Agilebot Robot SDK
Agilebot 로봇 SDK 매뉴얼

Python SDK

Python 기반의 고성능 로봇 제어 개발 키트로, 간결한 API 설계를 통해 지능형 로봇 애플리케이션을 빠르게 구축할 수 있습니다.

자세히 알아보기 →

C# SDK

.NET 생태계를 위한 엔터프라이즈급 로봇 제어 솔루션으로, 타입 안전한 강타입 API를 제공해 산업 자동화 시스템에 쉽게 통합할 수 있습니다.

자세히 알아보기 →

Python SDK

프롤로그

버전 기록

문서 버전	SDK 버전	출시일
V2.0	1.7.1.3	2025.07.07
V3.0	2.0.1.0	2026.01.26

Update Notes

로봇 버전 호환성

SDK는 Agilebot Scara, Puma 및 협동로봇 시리즈를 지원합니다. 로봇 소프트웨어가 설치된 장치에서 사용해야 합니다. 일부 함수는 버전에 따라 다른 결과를 반환할 수 있습니다. 자세한 내용은 4장을 참조하세요. SDK가 로봇팔에 연결되면 모션 제어 소프트웨어 버전을 확인합니다. 버전이 최소 요구 사항보다 낮으면 연결이 실패합니다. 권장 버전보다 낮을 경우 낮은 버전 메시지가 나타납니다. 로봇 소프트웨어 버전을 즉시 업데이트하십시오. 일부 SDK 인터페이스는 특정 컨트롤러 버전만 지원합니다. 각 인터페이스의 호환성을 확인하세요.

SDK 버전	로봇 소프트웨어 버전	지원현황
v1.7.0.X	Copper v7.6.X.X, Bronze v7.6.X.X	지원됨
v1.7.1.X	Copper v7.6.X.X, Bronze v7.6.X.X	지원됨
v2.0.1.X	Copper v7.7.X.X、Bronze v7.7.X.X	지원됨

1 소개 및 배포

1.1 로봇 시스템

SDK는 고객의 호스트 컴퓨터에 설치됩니다. 호스트 컴퓨터가 로봇을 작동하고 감지할 수 있는 인터페이스를 제공합니다. 이 시스템은 호스트 컴퓨터, 컨트롤러, 로봇 본체로 구성됩니다. 로봇 본체는 모터, 감속기, 엔코더 및 전송 메커니즘으로 구성된 기계 본체를 말합니다. 제어 캐비닛의 조작 패널에는 상태 표시 등, 제어 캐비닛 스위치, 통신 인터페이스 등이 장착되어 있습니다.

1.2 환경 요구사항

하드웨어 및 운영 체제:

- Windows 10 이상
  - x86_64 아키텍처
- Linux (Ubuntu 18.04 이상 권장)
  - x86_64 아키텍처
  - 팔 아키텍처

Python 버전:

- 3.8 이상

1.3 설치

환경 설정

- Python 환경을 다운로드합니다. Conda 사용을 권장합니다. 현재 시스템에 맞는 최신 Miniforge 또는 Miniconda를 설치하세요.
- Conda 다운로드 링크: Miniforge3
- 설치 후 Miniforge Prompt를 실행하고 다음 명령을 입력하여 conda를 초기화합니다.

```shell
conda init
```

- 설치 후 Miniforge Prompt 또는 시스템 터미널을 실행하고 다음 명령을 입력하여 새 Python 환경을 생성합니다.

```shell
conda create -n agilesdk python=3.10
```

- 다음 명령을 입력하여 방금 생성한 환경으로 진입합니다.

```shell
conda activate agilesdk
```

  - Conda 환경 설정이 불편하다면 `venv` 를 사용해 가상 환경을 생성하고 활성화하세요.

```shell
# 생성
python3.10 -m venv agilesdk
# Windows에서 환경 활성화
.\agilesdk\Scripts\activate
# Linux에서 환경 활성화
source agilesdk/bin/activate
```

- `cd` 명령으로 압축을 푼 디렉터리로 이동하고, 다음 명령으로 필요한 Python 패키지를 설치합니다.

```shell
cd 압축_해제_디렉터리
pip install -r requirements.txt
```

- pip 설치가 실패하면 `requirements.txt` 를 열고 각 패키지를 conda로 설치할 수 있습니다.

```shell
conda install package_name=<version>
```

IDE 설치

- 개발 환경으로는 PyCharm을 권장합니다.
- PyCharm 다운로드 링크: PyCharm
- PyCharm Community Edition을 다운로드하고 설치한 뒤 실행합니다.
- 새 Python 프로젝트를 만들고, 프로젝트 유형으로 "Pure Python"를 선택합니다.
- 인터프리터 유형으로 "Base Conda"를 선택하고, 경로를 앞서 생성한 `agilesdk` 환경으로 지정합니다.
- "생성"을 클릭하면 코드 작성을 시작할 수 있습니다.

SDK 설치 및 테스트

- 다음 명령으로 Python 버전의 로봇 SDK를 설치합니다. `x.x.x` 는 현재 SDK 버전 번호로 교체하세요.

```shell
pip install Agilebot.SDK.A-x.x.x-py3-none-any.whl
```

- `example` 폴더로 이동하여 다음 명령으로 예제를 실행합니다.

```shell
cd example
python 예제_프로그램_이름 (예: python arm/get_version.py)
```

- 예제를 실행할 때 호스트 컴퓨터는 반드시 로봇 네트워크에 연결되어 있어야 합니다.

2 어휘

용어	설명
펜던트를 가르치다	로봇에 부착되어 로봇을 가르치고 제어하는 데 사용되는 펜던트입니다.
SDK	로봇 프로그래밍 및 제어에 사용되는 소프트웨어 개발 키트
로봇 네트워크	로봇과 외부 컴퓨터 간의 네트워크 연결
제어 장치	동작 명령 실행, 센서 데이터 처리, 로봇 상태 관리를 담당하는 로봇의 제어 장치입니다.
로봇 팔	다수의 관절과 링크로 구성된 로봇의 주요 움직이는 부분
서보 시스템	로봇 관절의 움직임을 제어하는 모터 구동 시스템으로 정밀한 위치 및 속도 제어가 가능합니다.
가르침	로봇이나 티치펜던트의 수동 조작을 통해 로봇의 운동 궤적과 동작을 기록하는 과정
관절	로봇 팔의 다양한 링크를 연결하는 가동 부품으로, 각 관절은 1자유도에 해당합니다.
데카르트 좌표	공간에서 로봇의 위치와 방향을 설명하는 데 사용되는 서로 수직인 3개의 X, Y, Z 축을 기반으로 하는 3차원 좌표계
포즈	위치 좌표 및 회전 각도를 포함하여 공간에서 로봇의 위치와 방향의 조합
궤도	공간에서 이동하는 로봇 엔드 이펙터의 경로는 일반적으로 일련의 포즈 포인트로 구성됩니다.
유효 탑재량	로봇의 엔드 이펙터가 운반하는 무게와 물체는 로봇의 모션 성능과 정확성에 영향을 미칩니다.
좌표계	기본 좌표계, 도구 좌표계, 사용자 좌표계 등을 포함하여 로봇 위치 및 방향을 설명하는 데 사용되는 참조 시스템입니다.
OVC	전체 속도 제어(Overall Velocity Control)는 로봇의 전체 동작 속도 배율을 설정하는 데 사용됩니다.
OAC	로봇의 전체 가속 배율을 설정하는 데 사용되는 전체 가속도 제어
TF	로봇의 엔드툴을 원점으로 하는 좌표계인 Tool Frame
UF	편리한 프로그래밍 및 위치 지정을 위한 사용자 정의 좌표계인 사용자 프레임

용어	설명
TCS	티칭 좌표계(Teach Coordinate System)는 티칭 시 사용되는 좌표계입니다.
DH 매개변수	Denavit-Hartenberg 매개변수, 로봇 링크의 기하학적 관계를 설명하는 데 사용되는 표준 매개변수
PR 레지스터	포즈 레지스터(Pose Register)는 로봇 포즈 정보를 저장하는 데 사용되는 레지스터입니다.
MR 레지스터	모션 레지스터(Motion Register)는 모션 관련 매개변수를 저장하는 데 사용되는 레지스터입니다.
SR 레지스터	문자열 레지스터, 문자열 정보를 저장하는 데 사용되는 레지스터
R 레지스터	숫자 정보를 저장하는 데 사용되는 레지스터인 Real Register
MH 레지스터	Modbus Holding Register, Modbus 통신을 위한 Holding Register
MI 레지스터	Modbus 입력 레지스터, Modbus 통신을 위한 입력 레지스터
BAS	로봇 제어 프로그램을 작성하는 데 사용되는 고급 프로그래밍 언어인 기본 스크립트
스카라	선택적 컴플라이언스 조립 로봇 암은 4축 산업용 로봇의 일종입니다.
협동로봇	인간과 안전하게 협업할 수 있는 로봇으로, 일반적으로 힘 감지 및 충돌 감지 기능을 갖추고 있습니다.
산업용 로봇	산업 자동화 생산에 사용되는 로봇으로 일반적으로 고정밀, 고속, 고부하 능력을 갖추고 있습니다.
Copper	Agilebot의 협동로봇 제품 라인의 코드명
Bronze	Agilebot 산업용 로봇 제품 라인의 코드명

3 데이터 구조

3.1 StatusCodeEnum

설명

인터페이스 반환 상태 코드.

참고: 여기에 나열되지 않은 반환 코드가 나타나면 로봇 오류 코드 표를 참조하거나 값을 문자 그대로 해석하십시오.

가져오기

```python
from Agilebot import StatusCodeEnum
```

필드

Name	Enum Value	Description
OK	0	성공
INCOMPATIBLE_VERSION	-1	호환되지 않는 버전
CONNECTION_TIMEOUT	-3	연결 시간 초과
INTERFACE_NOTIMPLEMENTED	-4	현재 컨트롤러에 인터페이스가 구현되지 않았습니다.
INDEX_OUT_OF_RANGE	-5	범위를 벗어난 인덱스
UNSUPPORTED_FILETYPE	-6	지원되지 않는 파일 형식
UNSUPPORTED_PARAMETER	-7	지원되지 않는 로봇 매개변수
UNSUPPORTED_SIGNAL_TYPE	-8	지원되지 않는 IO 신호 유형
PROGRAM_NOT_FOUND	-9	프로그램을 찾을 수 없습니다

Name	Enum Value	Description
PROGRAM_POSE_NOT_FOUND	-10	프로그램 포즈를 찾을 수 없습니다
WRITE_PROGRAM_POSE_FAILED	-11	프로그램 포즈 작성 실패
GET_ALARM_CODE_FAILED	-12	알람 코드를 검색하지 못했습니다.
WRONG_POSITION_INFO	-13	컨트롤러의 잘못된 위치 정보
UNSUPPORTED_TRATYPE	-14	지원되지 않는 모션 유형
INVALID_DH_LIST	-15	잘못된 변환 매개변수 목록입니다. 개발자에게 문의하세요.
INTERVAL_PORTS_MUST_NOTNONE	-16	간격, 포트 목록, 레벨 기간은 비워둘 수 없습니다.
INVALID_IP_ADDRESS	-17	잘못된 IP 주소
INVALID_DH_PARAMETERS	-18	잘못된 DH 매개변수
INVALID_PAYLOAD_INFO	-19	잘못된 페이로드 정보
INVALID_FLYSHOT_CONFIG	-20	잘못된 플라이샷 구성 매개변수
FAILED_TO_DOWNLOAD_SAME_NAME_FILE	-21	같은 이름의 파일이 이미 존재하기 때문에 다운로드에 실패했습니다.
FAILED_TO_CONVERT_CART_TO_JOINT	-22	데카르트 좌표를 관절 좌표로 변환하지 못했습니다.
FAILED_TO_GET_ALL_INVERSE_KINEMATICS	-23	한 번의 회전으로 8가지 해결책을 얻지 못했습니다.
COORDINATE_LIMIT_EXCEEDED	-24	좌표계 한도를 초과했습니다. 최대값은 ID가 [1, 50]인 좌표계 50개입니다.
FILE_NOTEXIST	-25	파일이 존재하지 않습니다
INVALID_IO_LIST_PARAMETERS	-26	잘못된 IO 목록 매개변수
INVALID_PARAMETER	-27	잘못된 매개변수
NOT_FOUND	-28	요청한 데이터를 찾을 수 없습니다

Name	Enum Value	Description
LOCAL_PROXY_UNSUPPORTED	-29	현재 환경에서는 로컬 프록시가 지원되지 않습니다.
CONTROLLER_ERROR	-254	컨트롤러 오류입니다. 자세한 내용은 개발자에게 문의하세요.
SERVER_ERR	-255	기타 이유

3.2 RobotStatusEnum

설명

로봇 작동 상태.

가져오기

```python
from Agilebot import RobotStatusEnum
```

필드

Name	Enum Value	Description
ROBOT_UNKNOWN	-1	알 수 없는 상태
ROBOT_IDLE	0	로봇 유휴
ROBOT_RUNNING	1	로봇이 달리고 있다
ROBOT_TEACHING	2	로봇티칭
ROBOT_DRAG	3	드래그 모드의 로봇
ROBOT_FORCE_DRAG	4	강제 드래그 모드의 로봇
ROBOT_IDLE_TO_RUNNING	101	유휴 상태에서 실행 중으로 전환
ROBOT_IDLE_TO_TEACHING	102	유휴 상태에서 교육 상태로 전환

Name	Enum Value	Description
ROBOT_RUNNING_TO_IDLE	103	실행 중에서 유휴 상태로 전환
ROBOT_TEACHING_TO_IDLE	104	교육에서 유휴 상태로 전환

3.3 CtrlStatusEnum

설명

컨트롤러 작동 상태.

가져오기

```python
from Agilebot import CtrlStatusEnum
```

필드

Name	Enum Value	Description
CTRL_UNKNOWN	-1	알 수 없는 컨트롤러 상태
CTRL_INIT	0	컨트롤러 초기화 중
CTRL_ENGAGED	1	컨트롤러 활성화됨
CTRL_ESTOP	2	컨트롤러 비상 정지
CTRL_TERMINATED	3	컨트롤러가 종료되었습니다.
CTRL_ANY_TO_ESTOP	101	모든 상태에서 비상 정지로 전환
CTRL_ESTOP_TO_ENGAGED	102	비상 정지에서 활성화로 전환
CTRL_ESTOP_TO_TERMINATED	103	비상 정지에서 종료로 전환

3.4 ServoStatusEnum

설명

서보 컨트롤러 상태.

가져오기

```python
from Agilebot import ServoStatusEnum
```

필드

Name	Enum Value	Description
SERVO_UNKNOWN	-1	알 수 없는 서보 컨트롤러 상태
SERVO_IDLE	1	서보 컨트롤러 유휴
SERVO_RUNNING	2	서보 컨트롤러 실행 중
SERVO_DISABLE	3	서보 컨트롤러 비활성화됨
SERVO_WAIT_READY	4	준비를 기다리는 서보 컨트롤러
SERVO_WAIT_DOWN	5	종료를 기다리는 서보 컨트롤러
SERVO_INIT	10	서보 컨트롤러 초기화 중

3.5 SoftModeEnum

설명

로봇 PC 상태 모드.

가져오기

```python
from Agilebot import SoftModeEnum
```

필드

Name	Enum Value	Description
UNKNOWN	0	알려지지 않은
AUTO	1	자동 모드
MANUAL_LIMIT	2	수동 속도 제한 모드
MANUAL	3	수동 모드

3.6 PoseType

설명

로봇 포즈 유형.

가져오기

```python
from Agilebot import PoseType
```

필드

Name	Enum Value	Description
JOINT	0	Joint 공간
CART	1	데카르트 좌표

3.7 TCSType

설명

좌표계 유형.

가져오기

```python
from Agilebot import TCSType
```

필드

Name	Enum Value	Description
JOINT	0	Joint 공간
BASE	1	기본 좌표계
WORLD	2	세계 좌표계
USER	3	사용자 좌표계
TOOL	4	도구 좌표계
RTCP_USER	5	RTCP 사용자 좌표계
RTCP_TOOL	6	RTCP 도구 좌표계

3.8 Joint

설명

로봇 관절 위치를 설명하는 데이터 구조입니다.

가져오기

```python
from Agilebot import Joint
```

속성

Attribute	Type	Description
j1	float	조인트 1의 가치
j2	float	조인트 2의 가치

Attribute	Type	Description
j3	float	조인트 3의 가치
j4	float	조인트 4의 가치
j5	float	조인트 5의 가치
j6	float	조인트 6의 가치
j7	float	조인트 7의 가치
j8	float	관절 8의 가치
j9	float	조인트 9의 가치

3.9 Position

설명

로봇 데카르트 자세를 설명하는 데이터 구조입니다.

가져오기

```python
from Agilebot import Position
```

속성

Attribute	Type	Description
x	float	X축 좌표
y	float	Y축 좌표
z	float	Z축 좌표
a	float	A축 각도
b	float	B축 각도

Attribute	Type	Description
c	float	C축 각도

3.10 Posture

설명

로봇 자세 매개변수를 설명하는 데이터 구조입니다.

가져오기

```python
from Agilebot import Posture
```

속성

Attribute	Type	Description
turn_cycle	int[]	각 축의 회전 수입니다. 정수 범위 ..., -2,-1,0,1,2,... 0° 자세에서 0. 선형/arc 동작 중에 목표 회전 횟수가 자동으로 선택됩니다. 회전 ≥ 180°는 ≥ 1로 매핑되고, -179.99°와 179.99° 사이는 0으로 매핑되고, ≤ -180°는 ≤ -1로 매핑됩니다.
wrist_flip	int	손목 뒤집기 자세(-1/1). 6축 로봇(관절 5)에서 1은 손목이 아래쪽으로 뒤집히는 것을 의미하고, -1은 위쪽으로 뒤집히는 것을 의미합니다.
arm_up_down	int	팔 올리기/down 자세(-1/1). 6축 로봇(관절 3)에서 1은 팔이 관절 4와 2 사이의 선 위에 있음을 의미하고(J3 < 0), -1은 팔이 해당 선 아래에 있음을 의미합니다(J3 > 0).
arm_back_front	int	팔 앞/back 자세(-1/1). 6축 로봇(관절 1)에서 1은 팔이 앞에 있음(앞을 향할 때 축 1의 왼쪽에 있는 축 2)을 의미하고, -1은 팔이 뒤에 있음(축 1의 오른쪽에 있는 축 2)을 의미합니다.
arm_left_right	int	팔 왼쪽/right 자세(-1/1). 4축 SCARA 로봇(관절 2)에서 1은 팔이 오른쪽에 있음을 의미하고 -1은 팔이 왼쪽에 있음을 의미합니다.

3.11 BaseCartData

설명

로봇 데카르트 목표 포즈를 설명하는 데이터 구조입니다.

가져오기

```python
from Agilebot import BaseCartData
```

속성

Attribute	Type	Description
position	Position	데카르트 자세 데이터
posture	Posture	로봇 자세 매개변수

3.12 MotionPose

설명

로봇 자세를 설명하는 데이터 구조입니다. XYZ 방향의 거리는 밀리미터(mm) 단위로 측정되고 각도 데이터는 도(deg) 단위로 측정됩니다. 일부 버전에서는 각도 정보가 라디안으로 반환됩니다. 자세한 내용은 API 반환 설명을 참조하세요.

가져오기

```python
from Agilebot import MotionPose
```

속성

Attribute	Type	Description
cartData	BaseCartData	데카르트 데이터

Attribute	Type	Description
joint	Joint	Joint 데이터
pt	PoseType	포즈 유형, 기본값은 PoseType.JOINT 입니다.

3.13 DHparam

설명

로봇 DH 매개변수 목록.

가져오기

```python
from Agilebot import DHparam
```

속성

Attribute	Type	Description	Default Value
id	int	DH 매개변수의 ID	-1
d	float	Joint 거리	-1.0
a	float	링크 길이	-1.0
alpha	float	링크 비틀림 각도	-1.0
offset	float	Joint 오프셋	-1.0

3.14 PayloadInfo

설명

로봇 페이로드 정보.

가져오기

```python
from Agilebot import PayloadInfo
```

속성

Attribute	Type	Description
id	int	페이로드 ID
weight	float	페이로드 중량(kg)
comment	str	논평
mass_center	MassCenter	질량 중심
inertia_moment	InertiaMoment	관성 모멘트

매스센터

Field	Type	Description
x	float	엑스
y	float	와이
z	float	지

관성모멘트

Field	Type	Description
lx	float	LX
ly	float	리
lz	float	LZ

3.15 ProgramCartData

설명

데카르트 데이터를 프로그래밍합니다.

가져오기

```python
from Agilebot import ProgramCartData
```

속성

Attribute	Type	Description
baseCart	BaseCartData	데카르트 데이터
uf	int	사용중인 사용자 좌표계 ID
tf	int	사용 중인 공구 좌표계 ID

3.16 ProgramPoseData

설명

프로그램 포즈 데이터.

가져오기

```python
from Agilebot import ProgramPoseData
```

속성

Attribute	Type	Description
cartData	ProgramCartData	프로그램 포즈 데이터
joint	Joint	Joint 데이터
pt	PoseType	포즈 유형, 기본값은 PoseType.JOINT 입니다.

3.17 ProgramPose

설명

프로그램 포즈.

가져오기

```python
from Agilebot import ProgramPose
```

속성

Attribute	Type	Description
poseData	ProgramPoseData	프로그램 포즈 데이터
id	int	포즈 ID
name	str	포즈 이름
comment	str	포즈 코멘트

3.18 PoseRegisterData

설명

레지스터에 저장된 데이터를 나타내는 데 사용되는 레지스터 데이터 클래스입니다.

가져오기

```python
from Agilebot import PoseRegisterData
```

속성

Attribute	Type	Description
cartData	BaseCartData	데카르트 자세 데이터
joint	Joint	Joint 데이터
pt	PoseType	포즈 유형, 기본값은 PoseType.JOINT 입니다.

3.19 PoseRegister

설명

컨트롤러에 저장된 레지스터 데이터를 나타내는 데 사용되는 레지스터 클래스입니다.

가져오기

```python
from Agilebot import PoseRegister
```

속성

Attribute	Type	Description
poseRegisterData	PoseRegisterData	포즈 데이터 등록
id	int	아이디 등록
name	str	이름 등록
comment	str	댓글 등록

3.20 TransformStatusEnum

설명

궤적 변환 상태.

가져오기

```python
from Agilebot import TransformStatusEnum
```

필드

Name	Enum Value	Description
TRANSFORM_START	0	변환이 시작되었습니다
TRANSFORM_RUNNING	1	변환 진행 중
TRANSFORM_SUCCESS	2	변환 성공
TRANSFORM_FAILED	3	변환 실패
TRANSFORM_NOT_FOUND	4	변환을 찾을 수 없습니다.
TRANSFORM_UNKNOWN	5	알 수 없는 변환 상태

3.21 HW상태

설명

하드웨어 상태 열거.

