 매개 변수를 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 매개 변수를 딕셔너리 형식으로 획득합니다 .
인수	없음
반환값	[MotionParam] : Dict.

사용 예	# 예 : MotionParam 에서 동작 매개 변수를 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': 2}
	# 직접 값을 얻습니다 .
	lin_speed = m.mp2dict()['lin_speed'] → lin_speed=70.0

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

사용 예	# 예 : MotionParam 에서 동작 매개 변수를 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

클래스

i611Robot

로봇의 동작을 처리합니다 .

멤버 변수	
-	—

메소드		
abort()	로봇 프로그램을 중단합니다 .	P.50
adjust_mt()	Position 형 좌표값을 문자열로 변환할 때의 CC 값을 보정합니다 .	P.50
asyncm()	로봇 프로그램의 예측 동작 구간을 설정합니다 .	P.51
cause_user_error()	사용자 정의 에러를 발생시킵니다 .	P.52
changetool()	Tool 오프셋을 선택합니다 .	P.52
check_ready()	로봇이 자동 운전할 수 있는지를 확인합니다 .	P.53
close()	로봇과의 연결을 종료합니다 .	P.54
disable_mdo()	MDO 동작을 비활성화합니다 .	P.54
enable_interrupt()	감속 정지와 비상 정지의 예외 발생을 설정합니다 .	P.55
enable_mdo()	MDO 동작을 활성화합니다 .	P.56
exit()	로봇 프로그램을 강제종료합니다 .	P.57
get_hw_info()	모델명과 시리얼 번호를 얻습니다 .	P.57
get_system_port()	시스템 포트의 상태를 얻습니다 .	P.58
get_system_status()	시스템상태와 에러 ID 를 얻습니다 .	P.59
getjnt()	매니퓰레이터의 현재 위치를 Joint 형으로 얻습니다 .	P.59
getmotionparam()	현재 동작 매개 변수를 얻습니다 .	P.60
getpos()	매니퓰레이터의 현재 위치를 Position 형으로 얻습니다 .	P.60
home()	모든 축을 Joint 좌표 0 deg 으로 이동합니다 .	P.60
is_open()	i611Robot 의 오픈 상태를 확인합니다 .	P.61
is_pause()	로봇 프로그램의 일시 정지 중인 상태를 확인합니다 .	P.61
join()	예측된 로봇 프로그램의 동작 완료를 기다립니다 .	P.63
Joint2Position()	Joint 좌표값에서 Position 좌표값으로 변환합니다 .	P.63
line()	직선 보간 동작을 실행합니다 .	P.64
MCS_version()	로봇 라이브러리 버전을 얻습니다 .	P.65
motionparam()	동작 매개 변수를 설정합니다 .	P.65
move()	PTP 동작을 실행합니다 .	P.66
open()	로봇과의 연결을 시작합니다 . (초기화)	P.67
optline()	직선 보간 동작을 최적의 속도로 변속하면서 실시합니다 .	P.68
override()	오버라이드를 실행합니다 .	P.69
pause()	로봇 동작을 일시 정지합니다 .	P.69
Position2Joint()	Position 좌표값에서 Joint 좌표값으로 변환합니다 .	P.70

(이어서 i611Robot 메소드)

메소드		
release_stopevent()	발생 중인 예외 이벤트를 재설정합니다 .	P.70
reljntmove()	Joint 좌표계에서 상대 동작을 합니다 .	P.71
relline()	직교좌표계에서 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 처리를 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 .

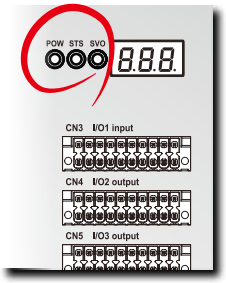
사용 예	# 예 1 : 인수를 생략합니다 .(초기값을 설정합니다 .)
	rb=i611Robot()
	
	# 예 2 : 리스트
	rb=i611Robot('127.0.0.1', 12345)
	
	# 예 3 : 키워드 인수 (모두 지정)
	rb= i611Robot(host='127.1.1.1', port=10000)
	
	# 예 4 : 키워드 인수 (host 만 지정)
	rb= i611Robot(host='127.1.1.1')
	
	# 예 5 : 키워드 인수 (port 만 지정)
	rb= i611Robot(port=3000)

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	rz 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 오프셋 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, 200.0, 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	

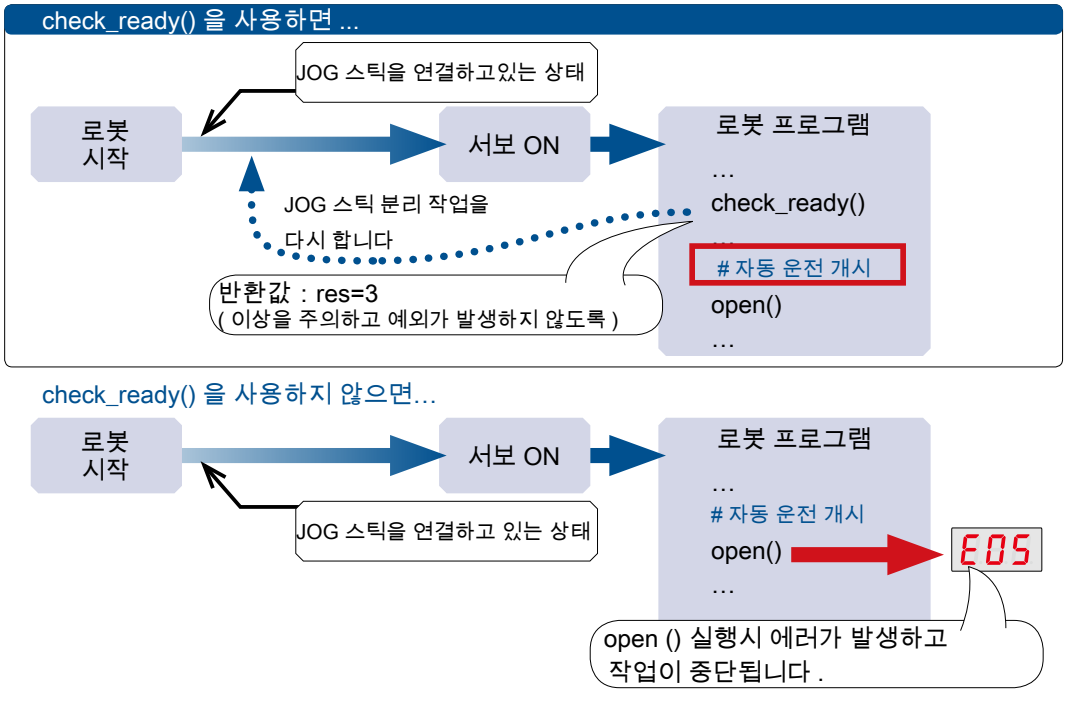
포인트!

check_ready() 메소드의 사용법

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

반환 값이 '0' 이 아닌 경우 로봇은 동작하지 않습니다 .
i611Robot 클래스의 생성자 또는 open () 실행시 예외가 발생합니다 .
이 메소드에서 상태를 확인하면 예외 발생을 방지 할 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 .

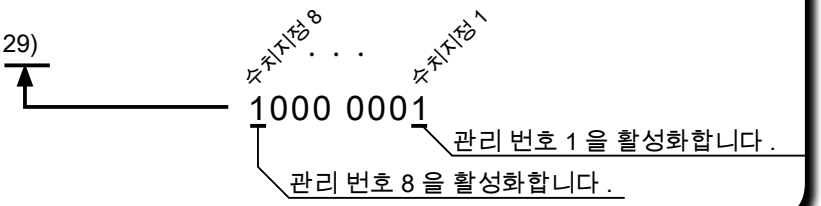
포인트 !