가져오기

```python
from Agilebot import HWState
```

필드

Name	Description
TOPIC_JOINT	로봇 공동 피드백 게시
TOPIC_CURRENT_CARTESIAN	현재 TCP 데카르트 포즈 게시
TOPIC_UF	현재 사용자 프레임 정보 게시

Name	Description
TOPIC_TF	현재 도구 프레임 정보 게시
TOPIC_VELOCITY	전역 속도 비율 게시
TOPIC_RUNNING_STATUS	컨트롤러 실행 상태 게시
TOPIC_INTERPRETER_STATUS	통역사 상태 게시
TOPIC_ROBOT_STATUS	로봇 상태 게시
TOPIC_CTRL_STATUS	컨트롤러 상태 게시
TOPIC_SERVO_STATUS	서보 컨트롤러 상태 게시
TOPIC_TRAJECTORY_RECORDS_STATUS	궤적기록 현황 공개

3.22 좌표계 유형

설명

로봇 좌표계 유형.

가져오기

```python
from Agilebot import CoordinateSystemType
```

필드

Name	Enum Value	Description
UserFrame	0	사용자 좌표계
ToolFrame	1	도구 좌표계

3.23 동등 어구

설명

도구 및 사용자 프레임을 포함한 좌표계 데이터.

가져오기

```python
from Agilebot import Coordinate
```

속성

Attribute	Type	Description
id	int	좌표계 ID
name	str	좌표계 이름
comment	str	좌표계 코멘트
data	Position	좌표 포즈 데이터

3.24 끌기 상태

설명

로봇 끌기 상태.

가져오기

```python
from Agilebot import DragStatus
```

속성

Attribute	Type	Description
cart_status	CartStatus	데카르트 축 드래그/lock 상태
joint_status	JointStatus	Joint 축 드래그/lock 상태

Attribute	Type	Description
is_continuous_drag	bool	연속 드래그 플래그

3.24.1 장바구니 상태

설명

데카르트 축 끌기 상태를 나타내는 데카르트 상태 클래스입니다.

가져오기

```python
from Agilebot import CartStatus
```

속성

Attribute	Type	Description
x	bool	X축 상태
y	bool	Y축 상태
z	bool	Z축 상태
a	bool	A축 상태
b	bool	B축 상태
c	bool	C축 상태

3.24.2 조인트상태

설명

각 로봇 관절의 상태를 나타내는 Joint 상태 클래스입니다. 모든 관절의 기본값은 True (사용 가능)입니다.

가져오기

```python
from Agilebot import JointStatus
```

속성

Attribute	Type	Description
j1	bool	J1의 상태
j2	bool	J2의 상태
j3	bool	J3의 상태
j4	bool	J4의 상태
j5	bool	J5의 상태
j6	bool	J6의 상태
j7	bool	J7의 상태
j8	bool	J8의 상태
j9	bool	J9의 상태

3.25 Modbus채널

설명

Modbus 채널 유형.

가져오기

```python
from Agilebot import ModbusChannel
```

필드

Name	Enum Value	Description
CONTROLLER_TCP_TO_485	2	TCP-RS485 모듈 채널
WRIST_485_0	3	손목 AI 채널
WRIST_485_1	4	손목 DO 채널
CONTROLLER_485	5	컨트롤러 485 채널

3.26 페이로드식별상태

설명

페이로드 식별 상태.

가져오기

```python
from Agilebot import PayloadIdentifyState
```

필드

Name	Enum Value	Description
PAYLOAD_CHECK_INIT	0	페이로드 확인 초기화 진행 중
PAYLOAD_CHECK_RUNNING	1	페이로드 확인 실행 중
PAYLOAD_CHECK_SUCCESS	2	페이로드 확인 성공
PAYLOAD_CHECK_FAILED	3	페이로드 확인 실패
PAYLOAD_IDENTIFY_INIT	4	페이로드 식별 초기화
PAYLOAD_IDENTIFY_RUNNING	5	페이로드 식별 실행 중
PAYLOAD_IDENTIFY_SUCCESS	6	페이로드 식별 성공
PAYLOAD_IDENTIFY_FAILED	7	페이로드 식별 실패

Name	Enum Value	Description
WRONG_DATA	-1	잘못된 데이터

3.27 이동 모드

설명

조그 이동 모드.

가져오기

```python
from Agilebot import MoveMode
```

필드

Name	Enum Value	Description
Continuous	0	연속 동작
Stepping	1	스텝모션

3.28 SignalType

설명

SignalType 는 로봇 제어 시스템이 각 신호 유형을 올바르게 식별하고 관리할 수 있도록 디지털, 사용자, 로봇, 그룹 및 아날로그 IO 신호를 구별하는 열거형 클래스입니다.

가져오기

```python
from Agilebot import SignalType
```

필드

Name	Enum Value	Description
DI	1	디지털 입력 신호
DO	2	디지털 출력 신호
UI	3	사용자 입력 신호
UO	4	사용자 출력 신호
RI	5	리모콘 입력 신호
RO	6	리모콘 출력 신호
GI	7	그룹 입력 신호
GO	8	그룹 출력 신호
TAI	9	도구 아날로그 입력 신호
TDI	10	도구 디지털 입력 신호
TDO	11	도구 디지털 출력 신호
AI	12	아날로그 입력 신호
AO	13	아날로그 출력 신호

3.29 신호값

설명

SignalValue 는 신호의 꺼짐 및 켜짐 상태를 나타내는 두 개의 상수 OFF 및 ON 를 사용하여 신호 on/off 값을 정의합니다.

가져오기

```python
from Agilebot import SignalValue
```

속성

Name	Value	Description
OFF	0	신호 꺼짐
ON	1	신호 켜짐

3.30 SerialParams

설명

직렬 통신 구성. Modbus-RTU/TCP~485 통신을 수행할 때 사용됩니다. 연결 매개변수를 설명하기 위해 인스턴스를 하위 수준 통신 인터페이스로 직접 전달할 수 있습니다.

가져오기

```python
from Agilebot import SerialParams, ModbusChannel, ModbusParity
```

생성자

```python
SerialParams(
    id: int = 1,
    channel: ModbusChannel = ModbusChannel.CONTROLLER_TCP_TO_485,
    ip: str = "10.27.1.80",
    port: int = 502,
    baud: int = 9600,
    data_bit: int = 8,
    stop_bit: int = 1,
    parity: ModbusParity = ModbusParity.NONE,
    timeout: int = 1000
)
```

속성

Attribute	Type	Description
id	int	디바이스 스테이션/슬레이브 ID

Attribute	Type	Description
channel	ModbusChannel	통신 채널 열거, 기본값은 CONTROLLER_TCP_TO_485
ip	str	TCP-485 사용 시 게이트웨이 IP 주소
port	int	TCP-485 사용 시 게이트웨이 포트
baud	int	전송 속도, 기본 9600 bps
data_bit	int	데이터 비트, 기본값 8
stop_bit	int	정지 비트, 기본값 1
parity	ModbusParity	패리티 설정, 기본 ModbusParity.NONE
timeout	int	Read/write 시간 제한(밀리초), 기본값 1000

3.31 SubPub 주제 유형

3.31.1 RobotTopicType

설명

로봇 상태를 구독하거나 게시하는 데 사용되는 로봇 실시간 주제 열거입니다.

가져오기

```python
from Agilebot import RobotTopicType
```

열거형 값

Name	Description
JOINT_POSITION	Joint 피드백(rad)
CARTESIAN_POSITION	활성 user/tool 프레임 아래의 TCP 포즈
TCP_WORLD_CARTESIAN	월드 프레임의 TCP 포즈

Name	Description
TCP_BASE_CARTESIAN	기본 프레임의 TCP 포즈
USER_FRAME	현재 사용자 프레임
TOOL_FRAME	현재 도구 프레임
VELOCITY_RATIO	전역 속도 비율(0-100%)
GLOBAL_ACC_RATIO	전역 가속 비율(0-100%)
CALIBRATE_STATUS	축 그룹 교정 상태
TRAJECTORY_RECORD	궤적 기록 상태
RUNNING_STATUS	컨트롤러 실행 상태
INTERPRETER_STATUS	통역사 상태(/paused/stopped/error 실행 중)
ROBOT_STATUS	로봇 전체 현황
CTRL_STATUS	컨트롤러 준비 상태
SERVO_STATUS	서보 활성화 상태
USER_OP_MODE	사용자 조작 모드
SOFT_OP_MODE	소프트 작동 모드
OPERATION_STATUS	운전상태
PERFORMANCE_GEAR	퍼포먼스 기어
POWER_STATUS	전원 상태
POWER_ON_STATUS	전원 켜짐 상태
SAFETY_ALARM	안전 경보 상태
SAFETY_PLANE_STATUS	안전면 상태
TOOL_POSTURE_LIMIT_STATUS	도구 자세 제한 상태
JOINTS_RECORD	Joint 레코드

Name	Description
TP_PROGRAM_STATUS	TP 프로그램 상태

3.31.2 RegTopicType

설명

내부 R / MR / SR / PR 레지스터에 액세스하기 위한 구독 주제 열거를 등록합니다.

가져오기

```python
from Agilebot import RegTopicType
```

열거형 값

Name	Value	Description
R	R	일반 레지스터
MR	MR	모션 레지스터
SR	SR	문자열 레지스터
PR	PR	Position 레지스터

3.31.3 IOTopicType

설명

디지털, 그룹, 아날로그, 로봇 및 도구 IO를 다루는 IO 구독 주제 열거입니다.

가져오기

```python
from Agilebot import IOTopicType
```

열거형 값

Name	Value	Description
DI	DI	디지털 입력
DO	DO	디지털 출력
GI	GI	그룹 입력
GO	GO	그룹 출력
RI	RI	리모콘 입력
RO	RO	리모콘 출력
UI	UI	사용자 입력
UO	UO	사용자 출력
TDI	TDI	도구 디지털 입력
TDO	TDO	도구 디지털 출력
TAI	TAI	도구 아날로그 입력
AI	AI	아날로그 입력
AO	AO	아날로그 출력

3.32 BasScript 관련 유형

BasScript 클래스는 다음 열거형 유형을 사용하여 다양한 매개변수와 구성 옵션을 정의합니다.

3.32.1 MovePoseType

설명

좌표 유형

가져오기

```python
from Agilebot import MovePoseType
```

열거형 값

Name	Enum Value	Description
PR	"PR"	PR

3.32.2 부드러운 유형

설명

부드러운 유형

가져오기

```python
from Agilebot import SmoothType
```

열거형 값

Name	Enum Value	Description
FINE	"괜찮은"	없음
SMOOTH_DISTANCE	"SD"	선의

3.32.3 속도 유형

설명

속도 유형

가져오기

```python
from Agilebot import SpeedType
```

열거형 값

Name	Enum Value	Description
VALUE	0	순수한
MR	1	상대적인

3.32.4 레지스터 유형

설명

레지스터 유형

가져오기

```python
from Agilebot import RegisterType
```

열거형 값

Name	Enum Value	Description
R	"아르 자형"	아르 자형
SR	"SR"	SR
MH	"MH"	MH
MI	"MI"	MI

3.32.5 IO 유형

설명

IO 유형

가져오기

```python
from Agilebot import IOType
```

열거형 값

Name	Enum Value	Description
DI	"디"	디
DO	"하다"	~하다
AI	"일체 포함"	일체 포함
AO	"아오"	AO
RI	"리"	RI
RO	"로"	RO
UI	"UI"	UI
UO	"우오"	우오
GI	"미군 병사"	미군 병사
GO	"가다"	가다
TAI	"타이"	타이
TAO	"타오"	타오
TDI	"TDI"	TDI
TDO	"TDO"	TDO

3.32.6 IOS상태

설명

IO 스위치 상태

가져오기

```python
from Agilebot import IOStatus
```

열거형 값

Name	Enum Value	Description
ON	"에"	~에
OFF	"끄다"	끄다
PULSE	"맥박"	맥박
POSITIVE_EDGE	"PE"	포지티브 엣지
NEGATIVE_EDGE	"네"	네거티브 엣지

3.32.7 수학 연산자

설명

수학 연산자

가져오기

```python
from Agilebot import MathOperator
```

열거형 값

Name	Enum Value	Description
ADD	"+"	추가하다
SUB	"-"	덜다
MUL	"*"	곱하다
DIV	"/"	나누다

3.32.8 부울연산자

설명

부울 연산자

가져오기

```python
from Agilebot import BooleanOperator
```

열거형 값

Name	Enum Value	Description
AND	"그리고"	그리고
OR	"또는"	또는
EQ	"="	동일한
NE	"<>"	같지 않음
GT	">"	보다 큼
GE	">="	이상
LT	"<"	미만
LE	"<="	작거나 같음

3.32.9 할당 유형

설명

과제 유형

가져오기

```python
from Agilebot import AssignType
```

열거형 값

Name	Enum Value	Description
R	"아르 자형"	아르 자형
MR	"MR"	MR

Name	Enum Value	Description
PR	"PR"	PR
PR_ELEMENT	"PR"	PR 요소
SR	"SR"	SR
UF	"UF"	UF
TF	"TF"	TF
MH	"MH"	MH
MI	"MI"	MI
DO	"하다"	~하다
RO	"로"	RO
GO	"가다"	가다
AO	"아오"	AO
TAO	"타오"	타오
TDO	"TDO"	TDO

3.32.10 기타 유형

설명

다른 유형

가져오기

```python
from Agilebot import OtherType
```

열거형 값

Name	Enum Value	Description
VALUE	0	값
STRING	1	끈
IO_STATUS	2	상태
CURRENT_POSE	3	현재 포즈

3.32.11 현재 포즈

설명

현재 포즈

가져오기

```python
from Agilebot import CurrentPose
```

열거형 값

Name	Enum Value	Description
J_POS	"J_POS"	Joint 값
L_POS	"L_POS"	데카르트 값

3.32.12 로드 유형

설명

부하 유형

가져오기

```python
from Agilebot import LoadType
```

열거형 값

Name	Enum Value	Description
R	"아르 자형"	아르 자형
SR	"SR"	SR
STRING	"끈"	끈
VALUE	"값"	값

3.32.13 Str 유형

설명

문자열 유형

가져오기

```python
from Agilebot import StrType
```

열거형 값

Name	Enum Value	Description
STRING	"끈"	끈
SR	"SR"	SR 레지스터

3.32.14 값 유형

설명

값 유형

가져오기

```python
from Agilebot import ValueType
```

열거형 값

Name	Enum Value	Description
VALUE	"값"	값
R	"아르 자형"	R 레지스터

3.32.15 매개변수 유형

설명

매개변수 유형

가져오기

```python
from Agilebot import ParamType
```

열거형 값

Name	Enum Value	Description
TF_NO	"TF_NO"	TF 번호
UF_NO	"UF_NO"	UF 번호
OVC	"OVC"	글로벌 속도
OAC	"OAC"	글로벌 가속
PAYLOAD_NO	"PAYLOAD_NO"	페이로드 번호

4 방법 및 예

4.1 기본 로봇 작동

개요

Arm 클래스는 Agilebot 로봇과 관련된 대부분의 고주파 인터페이스를 캡슐화하여 연결 관리, 상태 쿼리 및 제어 명령 전달을 처리합니다. 일반적인 사용법:

1. `Arm(local_proxy=False)` 인스턴스화
2. 컨트롤러/pendant와의 통신을 설정하려면 `connect()` 를 호출하세요.
3. 필요에 따라 모션, 상태 또는 I/O API를 호출합니다.
4. 마지막으로 `disconnect()` 또는 `release_access()` 를 호출하여 리소스를 확보하세요.

수동 구성이 필요하지 않습니다. 다음 작업이 자동으로 처리됩니다.

- SDK 버전 확인
- 컨트롤러 유형 식별
- 프록시 서비스 선택
- 선택한 통신 경로 로깅

메모

- 로봇 소프트웨어 버전이 7.7.0보다 낮은 경우 `Arm` 클래스를 인스턴스화할 때 `local_proxy=True` 를 확인하고 PC에 로컬 통신 기능이 있는지 확인하세요.
- 산업용 로봇의 경우 PC 모드를 활성 상태로 유지해야 하는 경우:
  - 펜던트 제어 권한을 신청하려면 연결 후 `acquire_access()` 를 호출하십시오.
  - 작업이 완료된 후 즉시 `release_access()` 에 전화하여 권한을 해제하세요.
  - 펜던트의 제어 권한을 장기간 점유하지 마십시오.

생성자

메서드 이름	**Arm(`local_proxy` : bool = False)**
설명	사용 가능한 모든 인터페이스를 캡슐화하는 Agilebot 로봇 클래스입니다.
요청 매개변수	`local_proxy` : bool 로컬 컨트롤러 프록시 사용 여부(기본값 `False` ; `True` : 로컬에서 프록시 시작, IP 주소 입력(도메인 이름 아님)만 지원, 로컬 통신 기능이 있는 PC 필요, `False` : 컨트롤러/pendant 내장 프록시 서비스 사용, 로봇 소프트웨어는 v7.7 이상이고 프록시 서비스가 설치되어 있어야 함).
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.1 로봇에 연결

메서드 이름	**connect(arm_controller_ip: str = `COBOT_IP` , teach_panel_ip: Optional[str] = None) -> StatusCodeEnum**
설명	Agilebot 로봇에 연결합니다. 이 방법은 협동적 /industrial 모델을 자동 감지하고, 티치 펜던트 IP를 처리하고, 적절한 프록시 서비스를 시작합니다.
요청 매개변수	`arm_controller_ip` : str 컨트롤러 IP(기본값은 `COBOT_IP`). `teach_panel_ip` : 선택 사항[str] 산업용 로봇에 대한 펜던트 IP 티칭(생략된 경우 협동 로봇에는 펜던트 IP가 필요하지 않고 산업용 로봇에는 기본 펜던트 IP 또는 컨트롤러 IP를 사용함).
반환 값	StatusCodeEnum: 함수 실행 결과
메모	- Local proxy is started only if `local_proxy=True` **and** a valid IP address is provided; Airbot domains only support direct connection and cannot work with the local proxy. - Invalid IP/domain values return `INVALID_IP_ADDRESS` . - A failure to contact the controller returns `CONTROLLER_CONNECTION_ERROR` with details.
호환 로봇 소프트웨어	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

메서드 이름	connect(arm_controller_ip: str = `COBOT_IP` , teach_panel_ip: Optional[str] = None) -> StatusCodeEnum
버전	

4.1.2 로봇과의 연결 상태 확인

메서드 이름	is_connected() -> bool
설명	로봇과의 연결이 유효한지 확인
요청 매개변수	매개변수 없음
반환 값	bool: 연결 상태, True: 연결이 유효함, False: 연결이 유효하지 않음
메모	The legacy `is_connect()` method is deprecated; use `is_connected()` instead.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.3 로봇과의 연결 끊기

메서드 이름	disconnect()
설명	Agilebot 로봇 연결 끊기
요청 매개변수	매개변수 없음
반환 값	반품 불가
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/connect_disconnect.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的基础通讯连接断开示例 / Example of a basic communication co
nnection
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 检查连接状态
# [EN] Check connection status
connect_status = arm.is_connected()
# [ZH] 打印结果
# [EN] Print the result
print(
    f"当前连接状态 / Current connection status: {connect_status}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.4 현재 로봇 모델 가져오기