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)
```



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 = i611Robt.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 中 0 : 서보 OFF 中
	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 中 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_por()[0]     # 로봇 프로그램 상태
    svon = rb.get_system_por()[1]        # 서보 상태
    emo = rb.get_system_por()[2]         # 비상 정지 상태
    hw_error = rb.get_system_por()[3]    # 시스템 정의 에러 ( 치명적 ) 상태
    sw_error = rb.get_system_por()[4]    # 시스템 정의 에러 상태
    abs_lost = rb.get_system_por()[5]    # ABS 소실 상태
    in_pause = rb.get_system_por()[6]    # 일시 정지 상태
    error = rb.get_system_por()[7]       # 시스템 에러 상태
```

메소드	get_system_status() i611 MCS	
기능	시스템 상태와 에러 ID 의 획득	
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 매개 변수를 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 현재위치를 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 처리의 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 매개 변수를 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 : [125.0, -717.5, 434.70181275976404, 3.508354649267438e-15, 4.296495291499103e-31, 180.0, < ... >, 7]

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 매개 변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 , 이후의 동작은 변경된 동작 매개 변수로 실행 됩니다 .
반환값	성공한 경우 : True [-] bool 실패한 경우 : 예외가 발생합니다 .

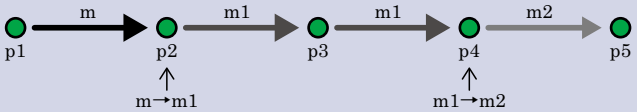
사용 예	# 예 1
	# 위치 좌표 p10 으로 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 으로 주어진 조건에 따릅니다 .
	rb.line(p10)
	
	# 예 2
	# 위치 좌표 p10 으로 향한 뒤 , p20 으로 직선 보간 운동을 합니다 .
	# 동작 조건은 , MotionParam 의 설정에 따릅니다 .
	rb.line(p10, p20)
	
	# 예 3
	# 동작 조건은 MotionParam 을 변경하여 , 위치 좌표 p10 으로 이동합니다 .
	# 이후 p20 으로 직선 보간 운동을 합니다 .
	mt=m.MotionParam(posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0)
	rb.line(mt, p10, p20)

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



메소드	MCS_version()	i611 MCS
기능	로봇 라이브러리의 버전을 획득합니다 .	
인수	없음	
반환값	[major, mainor, patch, build] : List	
	major [-] integer	Major Version
	minor [-] integer	Minor Version
	patch [-] integer	Patch Version
	build [-] integer	Build Version
사용 예	rb.MCS_version() → 실행 결과 [0, 3, 2, 4]	

메소드	motionparam()	i611 MCS
기능	동작 매개 변수를 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=1, 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 매개 변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 이후의 동작은 변경된 동작 매개 변수로 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
	# 동작 조건을 MotionParam 으로 변경하여 , 위치 좌표 p10 으로 이동 ,
	# 이후 p20 으로 PTP 이동을 합니다 .
	mt=m.MotionParam(posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0)
	rb.move(mt, p10, p20)
	
	# 예 4
	# 크로스오버 카운터 정보를 사용
	rb.use_mt(True)
	…
	rb.move(p10)
	…
	rb.close()

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 않은경우 크로스오버 카운터의 정보를 이용하지않는 동작입니다 .
크로스오버 카운터에 대한자세한 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)

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 실행횟수는 프로그램 중 1 회 뿐입니다 . 한번 close() 를 실행한 후 , 2 번째 open() 을 실행할 경우 예외가 발생합니다 .
루프문을 실행할 경우 , open() 은 루프 문의 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 매개 변수를 인수로 가집니다 . MotionParam 형을 인수로 가질 경우 , 이후의 동작은 변경된 동작 매개 변수로 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)
	
	# 예 3
	# 동작 조건을 MotionParam 으로 변경하여 , 위치 좌표 p10 으로 이동한 뒤 ,
	# p20 으로 최적 직선 보간 운동을 합니다 .
	mt=m.MotionParam(posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0)
	rb.optline(mt, 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 정지 합니다 .
            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(–50, –250, 350, 90, 0, 180)
	#Joint 형 좌표값으로 변경 (p10 → 변경 → j10)
	j10=rb.Position2Joint(p10)
	실행 결과
	p10 : [–50, –250, 350, 90, 0, 180]
	↓Position2Joint()
	j10 : [18.049733949948962, -21.510874537939305, 147.44314399114182, -180.0, 125.9322694532025, 161.95026605005106]