메서드 이름	get_arm_model_info() -> tuple[str, StatusCodeEnum]
설명	현재 로봇 모델 정보 가져오기
요청 매개변수	매개변수 없음
반환 값	str: 로봇 모델, 예: "GBT-C5A" StatusCodeEnum: 기능 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/get_arm_model_info.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的型号获取示例 / Example of model info acquisition
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 获取型号
# [EN] Get the robot model
model_info, ret = arm.get_arm_model_info()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print("获取型号成功 / Get robot model successfully")
    print(f"机器人型号 / Robot model: {model_info}")
else:
    print(
        f"获取型号失败，错误代码 / Get robot model failed, error code: {ret.err
msg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
```

4.1.5 로봇의 작동 상태를 가져옵니다.

메서드 이름	get_robot_status() -> tuple[RobotStatusEnum, StatusCodeEnum]
설명	Agilebot 로봇의 작동 상태를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	RobotStatusEnum: 로봇 동작 상태 StatusCodeEnum: 기능 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/get_robot_status.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的机器人状态获取示例 / Example of obtaining robot status
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取机器人运行状态
# [EN] Get the robot running status
state, ret = arm.get_robot_status()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "获取机器人运行状态成功 / Get robot running status successful"
    )
    print(
        f"机器人运行状态 / Robot running status: {state.msg}"
    )
else:
    print(
        f"获取机器人运行状态失败，错误代码 / Get robot running status failed, err
```

```
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.6 현재 컨트롤러 작동 상태 가져오기

메서드 이름	get_ctrl_status() -> tuple[CtrlStatusEnum, StatusCodeEnum]
설명	Agilebot 로봇 컨트롤러의 현재 작동 상태를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	CtrlStatusEnum: 컨트롤러 작동 상태 StatusCodeEnum: 기능 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/get_ctrl_status.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的获取运动控制器运行状态示例 / Example of the operating statu
s of the controller
"""

from Agilebot import Arm, StatusCodeEnum
```

```
# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取运动控制器运行状态
# [EN] Get the controller running status
state, ret = arm.get_ctrl_status()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "获取运动控制器运行状态成功 / Get controller running status successful"
    )
    print(
        f"运动控制器运行状态 / Controller running status: {state.msg}"
    )
else:
    print(
        f"获取运动控制器运行状态失败，错误代码 / Get controller running status fai
led, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
```

```
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.7 현재 서보 컨트롤러 상태 가져오기

메서드 이름	get_servo_status() -> tuple[ServoStatusEnum, StatusCodeEnum]
설명	Agilebot 로봇 서보 컨트롤러의 현재 상태를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	ServoStatusEnum: 서보 컨트롤러 상태 StatusCodeEnum: 기능 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/get_servo_status.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的伺服状态获取示例 / Example of obtaining servo status
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
```

```
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取伺服控制器状态
# [EN] Get the servo controller status
state, ret = arm.get_servo_status()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "获取伺服控制器状态成功 / Get servo controller status successful"
    )
    print(
        f"伺服控制器状态 / Servo controller status: {state.msg}"
    )
else:
    print(
        f"获取伺服控制器状态失败，错误代码 / Get servo controller status failed,
error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.8 로봇의 현재 소프트 모드 가져오기

메서드 이름	get_soft_mode() -> tuple[SoftModeEnum, StatusCodeEnum]
설명	Agilebot 로봇의 현재 소프트 모드를 가져옵니다(PC 모드 매뉴얼/auto 상태)
요청 매개변수	매개변수 없음
반환 값	SoftModeEnum: 소프트 모드 상태 StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.9 로봇의 현재 소프트 모드 설정

메서드 이름	set_soft_mode(soft_mode : SoftModeEnum) -> StatusCodeEnum
설명	로봇의 현재 소프트 모드 설정(PC 모드 매뉴얼/auto 상태)
요청 매개변수	soft_mode : SoftModeEnum 소프트 모드 값
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/soft_mode.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的机器人软状态获取示例 / Example of obtaining the soft state
of a robot
"""

from Agilebot import Arm, SoftModeEnum, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
```

```
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置机器人当前的软状态为手动限速模式
# [EN] Set the robot's current soft state to manual limit mode
ret = arm.set_soft_mode(SoftModeEnum.MANUAL_LIMIT)
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "设置机器人当前的软状态为手动限速模式成功 / Set the robot's current soft s
tate to manual limit mode successfully"
    )
else:
    print(
        f"设置机器人当前的软状态为手动限速模式失败，错误代码 / Set the robot's curre
nt soft state to manual limit mode failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取机器人当前的软状态
# [EN] Get the robot's current soft state
state, ret = arm.get_soft_mode()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "获取机器人当前的软状态成功 / Get the robot's current soft state success
fully"
```

```
    )
    print(
        f"机器人当前的软状态 / Robot current soft state: {state.name}"
    )
else:
    print(
        f"获取机器人当前的软状态失败，错误代码 / Get the robot's current soft stat
e failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.10 로봇 작동 모드 가져오기

메서드 이름	get_op_mode() -> tuple[SoftModeEnum, StatusCodeEnum]
설명	현재 로봇 작동 모드(예: robots/virtual 컨트롤러에 대한 수동, 자동 등 작동 권한 상태)를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	SoftModeEnum: 동작 모드 StatusCodeEnum: 함수 실행 결과
메모	컨트롤러가 유효하지 않은 모드를 반환하면 `SoftModeEnum.UNKNOWN` 로 대체됩니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.11 로봇 동작 모드 설정

메서드 이름	**set_op_mode(`soft_mode` : SoftModeEnum) -> StatusCodeEnum**
설명	로봇 동작 모드를 설정합니다. 가상 로봇/simulators.에만 지원됩니다.
요청 매개변수	`soft_mode` : 대상 SoftModeEnum(`UNKNOWN` 일 수 없음).
반환 값	StatusCodeEnum: 함수 실행 결과
메모	`SoftModeEnum.UNKNOWN` 를 전달하면 `UNSUPPORTED_PARAMETER` 가 반환됩니다.
호환 로봇 소프트웨어 버전	협업(Copper): 시뮬레이션 전용 산업용(Bronze): 시뮬레이션 전용

4.1.12 상위컴퓨터가 제어권한을 획득하다

메서드 이름	**acquire_access()**
설명	PC는 제어권한을 획득하여 로봇을 PC mode에 투입합니다. 모드를 활성 상태로 유지하기 위해 내부적으로 하트비트 스레드를 시작합니다.
요청 매개변수	매개변수 없음
반환 값	반품 불가
메모	산업용 로봇에만 이 호출이 필요합니다. 협동로봇과 P7A는 그렇지 않습니다.
호환 로봇 소프트웨어 버전	협업(Copper): 지원되지 않음 산업용(Bronze): v7.5.0.0+

4.1.13 상위 컴퓨터에서 제어 권한 해제

메서드 이름	release_access()
설명	PC 제어 권한을 해제하여 PC mode 하트비트를 중지하고 제어권을 티치 펜던트에 다시 넘겨줍니다.
요청 매개변수	매개변수 없음
반환 값	반품 불가
메모	산업용 로봇에만 이 호출이 필요합니다.
호환 로봇 소프트웨어 버전	협업(Copper): 지원되지 않음 산업용(Bronze): v7.5.0.0+

예제 코드

arm/access_operate.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的操作权限获取示例 / Example of obtaining operation permiss
ions
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")

# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
```

```
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取权限
# [EN] Acquire access
arm.acquire_access()

# [ZH] 返还权限
# [EN] Release access
arm.release_access()

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.14 로봇 컨트롤러 버전 받기

메서드 이름	get_controller_version() -> tuple[str, StatusCodeEnum]
설명	Agilebot 로봇의 현재 컨트롤러 버전을 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	str: 컨트롤러 버전 StatusCodeEnum: 함수 실행 결과
메모	버전이 권장 수준보다 낮으면 SDK는 연결 중에 업그레이드 경고를 인쇄합니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/get_version.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的运动控制器版本获取示例 / Example of obtaining the controller version
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取运动控制器版本
# [EN] Get the controller version
version_info, ret = arm.get_controller_version()
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "获取运动控制器版本成功 / Get controller version successful"
    )
    print(
        f"运动控制器版本 / Controller version: {version_info}"
    )
else:
    print(
        f"获取运动控制器版本失败，错误代码 / Get controller version failed, error code: {ret.errmsg}"
```

```
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.15 로봇 LED 표시등 설정

메서드 이름	switch_led_light(mode : bool) -> StatusCodeEnum
설명	Agilebot 로봇의 LED 표시등 스위치 제어
요청 매개변수	mode : 부울 LED 상태(참 켜짐, 거짓 꺼짐)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.1.3+ 산업용(Bronze): 지원되지 않음

예제 코드

arm/switch_led_light.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的机器人灯环状态获取示例 / Example of obtaining the status o
f the LED
"""

import time

from Agilebot import Arm, StatusCodeEnum
```

```
# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 控制LED灯光关闭
# [EN] Control LED light off
ret = arm.switch_led_light(mode=False)
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
    print(
        "LED灯光关闭成功 / Control LED light off successfully"
    )
else:
    print(
        f"LED灯光关闭失败，错误代码 / Control LED light off failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

time.sleep(5)

# [ZH] 控制LED灯光开启
# [EN] Control LED light on
ret = arm.switch_led_light(mode=True)
# [ZH] 检查是否成功
# [EN] Check if successful
if ret == StatusCodeEnum.OK:
```

```
    print(
        "LED灯光开启成功 / Control LED light on successfully"
    )
else:
    print(
        f"LED灯光开启失败，错误代码 / Control LED light on failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.16 로봇 서보 전원 켜기

메서드 이름	servo_on() -> StatusCodeEnum
설명	Agilebot 로봇의 서보 전원을 켜세요
요청 매개변수	매개변수 없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.17 로봇 서보 전원 끄기

메서드 이름	servo_off() -> StatusCodeEnum
설명	Agilebot 로봇의 서보 전원을 끄십시오.

메서드 이름	servo_off() -> StatusCodeEnum
요청 매개변수	매개변수 없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.1.18 로봇 서보 재설정

메서드 이름	servo_reset() -> StatusCodeEnum
설명	Agilebot 로봇의 서보 재설정
요청 매개변수	매개변수 없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/servo_operate.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的控制伺服操作示例 / Eexample of control servo operation ex
ample
"""

import time

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
```

```
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 机器人伺服重置
# [EN] Robot servo reset
ret = arm.servo_reset()
if ret == StatusCodeEnum.OK:
    print("伺服重置成功 / Servo reset successfully")
else:
    print(
        f"伺服重置失败，错误代码 / Servo reset failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)
time.sleep(5)

# [ZH] 机器人伺服下电
# [EN] Robot servo power off
ret = arm.servo_off()
if ret == StatusCodeEnum.OK:
    print("伺服下电成功 / Servo power off successfully")
else:
    print(
        f"伺服下电失败，错误代码 / Servo power off failed, error code: {ret.err
msg}"
    )
    arm.disconnect()
    exit(1)
time.sleep(5)
```

```
# [ZH] 机器人伺服上电
# [EN] Robot servo power on
ret = arm.servo_on()
if ret == StatusCodeEnum.OK:
    print("伺服上电成功 / Servo power on successfully")
else:
    print(
        f"伺服上电失败，错误代码 / Servo power on failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.1.19 비상 정지

메서드 이름	estop() -> StatusCodeEnum
설명	Agilebot 로봇이 즉시 비상 정지를 수행하도록 합니다.
요청 매개변수	매개변수 없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

arm/estop.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Arm类下的紧急停止示例 / Example of emergency stop
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 机器人紧急停止
# [EN] Robot emergency stop
ret = arm.estop()
if ret == StatusCodeEnum.OK:
    print(
        "机器人紧急停止成功 / Robot emergency stop successfully"
    )
else:
    print(
        f"机器人紧急停止失败，错误代码 / Robot emergency stop failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
```

```
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2 로봇 모션 제어 및 상태

개요

Motion 클래스는 Agilebot 로봇의 모션 제어를 위한 핵심 개체로, 다음 핵심 기능을 캡슐화하는 역할을 합니다.

- Speed/acceleration 매개변수 관리
- 좌표계 관리
- 포인트 전환
- 궤적 모션 제어
- 끌어서 가르치기
- 실시간 제어
- 페이로드 관리

일반적인 작업흐름

Arm 연결이 설정된 후 `arm.motion` 를 통해 Motion 인스턴스를 얻으면 별도의 초기화가 필요하지 않습니다.

4.2.1 로봇 매개변수 가져오기

4.2.1.1 OVC 전체 속도 계수 가져오기

메서드 이름	**motion.get_OVC() -> tuple[float, StatusCodeEnum]**
설명	0에서 1 사이의 로봇의 현재 OVC 전체 속도 계수를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	float: 속도 비율(0~1 범위) StatusCodeEnum: 함수 실행 결과

메서드 이름	motion.get_OVC() -> tuple[float, StatusCodeEnum]
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.1.2 OAC 전체 가속 계수 가져오기

메서드 이름	motion.get_OAC() -> tuple[float, StatusCodeEnum]
설명	0에서 1.2 범위의 로봇의 현재 OAC 전체 가속 계수를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	float: 가속 비율(0~1.2 범위) StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.1.3 현재 TF 공구 좌표계 번호 가져오기

메서드 이름	motion.get_TF() -> tuple[int, StatusCodeEnum]
설명	로봇이 사용하는 현재 TF 도구 좌표계 번호를 가져옵니다. 유효 범위는 0~50입니다.
요청 매개변수	매개변수 없음
반환 값	int: 공구 좌표계 번호 StatusCodeEnum: 기능 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.1.4 현재 UF 사용자 좌표계 번호 가져오기

메서드 이름	motion.get_UF() -> tuple[int, StatusCodeEnum]
설명	로봇이 사용하는 현재 UF 사용자 좌표계 번호를 가져옵니다. 유효 범위는 0~50입니다.
요청 매개변수	매개변수 없음

메서드 이름	motion.get_UF() -> tuple[int, StatusCodeEnum]
반환 값	int: 사용자 좌표계 번호 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.1.5 현재 TCS 교육 좌표계 가져오기

메서드 이름	motion.get_TCS() -> tuple[TCSType, StatusCodeEnum]
설명	로봇이 사용하는 현재 TCS 티칭 좌표계를 가져옵니다. 자세한 내용은 TCSType를 참조하세요.
요청 매개변수	매개변수 없음
반환 값	TCSType: 티칭 좌표계 번호 StatusCodeEnum: 기능 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_param.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 全局参数获取示例 / Example of system parameter acquisition
"""

from Agilebot import Arm, StatusCodeEnum, TCSType

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
```

```
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取机器人各参数并打印
# [EN] Get robot parameters and print
res, ret = arm.motion.get_OVC()
if ret == StatusCodeEnum.OK:
    print(
        "获取全局速度比率成功 / Get global velocity ratio successful"
    )
else:
    print(
        f"获取全局速度比率失败，错误代码 / Get global velocity ratio failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
print(f"全局速度比率 / Global velocity ratio: {res}")

res, ret = arm.motion.get_OAC()
if ret == StatusCodeEnum.OK:
    print(
        "获取全局加速度比率成功 / Get global acceleration ratio successful"
    )
else:
    print(
        f"获取全局加速度比率失败，错误代码 / Get global acceleration ratio faile
d, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
print(f"全局加速度比率 / Global acceleration ratio: {res}")
```

```
res, ret = arm.motion.get_TCS()
if ret == StatusCodeEnum.OK:
    print(
        "获取示教坐标系类型成功 / Get teaching coordinate system type successfu
l"
    )
else:
    print(
        f"获取示教坐标系类型失败，错误代码 / Get teaching coordinate system type
failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
print(
    f"示教坐标系类型 / Teaching coordinate system type: {TCSType(res).name}"
)

res, ret = arm.motion.get_UF()
if ret == StatusCodeEnum.OK:
    print(
        "获取用户坐标系成功 / Get user coordinate system successful"
    )
else:
    print(
        f"获取用户坐标系失败，错误代码 / Get user coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
print(f"用户坐标系 / User coordinate system: {res}")

res, ret = arm.motion.get_TF()
if ret == StatusCodeEnum.OK:
    print(
        "获取工具坐标系成功 / Get tool coordinate system successful"
    )
else:
    print(
        f"获取工具坐标系失败，错误代码 / Get tool coordinate system failed, erro
r code: {ret.errmsg}"
    )
```

```
    arm.disconnect()
    exit(1)
print(f"工具坐标系 / Tool coordinate system: {res}")

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.2 로봇 매개변수 설정

4.2.2.1 OVC 전체 속도 계수 설정

메서드 이름	**motion.set_OVC(`value` : float) -> StatusCodeEnum**
설명	로봇의 OVC 전체 속도 계수를 설정합니다.
요청 매개변수	`value` : 부동 속도 비율(범위 0~1, > 0)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.2.2 OAC 전체 가속 계수 설정

메서드 이름	**motion.set_OAC(`value` : float) -> StatusCodeEnum**
설명	로봇의 OAC 전체 가속 계수 설정
요청 매개변수	`value` : 부동 가속도 비율 (범위 0.01~1.2)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.2.3 현재 TF 공구 좌표계 설정

메서드 이름	**motion.set_TF(`value` : int) -> StatusCodeEnum**
설명	로봇이 사용하는 현재 TF 도구 좌표계 설정
요청 매개변수	`value` : int 공구 좌표계 번호 (범위 0~50)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.2.4 현재 UF 사용자 좌표계 설정

메서드 이름	**motion.set_UF(`value` : int) -> StatusCodeEnum**
설명	로봇이 사용하는 현재 UF 사용자 좌표계를 설정합니다.
요청 매개변수	`value` : int 사용자 좌표계 번호 (범위 0~50)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.2.5 현재 TCS 티칭 좌표계 설정

메서드 이름	**motion.set_TCS(`value` : TCSType) -> StatusCodeEnum**
설명	로봇이 사용하는 현재 TCS 티칭 좌표계를 설정합니다. 자세한 내용은 TCSType를 참조하세요.
요청 매개변수	`value` : TCSType TCS 티칭 좌표계
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/set_param.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 全局参数设置示例 / Example of system parameter setting
"""

from Agilebot import Arm, StatusCodeEnum, TCSType

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置机器人各参数
# [EN] Set robot parameters
ret = arm.motion.set_OVC(0.7)
if ret == StatusCodeEnum.OK:
    print(
        "设置全局速度比率成功 / Set global velocity ratio successful"
    )
else:
    print(
        f"设置全局速度比率失败，错误代码 / Set global velocity ratio failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

ret = arm.motion.set_OAC(0.7)
if ret == StatusCodeEnum.OK:
    print(
        "设置全局加速度比率成功 / Set global acceleration ratio successful"
    )
else:
    print(
        f"设置全局加速度比率失败，错误代码 / Set global acceleration ratio faile
d, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

ret = arm.motion.set_TCS(TCSType.TOOL)
if ret == StatusCodeEnum.OK:
    print(
        "设置示教坐标系类型成功 / Set teaching coordinate system type successfu
l"
    )
else:
    print(
        f"设置示教坐标系类型失败，错误代码 / Set teaching coordinate system type
failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

ret = arm.motion.set_UF(0)
if ret == StatusCodeEnum.OK:
    print(
        "设置用户坐标系成功 / Set user coordinate system successful"
    )
else:
    print(
        f"设置用户坐标系失败，错误代码 / Set user coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
ret = arm.motion.set_TF(0)
if ret == StatusCodeEnum.OK:
    print(
        "设置工具坐标系成功 / Set tool coordinate system successful"
    )
else:
    print(
        f"设置工具坐标系失败，错误代码 / Set tool coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.3 데카르트 자세를 Joint 자세로 변환

메서드 이름	**motion.convert_cart_to_joint(`pose` : MotionPose, `uf_index` : int = 0, `tf_index` : int = 0) -> tuple[MotionPose, StatusCodeEnum]**
설명	포즈 데이터를 데카르트 좌표에서 관절 좌표로 변환
요청 매개변수	`pose` : MotionPose 데카르트 포즈 (PoseType.CART; 자세가 생략되면 SDK가 가능한 자세를 자동으로 해결합니다.) `uf_index` : int 사용자 좌표계 ID (기본값 0) `tf_index` : int 공구 좌표계 ID (기본값 0)
반환 값	MotionPose: 로봇 포즈 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.2.4 Joint 포즈를 데카르트 포즈로 변환

메서드 이름	**motion.convert_joint_to_cart(`pose` : MotionPose, `uf_index` : int = 0, `tf_index` : int = 0) -> tuple[MotionPose, StatusCodeEnum]**
설명	관절 좌표를 데카르트 좌표로 변환
요청 매개변수	`pose` : MotionPose Joint 포즈 `uf_index` : int 사용자 좌표계 ID(기본값 0) `tf_index` : int 공구 좌표계 ID(기본값 0)
반환 값	MotionPose: 로봇 포즈 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/convert_pose.py

```py
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 转换关节值坐标使用示例 / Example of converting joint coordinates
"""

from Agilebot import (
    Arm,
    MotionPose,
    PoseType,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
```

```
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化位姿
# [EN] Initialize pose
motion_pose = MotionPose()
motion_pose.pt = PoseType.CART
motion_pose.cartData.position.x = 221.5
motion_pose.cartData.position.y = -494.1
motion_pose.cartData.position.z = 752.0
motion_pose.cartData.position.a = -89.1
motion_pose.cartData.position.b = 31.6
motion_pose.cartData.position.c = -149.3

# [ZH] 转换关节值坐标
# [EN] Convert to joint coordinates
joint_pose, ret = arm.motion.convert_cart_to_joint(
    motion_pose
)
if ret == StatusCodeEnum.OK:
    print(
        "转换关节值坐标成功 / Convert to joint coordinates successful"
    )
else:
    print(
        f"转换关节值坐标失败，错误代码 / Convert to joint coordinates failed, er
ror code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果位姿
# [EN] Print result pose
```

```
print(f"位姿类型 / Pose type: {joint_pose.pt}")
print(
    f"轴位置 / Axis position: \n"
    f"J1:{joint_pose.joint.j1}\n"
    f"J2:{joint_pose.joint.j2}\n"
    f"J3:{joint_pose.joint.j3}\n"
    f"J4:{joint_pose.joint.j4}\n"
    f"J5:{joint_pose.joint.j5}\n"
    f"J6:{joint_pose.joint.j6}"
)

# [ZH] 转换笛卡尔坐标
# [EN] Convert to Cartesian coordinates
cart_pose, ret = arm.motion.convert_joint_to_cart(
    joint_pose
)
if ret == StatusCodeEnum.OK:
    print(
        "转换笛卡尔坐标成功 / Convert to Cartesian coordinates successful"
    )
else:
    print(
        f"转换笛卡尔坐标失败，错误代码 / Convert to Cartesian coordinates faile
d, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果位姿
# [EN] Print result pose
print(f"位姿类型 / Pose type: {cart_pose.pt}")
print(
    f"笛卡尔位置 / Cartesian position: \n"
    f"X:{cart_pose.cartData.position.x}\n"
    f"Y:{cart_pose.cartData.position.y}\n"
    f"Z:{cart_pose.cartData.position.z}\n"
    f"A:{cart_pose.cartData.position.a}\n"
    f"B:{cart_pose.cartData.position.b}\n"
    f"C:{cart_pose.cartData.position.c}"
)

# [ZH] 断开捷勃特机器人连接
```

```
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.5 Joint 스페이스 모션

메서드 이름	**motion.move_joint(`pose` : MotionPose, `vel` : float = 1, `acc` : float = 1) -> StatusCodeEnum**
설명	관절 공간에서 가장 빠른 경로를 이용하여 로봇 엔드 이펙터가 지정된 위치로 이동하도록 제어
요청 매개변수	`pose` : MotionPose 데카르트 또는 조인트 공간의 점 `vel` : 부동 속도 비율(범위 0~1) `acc` : 부동 가속 비율(범위 0~1.2)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/move_joint.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 关节运动使用示例 / Example of joint movement usage
"""

from Agilebot import (
    Arm,
    MotionPose,
    PoseType,
    StatusCodeEnum,
    Posture,
```

```
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化位姿
# [EN] Initialize pose
motion_pose = MotionPose()
motion_pose.pt = PoseType.CART
motion_pose.cartData.position.x = 121.3
motion_pose.cartData.position.y = 416.386
motion_pose.cartData.position.z = 74.104
motion_pose.cartData.position.a = -180
motion_pose.cartData.position.b = 0
motion_pose.cartData.position.c = 0
motion_pose.cartData.posture = Posture()
motion_pose.cartData.posture.arm_back_front = -1
motion_pose.cartData.posture.arm_left_right = -1
motion_pose.cartData.posture.arm_up_down = -1
motion_pose.cartData.posture.wrist_flip = -1


# [ZH] 发送运动请求
# [EN] Send motion request
ret = arm.motion.move_joint(motion_pose, vel=1, acc=1)
if ret == StatusCodeEnum.OK:
    print(
```

```
            "运动指令下发成功 / Joint motion command issued successfully"
        )
    else:
        print(
            f"运动指令下发失败，错误代码 / Joint motion command issued failed, error
    code: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