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
	dg4 [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, 90, 0 )
# 동작 매개 변수를 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 오프셋 양
	dy [mm] float	Y 축 방향으로 오프셋 양
	dz [mm] float	Z 축 방향으로 오프셋 양
	drz [deg] float	Rz 축을 중심으로 오프셋 양
	drz [deg] float	Ry 축을 중심으로 오프셋 양
	drx [deg] float	Rx 축을 중심으로 오프셋 양
반환값	없음	

사용 예

```
#Position 형의 좌표값을 준비합니다 .(*)
    P10 = Position( 95, –280, 240, 154, 80, –114 )
# 동작 매개 변수를 설정합니다 .
    m=MotionParam( jnt_speed=10, lin_speed=70, overlap=30 )
    rb.motionparam( m )
    ...

    rb.move(P10)
# 직교 좌표계에서 X 축방향으로 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() 만으로 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 정지 합니다 .

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 !

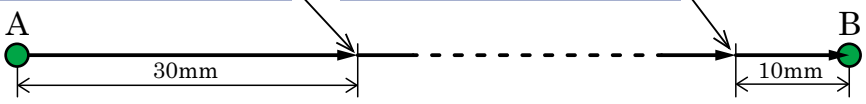
MDO 동작

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 오프셋 양
	offy [mm] float	Tool 좌표계에서 Y 축의 Tool 오프셋 양
	offz [mm] float	Tool 좌표계에서 Z 축의 Tool 오프셋 양
	offrz [deg] float	Tool 좌표계에서 Rz 축을 중심으로 회전하는 Tool 오프셋 양
	offry [deg] float	Tool 좌표계에서 Ry 축을 중심으로 회전하는 Tool 오프셋 양
	offrx [deg] float	Tool 좌표계에서 Rx 축을 중심으로 회전하는 Tool 오프셋 양
	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, 200.0, 0.0, 0.0, 0.0)
	# 예 2 : 키워드
	rb.settool(id=1, offx=0.0, offy=0.0, offz=200.0, 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 사용하는 것으로 sleep 중에도 로봇의 로봇 관계의 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
정지 시킬 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, 240, 154, 80, –114)
	
	# 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)	

포인트!

크로스오버 카운터에 관한 메소드

크로스오버 카운터를 활성화시킬 경우 하기의 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	

포인트!

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 이외의 일시 정지를 금지한다 .」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

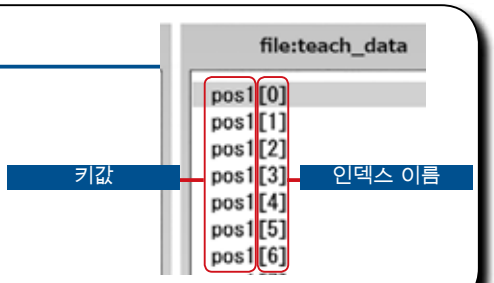
생성자	Teachdata()	Teach data
기능	교시 데이터를 로드하여 Teachdata 클래스의 인스턴스를 작성합니다 .	
인수	fname [-] string	교시 데이터의 파일 이름 초기값 : '/home/i611usr/teach_data'
반환값	자신에게 참조 (Teachdata 클래스 객체)	

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 단독 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()

· 데이터를 업데이트하는 경우 close () 할 때도 내부에서 실행하고 있습니다 .
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	# 키 = 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, –47.308, 10.110]	

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	# 키값 : "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 연결 없음)
	baseid [-] integer Base ID 반환값 : 0 – 3 (0 은 연결 없음)
	comment [-] string Comment (최대 32 문자 , 없는 경우 ' ')
사용 예	# 키 = 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, 459.94, 120.61, 53.890, 4.720, –142.88, < ... >, 6]
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 자세 (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 미지의 버전 (R1.0.0 이상) 의 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 와 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, 84, –28)
	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()	보정하는 셀의 위치를 보정합니다 .	P.94
get_pos()	셀의 위치 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(–250, –250, 400)
	pos_1=Position(–170, –250, 400)
	pos_2=Position(–250, –180, 400)
	pos_3=Position(–170, –180, 400)
	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, 400 )
    pos_1=Position( –170, –250, 400 )
    pos_2=Position( –250, –180, 400 )
    pos_3=Position( –170, –180, 400 )
    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 !

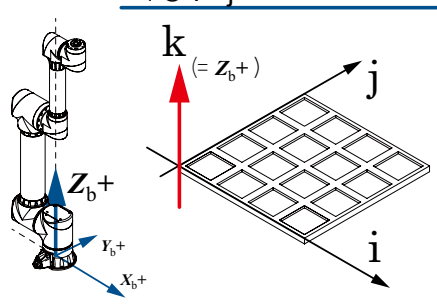
팔레트의 수직 방향

교시 포인트의 배치에 의해 팔레트 수직 방향 (k 방향) 의 양의 방향이 바뀝니다 .

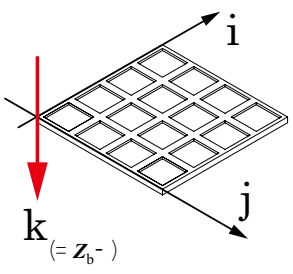
예① $\vec{i} \times \vec{j}$: 평면 (팔레트면) 에 수직인 벡터 z + 입니다 .

예② $\vec{j} \times \vec{i}$: 예①과 반대 방향의 z - 입니다 .

예 ① $\vec{i} \times \vec{j}$



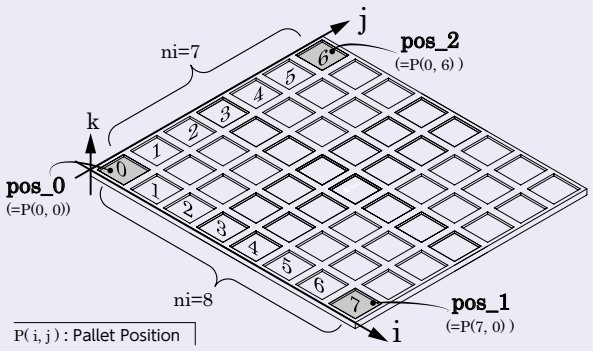
예 ② $\vec{j} \times \vec{i}$



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, 400 )
    pos_1=Position( –170, –250, 400 )
    pos_2=Position( –250, –180, 400 )

#3 점 교시 데이터를 사용한 팔레트를 정의합니다 .
    pal.init_3( pos_0, pos_1, pos_2, 8, 7 )

    rb.close()
```

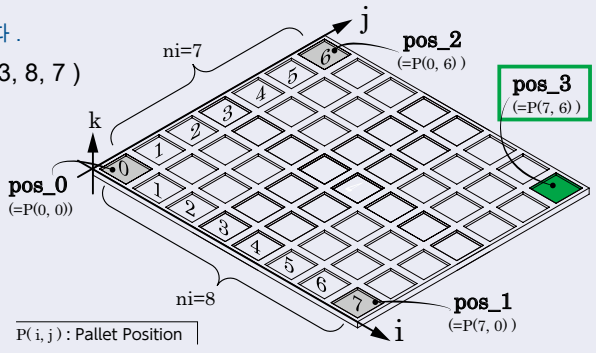


*) 실제 동작은 교시한 포인트를 팔레트 각각의 좌표 정의에 사용하십시오 .

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, 400 )
    pos_1=Position( –170, –250, 400 )
    pos_2=Position( –250, –180, 400 )
    pos_3=Position( –170, –180, 400 )

#4 점 교시 데이터를 사용한 팔레트를 정의합니다 .
    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	일시정지 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	비상정지 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	일시정지 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 : 일시정지	
인수	없음	
반환값	res0 [-] bool	True : 성공 False : 실패
사용 예	rbs.cmd_pause()	

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()	

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 : 비상정지 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')	

set_robtask() 메소드로는 로봇 프로그램을 등록만 가능합니다 . 로봇 프로그램 기동에는 사용할 수 없습니다 .