    # [ZH] 断开捷勃特机器人连接
    # [EN] Disconnect from the Agilebot robot
    arm.disconnect()
    print(
        "机器人断开连接成功 / Robot disconnected successfully"
    )
```

4.2.6 선형 운동

메서드 이름	**motion.move_line(`pose` : MotionPose, `vel` : float = 100, `acc` : float = 1) -> StatusCodeEnum**
설명	지정된 위치까지 직선으로 이동하도록 로봇 엔드 이펙터를 제어합니다.
요청 매개변수	`pose` : MotionPose 데카르트 또는 조인트 공간의 점 `vel` : 부동 TCP 속도(범위 1~4000mm/s) `acc` : 부동 가속 비율(범위 0~1.2)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/move_line.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 直线运动使用示例 / Example of linear motion usage
"""

from Agilebot import (
    Arm,
    MotionPose,
    PoseType,
    StatusCodeEnum,
)

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化位姿
# [EN] Initialize pose
motion_pose = MotionPose()
motion_pose.pt = PoseType.JOINT
motion_pose.joint.j1 = 0
motion_pose.joint.j2 = 0
motion_pose.joint.j3 = 60
motion_pose.joint.j4 = 60
motion_pose.joint.j5 = 0
motion_pose.joint.j6 = 0
```

```
# [ZH] 发送运动请求
# [EN] Send motion request
ret = arm.motion.move_line(motion_pose, vel=100, acc=0.5)
if ret == StatusCodeEnum.OK:
    print(
        "直线运动指令下发成功 / Line motion command issued successfully"
    )
else:
    print(
        f"直线运动指令下发失败，错误代码 / Line motion command issued failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 结束后断开机器人连接
# [EN] Disconnect from the robot after completion
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.7 원형 운동

메서드 이름	**motion.move_circle(** `pose1` **: MotionPose,** `pose2` **: MotionPose,** `vel` **: float = 100,** `acc` **: float = 1.0) -> StatusCodeEnum**
설명	로봇 엔드 이펙터를 제어하여 원형 궤적을 따라 지정된 위치로 이동합니다.
요청 매개변수	`pose1` : MotionPose 중간 포즈 `pose2` : MotionPose 대상 포즈 `vel` : 부동 TCP 속도 (범위 1~4000 mm/s) `acc` : 부동 가속 비율 (범위 0~1.2)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/move_circle.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 圆弧运动使用示例 / Example of circular arc motion usage
"""

import time

from Agilebot import (
    Arm,
    MotionPose,
    PoseType,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化位姿
# [EN] Initialize pose
motion_pose = MotionPose()
motion_pose.pt = PoseType.CART
```

```
motion_pose.cartData.position.x = 377.000
motion_pose.cartData.position.y = 202.820
motion_pose.cartData.position.z = 507.155
motion_pose.cartData.position.c = 0
motion_pose.cartData.position.b = 0
motion_pose.cartData.position.a = 0

# [ZH] 运动到初始点
# [EN] Move to initial position
ret = arm.motion.move_joint(motion_pose)
if ret == StatusCodeEnum.OK:
    print(
        "运动到初始点指令下发成功 / Move to initial position command issued succ
essfully"
    )
else:
    print(
        f"运动到初始点指令下发失败，错误代码 / Move to initial position command i
ssued failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 修改为运动中间点
# [EN] Modify to intermediate motion point
motion_pose.cartData.position.x = 488.300
motion_pose.cartData.position.y = 359.120
motion_pose.cartData.position.z = 507.155

# [ZH] 运动终点
# [EN] End position
motion_pose2 = MotionPose()
motion_pose2.pt = PoseType.CART
motion_pose2.cartData.position.x = 629.600
motion_pose2.cartData.position.y = 509.270
motion_pose2.cartData.position.z = 507.155
motion_pose2.cartData.position.c = 0
motion_pose2.cartData.position.b = 0
motion_pose2.cartData.position.a = 0

# [ZH] 等待运动结束
# [EN] Wait for motion to complete
```

```
time.sleep(10)

# [ZH] 开始运动
# [EN] Start motion
ret_code = arm.motion.move_circle(
    motion_pose, motion_pose2, vel=100
)
if ret_code == StatusCodeEnum.OK:
    print(
        "圆弧运动指令下发成功 / Circle motion command issued successfully"
    )
else:
    print(
        f"圆弧运动指令下发失败，错误代码 / Circle motion command issued failed, e
rror code: {ret_code.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.8 현재 포즈 가져오기

메서드 이름	**motion.get_current_pose(** `pose_type` **: PoseType,** `uf_index` **: int = 0,** `tf_index` **: int = 0) -> tuple[MotionPose, StatusCodeEnum]**
설명	직교 좌표계 또는 관절 좌표계에 있을 수 있는 로봇의 현재 자세를 가져옵니다.
요청 매개변수	`pose_type` :PoseType 포즈 유형 `uf_index` : int 사용자 좌표계 ID(PoseType.CART에만 해당, 기본값 0) `tf_index` : int 도구 좌표계 ID(PoseType.CART에만 해당, 기본값 0)

메서드 이름	**motion.get_current_pose(`pose_type` : PoseType, `uf_index` : int = 0, `tf_index` : int = 0) -> tuple[MotionPose, StatusCodeEnum]**
반환 값	MotionPose: 로봇 포즈 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_current_pose.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 当前关节位姿获取示例 / Example of current joint pose acquisition
"""

from Agilebot import Arm, PoseType, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取当前位姿
# [EN] Get current pose
```

```
motion_pose, ret = arm.motion.get_current_pose(
    PoseType.JOINT
)
if ret == StatusCodeEnum.OK:
    print("获取关节位姿成功 / Get joint pose successful")
else:
    print(
        f"获取关节位姿失败，错误代码 / Get joint pose failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果位姿
# [EN] Print result pose
print(f"位姿类型 / Pose type: {motion_pose.pt}")
print(
    f"轴位置 / Axis position:\n"
    f"J1:{motion_pose.joint.j1}\n"
    f"J2:{motion_pose.joint.j2}\n"
    f"J3:{motion_pose.joint.j3}\n"
    f"J4:{motion_pose.joint.j4}\n"
    f"J5:{motion_pose.joint.j5}\n"
    f"J6:{motion_pose.joint.j6}"
)

# [ZH] 获取当前位姿
# [EN] Get current pose
motion_pose, ret = arm.motion.get_current_pose(
    PoseType.CART, 0, 0
)
if ret == StatusCodeEnum.OK:
    print(
        "获取笛卡尔位姿成功 / Get Cartesian pose successful"
    )
else:
    print(
        f"获取笛卡尔位姿失败，错误代码 / Get Cartesian pose failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 打印结果位姿
# [EN] Print result pose
print(f"位姿类型 / Pose type: {motion_pose.pt}")
print(
    f"坐标位置 / Coordinate position:\n"
    f"X:{motion_pose.cartData.position.x}\n"
    f"Y:{motion_pose.cartData.position.y}\n"
    f"Z:{motion_pose.cartData.position.z}\n"
    f"A:{motion_pose.cartData.position.a}\n"
    f"B:{motion_pose.cartData.position.b}\n"
    f"C:{motion_pose.cartData.position.c}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.9 DH 매개변수 가져오기

메서드 이름	motion.get_DH_param() -> tuple[list[DHparam], StatusCodeEnum]
설명	로봇의 DH 매개변수 가져오기
요청 매개변수	매개변수 없음
반환 값	list(DHparam): List of DH parameters StatusCodeEnum: Function execution result
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): 지원되지 않음

4.2.10 DH 매개변수 설정

메서드 이름	motion.set_DH_param(dh_list : list[DHparam]) -> StatusCodeEnum
설명	로봇의 DH 매개변수 설정
요청 매개변수	dh_list : list(DHparam) DH parameter list
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/DH_param.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: DH参数设置使用示例 / Example of DH parameter setting usage
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 获取DH参数
# [EN] Get DH parameters
res, ret = arm.motion.get_DH_param()
if ret == StatusCodeEnum.OK:
    print("获取DH参数成功 / Get DH parameters successful")
else:
    print(
        f"获取DH参数失败，错误代码 / Get DH parameters failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置DH参数
# [EN] Set DH parameters
ret = arm.motion.set_DH_param(res)
if ret == StatusCodeEnum.OK:
    print("设置DH参数成功 / Set DH parameters successful")
else:
    print(
        f"设置DH参数失败，错误代码 / Set DH parameters failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
for param in res:
    print(
        f"DH参数的ID / DH parameter ID: {param.id}\n"
        f"杆件长度 / Link length: {param.a}\n"
        f"杆件扭角 / Link twist angle: {param.alpha}\n"
        f"关节距离 / Joint distance: {param.d}\n"
        f"关节转角 / Joint angle: {param.offset}"
    )

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
```

```
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.11 축 잠금 상태 가져오기

메서드 이름	**motion.get_drag_set() -> tuple[DragStatus, StatusCodeEnum]**
설명	동작 티칭에만 적용되는 현재 로봇 축 잠금 상태를 가져옵니다.
요청 매개변수	매개변수 없음
반환 값	DragStatus: 축 잠금 상태, True는 축이 이동 가능함을 나타내고, False는 잠겨 있음을 나타냅니다. StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/get_drag_set.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 拖动设置使用示例 / Example of drag Settings usage
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
```

```
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取当前轴锁定状态
# [EN] Get current axis lock status
drag_status, ret = arm.motion.get_drag_set()
if ret == StatusCodeEnum.OK:
    print("获取拖动设置成功 / Get drag set successful")
else:
    print(
        f"获取拖动设置失败，错误代码 / Get drag set failed, error code: {ret.err
msg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"当前X轴拖动状态 / Current X axis drag status: {drag_status.cart_status.
x}\n"
    f"当前Y轴拖动状态 / Current Y axis drag status: {drag_status.cart_status.
y}\n"
    f"当前Z轴拖动状态 / Current Z axis drag status: {drag_status.cart_status.
z}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.12 로봇 축 잠금 상태 설정

메서드 이름	**motion.set_drag_set(drag_status : DragStatus) -> StatusCodeEnum**
설명	티칭 동작에만 적용되는 현재 로봇 축 잠금 상태를 설정합니다.
요청 매개변수	drag_status : DragStatus 축 잠금 상태(기본값 모두 True: 잠금 해제)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/set_drag_set.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 拖动状态设置实例 / Example of dragging status setting
"""

from Agilebot import (
    Arm,
    DragStatus,
    StatusCodeEnum,
    TCSType,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
```

```
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置示教坐标系
# [EN] Set teaching coordinate system
arm.motion.set_TCS(TCSType.BASE)
if ret == StatusCodeEnum.OK:
    print("操作成功 / Operation successful")
else:
    print(
        f"操作失败，错误代码 / Operation failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置要锁定的轴
# [EN] Set axes to be locked
drag_status = DragStatus()
drag_status.cart_status.x = False
drag_status.cart_status.y = False
# [ZH] 设置连续拖动开关
# [EN] Set continuous drag switch
drag_status.is_continuous_drag = True

# [ZH] 设置轴锁定状态
# [EN] Set axis lock status
ret = arm.motion.set_drag_set(drag_status)
if ret == StatusCodeEnum.OK:
    print("设置拖动状态成功 / Set drag status successful")
else:
    print(
        f"设置拖动状态失败，错误代码 / Set drag status failed, error code: {ret.
errmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 打印结果
# [EN] Print result
print(
    f"当前X轴拖动状态 / Current X axis drag status: {drag_status.cart_status.
x}\n"
    f"当前Y轴拖动状态 / Current Y axis drag status: {drag_status.cart_status.
y}\n"
    f"当前Z轴拖动状态 / Current Z axis drag status: {drag_status.cart_status.
z}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.13 로봇 드래그 활성화

메서드 이름	**motion.enable_drag(drag_state : bool) -> StatusCodeEnum**
설명	로봇 끌기 활성화 또는 비활성화
요청 매개변수	drag_state : bool 드래그 모드 스위치(True 드래그 모드 입력, False 드래그 모드 종료)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/enable_drag.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
```

```
Instruction: 拖动示教使用示例 / example of drag teaching usage
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 进入拖动示教
# [EN] Enter drag teaching mode
ret = arm.motion.enable_drag(True)
if ret == StatusCodeEnum.OK:
    print(
        "进入拖动示教成功 / Enter drag teaching successful"
    )
else:
    print(
        f"进入拖动示教失败，错误代码 / Enter drag teaching failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 退出拖动示教
# [EN] Exit drag teaching mode
ret = arm.motion.enable_drag(False)
if ret == StatusCodeEnum.OK:
```

```
    print(
        "退出拖动示教成功 / Exit drag teaching successful"
    )
else:
    print(
        f"退出拖动示教失败，错误代码 / Exit drag teaching failed, error code: {r
et.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.14 실시간 Position 제어 모드 진입

메서드 이름	motion.enter_position_control() -> StatusCodeEnum
설명	실시간 위치 제어 모드로 진입하여 로봇의 정밀한 위치 제어가 가능합니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과
메모	실시간 제어 모드로 진입한 후 UDP를 통해 제어 명령을 전송해야 합니다.
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.15 실시간 Position 제어 모드 종료

메서드 이름	motion.exit_position_control() -> StatusCodeEnum
설명	실시간 위치 제어 모드를 종료하고 기본 로봇 제어 상태로 복귀합니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과
메모	종료 후 로봇은 더 이상 실시간 제어 명령을 수락하지 않습니다.
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/position_control.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 实时位置控制模式使用示例 / Example of the real-time location contro
l mode usage
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 进入实时位置控制模式
# [EN] Enter real-time position control mode
ret = arm.motion.enter_position_control()
if ret == StatusCodeEnum.OK:
    print(
        "进入实时位置控制模式成功 / Enter real-time position control mode succes
sful"
    )
else:
    print(
        f"进入实时位置控制模式失败，错误代码 / Enter real-time position control m
ode failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 在此插入发送UDP数据控制机器人代码
# [EN] Insert UDP data sending code to control robot here

# [ZH] 退出实时位置控制模式
# [EN] Exit real-time position control mode
ret = arm.motion.exit_position_control()
if ret == StatusCodeEnum.OK:
    print(
        "退出实时位置控制模式成功 / Exit real-time position control mode success
ful"
    )
else:
    print(
        f"退出实时位置控制模式失败，错误代码 / Exit real-time position control mo
de failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
```

```
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.16 UDP 피드백 매개변수 설정

메서드 이름	**motion.set_udp_feedback_params(** `flag` **: bool,** `ip` **: str,** `interval` **: int,** `feedback_type` **: int,** `DO_list` **: list[int] = None) -> StatusCodeEnum**
설명	지정된 IP 주소로 데이터를 푸시하기 위한 UDP 피드백 매개변수 구성
요청 매개변수	`flag` : bool UDP 데이터 푸시를 활성화합니다. `ip` : str 수신기 IP 주소; `interval` : int 전송 간격(ms); `feedback_type` : int 피드백 형식(0: XML, 1: JSON, 2: PROTO); `DO_list` : list[int] DO 신호 목록(최대 10개, 선택 사항)
반환 값	StatusCodeEnum: 함수 실행 결과
메모	매개변수 설정은 UDP 데이터 푸시 기능이 활성화된 경우에만 유효합니다.
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

예제 코드

motion/UDP_pose_feedback.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: UDP位置反馈示例 / Example of using UDP to receive robot pose feed
back
"""

import json
import socket

from Agilebot.IR.A.arm import Arm
```

```python
from Agilebot.IR.A.status_code import StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置udp反馈参数
# [EN] Set UDP feedback parameters
udp_local_ip = "10.27.1.225"
udp_feedback_interval = 4
udp_feedback_type = 1  # 0: xml, 1: json, 2: proto
udp_do_list = []
ret = arm.motion.set_udp_feedback_params(
    True,
    udp_local_ip,
    udp_feedback_interval,
    udp_feedback_type,
    udp_do_list,
)
if ret == StatusCodeEnum.OK:
    print(
        "设置UDP反馈参数成功 / Set UDP feedback parameters successful"
    )
else:
    print(
        f"设置UDP反馈参数失败，错误代码 / Set UDP feedback parameters failed, er
ror code: {ret.errmsg}"
    )
```

```
    arm.disconnect()
    exit(1)

# [ZH] 使用UDP连接机器人并接收反馈数据
# [EN] Use UDP to connect to the robot and receive feedback data
udp_feedback_port = 5605  # 按控制器UDP反馈端口配置修改 / Update to match contro
ller UDP feedback port
udp_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_sock.settimeout(2.0)
udp_sock.bind((udp_local_ip, udp_feedback_port))


# [ZH] 接收UDP反馈数据示例
# [EN] Example of receiving UDP feedback data
for _ in range(50):
    try:
        data, addr = udp_sock.recvfrom(2048)
    except socket.timeout:
        print("UDP接收超时 / UDP receive timeout")
        continue
    if udp_feedback_type == 1:
        try:
            payload = json.loads(
                data.decode("utf-8", errors="ignore")
            )
            print(f"UDP反馈 / UDP feedback from {addr}\n")
            print(
                f"关节坐标 / AIPos: {payload.get('AIPos')}\n"
            )
            print(
                f"笛卡尔坐标 / RIst: {payload.get('RIst')}\n"
            )
        except json.JSONDecodeError:
            print(
                f"UDP反馈解析失败 / Failed to parse UDP feedback: {data}"
            )
    else:
        print(
            f"UDP反馈原始数据 / UDP raw feedback from {addr}: {data}"
        )

# [ZH] 断开捷勃特机器人连接
```

```
# [EN] Disconnect from Agilebot robot
udp_sock.close()
ret = arm.motion.set_udp_feedback_params(
    False,
    udp_local_ip,
    udp_feedback_interval,
    udp_feedback_type,
    udp_do_list,
)
if ret == StatusCodeEnum.OK:
    print(
        "设置UDP反馈参数成功 / Set UDP feedback parameters successful"
    )
else:
    print(
        f"设置UDP反馈参数失败，错误代码 / Set UDP feedback parameters failed, er
ror code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

데이터 푸시 설명

이름	필드	설명
RIst: Cartesian Position	엑스	공구 좌표계의 X 방향 값(밀리미터)
	와이	공구 좌표계의 Y 방향 값(밀리미터)
	지	공구 좌표계의 Z 방향 값(밀리미터)
	에이	공구 좌표계에서 X축을 중심으로 한 회전(도)
	비	공구 좌표계에서 Y축을 중심으로 한 회전(도)

이름	필드	설명
	기음	공구 좌표계에서 Z축을 중심으로 한 회전(도)
AIPos: Joint Position	A1-A6	6개 관절의 값(도)
EIPos: Additional Axis Data	EIPos	추가 축 데이터
WristBtnState: Wrist Button State	버튼 상태	1 = 버튼이 눌림, 0 = 버튼이 해제됨
	드래그모델	드래그 버튼 상태
	레코드조인트	교육 기록 버튼 상태
	일시중지이력서	일시정지/Resume 버튼 상태
Digout: DO Output	디그아웃	디지털 출력 상태(DO)
ProgramStatus: Program Status	ProgId	프로그램 ID
	상태	인터프리터 실행 상태: 0 = INTERPRETER_IDLE 1 = INTERPRETER_EXECUTE 2 = INTERPRETER_PAUSED
	XPath	`program_name:line_number` 형식의 프로그램 조각 반환 값
IPOC: Timestamp	IPOC	타임스탬프

4.2.17 로봇 소프트 제한 가져오기

메서드 이름	**motion.get_user_soft_limit() -> tuple[list[list[float]], StatusCodeEnum]**
설명	로봇의 현재 소프트 제한을 가져옵니다.
요청 매개변수	없음

메서드 이름	**motion.get_user_soft_limit() -> tuple[list[list[float]], StatusCodeEnum]**
반환 값	List(List(float)): Robot soft limits information, the first layer of the list represents each axis, and the second layer represents the lower and upper limits of each axis StatusCodeEnum: Function execution result
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_user_soft_limit.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 用户软限位设置获取示例 / Example of obtaining the user's soft limi
t setting
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 获取当前机器人软限位信息
# [EN] Get current robot soft limit information
res, ret = arm.motion.get_user_soft_limit()
if ret == StatusCodeEnum.OK:
    print(
        "获取用户软限位成功 / Get user soft limit successful"
    )
else:
    print(
        f"获取用户软限位失败，错误代码 / Get user soft limit failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"当前机器人软限位信息 / Current robot soft limit information: {res}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18 페이로드 관련 인터페이스

4.2.18.1 현재 활성화된 페이로드 ID 가져오기

메서드 이름	motion.payload.get_current_payload() -> tuple[int, StatusCodeEnum]
설명	현재 활성화된 페이로드 ID 가져오기
요청 매개변수	없음

메서드 이름	motion.payload.get_current_payload() -> tuple[int, StatusCodeEnum]
반환 값	int: 페이로드 ID StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_current_payload.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 获取当前负载示例 / Example of get the current load
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取当前激活负载
# [EN] Get current active payload
current_payload_id, ret = (
```

```
    arm.motion.payload.get_current_payload()
)
if ret == StatusCodeEnum.OK:
    print(
        "获取当前负载成功 / Get current payload successful"
    )
else:
    print(
        f"获取当前负载失败，错误代码 / Get current payload failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"当前激活负载ID为 / Current active payload ID: {current_payload_id}"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.2 ID로 페이로드 정보 얻기

메서드 이름	**motion.payload.get_payload_by_id(`payload_id` : int) -> tuple[PayloadInfo, StatusCodeEnum]**
설명	지정된 ID로 페이로드 정보 가져오기
요청 매개변수	`payload_id` : 정수 페이로드 ID
반환 값	PayloadInfo: 페이로드 정보 StatusCodeEnum: 함수 실행 결과

메서드 이름	motion.payload.get_payload_by_id(payload_id : int) -> tuple[PayloadInfo, StatusCodeEnum]
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_payload_by_id.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 根据ID获取负载参数示例 / Example of obtaining load parameters base
d on ID
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取负载
# [EN] Get payload
res, ret = arm.motion.payload.get_payload_by_id(6)
if ret == StatusCodeEnum.OK:
```

```
    print("获取负载成功 / Get payload successful")
else:
    print(
        f"获取负载失败，错误代码 / Get payload failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"负载ID / Payload ID: {res.id}\n"
    f"负载质量 / Payload mass: {res.weight}\n"
    f"负载注释 / Payload comment: {res.comment}\n"
    f"负载质心 / Payload mass center: {res.mass_center.x}, {res.mass_center.
y}, {res.mass_center.z}\n"
    f"负载转动惯量 / Payload inertia moment: {res.inertia_moment.lx}, {res.ine
rtia_moment.ly}, {res.inertia_moment.lz}\n"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.3 지정된 페이로드 활성화

메서드 이름	motion.payload.set_current_payload(payload_id : int) -> StatusCodeEnum
설명	지정된 ID로 페이로드 활성화
요청 매개변수	payload_id : 정수 페이로드 ID
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/set_current_payload.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 当前负载设置示例 / Example of the current load settings
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 指定编号激活负载
# [EN] Activate payload by specified ID
ret = arm.motion.payload.set_current_payload(1)
if ret == StatusCodeEnum.OK:
    print(
        "设置当前负载成功 / Set current payload successful"
    )
else:
    print(
        f"设置当前负载失败，错误代码 / Set current payload failed, error code:
{ret.errmsg}"
```

```
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.4 사용자 정의 페이로드 추가

메서드 이름	**motion.payload.add_payload(payload_info : PayloadInfo) -> StatusCodeEnum**
설명	로봇 컨트롤러에 사용자 정의 페이로드 추가
요청 매개변수	payload_info : PayloadInfo 사용자 정의 페이로드 정보
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/add_payload.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 添加负载使用示例 / Example of Add load usage
"""

from Agilebot import Arm, PayloadInfo, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