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
일시정지	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
비상정지 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
일시정지 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	비상정지 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  모듈의 임포트 ######################
    from i611_MCS import *
## 2 . 초기 설정 ② : 동작 조건 설정 ################################
    # i611 로봇 Constructor
        rb = i611Robot()
    # 월드 좌표계의 정의
        _BASE = Base()
    # 로봇과 접속 개시 , 초기화
        rb.open( True )
    # 동작 중 비상정지 입력에 의한 예외 발생 활성화
        rb.enable_interrupt( 1, True )
...
# 예외 처리 ----------------------------------------------------------------------------------

# 예 1 : Exception 검출하여 정상 종료
        try:
        …
        except Robot_emo:   # 비상정지 시 발생하는 예외
            # 필요한 에러처리 ( 종료처리 ) 기술

# 예 2 : Exception 검출하여 에러 발생 종료
        try:
        …
        except Robot_emo:   # 비상정지 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 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 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 모듈의 임포트 ####################
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##############################
	# i611 로봇 Constructor
	rb = i611Robot()
	# 월드 좌표계의 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 에러처리 (종료처리) 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 모듈의 임포트 ####################
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##############################
	# i611 로봇 Constructor
	rb = i611Robot()
	# 월드 좌표계의 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 에러처리 (종료처리) 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 모듈의 임포트 ##########
	from i611_MCS import *
	
	## 2 . 초기 설정② : 동작 조건 설정 ##########
	# i611 로봇 Constructor
	rb = i611Robot()
	# 월드 좌표계의 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 에러처리 (종료처리) 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  모듈의 임포트 ######################
    from i611_MCS import *

## 2 . 초기 설정② : 동작 조건 설정 ################################
    # i611 로봇 Constructor
        rb = i611Robot()
    # 월드 좌표계의 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 에러처리 ( 종료처리 ) 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 (종료)
반환값	[*port] : List	
	*port [-] string	지정된 입력 포트의 실행 결과를 '0' 또는 '1' 로 돌려줍니다 입력 포트를 여러 개 지정하는 경우 , 목록에서 돌아갑니다 .
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 변경하지 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 포트의 어드레스 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 포트의 어드레스 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
	dout(16, '11111')

함수	IOinit() i611 IO
기능	I/O 초기화 (*)
인수	[IPaddress, port] : List
	IPaddress [-] string — IP 어드레스 / 초기값 : '127.0.0.1'
	port [-] integer — 포트 번호 / 초기값 : 12345
	인수를 생략하면 기본값으로 설정됩니다
반환값	없음

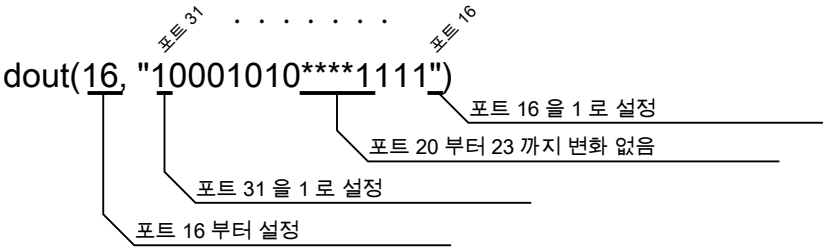
사용 예	# 예 1 : 인수 생략하는 경우 (초기값으로 설정됨)
	IOinit()
	
	# 예 2 : 리스트 (전체)
	IOinit('127.1.1.1', 12345)

*)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
	if wait(8, '1', 10)[0] == 1:
	if wait(9, '1', 10)[1] == '1':
	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 기호를 지정합니다 . 이를 통해 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 – 0x2800 「 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 (*)	256 개
0x1C00 – 0x23F8	float	256 개

(*) 4 바이트

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 모두가 업데이트되기 때문에 새로운 정보가 저장됩니다 .