```

```
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化负载
# [EN] Initialize payload
new_payload = PayloadInfo()
new_payload.id = 6
new_payload.comment = "Test"
new_payload.weight = 5
new_payload.mass_center.x = -151
new_payload.mass_center.y = 1.0
new_payload.mass_center.z = 75
new_payload.inertia_moment.lx = 0.11
new_payload.inertia_moment.ly = 0.61
new_payload.inertia_moment.lz = 0.54

# [ZH] 添加负载
# [EN] Add payload
ret = arm.motion.payload.add_payload(new_payload)
if ret == StatusCodeEnum.OK:
    print("添加负载成功 / Add payload successful")
else:
    print(
        f"添加负载失败，错误代码 / Add payload failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.5 페이로드 삭제

메서드 이름	**motion.payload.delete_payload(** payload_id **: int) -> StatusCodeEnum**
설명	컨트롤러에서 사용자 정의 페이로드 삭제
요청 매개변수	payload_id : 정수 페이로드 ID
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	참고: 현재 활성화된 페이로드는 삭제할 수 없습니다. 활성화된 페이로드를 삭제하려면 먼저 다른 페이로드를 활성화한 후 현재 페이로드를 삭제하세요.

예제 코드

motion/delete_payload.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 删除负载使用示例 / Example of delete the load usage
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
```

```
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 删除指定ID负载
# [EN] Delete payload with specified ID
ret = arm.motion.payload.delete_payload(6)
if ret == StatusCodeEnum.OK:
    print("删除负载成功 / Delete payload successful")
else:
    print(
        f"删除负载失败，错误代码 / Delete payload failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print("删除负载6成功 / Delete payload 6 successful")

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.6 페이로드 업데이트

메서드 이름	**motion.payload.update_payload(payload_info : PayloadInfo) -> StatusCodeEnum**
설명	기존 사용자 정의 페이로드 정보 업데이트
요청 매개변수	payload_info : PayloadInfo 사용자 정의 업데이트 페이로드 정보
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/update_payload.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 更新负载使用示例 / Example of updating the load
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 获取负载
# [EN] Get payload
payload_info, ret_code = (
    arm.motion.payload.get_payload_by_id(6)
)
payload_info.comment = "Test"
payload_info.weight = 10
payload_info.mass_center.x = -100
payload_info.mass_center.y = 10
payload_info.mass_center.z = 10
payload_info.inertia_moment.lx = 10
payload_info.inertia_moment.ly = 10
payload_info.inertia_moment.lz = 10

# [ZH] 更新负载
# [EN] Update payload
ret = arm.motion.payload.update_payload(payload_info)
if ret == StatusCodeEnum.OK:
    print("更新负载成功 / Update payload successful")
else:
    print(
        f"更新负载失败，错误代码 / Update payload failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"负载ID / Payload ID: {payload_info.id}\n"
    f"负载质量 / Payload mass: {payload_info.weight}\n"
    f"负载质心 / Payload mass center: {payload_info.mass_center.x}, {payload_
info.mass_center.y}, {payload_info.mass_center.z}\n"
    f"负载转动惯量 / Payload inertia moment: {payload_info.inertia_moment.lx},
{payload_info.inertia_moment.ly}, {payload_info.inertia_moment.lz}\n"
    f"负载注释 / Payload comment: {payload_info.comment}\n"
)

# [ZH] 断开捷勃特机器人连接
```

```
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.7 모든 페이로드 가져오기

메서드 이름	motion.payload.get_all_payload() -> tuple[list, StatusCodeEnum]
설명	모든 페이로드 정보 가져오기
요청 매개변수	없음
반환 값	list : 모든 페이로드 정보 목록 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

motion/get_all_payload.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 所有负载获取示例 / Example of all load acquisition
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化Arm类
# [EN] Initialize Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to controller
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
```

```
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取所有负载
# [EN] Get all payloads
res, ret = arm.motion.payload.get_all_payload()
if ret == StatusCodeEnum.OK:
    print("获取所有负载成功 / Get all payloads successful")
else:
    print(
        f"获取所有负载失败，错误代码 / Get all payloads failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
for payload in res:
    print(
        f"负载ID / Payload ID: {payload[0]}\n负载注释 / Payload comment: {payl
oad[1]}\n"
    )

# [ZH] 结束后断开机器人连接
# [EN] Disconnect from robot after completion
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.2.18.8 축 3 수평 레벨 확인

메서드 이름	**motion.payload.check_axis_three_horizontal() -> tuple[float, StatusCodeEnum]**
설명	축 3이 수평인지 확인
요청 매개변수	없음
반환 값	float: 축 3의 수평 각도(도) StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음
메모	페이로드 식별을 진행하려면 수평 각도가 -1과 1 사이여야 합니다.

4.2.18.9 페이로드 식별 상태 가져오기

메서드 이름	**motion.payload.get_payload_identify_state() -> tuple[PayloadIdentifyState, StatusCodeEnum]**
설명	페이로드 식별 상태 가져오기
요청 매개변수	없음
반환 값	PayloadIdentifyState 페이로드 식별 상태 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.18.10 페이로드 식별 시작

메서드 이름	**motion.payload.start_payload_identify(** `weight` **: float,** `angle` **: float) -> StatusCodeEnum**
설명	페이로드 식별 프로세스 시작
요청 매개변수	`weight` : 부동 페이로드 가중치(알 수 없는 경우 -1 사용); `angle` : 플로트 축 6 허용 회전 각도(30~90도)

메서드 이름	**motion.payload.start_payload_identify(`weight` : float, `angle` : float) -> StatusCodeEnum**
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음
메모	로봇은 페이로드 식별을 시작하기 전에 페이로드 식별 상태로 들어가야 합니다.

4.2.18.11 페이로드 식별 결과 얻기

메서드 이름	**motion.payload.payload_identify_result() -> tuple[PayloadInfo, StatusCodeEnum]**
설명	페이로드 식별 결과 얻기
요청 매개변수	없음
반환 값	PayloadInfo: 페이로드 식별 결과 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.18.12 간섭 확인 시작

메서드 이름	**motion.payload.interference_check_for_payload_identify(`weight` : float, `angle` : float) -> StatusCodeEnum**
설명	페이로드 식별을 위한 간섭 검사 시작
요청 매개변수	`weight` : 부동 페이로드 무게; `angle` : 플로트 6축 회전 각도(30~90도)
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.18.13 페이로드 식별 상태 입력

메서드 이름	**motion.payload.payload_identify_start() -> StatusCodeEnum**
설명	페이로드 식별 상태 입력
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.18.14 페이로드 식별 상태 종료

메서드 이름	**motion.payload.payload_identify_done() -> StatusCodeEnum**
설명	페이로드 식별 상태 종료
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

4.2.18.15 페이로드 식별 프로세스 완료

메서드 이름	**motion.payload.payload_identify(** `weight` **: float,** `angle` **: float) -> tuple[PayloadInfo, StatusCodeEnum]**
설명	위에 언급된 모든 인터페이스를 포함하여 페이로드 식별 프로세스를 완료합니다. 특별한 요구 사항이 없는 일반 페이로드 식별의 경우 이 인터페이스로 충분합니다.
요청 매개변수	`weight` : 부동 페이로드 가중치(알 수 없는 경우 -1 사용); `angle` : 플로트 6축 회전 각도(30~90도)
반환 값	PayloadInfo: 페이로드 식별 결과 StatusCodeEnum: 함수 실행 결과
호환 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): 지원되지 않음

메서드 이름	motion.payload.payload_identify(weight : float, angle : float) -> tuple[PayloadInfo, StatusCodeEnum]
메모	반환된 페이로드는 로봇에 추가되거나 로봇의 기존 페이로드에 기록될 수 있습니다. 전체 프로세스 단계는 다음과 같습니다. 1. 지정된 수평 위치로 이동하여 수평인지 확인 2. 페이로드 식별 상태 입력 3. 페이로드 식별 시작 4. 페이로드 식별 결과 가져오기 5. 페이로드 식별 상태 종료

예제 코드

motion/payload_identify.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 负载识别使用示例 / Example of load identification usage
"""

import time

from Agilebot import (
    Arm,
    MotionPose,
    PoseType,
    ServoStatusEnum,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
```

```
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

motion_pose = MotionPose()
motion_pose.pt = PoseType.JOINT

motion_pose.joint.j1 = 0
motion_pose.joint.j2 = 0
motion_pose.joint.j3 = 0
motion_pose.joint.j4 = 0
motion_pose.joint.j5 = 0
motion_pose.joint.j6 = 0

# [ZH] 运动到指定点
# [EN] Move to specified position
code = arm.motion.move_joint(motion_pose)
if ret == StatusCodeEnum.OK:
    print(
        "运动到指定点成功 / Move to specified position successful"
    )
else:
    print(
        f"运动到指定点失败，错误代码 / Move to specified position failed, error
code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

while True:
    # [ZH] 获取伺服状态
    # [EN] Get servo status
    state, ret = arm.get_servo_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取伺服状态成功 / Get servo status successful"
        )
    else:
```

```
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

    if state == ServoStatusEnum.SERVO_IDLE:
        break
    else:
        time.sleep(1)

# [ZH] 开始负载测定并获取结果
# [EN] Start payload identification and get result
res, ret = arm.motion.payload.payload_identify(-1, 90)
if ret == StatusCodeEnum.OK:
    print(
        "负载识别成功 / Payload identification successful"
    )
else:
    print(
        f"负载识别失败，错误代码 / Payload identification failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.3 로봇 프로그램 운영 수업

개요

Execution 클래스는 다음 핵심 기능을 담당하는 로봇 프로그램 및 모션 작업에 대한 통합 예약 인터페이스를 제공합니다.

- Start/stop/pause/resume 티치 펜던트 프로그램
- 동시에 실행되는 작업 목록 관리
- BAS 스크립트 및 기타 사용자 정의 워크플로우 실행

Arm과 Motion의 연결과 모션 기능을 결합한 Execution은 호스트 측에서 컨트롤러의 프로그램 흐름을 트리거하고 제어하는 역할을 담당합니다.

4.3.1 지정된 프로그램 실행

메서드 이름	**execution.start(`program_name` : str) -> StatusCodeEnum**
설명	지정된 프로그램을 실행합니다.
요청 매개변수	`program_name` : str 실행할 프로그램의 이름입니다.
반환 값	StatusCodeEnum: 함수 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.3.2 현재 실행 중인 프로그램 중지

메서드 이름	**execution.stop(`program_name` : str = '') -> StatusCodeEnum**
설명	현재 실행 중인 프로그램을 중지하거나 로봇의 현재 동작을 중지합니다.

메서드 이름	**execution.stop(program_name : str = '') -> StatusCodeEnum**
요청 매개변수	program_name : str 프로그램 이름(기본적으로 빈 문자열, 현재 실행 중인 프로그램 또는 모션 명령을 중지함).
반환 값	StatusCodeEnum: 함수 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.3.3 실행 중인 모든 프로그램의 세부 정보 반환

메서드 이름	**execution.all_running_programs() -> tuple[list, StatusCodeEnum]**
설명	스레드 ID, 프로그램 이름, xpath, 프로그램 상태 및 프로그램 유형을 포함하여 실행 중인 모든 프로그램에 대한 자세한 정보를 반환합니다.
요청 매개변수	없음
반환 값	list: 실행 중인 프로그램 세부 정보 목록으로, 각 요소에는 thread_id , program_name , xpath , program_status , program_type 등이 포함됩니다. StatusCodeEnum: 함수 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.3.4 프로그램 실행 일시 중지

메서드 이름	**execution.pause(program_name : str = '') -> StatusCodeEnum**
설명	현재 실행 중인 프로그램이나 로봇의 현재 동작을 일시 정지합니다.
요청 매개변수	program_name : str 프로그램 이름(기본 빈 문자열, 현재 실행 중인 프로그램 또는 모션 명령을 일시 중지함).

메서드 이름	**execution.pause(program_name : str = '') -> StatusCodeEnum**
반환 값	StatusCodeEnum: 함수 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.3.5 프로그램 실행 재개

메서드 이름	**execution.resume(program_name : str = '') -> StatusCodeEnum**
설명	일시정지된 상태의 프로그램을 계속 실행합니다.
요청 매개변수	program_name : str 프로그램 이름(기본적으로 빈 문자열, 현재 일시 중지된 프로그램 또는 모션 명령을 다시 시작함).
반환 값	StatusCodeEnum: 함수 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

execution/program_execution.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 自定义程序使用示例 / Example of custom program usage
"""

import time

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
```

```python
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connection successful")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

program_name = "test"

# [ZH] 执行程序
# [EN] Execute program
ret = arm.execution.start(program_name)
if ret == StatusCodeEnum.OK:
    print("程序启动成功 / Program start successful")
else:
    print(
        f"程序启动失败，错误代码 / Program start failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取所有正在运行的程序
# [EN] Get all running programs
programs_list, ret = arm.execution.all_running_programs()
if ret == StatusCodeEnum.OK:
    print(
        "获取运行程序列表成功 / Get running programs list successful"
    )
else:
    print(
        f"获取运行程序列表失败，错误代码 / Get running programs list failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)
for program in programs_list:
    print(f"正在运行的程序名：{program}")

time.sleep(2)

# [ZH] 暂停程序
# [EN] Pause program
ret = arm.execution.pause(program_name)
if ret == StatusCodeEnum.OK:
    print("程序暂停成功 / Program pause successful")
else:
    print(
        f"程序暂停失败，错误代码 / Program pause failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

time.sleep(2)

# [ZH] 恢复程序
# [EN] Resume program
ret = arm.execution.resume(program_name)
if ret == StatusCodeEnum.OK:
    print("程序恢复成功 / Program resume successful")
else:
    print(
        f"程序恢复失败，错误代码 / Program resume failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)

time.sleep(2)

# [ZH] 停止程序
# [EN] Stop program
ret = arm.execution.stop(program_name)
if ret == StatusCodeEnum.OK:
    print("程序停止成功 / Program stop successful")
else:
```

```
    print(
        f"程序停止失败，错误代码 / Program stop failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.3.6 BAS 스크립트 프로그램 실행

메서드 이름	**execution.execute_bas_script(bas_script : BasScript |목록[str]) -> StatusCodeEnum**
설명	사용자 정의 BAS 스크립트 프로그램을 실행합니다.
요청 매개변수	bas_script : BasScript 또는 list[str] 사용자 정의 BAS 스크립트 프로그램입니다.
반환 값	StatusCodeEnum: 함수 실행 결과입니다.
메모	BAS 스크립트 프로그램의 일시 중지, 재개 및 중지 방법은 일반 프로그램과 동일합니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): v7.6.0.0+

execution/bas_script_execution.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: Bas脚本创建和使用示例 / Example of Bas script creation and usage
"""
```

```
from Agilebot import *

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 创建BasScript对象
# [EN] Create BasScript object
bs = BasScript(name="bas_test")
ret = bs.assign_value(AssignType.R, 1, OtherType.VALUE, 10)
ret = bs.move_joint(
    pose_type=MovePoseType.PR,
    pose_index=1,
    speed_type=SpeedType.VALUE,
    speed_value=100,
    smooth_type=SmoothType.SMOOTH_DISTANCE,
    smooth_distance=200.5,
)
ret = bs.wait_time(ValueType.VALUE, 10)
if ret == StatusCodeEnum.OK:
    print(
        "创建BasScript对象成功 / Create BasScript object successfully"
    )
else:
    print(
        f"创建BasScript对象失败，错误代码 / Create BasScript object failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 执行脚本
# [EN] Execute script
ret = arm.execution.execute_bas_script(bs)
if ret == StatusCodeEnum.OK:
    print("执行脚本成功 / Execute script successfully")
else:
    print(
        f"执行脚本失败，错误代码 / Execute script failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)


# [ZH] 结束后断开机器人连接
# [EN] Disconnect from the robot after completion
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.4 프로그램 포즈 읽기 및 쓰기

개요

ProgramPose 인터페이스는 로봇 티치 펜던트 프로그램에서 포즈 포인트를 읽고, 쓰고, 변환하는 데 사용됩니다. `program_pose` 모듈을 통해 다음을 수행할 수 있습니다.

- 특정 프로그램 및 포인트 인덱스 찾기
- CRUD 작업 수행
- 데카르트 표현과 결합 표현 간 전환

이를 통해 호스트 PC 시나리오에서 프로그램 포인트의 일괄 유지 관리 또는 오프라인 편집이 용이해집니다.

4.4.1 프로그램에서 지정된 포즈의 값 가져오기

메서드 이름	**program_pose.read(`program_name` : str, `index` : int, `program_type` : str = USER_PROGRAM) -> tuple[ProgramPose, StatusCodeEnum]**
설명	프로그램에서 지정된 인덱스를 가진 포즈의 값을 가져옵니다.
요청 매개변수	`program_name` : str 프로그램 이름. `index` : int 포즈 인덱스. `program_type` : str 프로그램 유형(기본값 USER_PROGRAM).
반환 값	ProgramPose: 포즈 정보. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.4.2 프로그램에서 지정된 포즈의 값 수정

메서드 이름	**program_pose.write(** `program_name` **: str,** `index` **: int,** `value` **: ProgramPose,** `program_type` **: str = USER_PROGRAM) -> StatusCodeEnum**
설명	지정된 프로그램에서 지정된 인덱스로 포즈의 값을 수정합니다.
요청 매개변수	`program_name` : str 프로그램 이름. `index` : int 포즈 인덱스. `value` : ProgramPose 업데이트할 포즈 값입니다. `program_type` : str 프로그램 유형(기본값 USER_PROGRAM).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.4.3 지정된 프로그램에서 모든 포즈 가져오기

메서드 이름	**program_pose.read_all_poses(** `program_name` **: str,** `program_type` **: str = USER_PROGRAM) -> tuple[list[ProgramPose], StatusCodeEnum]**
설명	지정된 프로그램에서 모든 포즈 정보를 가져옵니다.
요청 매개변수	`program_name` : str 프로그램 이름. `program_type` : str 프로그램 유형(기본값 USER_PROGRAM).
반환 값	list[ProgramPose]: 포즈 정보 목록입니다. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.4.4 프로그램에 포즈 항목 추가

메서드 이름	**program_pose.add(** `program_name` **: str,** `index` **: int,** `value` **: ProgramPose,** `program_type` **: str = USER_PROGRAM) -> StatusCodeEnum**
설명	지정된 프로그램의 지정된 인덱스에 새 포즈를 추가합니다. 인덱스가 이미 존재하는 경우 오류를 반환합니다.
요청 매개변수	`program_name` : str 프로그램 이름. `index` : int 포즈 인덱스. `value` : ProgramPose 추가할 포즈 값입니다. `program_type` : str 프로그램 유형(기본값 USER_PROGRAM).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.4.5 프로그램에 하나 이상의 로봇 자세 값 쓰기

메서드 이름	**program_pose.write_poses(** `program_name` **: str,** `poses_to_update` **: list[ProgramPose]) -> StatusCodeEnum**
설명	프로그램에 하나 이상의 로봇 포즈 값을 작성합니다. 참고: 기존 포인트만 업데이트할 수 있습니다. new points cannot be added.
요청 매개변수	`program_name` : str 프로그램 이름. `poses_to_update` : list[ProgramPose] 업데이트될 포즈 목록입니다.
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.4.6 로봇 프로그램 포즈 유형 변환

메서드 이름	**ProgramPoses.convert_pose(`pose` : ProgramPose, `from_type` : PoseType, `to_type` : PoseType) -> tuple[ProgramPose, StatusCodeEnum]**
설명	프로그램에서 관절 좌표와 데카르트 공간 좌표 사이의 로봇 포즈 값을 변환합니다.
요청 매개변수	`pose` : ProgramPose 변환할 포즈 값입니다. `from_type` : PoseType 변환 전 타입입니다. `to_type` : PoseType 변환 후 원하는 유형입니다.
반환 값	ProgramPose: 포즈 정보. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

program_pose.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 机器人位姿使用示例 / Example of robot pose usage
"""

from Agilebot import Arm, PoseType, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
```

```
    arm.disconnect()
    exit(1)

program_name = "test_prog"

# [ZH] 读取所有位姿
# [EN] Read all poses
poses, ret = arm.program_pose.read_all_poses(program_name)
if ret == StatusCodeEnum.OK:
    print("读取所有位姿成功 / Read all poses successfully")
    # [ZH] 打印位姿信息
    # [EN] Print pose information
    for p in poses:
        print(
            f"位姿ID / Pose ID：{p.id}\n位姿名称 / Pose name：{p.name}"
        )
else:
    print(
        f"读取所有位姿失败，错误代码 / Read all poses failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取单个位姿
# [EN] Read a single pose
pose, ret = arm.program_pose.read(program_name, 1)
if ret == StatusCodeEnum.OK:
    print(
        "读取单个位姿成功 / Read single pose successfully"
    )
    # [ZH] 打印位姿信息
    # [EN] Print pose information
    print(
        f"位姿ID / Pose ID：{pose.id}\n"
        f"位姿名称 / Pose name：{pose.name}\n"
        f"位姿类型 / Pose type：{pose.poseData.pt}\n"
        f"X：{pose.poseData.cartData.baseCart.position.x}\n"
        f"Y：{pose.poseData.cartData.baseCart.position.y}\n"
        f"Z：{pose.poseData.cartData.baseCart.position.z}\n"
        f"J1：{pose.poseData.joint.j1}\n"
        f"J2：{pose.poseData.joint.j2}\n"
```

```
            f"J3：{pose.poseData.joint.j3}\n"
        )
else:
        print(
            f"读取单个位姿失败，错误代码 / Read single pose failed, error code: {re
t.errmsg}"
        )
        arm.disconnect()
        exit(1)

# [ZH] 修改位姿
# [EN] Modify pose
pose.comment = "SDK_TEST_COMMENT"
ret = arm.program_pose.write(program_name, 1, pose)
if ret == StatusCodeEnum.OK:
    print("修改位姿成功 / Modify pose successfully")
else:
    print(
        f"修改位姿失败，错误代码 / Modify pose failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 转换位姿
# [EN] Convert pose
converted_pose, ret = arm.program_pose.convert_pose(
    pose, PoseType.CART, PoseType.JOINT
)
if ret == StatusCodeEnum.OK:
    print("转换位姿成功 / Convert pose successfully")
else:
    print(
        f"转换位姿失败，错误代码 / Convert pose failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印位姿信息
# [EN] Print pose information
print(
```

```
    f"位姿ID / Pose ID：{converted_pose.id}\n"
    f"位姿名称 / Pose name：{converted_pose.name}\n"
    f"位姿类型 / Pose type：{converted_pose.poseData.pt}\n"
    f"X：{converted_pose.poseData.cartData.baseCart.position.x}\n"
    f"Y：{converted_pose.poseData.cartData.baseCart.position.y}\n"
    f"Z：{converted_pose.poseData.cartData.baseCart.position.z}\n"
    f"J1：{converted_pose.poseData.joint.j1}\n"
    f"J2：{converted_pose.poseData.joint.j2}\n"
    f"J3：{converted_pose.poseData.joint.j3}\n"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.5 IO 신호

개요

신호 모듈은 다음 핵심 기능을 캡슐화하는 컨트롤러의 I/O,에 대한 통합 read/write 인터페이스를 제공합니다.

- Digital/analog 입출력 제어
- 다중 채널 일괄 작업
- 시간에 따른 트리거링 기능

`signals` 를 통해 다음을 수행할 수 있습니다.

- 현재 신호 상태 쿼리
- 일괄 쓰기 DO/RO/GO 포트
- 지정된 간격으로 펄스 생성

이를 통해 외부 그리퍼, 센서 또는 생산 라인 장비와의 조정이 가능합니다.

4.5.1 지정된 유형 및 포트의 IO 값 읽기

메서드 이름	**signals.read(`signal_type` : SignalType, `index` : int) -> tuple[float, StatusCodeEnum]**
설명	지정된 유형 및 포트의 IO 값을 읽습니다 (DI/DO/UI/UO/RI/RO/GI/GO/TAI/TDI/TDO/AI/AO 지원).
요청 매개변수	`signal_type` : SignalType IO 유형. `index` : int IO 인덱스(1부터 시작).
반환 값	부동 소수점: IO 값. DI/DO/RI/RO/TAI/TDI/TDO/AI/AO는 0 또는 1을 반환하고 GI/GO는 정수 값을 반환합니다(Off의 경우 음수). StatusCodeEnum: 함수 실행 결과.

메서드 이름	**signals.read(`signal_type` : SignalType, `index` : int) -> tuple[float, StatusCodeEnum]**
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	UI/UO는 읽기만 가능하고 쓰기는 불가능합니다.

4.5.2 지정된 유형 및 포트에 IO 값 쓰기

메서드 이름	**signals.write(`signal_type` : SignalType, `index` : int, `value` : float) -> StatusCodeEnum**
설명	지정된 유형 및 포트의 IO 값을 씁니다. 현재는 DO/RO/GO/TDO/AO만 지원됩니다.
요청 매개변수	`signal_type` : SignalType IO 유형. `index` : int IO 인덱스(1부터 시작). `value` : 부동 IO 값(DO/RO/TDO에서는 0/1;를 허용하고 GO에서는 정수를 기대하며 AO에서는 부동 아날로그를 허용합니다).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

signals/signals.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 单信号IO读写示例 / Example of single-signal I/O reading and writi
ng
"""

from Agilebot import (
    Arm,
```

```
    SignalType,
    SignalValue,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取IO
# [EN] Read IO
do_value, ret = arm.signals.read(SignalType.DO, 1)
if ret == StatusCodeEnum.OK:
    print("读取IO成功 / Read IO successfully")
    print(f"DO 1 状态 / DO 1 status：{do_value}")
else:
    print(
        f"读取IO失败，错误代码 / Read IO failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 写入IO
# [EN] Write IO
ret = arm.signals.write(SignalType.DO, 1, SignalValue.ON)