포인트 !

메모리 내용 저장

메모리 내용은 파일에 저장하는 것을 권장합니다 .
저장하는 경우 작성한 후 flush () 또는 close () 를 호출합니다 .
(flush () : 파일 버퍼의 강제 플래시)

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 포트를 할당 (running) 또는 -1	업데이트 시	R
+0x084E	assign_out(svon)	2	short	assign_port[5]	출력 포트를 할당 (svon) 또는 -1	업데이트 시	R
+0x0850	assign_out(emo)	2	short	assign_port[6]	출력 포트를 할당 (emo) 또는 -1	업데이트 시	R
+0x0852	assign_out(hw_error)	2	short	assign_port[7]	출력 포트를 할당 (hw_error) 또는 -1	업데이트 시	R
+0x0854	assign_out(sw_error)	2	short	assign_port[8]	출력 포트를 할당 (sw_error) 또는 -1	업데이트 시	R
+0x0856	assign_out(abs_lost)	2	short	assign_port[9]	출력 포트를 할당 (abs_lost) 또는 -1	업데이트 시	R
+0x0858	assign_out(in_pause)	2	short	assign_port[10]	출력 포트를 할당 (in_pause) 또는 -1	업데이트 시	R
+0x085A	assign_out(error)	2	short	assign_port[11]	출력 포트를 할당 (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)	–	–	–	–	–	–

조작의 정보 (rbcfg)

오프셋	데이터	바이트	데이터 형	변수명	내용	업데이트주기	사용자 엑세스
+0x2C00	manip_type	36	char (string)	manip_type	조작 정보	업데이트 없음	R
+0x2C24	manip_serial	36	char (string)	manip_serial	조작 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	지령 값 (직교 좌표) X[mm]	8	double	cmdx	현재 지령 값	1ms	R
+0x3008	지령 값 (직교 좌표) Y[mm]	8	double	cmdy	현재 지령 값	1ms	R
+0x3010	지령 값 (직교 좌표) Z[mm]	8	double	cmdz	현재 지령 값	1ms	R
+0x3018	지령 값 (직교 좌표) Rz[deg]	8	double	cmdrz	현재 지령 값	1ms	R
+0x3020	지령 값 (직교 좌표) Ry[deg]	8	double	cmdry	현재 지령 값	1ms	R
+0x3028	지령 값 (직교 좌표) Rx[deg]	8	double	cmdrx	현재 지령 값	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	지령 값 (Joint) J1[deg]	8	double	joint[0]	현재 지령 값	1ms	R
+0x3058	지령 값 (Joint) J2[deg]	8	double	joint[1]	현재 지령 값	1ms	R
+0x3060	지령 값 (Joint) J3[deg]	8	double	joint[2]	현재 지령 값	1ms	R
+0x3068	지령 값 (Joint) J4[deg]	8	double	joint[3]	현재 지령 값	1ms	R
+0x3070	지령 값 (Joint) J5[deg]	8	double	joint[4]	현재 지령 값	1ms	R
+0x3078	지령 값 (Joint) J6[deg]	8	double	joint[5]	현재 지령 값	1ms	R
+0x3080	(예약)	–	–	–	–	–	–
+0x3090	속도	8	double	velocity	현재 지령 값	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 상태 (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 .

포인트!

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 운전시에 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 전송 한 로봇 프로그램을 실행합니다 .
주로 테스트 및 디버깅에 사용합니다 .

(연결 방법은 취급설명서「C 교시」을 참조하십시오)

주 의

	init.py 에는 전원 투입과 동시에 자동 운전을 시작하는 로봇 프로그램을 작성하지 마십시오 .	
	로봇을 손상시킬 수있는 기계와 로봇을 조합하거나 공유할 경우 , 다른 기계의 작업 공간 외부에서 모든 기능과 동작 프로그램을 개별적으로 시험 할 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	userporg_out.log
오류 출력		userporg_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 작업 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)