if ret == StatusCodeEnum.OK:
    print("写入IO成功 / Write IO successfully")
else:
    print(
        f"写入IO失败，错误代码 / Write IO failed, error code: {ret.errmsg}"
```

```
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.5.3 일괄 읽기 DO(디지털 출력) 포트 값

메서드 이름	**signals.multi_read(signal_type : SignalType, io_list : list) -> tuple[list, StatusCodeEnum]**
설명	DO(디지털 출력) 포트 값을 일괄적으로 읽습니다.
요청 매개변수	signal_type : SignalType IO 유형(DO만 해당). io_list : 포트 번호를 나열합니다(비워둘 수 없음).
반환 값	목록: [port1, state1, port2, state2, ...] 형식의 컨트롤러 응답 StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	UI/UO는 읽기만 가능하고 쓰기는 불가능합니다.

4.5.4 일괄 쓰기 DO(디지털 출력) 신호

메서드 이름	**signals.multi_write(signal_type : SignalType, io_list : list) -> StatusCodeEnum**
설명	DO(디지털 출력) 신호를 일괄 쓰기합니다.

메서드 이름	**signals.multi_write(** `signal_type` **: SignalType,** `io_list` **: list) -> StatusCodeEnum**
요청 매개변수	`signal_type` : SignalType IO 유형(DO만 해당). `io_list` : Port/value 쌍을 나열합니다(예: [port1, state1, port2, state2]; 길이가 짝수이고 > 0).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

signals/multi_read_write.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 多信号IO读写示例 / Example of multi-signal I/O reading and writin
g
"""

from Agilebot import (
    Arm,
    SignalType,
    SignalValue,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
```

```
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取IO
# [EN] Read IO
do_value, ret = arm.signals.multi_read(
    SignalType.DO, [1, 2]
)
if ret == StatusCodeEnum.OK:
    print("读取IO成功 / Read IO successfully")
    print(f"DO 1 状态 / DO 1 status：{do_value}")
else:
    print(
        f"读取IO失败，错误代码 / Read IO failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 写入IO
# [EN] Write IO
ret = arm.signals.multi_write(
    SignalType.DO, [1, SignalValue.ON, 2, SignalValue.ON]
)
if ret == StatusCodeEnum.OK:
    print("写入IO成功 / Write IO successfully")
else:
    print(
        f"写入IO失败，错误代码 / Write IO failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.5.5 시간 간격으로 IO 채널 트리거

메서드 이름	**signals.trigger_io_with_intervals(in_port : int = -1, intervals : list, out_ports : list, pulse_duration : int) -> StatusCodeEnum**
설명	일련의 시간 간격에 따라 여러 출력 포트를 트리거하고 선택적으로 시작하기 전에 DI 포트의 상승 에지를 기다립니다.
요청 매개변수	in_port : int 입력 포트(기본값 -1, 입력 트리거 없음). intervals : 시간 간격(ms)을 나열합니다. out_ports : 출력 포트를 나열합니다. pulse_duration : int 펄스 지속 시간(ms).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.6 등록정보

개요

레지스터 모듈은 호스트 컴퓨터가 컨트롤러 레지스터에서 읽고 쓸 수 있는 통합 게이트웨이를 제공하여 여러 레지스터 유형에 대한 작업을 지원합니다.

핵심 기능

- 숫자 레지스터 읽기 및 쓰기 지원(R)
- 모션 레지스터 읽기 및 쓰기 지원(MR)
- 문자열 레지스터 읽기 및 쓰기 지원(SR)
- 포즈 레지스터 읽기 및 쓰기 지원(PR)
- Modbus 레지스터 읽기 및 쓰기 지원(MH 보유 레지스터, MI 입력 레지스터)

사용 사례

- 프로그램 런타임 중 매개변수 전달
- 로봇 상태 정보 동기화
- 외부 시스템과 구성 데이터 공유
- 로봇과 외부 장치 간의 대화형 제어 구현

4.6.1 R 숫자 레지스터 작업

4.6.1.1 R 레지스터 값 읽기

메서드 이름	register.read_R(index : int, with_meta : bool = False) -> tuple[float |등록, StatusCodeEnum]
설명	R 숫자 레지스터의 값을 읽습니다.

메서드 이름	**register.read_R(index : int, with_meta : bool = False) -> tuple[float |등록, StatusCodeEnum]**
요청 매개변수	index : int R 읽을 레지스터 번호 with_meta : bool 레지스터 메타데이터 반환 여부(name/comment)
반환 값	float: R 레지스터 숫자값(with_meta=False) 레지스터: 레지스터 객체(with_meta=True) StatusCodeEnum: 연산 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.1.2 R 레지스터 값 쓰기

메서드 이름	**register.write_R(index : int, value : float) -> StatusCodeEnum** **register.write_R(register : Register) -> StatusCodeEnum**
설명	R 숫자 레지스터의 값을 씁니다.
요청 매개변수	index : 쓸 int R 레지스터 번호 value : 쓸 R 레지스터 숫자 값 register : 레지스터 객체 (id/name/comment/value)
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.1.3 R 레지스터 삭제

메서드 이름	**register.delete_R(index : int) -> StatusCodeEnum**
설명	지정된 R 숫자 레지스터를 삭제합니다.
요청 매개변수	index : 삭제할 int R 레지스터 번호
반환 값	StatusCodeEnum: 작업 실행 결과 삭제
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

registers/R.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: R寄存器读写示例 / Example of reading and writing the R register
"""

from Agilebot import Arm, Register, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 添加R寄存器
# [EN] Add R register
ret = arm.register.write_R(5, 8.6)
if ret == StatusCodeEnum.OK:
    print("写入R寄存器成功 / Write R register successful")
else:
    print(
        f"写入R寄存器失败，错误代码 / Write R register failed, error code: {ret.
errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 读取R寄存器
# [EN] Read R register
res, ret = arm.register.read_R(5)
if ret == StatusCodeEnum.OK:
    print("读取R寄存器成功 / Read R register successful")
else:
    print(
        f"读取R寄存器失败，错误代码 / Read R register failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print the result
print(f"R寄存器值 / R register value: {res}")

# [ZH] 使用Register对象写入R寄存器
# [EN] Write R register with Register object
reg = Register()
reg.id = 5
reg.value = 8.6
reg.name = "R5"
reg.comment = "example"
ret = arm.register.write_R(reg)
if ret == StatusCodeEnum.OK:
    print(
        "使用Register写入R寄存器成功 / Write R register with Register successfu
l"
    )
else:
    print(
        f"使用Register写入R寄存器失败，错误代码 / Write R register with Register
failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取R寄存器元信息
# [EN] Read R register meta info
```

```
reg_meta, ret = arm.register.read_R(5, with_meta=True)
if ret == StatusCodeEnum.OK:
    print(
        "读取R寄存器元信息成功 / Read R register meta successful"
    )
    print(
        f"名称 / Name: {reg_meta.name}, 注释 / Comment: {reg_meta.comment}"
    )
else:
    print(
        f"读取R寄存器元信息失败，错误代码 / Read R register meta failed, error co
de: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 删除R寄存器
# [EN] Delete R register
ret = arm.register.delete_R(5)
if ret == StatusCodeEnum.OK:
    print("删除R寄存器成功 / Delete R register successful")
else:
    print(
        f"删除R寄存器失败，错误代码 / Delete R register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.6.2 MR 모션 레지스터 작업

4.6.2.1 MR 레지스터 값 읽기

메서드 이름	**register.read_MR(index : int, with_meta : bool = False) -> tuple[int |모션 레지스터, StatusCodeEnum]**
설명	MR 모션 레지스터의 값을 읽습니다.
요청 매개변수	index : int MR 읽을 레지스터 번호 with_meta : bool 레지스터 메타데이터 반환 여부(name/comment)
반환 값	int: MR 레지스터 숫자값(with_meta=False) MotionRegister: 레지스터 객체(with_meta=True) StatusCodeEnum: 연산 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.2.2 MR 레지스터 값 쓰기

메서드 이름	**register.write_MR(index : int, value : int) -> StatusCodeEnum** **register.write_MR(register : MotionRegister) -> StatusCodeEnum**
설명	MR 모션 레지스터의 값을 씁니다.
요청 매개변수	index : 쓸 int MR 레지스터 번호 value : int MR 쓸 레지스터 숫자 값 register : MotionRegister 객체(id/name/comment/value)
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.2.3 MR 등록 삭제

메서드 이름	**register.delete_MR(index : int) -> StatusCodeEnum**
설명	지정된 MR 모션 레지스터를 삭제합니다.
요청 매개변수	index : int MR 삭제할 레지스터 번호

메서드 이름	register.delete_MR(index : int) -> StatusCodeEnum
반환 값	StatusCodeEnum: 작업 실행 결과 삭제
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

registers/MR.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: MR寄存器读写示例 / Example of reading and writing MR Registers
"""

from Agilebot import Arm, MotionRegister, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 添加MR寄存器
# [EN] Add MR register
ret = arm.register.write_MR(5, 8)
```

```
if ret == StatusCodeEnum.OK:
    print("写入MR寄存器成功 / Write MR register successful")
else:
    print(
        f"写入MR寄存器失败，错误代码 / Write MR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取MR寄存器
# [EN] Read MR register
res, ret = arm.register.read_MR(5)
if ret == StatusCodeEnum.OK:
    print("读取MR寄存器成功 / Read MR register successful")
else:
    print(
        f"读取MR寄存器失败，错误代码 / Read MR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print the result
print(f"MR寄存器值 / MR register value: {res}")

# [ZH] 使用MotionRegister对象写入MR寄存器
# [EN] Write MR register with MotionRegister object
reg = MotionRegister()
reg.id = 5
reg.value = 8
reg.name = "MR5"
reg.comment = "example"
ret = arm.register.write_MR(reg)
if ret == StatusCodeEnum.OK:
    print(
        "使用MotionRegister写入MR寄存器成功 / Write MR register with MotionRegi
ster successful"
    )
else:
    print(
```

```
        f"使用MotionRegister写入MR寄存器失败，错误代码 / Write MR register with
MotionRegister failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取MR寄存器元信息
# [EN] Read MR register meta info
reg_meta, ret = arm.register.read_MR(5, with_meta=True)
if ret == StatusCodeEnum.OK:
    print(
        "读取MR寄存器元信息成功 / Read MR register meta successful"
    )
    print(
        f"名称 / Name: {reg_meta.name}, 注释 / Comment: {reg_meta.comment}"
    )
else:
    print(
        f"读取MR寄存器元信息失败，错误代码 / Read MR register meta failed, error
code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 删除MR寄存器
# [EN] Delete MR register
ret = arm.register.delete_MR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除MR寄存器成功 / Delete MR register successful"
    )
else:
    print(
        f"删除MR寄存器失败，错误代码 / Delete MR register failed, error code: {r
et.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
```

```
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.6.3 SR 문자열 레지스터 작업

4.6.3.1 SR 레지스터 값 읽기

메서드 이름	**register.read_SR(index : int, with_meta : bool = False) -> tuple[str |문자열 레지스터, StatusCodeEnum]**
설명	SR 문자열 레지스터의 값을 읽습니다.
요청 매개변수	index : int SR 읽을 레지스터 번호 with_meta : bool 레지스터 메타데이터 반환 여부(name/comment)
반환 값	str: SR 레지스터 문자열 값(with_meta=False) StringRegister: 레지스터 객체(with_meta=True) StatusCodeEnum: 연산 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.3.2 SR 레지스터 값 쓰기

메서드 이름	**register.write_SR(index : int, value : str) -> StatusCodeEnum** **register.write_SR(register : StringRegister) -> StatusCodeEnum**
설명	SR 문자열 레지스터의 값을 씁니다.
요청 매개변수	index : 쓸 int SR 레지스터 번호 value : str SR 쓸 레지스터 문자열 값 register : StringRegister 객체(id/name/comment/value)
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.3.3 SR 등록 삭제

메서드 이름	register.delete_SR(index : int) -> StatusCodeEnum
설명	지정된 SR 문자열 레지스터를 삭제합니다.
요청 매개변수	index : int SR 삭제할 레지스터 번호
반환 값	StatusCodeEnum: 작업 실행 결과 삭제
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

registers/SR.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: SR寄存器读写示例 / Example of reading and writing SR registers
"""

from Agilebot import Arm, StatusCodeEnum, StringRegister

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 添加SR寄存器
# [EN] Add SR register
ret = arm.register.write_SR(5, "pytest")
if ret == StatusCodeEnum.OK:
    print("写入SR寄存器成功 / Write SR register successful")
else:
    print(
        f"写入SR寄存器失败，错误代码 / Write SR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取SR寄存器
# [EN] Read SR register
res, ret = arm.register.read_SR(5)
if ret == StatusCodeEnum.OK:
    print("读取SR寄存器成功 / Read SR register successful")
else:
    print(
        f"读取SR寄存器失败，错误代码 / Read SR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print the result
print(f"SR寄存器值 / SR register value: {res}")

# [ZH] 使用StringRegister对象写入SR寄存器
# [EN] Write SR register with StringRegister object
reg = StringRegister()
reg.id = 5
reg.value = "pytest"
reg.name = "SR5"
reg.comment = "example"
ret = arm.register.write_SR(reg)
if ret == StatusCodeEnum.OK:
    print(
```

```
        "使用StringRegister写入SR寄存器成功 / Write SR register with StringRegister successful"
    )
else:
    print(
        f"使用StringRegister写入SR寄存器失败，错误代码 / Write SR register with StringRegister failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取SR寄存器元信息
# [EN] Read SR register meta info
reg_meta, ret = arm.register.read_SR(5, with_meta=True)
if ret == StatusCodeEnum.OK:
    print(
        "读取SR寄存器元信息成功 / Read SR register meta successful"
    )
    print(
        f"名称 / Name: {reg_meta.name}, 注释 / Comment: {reg_meta.comment}"
    )
else:
    print(
        f"读取SR寄存器元信息失败，错误代码 / Read SR register meta failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 删除SR寄存器
# [EN] Delete SR register
ret = arm.register.delete_SR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除SR寄存器成功 / Delete SR register successful"
    )
else:
    print(
        f"删除SR寄存器失败，错误代码 / Delete SR register failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.6.4 PR 포즈 레지스터 작업

4.6.4.1 PR 레지스터 값 읽기

메서드 이름	**register.read_PR(index : int, with_meta : bool = False) -> tuple[PoseRegister, StatusCodeEnum]**
설명	PR 포즈 레지스터의 값을 읽습니다.
요청 매개변수	index : int PR 읽을 레지스터 번호 with_meta : bool 레지스터 메타데이터 반환 여부(name/comment)
반환 값	PoseRegister: PR 포즈 데이터 등록(with_meta=True 인 경우 이름/comment 포함) StatusCodeEnum: 작업 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.4.2 PR 레지스터 값 쓰기

메서드 이름	**register.write_PR(value : PoseRegister, with_meta : bool = False) -> StatusCodeEnum**
설명	PR 포즈 레지스터의 값을 씁니다.
요청 매개변수	value : PoseRegister PR 쓸 포즈 데이터를 등록합니다. with_meta : bool 이름 쓸지 여부/comment
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과

메서드 이름	register.write_PR(value : PoseRegister, with_meta : bool = False) -> StatusCodeEnum
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.1+ 산업용(Bronze): v7.6.0.0+

4.6.4.3 PR 등록 삭제

메서드 이름	register.delete_PR(index : int) -> StatusCodeEnum
설명	지정된 PR 포즈 레지스터를 삭제합니다.
요청 매개변수	index : int PR 삭제할 레지스터 번호
반환 값	StatusCodeEnum: 작업 실행 결과 삭제
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

registers/PR.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: PR寄存器读写示例 / Example of reading and writing to the PR regis
ter
"""

from Agilebot import (
    Arm,
    PoseRegister,
    PoseType,
    Posture,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
```

```
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 添加PR寄存器
# [EN] Add PR register
# [ZH] 创建位姿
# [EN] Create pose
pose_register = PoseRegister()
posture = Posture()
posture.arm_back_front = 1
pose_register.poseRegisterData.cartData.posture = posture
pose_register.id = 5
pose_register.poseRegisterData.pt = PoseType.CART
pose_register.poseRegisterData.cartData.position.x = 100
pose_register.poseRegisterData.cartData.position.y = 200
pose_register.poseRegisterData.cartData.position.z = 300
ret = arm.register.write_PR(pose_register)
if ret == StatusCodeEnum.OK:
    print("写入PR寄存器成功 / Write PR register successful")
else:
    print(
        f"写入PR寄存器失败，错误代码 / Write PR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取PR寄存器
# [EN] Read PR register
```

```
res, ret = arm.register.read_PR(5)
if ret == StatusCodeEnum.OK:
    print("读取PR寄存器成功 / Read PR register successful")
else:
    print(
        f"读取PR寄存器失败，错误代码 / Read PR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"位姿寄存器ID / Pose register ID: {res.id}\n"
    f"位姿类型 / Pose type: {res.poseRegisterData.pt}\n"
    f"X / X: {res.poseRegisterData.cartData.position.x}\n"
    f"Y / Y: {res.poseRegisterData.cartData.position.y}\n"
    f"Z / Z: {res.poseRegisterData.cartData.position.z}\n"
    f"C / C: {res.poseRegisterData.cartData.position.c}\n"
    f"B / B: {res.poseRegisterData.cartData.position.b}\n"
    f"A / A: {res.poseRegisterData.cartData.position.a}\n"
)

# [ZH] 写入PR寄存器元信息
# [EN] Write PR register meta info
pose_register.name = "PR5"
pose_register.comment = "example"
ret = arm.register.write_PR(pose_register, with_meta=True)
if ret == StatusCodeEnum.OK:
    print("写入PR寄存器成功 / Write PR register successful")
else:
    print(
        f"写入PR寄存器失败，错误代码 / Write PR register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取PR寄存器元信息
# [EN] Read PR register meta info
reg_meta, ret = arm.register.read_PR(5, with_meta=True)
```

```
if ret == StatusCodeEnum.OK:
    print(
        "读取PR寄存器元信息成功 / Read PR register meta successful"
    )
    print(
        f"名称 / Name: {reg_meta.name}, 注释 / Comment: {reg_meta.comment}"
    )
else:
    print(
        f"读取PR寄存器元信息失败，错误代码 / Read PR register meta failed, error
code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 删除PR寄存器
# [EN] Delete PR register
ret = arm.register.delete_PR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除PR寄存器成功 / Delete PR register successful"
    )
else:
    print(
        f"删除PR寄存器失败，错误代码 / Delete PR register failed, error code: {r
et.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.6.5 Modbus 레지스터(MH 홀딩 레지스터, MI 입력 레지스터)

4.6.5.1 MH 레지스터 값 읽기

메서드 이름	**register.read_MH(index : int) -> tuple[int, StatusCodeEnum]**
설명	MH 보유 레지스터의 값을 읽습니다.
요청 매개변수	index : int MH 읽을 레지스터 번호
반환 값	int: MH 레지스터 숫자값 StatusCodeEnum: 연산 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.6.5.2 MH 레지스터 값 쓰기

메서드 이름	**register.write_MH(index : int, value : int) -> StatusCodeEnum**
설명	MH 보유 레지스터의 값을 씁니다.
요청 매개변수	index : int MH 쓸 레지스터 번호 value : int MH 쓸 레지스터 숫자값
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.6.5.3 MI 레지스터 값 읽기

메서드 이름	**register.read_MI(index : int) -> tuple[int, StatusCodeEnum]**
설명	MI 입력 레지스터의 값을 읽습니다.
요청 매개변수	index : int MI 읽을 레지스터 번호
반환 값	int: MI 레지스터 숫자값 StatusCodeEnum: 연산 실행 결과 읽기
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.6.5.4 MI 레지스터 값 쓰기

메서드 이름	register.write_MI(index : int, value : int) -> StatusCodeEnum
설명	MI 입력 레지스터의 값을 씁니다.
요청 매개변수	index : int MI 쓸 레지스터 번호 value : int MI 쓸 레지스터 숫자값
반환 값	StatusCodeEnum: 쓰기 작업 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

예제 코드

registers/MH_MI.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: MH寄存器及MI寄存器读写示例 / Example of reading and writing to the
MH register and MI register
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
```

```
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取MH寄存器
# [EN] Read MH register
res, ret = arm.register.read_MH(1)
if ret == StatusCodeEnum.OK:
    print("读取MH寄存器成功 / Read MH register successful")
else:
    print(
        f"读取MH寄存器失败，错误代码 / Read MH register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(f"MH寄存器 / MH register: {res}")

# [ZH] 写入MH寄存器
# [EN] Write MH register
ret = arm.register.write_MH(1, 16)
if ret == StatusCodeEnum.OK:
    print("写入MH寄存器成功 / Write MH register successful")
else:
    print(
        f"写入MH寄存器失败，错误代码 / Write MH register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 读取MI寄存器
# [EN] Read MI register
res, ret = arm.register.read_MI(1)
if ret == StatusCodeEnum.OK:
    print("读取MI寄存器成功 / Read MI register successful")
else:
    print(
        f"读取MI寄存器失败，错误代码 / Read MI register failed, error code: {re
```

```
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(f"MI寄存器 / MI register: {res}")

# [ZH] 写入MI寄存器
# [EN] Write MI register
ret = arm.register.write_MI(1, 18)
if ret == StatusCodeEnum.OK:
    print("写入MI寄存器成功 / Write MI register successful")
else:
    print(
        f"写入MI寄存器失败，错误代码 / Write MI register failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.7 궤도 제어

개요

Trajectory 모듈은 오프라인 궤도 실행, Trajectory/path 기록, 변환 및 재생을 위한 전체 주기 인터페이스를 제공하여 복잡한 궤도 사용자 정의 및 재생을 지원합니다.

핵심 기능

- 오프라인 궤적 파일 설정 및 실행 지원
- 안전한 속도로 로봇을 오프라인 궤적 시작점까지 이동 지원
- CSV를 궤적 형식 파일로 변환 지원
- Trajectory/path 녹화, 재생, 삭제, 관리 지원
- 궤적 목록 및 시작 위치 획득 지원
- 경로 플래너 매개변수 구성 및 쿼리 지원
- 궤적을 따른 모션 지원/paths

사용 사례

- 복잡한 궤적의 정밀한 재현 구현
- 로봇 모션 궤적 기록 및 재생
- 외부 궤적 데이터를 로봇 실행 가능 형식으로 변환
- 경로 계획 매개변수를 사용자 정의하여 모션 성능 최적화
- 궤적/path 파일 일괄 관리 및 쿼리

4.7.1 실행할 오프라인 궤적 파일 설정

메서드 이름	**trajectory.set_offline_trajectory_file(`path` : str) -> StatusCodeEnum**
설명	실행할 오프라인 궤적 파일을 설정합니다.

메서드 이름	**trajectory.set_offline_trajectory_file(`path` : str) -> StatusCodeEnum**
요청 매개변수	`path` : str 오프라인 궤적 파일 이름(형식은 참고 사항 참조).
반환 값	StatusCodeEnum: 함수 실행 결과.
메모	궤적 파일은 일반 텍스트입니다: - 라인 1: `6` (축), `0.001` (시간 간격, s), `8093` (포인트 개수). - 라인 2: 6개 축의 초기 위치. - 라인 3-8095: 궤적 점 포함 위치/velocity/acceleration/torque 피드포워드/ `do_port` / `do_state` . - `do_port` 범위 1-24; `-1` 는 IO 트리거가 없음을 의미합니다. `do_port = 1` 와 `do_state = 1` 는 `do1` 를 ON으로 설정하고, `do_state = 0` 는 `do1` 를 OFF로 설정합니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.2 안전한 속도로 출발점까지 이동

메서드 이름	**trajectory.prepare_offline_trajectory() -> StatusCodeEnum**
설명	안전한 속도로 로봇을 오프라인 궤적의 시작점으로 이동시킵니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.3 오프라인 궤적 실행 시작

메서드 이름	**trajectory.execute_offline_trajectory() -> StatusCodeEnum**
설명	오프라인 궤적 프로그램 실행을 시작합니다.

메서드 이름	**trajectory.execute_offline_trajectory() -> StatusCodeEnum**
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.4 궤적 파일 변환

메서드 이름	**trajectory.transform_csv_to_trajectory(** `file_name` **: str,** `separator` **: str = " ",** `io_flag` **: str = "2")**
설명	궤적 CSV 파일을 컨트롤러 궤적 형식으로 변환하고 이를 컨트롤러의 궤적 디렉터리에 저장합니다.
요청 매개변수	`file_name` : str CSV 궤적 파일 이름(`.csv` 제외). `separator` : str 구분 기호(공백 또는 쉼표, 공백 -> `.trajectory` , 쉼표 -> `.csv`). `io_flag` : 문자열 IO 소스(1: 기본값 `do_port = -1` , `do_state = 0` , 2: 사용자 IO).
반환 값	str: 변환된 궤적 파일의 경로 StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.5 궤적 파일 변환 상태 쿼리

메서드 이름	**trajectory.check_transform_status(** `file_name` **: str) -> tuple[TransformStatusEnum, StatusCodeEnum]**
설명	궤적 파일 변환의 현재 상태를 쿼리합니다.
요청 매개변수	`file_name` : str 궤적 파일 경로(`transform_csv_to_trajectory` 에서 반환).

메서드 이름	**trajectory.check_transform_status(file_name : str) -> tuple[TransformStatusEnum, StatusCodeEnum]**
반환 값	TransformStatusEnum: 변환 상태 StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

trajectory/offline_trajectory.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 离线轨迹使用示例 / Example of offline trajectory usage
"""

import time

from Agilebot import (
    Arm,
    RobotStatusEnum,
    ServoStatusEnum,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
```

```
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置离线轨迹文件
# [EN] Set the offline trajectory file
ret = arm.trajectory.set_offline_trajectory_file(
    "test_torque.trajectory"
)
if ret == StatusCodeEnum.OK:
    print(
        "设置离线轨迹文件成功 / Set offline trajectory file successful"
    )
else:
    print(
        f"设置离线轨迹文件失败，错误代码 / Set offline trajectory file failed, er
ror code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 准备进行离线轨迹运行
# [EN] Prepare for offline trajectory execution
ret = arm.trajectory.prepare_offline_trajectory()
if ret == StatusCodeEnum.OK:
    print(
        "准备离线轨迹运行成功 / Prepare offline trajectory execution successfu
l"
    )
else:
    print(
        f"准备离线轨迹运行失败，错误代码 / Prepare offline trajectory execution f
ailed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 等待控制器到位
# [EN] Wait for the controller to be ready
while True:
```

```
    robot_status, ret = arm.get_robot_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取机器人状态成功 / Get robot status successful"
        )
    else:
        print(
            f"获取机器人状态失败，错误代码 / Get robot status failed, error cod
e: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"robot_status arm: {robot_status}")
    servo_status, ret = arm.get_servo_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取伺服状态成功 / Get servo status successful"
        )
    else:
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"伺服状态 / Servo status: {servo_status}")
    if (
        robot_status == RobotStatusEnum.ROBOT_IDLE
        and servo_status == ServoStatusEnum.SERVO_IDLE
    ):
        break
    time.sleep(2)

# [ZH] 执行离线轨迹
# [EN] Execute the offline trajectory
ret = arm.trajectory.execute_offline_trajectory()
if ret == StatusCodeEnum.OK:
    print(
        "执行离线轨迹成功 / Execute offline trajectory successful"
    )
else:
    print(
```

```
        f"执行离线轨迹失败，错误代码 / Execute offline trajectory failed, error
code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.7.6 궤적 기록 시작

메서드 이름	**trajectory.trajectory_record_begin(`name` : str) -> StatusCodeEnum**
설명	로봇에게 궤적 기록을 시작하도록 지시합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.7 궤적 기록 중지

메서드 이름	**trajectory.trajectory_record_finish(`name` : str) -> StatusCodeEnum**
설명	로봇에게 궤적 기록을 중지하도록 지시합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	StatusCodeEnum: 함수의 실행 결과.

메서드 이름	**trajectory.trajectory_record_finish(`name` : str) -> StatusCodeEnum**
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.8 궤적 재생 시작

메서드 이름	**trajectory.trajectory_replay_start(`name` : str) -> StatusCodeEnum**
설명	로봇에게 궤적 재생을 시작하도록 지시합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.9 궤적 재생 중지

메서드 이름	**trajectory.trajectory_replay_stop(`name` : str) -> StatusCodeEnum**
설명	로봇에게 궤적 재생을 중지하도록 지시합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.10 궤적 삭제

메서드 이름	**trajectory.trajectory_record_delete(`name` : str) -> StatusCodeEnum**
설명	로봇에 저장된 특정 궤적을 삭제합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.11 기록된 궤적 목록 가져오기

메서드 이름	**trajectory.get_trajectory_record_list() -> tuple[list[str], StatusCodeEnum]**
설명	기록된 궤적 목록을 검색합니다.
매개변수	없음
반품	[list[str]]: 궤적 프로그램 이름 목록입니다. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.7.12 기록된 궤적 시작 포즈 가져오기

메서드 이름	**trajectory.get_trajectory_record_start_pose(`name` : str) -> tuple[MotionPose, StatusCodeEnum]**
설명	기록된 궤적의 시작 포즈를 검색합니다.
매개변수	`name` : 궤적 프로그램 이름.
반품	MotionPose: 궤적의 시작 포즈. StatusCodeEnum: 함수의 실행 결과.

메서드 이름	trajectory.get_trajectory_record_start_pose(name : str) -> tuple[MotionPose, StatusCodeEnum]
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

trajectory/trajectory_record.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 轨迹记录相关使用示例 / Example of real-time trajectory records usage
"""

import time

from Agilebot import (
    Arm,
    RobotStatusEnum,
    ServoStatusEnum,
    StatusCodeEnum,
)
from Agilebot.IR.A.hardware_state import (
    HardwareState,
    HWState,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
```

```
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 订阅轨迹记录状态
# [EN] Subscribe to the trajectory record status
hw_state = HardwareState("10.27.1.254")
hw_state.subscribe(
    topic_list=[HWState.TOPIC_TRAJECTORY_RECORDS_STATUS]
)

# [ZH] 开始记录轨迹
# [EN] Start recording trajectory
ret = arm.trajectory.trajectory_record_begin("test")
if ret == StatusCodeEnum.OK:
    print(
        "开始轨迹记录成功 / Start trajectory record successful"
    )
else:
    print(
        f"开始轨迹记录失败，错误代码 / Start trajectory record failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
time.sleep(10)

res = hw_state.recv()
print(f"轨迹记录状态 / Trajectory record status: {res}")

# [ZH] 结束记录轨迹
# [EN] End recording trajectory
ret = arm.trajectory.trajectory_record_finish("test")
if ret == StatusCodeEnum.OK:
    print(
        "结束轨迹记录成功 / End trajectory record successful"
    )
else:
```

```
    print(
        f"结束轨迹记录失败，错误代码 / End trajectory record failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

res = hw_state.recv()
print(f"轨迹记录状态 / Trajectory record status: {res}")

# [ZH] 获取轨迹列表
# [EN] Get trajectory list
record_list, ret = (
    arm.trajectory.get_trajectory_record_list()
)
if ret == StatusCodeEnum.OK:
    print(
        "获取轨迹记录列表成功 / Get trajectory record list successful"
    )
else:
    print(
        f"获取轨迹记录列表失败，错误代码 / Get trajectory record list failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
assert "test" in record_list

# [ZH] 获取轨迹起始位姿
# [EN] Get trajectory start pose
pose, ret = arm.trajectory.get_trajectory_record_start_pose(
    "test"
)
if ret == StatusCodeEnum.OK:
    print(
        "获取轨迹起始位姿成功 / Get trajectory start pose successful"
    )
else:
    print(
        f"获取轨迹起始位姿失败，错误代码 / Get trajectory start pose failed, erro
r code: {ret.errmsg}"
    )
```

```
    arm.disconnect()
    exit(1)

# [ZH] 移动到起始位姿
# [EN] Move to the start pose
ret = arm.motion.move_joint(pose)
if ret == StatusCodeEnum.OK:
    print("关节运动成功 / Joint motion successful")
else:
    print(
        f"关节运动失败，错误代码 / Joint motion failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 等待控制器到位
# [EN] Wait for the controller to be ready
while True:
    robot_status, ret = arm.get_robot_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取机器人状态成功 / Get robot status successful"
        )
    else:
        print(
            f"获取机器人状态失败，错误代码 / Get robot status failed, error cod
e: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"robot_status arm: {robot_status}")
    servo_status, ret = arm.get_servo_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取伺服状态成功 / Get servo status successful"
        )
    else:
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
```

```
            arm.disconnect()
            exit(1)
        print(f"伺服状态 / Servo status: {servo_status}")
        if (
            robot_status == RobotStatusEnum.ROBOT_IDLE
            and servo_status == ServoStatusEnum.SERVO_IDLE
        ):
            break
        time.sleep(2)

# [ZH] 开始回放轨迹
# [EN] Start replay trajectory
ret = arm.trajectory.trajectory_replay_start("test")
if ret == StatusCodeEnum.OK:
    print(
        "开始轨迹回放成功 / Start trajectory replay successful"
    )
else:
    print(
        f"开始轨迹回放失败，错误代码 / Start trajectory replay failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

time.sleep(5)

# [ZH] 停止回放轨迹
# [EN] Stop replay trajectory
ret = arm.trajectory.trajectory_replay_stop("test")
if ret == StatusCodeEnum.OK:
    print(
        "停止轨迹回放成功 / Stop trajectory replay successful"
    )
else:
    print(
        f"停止轨迹回放失败，错误代码 / Stop trajectory replay failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
```

```
# [ZH] 删除轨迹
# [EN] Delete trajectory
ret = arm.trajectory.trajectory_record_delete("test")
if ret == StatusCodeEnum.OK:
    print(
        "删除轨迹记录成功 / Delete trajectory record successful"
    )
else:
    print(
        f"删除轨迹记录失败，错误代码 / Delete trajectory record failed, error co
de: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.7.13 기록 시작 궤적/경로

메서드 이름	**trajectory.path_record_begin(`name` : str, `comment` : str, `param` : float, `angle` : float = 1) -> StatusCodeEnum**
설명	궤적 테이블(*.traj*) 또는 경로 테이블(*.path*) 기록을 시작합니다.
매개변수	`name` : 궤적/경로 파일 이름은 *.traj* 또는 *.path*로 끝나야 합니다. `comment` : 궤적/경로에 대한 설명. `param` : - *.path* 테이블의 경우: 선형 동작 거리 임계값(mm). - *.traj* 테이블의 경우: 기록 간격(ms). `angle` : 회전 각도 임계값 (°), `name` 가 *.path* 파일인 경우에만 유효합니다.
반품	StatusCodeEnum: 함수 실행 결과.

메서드 이름	**trajectory.path_record_begin(** `name` **: str,** `comment` **: str,** `param` **: float,** `angle` **: float = 1) -> StatusCodeEnum**
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.14 궤적/경로 기록 중지

메서드 이름	**trajectory.path_record_finish() -> StatusCodeEnum**
설명	현재 궤적/경로 테이블 기록을 중지합니다.
매개변수	없음
반품	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.15 궤적/경로의 시작 포즈 가져오기

메서드 이름	**trajectory.get_path_start_pose(** `name` **: str) -> tuple[MotionPose, StatusCodeEnum]**
설명	지정된 궤적/경로 파일의 시작 포즈를 검색합니다.
매개변수	`name` : 궤적/경로 파일 이름.
반품	`MotionPose` : 시작 포즈. StatusCodeEnum: 함수의 실행 결과입니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.16 궤적/경로 상태 가져오기

메서드 이름	**trajectory.get_path_state(`path_list` : list[str]) -> tuple[dict[str, int], StatusCodeEnum]**
설명	궤적/경로 파일의 현재 상태를 일괄 쿼리합니다.
매개변수	`path_list` : 쿼리할 궤적/경로 파일 이름 목록입니다.
반품	`dict[str, int]` : 파일명 → 상태 코드 매핑. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.17 경로 계획자 매개변수 설정

메서드 이름	**trajectory.set_path_planner_parameter(`transition_time` : float, `scaling_factor` : float) -> StatusCodeEnum**
설명	동작의 부드러움과 속도/가속도 분포에 영향을 미치는 경로 계획 매개변수를 구성합니다.
매개변수	`transition_time` : 블렌딩 시간(0~1; 더 큰 = 더 부드러운 가속/decel). `scaling_factor` : 경로 매개변수 *s* 가중치(0~1; 0: 균일한 시간 스케일링, 1: 최대 변위 스케일링, 0.5: 동일 가속, 0.33: 동일 저크).
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.18 경로 계획자 매개변수 가져오기

메서드 이름	**trajectory.get_path_planner_parameter() -> tuple[float, float, StatusCodeEnum]**
설명	현재 경로 계획자 매개변수를 읽습니다.
매개변수	없음

메서드 이름	**trajectory.get_path_planner_parameter() -> tuple[float, float, StatusCodeEnum]**
반품	`transition_time` : 블렌딩 시간, 0~1. `scaling_factor` : 재분배 가중치, 0~1. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

4.7.19 궤적/경로를 따라 이동

메서드 이름	**trajectory.move_path(** `name` **: str,** `vel` **: float = 100,** `acc` **: float = 1) -> StatusCodeEnum**
설명	로봇 엔드 이펙터에게 지정된 궤적/경로 파일을 따라 이동하도록 명령합니다.
매개변수	`name` : 궤적/path 파일 이름. `vel` : TCP 속도(0~5000mm/s). `acc` : 가속 배율(0~1.2).
반품	StatusCodeEnum: 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.1.0+ 산업용(Bronze): 지원되지 않음

예제 코드

trajectory/path_record.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 路径记录相关使用示例 / Example of usage related to path records
"""

import time
```

```
from Agilebot import (
    Arm,
    MoveMode,
    RobotStatusEnum,
    ServoStatusEnum,
    StatusCodeEnum,
)

# [ZH] 初始化机械臂并连接到指定IP地址
# [EN] Initialize the robotic arm and connect to the specified IP address
arm = Arm()
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 开始记录轨迹，指定文件名、轨迹名、记录模式和覆盖选项
# [EN] Start trajectory recording, specifying filename, trajectory name, rec
ording mode and overwrite option
ret = arm.trajectory.path_record_begin(
    "test_path.path", "Path Test", 10, 1
)
if ret == StatusCodeEnum.OK:
    print("开始路径记录成功 / Start path record successful")
else:
    print(
        f"开始路径记录失败，错误代码 / Start path record failed, error code: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 检查记录状态，传入轨迹文件名列表
# [EN] Check recording status, passing in trajectory filename list
state, ret = arm.trajectory.get_path_state(
    ["test_path.path"]
```

```
)
if ret == StatusCodeEnum.OK:
    print("获取路径状态成功 / Get path state successful")
else:
    print(
        f"获取路径状态失败，错误代码 / Get path state failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
if state["test_path.path"] == 0:
    print("路径表记录中 / Path table recording in progress")

# [ZH] 示教运动
# [EN] Teaching motion
ret = arm.jogging.move(3, MoveMode.Continuous)
if ret == StatusCodeEnum.OK:
    print("点动运动成功 / Jogging movement successful")
else:
    print(
        f"点动运动失败，错误代码 / Jogging movement failed, error code: {ret.er
rmsg}"
    )
    arm.disconnect()
    exit(1)
# 等待3秒，让机械臂持续运动
time.sleep(2)
# 停止机械臂运动
arm.jogging.stop()
time.sleep(2)
ret = arm.jogging.move(-3, MoveMode.Continuous)
if ret == StatusCodeEnum.OK:
    print("点动运动成功 / Jogging movement successful")
else:
    print(
        f"点动运动失败，错误代码 / Jogging movement failed, error code: {ret.er
rmsg}"
    )
    arm.disconnect()
    exit(1)
# 等待3秒，让机械臂持续运动
time.sleep(2)
```

```
# 停止机械臂运动
arm.jogging.stop()
time.sleep(2)

# [ZH] 结束轨迹记录
# [EN] Finish trajectory recording
ret = arm.trajectory.path_record_finish()
if ret == StatusCodeEnum.OK:
    print(
        "结束路径记录成功 / Finish path record successful"
    )
else:
    print(
        f"结束路径记录失败，错误代码 / Finish path record failed, error code: {r
et.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 检查记录状态，传入轨迹文件名列表
# [EN] Check recording status, passing in trajectory filename list
state, ret = arm.trajectory.get_path_state(
    ["test_path.path"]
)
if ret == StatusCodeEnum.OK:
    print("获取路径状态成功 / Get path state successful")
else:
    print(
        f"获取路径状态失败，错误代码 / Get path state failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
if state["test_path.path"] == 1:
    print("路径表记录完成 / Path table recording completed")

# [ZH] 获取记录轨迹的起始位置姿态
# [EN] Get the starting position pose of the recorded trajectory
pose, ret = arm.trajectory.get_path_start_pose(
    "test_path.path"
)
if ret == StatusCodeEnum.OK:
```

```
    print(
        "获取路径起始位姿成功 / Get path start pose successful"
    )
else:
    print(
        f"获取路径起始位姿失败，错误代码 / Get path start pose failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 移动到起始位姿
# [EN] Move to the start pose
ret = arm.motion.move_joint(pose)
if ret == StatusCodeEnum.OK:
    print("关节运动成功 / Joint motion successful")
else:
    print(
        f"关节运动失败，错误代码 / Joint motion failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 等待控制器到位
# [EN] Wait for the controller to be ready
while True:
    robot_status, ret = arm.get_robot_status()
    # [ZH] 检查机器人状态获取是否成功
    # [EN] Check if robot status acquisition is successful
    if ret == StatusCodeEnum.OK:
        print(
            "获取机器人状态成功 / Get robot status successful"
        )
    else:
        print(
            f"获取机器人状态失败，错误代码 / Get robot status failed, error cod
e: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"robot_status arm: {robot_status}")
```

```
    servo_status, ret = arm.get_servo_status()
    # [ZH] 检查伺服状态获取是否成功
    # [EN] Check if servo status acquisition is successful
    if ret == StatusCodeEnum.OK:
        print(
            "获取伺服状态成功 / Get servo status successful"
        )
    else:
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"servo status arm: {servo_status}")
    if (
        robot_status == RobotStatusEnum.ROBOT_IDLE
        and servo_status == ServoStatusEnum.SERVO_IDLE
    ):
        break
    time.sleep(2)

# [ZH] 设置轨迹规划参数（速度比例和加速度比例）
# [EN] Set trajectory planning parameters (velocity ratio and acceleration r
atio)
ret = arm.trajectory.set_path_planner_parameter(
    0.5, 0.3333333
)
if ret == StatusCodeEnum.OK:
    print(
        "设置路径规划参数成功 / Set path planner parameter successful"
    )
else:
    print(
        f"设置路径规划参数失败，错误代码 / Set path planner parameter failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取轨迹规划参数以验证设置是否成功
# [EN] Get trajectory planning parameters to verify if the setting is succes
```

```
sful
param1, param2, ret = (
    arm.trajectory.get_path_planner_parameter()
)
if ret == StatusCodeEnum.OK:
    print(
        "获取路径规划参数成功 / Get path planner parameter successful"
    )
else:
    print(
        f"获取路径规划参数失败，错误代码 / Get path planner parameter failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 移动到记录的轨迹，指定轨迹文件名、速度和模式
# [EN] Move to the recorded trajectory, specifying trajectory filename, spee
d and mode
ret = arm.trajectory.move_path("test_path.path", 2000, 1)
if ret == StatusCodeEnum.OK:
    print("路径运动成功 / Path motion successful")
else:
    print(
        f"路径运动失败，错误代码 / Path motion failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)
time.sleep(3)

# [ZH] 断开与机械臂的连接
# [EN] Disconnect from the robotic arm
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.8 알람 정보

개요

알람 모듈은 로봇 알람 정보를 읽고, 재설정하고, 쿼리하는 기능을 제공하여 작동 중 발생할 수 있는 이상 현상을 모니터링하고 처리할 수 있습니다.

핵심 기능

- 알람 재설정 지원
- 모든 활성 알람 수신 지원
- 우선순위가 가장 높은 알람 수신 지원
- 알람 출력 언어 지정 지원

사용 사례

- 로봇 동작 상태를 모니터링하고 이상 징후를 적시에 감지
- 로봇 작동 중 오류 및 알람 처리
- 로봇 알람 이력 기록 및 분석
- 자동화된 경보 처리 프로세스 구현
- 로봇 모니터링 시스템에 통합

4.8.1 알람 재설정

메서드 이름	alarm.reset() -> StatusCodeEnum
설명	현재 오류를 재설정합니다./alarms.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과.

메서드 이름	alarm.reset() -> StatusCodeEnum
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.8.2 모든 활성 알람 가져오기

메서드 이름	alarm.get_all_active_alarms(`language` : LanguageType) -> tuple[list, StatusCodeEnum]
설명	현재 활성화된 모든 알람을 가져옵니다.
요청 매개변수	`language` : LanguageType 출력 언어(`LanguageType.English` 또는 `LanguageType.Chinese`).
반환 값	목록: 활성 알람 항목. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.8.3 최우선 순위 알람 받기

메서드 이름	alarm.get_top_alarm() -> tuple[ROBOT_ALARM, StatusCodeEnum]
설명	가장 높은 우선순위로 알람을 가져옵니다.
요청 매개변수	없음
반환 값	`ROBOT_ALARM` : 우선순위가 가장 높은 경보입니다. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

alarm.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 告警功能使用示例 / Example of using the alarm function
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()

# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取所有的活动的报警
# [EN] Get all active alarms
alarms, ret = arm.alarm.get_all_active_alarms()
if ret == StatusCodeEnum.OK:
    print(
        "获取报警信息成功 / Get alarm information successfully"
    )
    for alarm in alarms:
        print(alarm)
else:
    print(
        f"获取报警信息失败，错误代码 / Failed to get alarm information, error co
de: {ret.errmsg}"
    )
```

```
    arm.disconnect()
    exit(1)

# [ZH] 获取所有的活动的报警
# [EN] Get all active alarms
alarm, ret = arm.alarm.get_top_alarm()
if ret == StatusCodeEnum.OK:
    print(
        "获取报警信息成功 / Get alarm information successfully"
    )
    for alarm in alarm:
        print(alarm)
else:
    print(
        f"获取报警信息失败，错误代码 / Failed to get alarm information, error co
de: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 重置报警
# [EN] Reset alarms
ret = arm.alarm.reset()
if ret == StatusCodeEnum.OK:
    print("报警重置成功 / Alarm reset successfully")
else:
    print(
        f"报警重置失败，错误代码 / Alarm reset failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.9 파일 관리자

개요

FileManager는 호스트 컴퓨터와 로봇 컨트롤러 간의 로봇 프로그램, 궤적, 임시 파일 등의 리소스를 업로드, 다운로드, 삭제 또는 검색하는 데 사용되며 다양한 파일 유형에 대한 관리 및 운영을 지원합니다.

핵심 기능

- 로봇 컨트롤러에 로컬 파일 업로드 지원
- 로컬 디렉터리에 로봇 파일 다운로드 지원
- 로봇 컨트롤러에서 파일 삭제 지원
- 파일 이름 패턴으로 파일 검색 지원
- 여러 파일 형식 관리 지원(USER_PROGRAM, BLOCK_PROGRAM, TRAJECTORY, ROBOT_TMP)
- 업로드 중 덮어쓰기 제어 지원/download

사용 사례

- 로봇 컨트롤러에 프로그램 배포
- 로봇의 프로그램 및 궤적 파일 백업
- 생산 라인의 디버그 데이터 동기화
- 로봇 파일 일괄 관리 및 쿼리
- 임시 데이터 파일을 로봇에 업로드
- 로봇 로그 또는 출력 파일을 로컬로 다운로드

4.9.1 로봇에 로컬 파일 업로드

메서드 이름	**file_manager.upload(** `file_path` **: str,** `file_type` **: str,** `overwriting` **: bool = False) -> StatusCodeEnum**
설명	로봇 컨트롤러에 로컬 파일을 업로드합니다.

메서드 이름	**file_manager.upload(`file_path` : str, `file_type` : str, `overwriting` : bool = False) -> StatusCodeEnum**
요청 매개변수	`file_path` : str 로컬 파일 절대 경로. `file_type` : str 파일 형식(USER_PROGRAM: `file_path` 는 프로그램 이름 경로입니다. 동일한 디렉터리에서 `json` / `xml` 를 업로드합니다. BLOCK_PROGRAM: `file_path` 는 블록 프로그램 이름 경로입니다. 동일한 디렉터리에서 `block` / `json` / `xml` 를 업로드합니다. TRAJECTORY: `file_path` 는 전체 궤적 파일 경로입니다. ROBOT_TMP: `file_path` 는 전체 임시 파일 경로입니다. `overwriting` : bool 같은 이름의 파일을 덮어씁니다(기본값 `False`).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	`USER_PROGRAM` 및 `BLOCK_PROGRAM` 의 경우 프로그램 이름만 지정하십시오(예: 확장자가 없는 파일 이름).

4.9.2 로봇 파일을 로컬로 다운로드

메서드 이름	**file_manager.download(`file_name` : str, `file_path` : str, `file_type` : str, `overwriting` : bool=False) -> StatusCodeEnum**
설명	로봇에서 로컬 디렉터리로 파일을 다운로드합니다.
요청 매개변수	`file_name` : str 파일 이름. `file_path` : str 로컬 저장 디렉터리. `file_type` : str 파일 형식(`BLOCK_PROGRAM` 다운로드는 지원되지 않음). `overwriting` : bool 같은 이름의 파일을 덮어씁니다.
반환 값	StatusCodeEnum: 함수 실행 결과.

메서드 이름	**file_manager.download(file_name : str, file_path : str, file_type : str, overwriting : bool=False) -> StatusCodeEnum**
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	USER_PROGRAM 의 경우 프로그램 이름만 지정하십시오(예: 확장자가 없는 파일 이름). 시스템은 일치하는 .json / .xml .를 다운로드합니다. TRAJECTORY 의 경우 .trajectory 로 전체 파일 이름을 제공합니다. ROBOT_TMP 의 경우 접미사를 포함한 전체 파일 이름을 제공하세요(예: .csv).

4.9.3 로봇에서 파일 삭제

메서드 이름	**file_manager.delete(file_name : str, file_type : str) -> StatusCodeEnum**
설명	로봇 컨트롤러에서 파일을 삭제합니다.
요청 매개변수	file_name : str 파일 이름. file_type : str 파일 형식.
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+
메모	USER_PROGRAM / BLOCK_PROGRAM ,의 경우 프로그램 이름만 지정하십시오(예: 확장자가 없는 파일 이름). TRAJECTORY / ROBOT_TMP ,의 경우 접미사와 함께 전체 파일 이름을 제공하십시오.

4.9.4 파일 이름 패턴으로 파일 검색

메서드 이름	file_manager.search(pattern : str, file_list : list) -> StatusCodeEnum
설명	파일 이름 패턴과 일치하는 컨트롤러에서 파일을 검색합니다.
요청 매개변수	pattern : str 파일 이름 일치 패턴입니다. file_list : list 출력 목록(일치하는 파일 이름을 받습니다).
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

file_manager/file_manager.py

```py
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 文件操作，上传下载示例 / Example of file operation, upload and dow
nload
"""

from pathlib import Path

from Agilebot import (
    ROBOT_TMP,
    TRAJECTORY,
    USER_PROGRAM,
    FileManager,
    StatusCodeEnum,
)

current_path = Path(__file__)
path = str(current_path)

# [ZH] 连接文件管理服务
# [EN] Connect to the file management service
file_manager = FileManager("10.27.1.254")
```

```
# [ZH] 上传其他文件
# [EN] Upload other files
tmp_file_path = path.replace("file_manager.py", "test.csv")
ret = file_manager.upload(tmp_file_path, ROBOT_TMP, True)
if ret == StatusCodeEnum.OK:
    print("文件上传成功 / File upload successful")
else:
    print(
        f"文件上传失败，错误代码 / File upload failed, error code: {ret.errms
g}"
    )
    exit(1)

# [ZH] 上传程序
# [EN] Upload a program
prog_file_path = path.replace(
    "file_manager.py", "test_prog"
)
ret = file_manager.upload(
    prog_file_path, USER_PROGRAM, True
)
if ret == StatusCodeEnum.OK:
    print("程序上传成功 / Program upload successful")
else:
    print(
        f"程序上传失败，错误代码 / Program upload failed, error code: {ret.errm
sg}"
    )
    exit(1)

# [ZH] 上传轨迹
# [EN] Upload a trajectory
trajectory_file_path = path.replace(
    "file_manager.py", "test_torque.trajectory"
)
ret = file_manager.upload(
    trajectory_file_path, TRAJECTORY, True
)
if ret == StatusCodeEnum.OK:
    print("轨迹上传成功 / Trajectory upload successful")
else:
    print(
```

```
        f"轨迹上传失败，错误代码 / Trajectory upload failed, error code: {ret.e
rrmsg}"
    )
    exit(1)

# [ZH] 搜索文件
# [EN] Search for files
file_list = list()
ret = file_manager.search("test.csv", file_list)
if ret == StatusCodeEnum.OK:
    print("文件搜索成功 / File search successful")
else:
    print(
        f"文件搜索失败，错误代码 / File search failed, error code: {ret.errms
g}"
    )
    exit(1)
print("搜索文件：", file_list)

# [ZH] 下载文件
# [EN] Download a file
download_file_path = path.replace(
    "file_manager.py", "download"
)
ret = file_manager.download(
    "test_torque", download_file_path, file_type=TRAJECTORY
)
if ret == StatusCodeEnum.OK:
    print("文件下载成功 / File download successful")
else:
    print(
        f"文件下载失败，错误代码 / File download failed, error code: {ret.errms
g}"
    )
    exit(1)

# [ZH] 删除文件
# [EN] Delete a file
ret = file_manager.delete(
    "test_torque.trajectory", TRAJECTORY
)
if ret == StatusCodeEnum.OK:
```

```
    print("文件删除成功 / File delete successful")
else:
    print(
        f"文件删除失败，错误代码 / File delete failed, error code: {ret.errms
g}"
    )
    exit(1)
```

4.10 좌표계

개요

CoordinateSystem 모듈은 로봇의 사용자 프레임(UF) 및 도구 프레임(TF)을 관리하여 프레임을 추가, 삭제, 업데이트, 쿼리 및 계산할 수 있는 통합 API를 제공합니다.

핵심 기능

- user/tool 좌표계 요약 정보 목록 획득 지원
- user/tool 좌표계 추가, 삭제, 업데이트 및 쿼리 지원
- 여러 입력 포즈를 기반으로 user/tool 좌표계 계산 지원
- 컨트롤러의 프레임 목록을 유지하기 위한 통합 좌표계 관리 인터페이스 제공
- 학습된 점으로부터 새로운 좌표계를 신속하게 해결하기 위한 지원

사용 사례

- 다중 스테이션 설정을 디버깅할 때 일관된 공간 참조 유지
- 도구 전환 시 다양한 도구에 대한 좌표계 관리
- 로봇 좌표계 일괄 관리 및 쿼리
- 학습된 점으로부터 새로운 좌표계를 자동으로 계산
- 복잡한 작업에 좌표계 전환 구현

4.10.1 User/Tool 좌표 목록 가져오기

메서드 이름	**coordinate_system.UF/TF.get_coordinate_list() -> tuple[List[Coordinate], StatusCodeEnum]**
설명	user/tool 좌표계의 요약 목록을 가져옵니다.
요청 매개변수	없음

메서드 이름	**coordinate_system.UF/TF.get_coordinate_list() -> tuple[List[Coordinate], StatusCodeEnum]**
반환 값	UserCoordSummaryList: 좌표계 요약 정보 목록입니다. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.10.2 User/Tool 좌표계 추가

메서드 이름	**coordinate_system.UF/TF.add(** `coordinate` **: Coordinate) -> StatusCodeEnum**
설명	현재 user/tool 좌표 세트에 좌표계를 추가합니다.
요청 매개변수	`coordinate` : Coordinate 추가할 좌표계 정보입니다.
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.10.3 User/Tool 좌표계 삭제

메서드 이름	**coordinate_system.UF/TF.delete(** `index` **: int) -> StatusCodeEnum**
설명	현재 user/tool 좌표 세트에서 좌표계를 삭제합니다.
요청 매개변수	`index` : int 좌표계 인덱스입니다.
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.10.4 User/Tool 좌표계 업데이트

메서드 이름	**coordinate_system.UF/TF.update(`coordinate` : Coordinate) -> StatusCodeEnum**
설명	현재 user/tool 좌표 세트의 좌표계를 업데이트합니다.
요청 매개변수	`coordinate` : Coordinate 업데이트된 좌표계 정보입니다.
반환 값	StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.10.5 User/Tool 좌표계 가져오기

메서드 이름	**coordinate_system.UF/TF.get(`index` : int) -> tuple[Coordinate, StatusCodeEnum]**
설명	현재 user/tool 좌표 세트에서 지정된 좌표계를 가져옵니다.
요청 매개변수	`index` : int 좌표계 인덱스입니다.
반환 값	Coordinate: 좌표계 정보. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.10.6 User/Tool 좌표계 계산

메서드 이름	**coordinate_system.UF/TF.calculate(pose: List[Position]) -> tuple[Position, StatusCodeEnum]**
설명	입력된 포즈를 기반으로 user/tool 좌표 정보를 계산하고, 계산된 포즈를 반환합니다.

메서드 이름	**coordinate_system.UF/TF.calculate(pose: List[Position]) -> tuple[Position, StatusCodeEnum]**
요청 매개변수	`pose` : List[Position] 입력 포즈 목록입니다.
반환 값	Position: 계산된 포즈 정보. StatusCodeEnum: 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

coordinate_system.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 坐标系系统使用示例 / Example of coordinate system usage
"""

from Agilebot import (
    Arm,
    Coordinate,
    Position,
    StatusCodeEnum,
)

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the robot
arm = Arm()

# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret != StatusCodeEnum.OK:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

print("机器人连接成功 / Robot connected successfully")

# [ZH] 定义位姿数据用于计算工具坐标系
# [EN] Define pose data for calculating tool coordinate system
pose_data = [
    Position(
        341.6861424047297,
        -33.70972073115479,
        430.1721970894897,
        0.001,
        6.745,
        -180.000,
    ),
    Position(
        365.4597874970455,
        77.95089759481547,
        441.39040857936857,
        -7.343,
        12.620,
        138.857,
    ),
    Position(
        410.64702354574865,
        10.394172666192766,
        468.26089261578807,
        18.719,
        29.151,
        155.585,
    ),
    Position(
        483.2519847999948,
        112.71925218513972,
        448.39071038067624,
        33.947,
        69.714,
        133.597,
    ),
]

# [ZH] 根据位姿数据计算工具坐标系
```

```
# [EN] Calculate tool coordinate system based on pose data
pose, ret = arm.coordinate_system.TF.calculate(pose_data)
if ret != StatusCodeEnum.OK:
    print(
        f"计算工具坐标系失败，错误代码 / Calculate tool coordinate system faile
d, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "计算工具坐标系成功 / Calculate tool coordinate system successfully"
)
print(
    f"计算得到的位姿 / Calculated pose：X={pose.x}, Y={pose.y}, Z={pose.z}, A=
{pose.a}, B={pose.b}, C={pose.c}"
)

# [ZH] 创建工具坐标系对象
# [EN] Create tool coordinate system object
tf = Coordinate(5, "test_tf", "测试工具坐标系", pose)

# [ZH] 删除可能存在的ID为5的坐标系（避免冲突）
# [EN] Delete coordinate system with ID 5 if exists (avoid conflict)
arm.coordinate_system.TF.delete(5)

# [ZH] 添加工具坐标系名字 / Add tool coordinate system
ret = arm.coordinate_system.TF.add(tf)
if ret != StatusCodeEnum.OK:
    print(
        f"添加工具坐标系失败，错误代码 / Add tool coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "添加工具坐标系成功 / Add tool coordinate system successfully"
)

# [ZH] 获取工具坐标系列表
# [EN] Get tool coordinate system list
```

```
tf_list, ret = (
    arm.coordinate_system.TF.get_coordinate_list()
)
if ret != StatusCodeEnum.OK:
    print(
        f"获取工具坐标系列表失败，错误代码 / Get tool coordinate system list fail
ed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "获取工具坐标系列表成功 / Get tool coordinate system list successfully"
)
print(
    f"工具坐标系列表 / Tool coordinate system list: {tf_list}"
)

# [ZH] 获取指定的工具坐标系
# [EN] Get a specific tool coordinate system
tf, ret = arm.coordinate_system.TF.get(5)
if ret != StatusCodeEnum.OK:
    print(
        f"获取工具坐标系失败，错误代码 / Get tool coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "获取工具坐标系成功 / Get tool coordinate system successfully"
)
print(f"  TF ID: {tf.id}")
print(f"  TF Name: {tf.name}")
print(f"  TF Comment: {tf.comment}")
print(f"  X: {tf.data.x}, Y: {tf.data.y}, Z: {tf.data.z}")
print(f"  A: {tf.data.a}, B: {tf.data.b}, C: {tf.data.c}")

# [ZH] 更新工具坐标系的名称
# [EN] Update tool coordinate system name
tf.name = "updated_test_tf"
ret = arm.coordinate_system.TF.update(tf)
```

```
if ret != StatusCodeEnum.OK:
    print(
        f"更新工具坐标系失败，错误代码 / Update tool coordinate system failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "更新工具坐标系成功 / Update tool coordinate system successfully"
)

# [ZH] 删除工具坐标系
# [EN] Delete tool coordinate system
ret = arm.coordinate_system.TF.delete(5)
if ret != StatusCodeEnum.OK:
    print(
        f"删除工具坐标系失败，错误代码 / Delete tool coordinate system failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

print(
    "删除工具坐标系成功 / Delete tool coordinate system successfully"
)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.11 모드버스 기능

개요

Modbus 모듈은 Modbus 마스터 역할을 할 때 슬레이브 관리, read/write, 등록 및 직렬 포트 매개변수 구성, PLC, I/O 확장 모듈 및 기타 Modbus 장치와의 연결을 용이하게 하는 로봇의 기능을 캡슐화합니다.

핵심 기능

- 채널 및 슬레이브 ID로 Modbus 슬레이브 인스턴스 획득 지원
- 슬레이브에서 Modbus 코일 레지스터 읽기 및 쓰기 지원
- 슬레이브에서 Modbus 보유 레지스터 읽기 및 쓰기 지원
- 슬레이브에서 Modbus 개별 입력 레지스터 읽기 지원
- 슬레이브에서 Modbus 입력 레지스터 읽기 지원
- 직렬 통신 매개변수 설정 및 쿼리 지원

사용 사례

- PLC 장치와의 데이터 교환
- I/O 확장 모듈 제어
- 로봇과 기타 Modbus 장치 간의 연계 구현
- Modbus 직렬 통신 매개변수 구성 및 관리

메모

Modbus 기능은 로봇 소프트웨어의 버스 구성과 충돌하므로 동시에 사용할 수 없습니다.

4.11.1 Modbus 슬레이브 인스턴스 가져오기

메서드 이름	**modbus.get_slave(`channel` : ModbusChannel, `slave_id` : int, `master_id` : int = 0) -> 'Modbus.Slave'**
설명	지정된 채널 및 슬레이브 ID에 대한 Modbus 슬레이브 인스턴스를 가져옵니다.
요청 매개변수	`channel` : ModbusChannel Modbus 채널 `slave_id` : int 슬레이브 ID `master_id` : int 마스터 ID(기본값 0)
반환 값	Modbus.Slave: Modbus 슬레이브 인스턴스
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.2 슬레이브에서 Modbus 코일 레지스터 읽기

메서드 이름	**slave.read_coils(`address` : int, `number` : int) -> tuple[list[int], StatusCodeEnum]**
설명	슬레이브에서 Modbus 코일 레지스터를 읽습니다.
요청 매개변수	`address` : int 레지스터 주소 `number` : int 레지스터 개수(최대 120)
반환 값	list[int]: 레지스터 값 StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.3 슬레이브의 Modbus 코일 레지스터에 쓰기

메서드 이름	**slave.write_coils(`address` : int, `value` : list[int]) -> StatusCodeEnum**
설명	슬레이브의 Modbus 코일 레지스터에 씁니다.

메서드 이름	**slave.write_coils(address : int, value : list[int]) -> StatusCodeEnum**
요청 매개변수	address : int 레지스터 주소 value : list[int] 레지스터 값
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.4 슬레이브에서 Modbus 보유 레지스터 읽기

메서드 이름	**slave.read_holding_regs(address : int, number : int) -> tuple[list[int], StatusCodeEnum]**
설명	슬레이브에서 Modbus 보유 레지스터를 읽습니다.
요청 매개변수	address : int 레지스터 주소 number : int 레지스터 개수(최대 120)
반환 값	list[int]: 레지스터 값 StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.5 슬레이브의 Modbus 보유 레지스터에 쓰기

메서드 이름	**slave.write_holding_regs(address : int, value : list[int]) -> StatusCodeEnum**
설명	슬레이브의 Modbus 보유 레지스터에 씁니다.
요청 매개변수	address : int 레지스터 주소 value : list[int] 레지스터 값

메서드 이름	**slave.write_holding_regs(address : int, value : list[int]) -> StatusCodeEnum**
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.6 슬레이브에서 Modbus 개별 입력 레지스터 읽기

메서드 이름	**slave.read_discrete_inputs(address : int, number : int) -> tuple[list[int], StatusCodeEnum]**
설명	슬레이브에서 Modbus 개별 입력 레지스터를 읽습니다.
요청 매개변수	address : int 레지스터 주소 number : int 레지스터 개수(최대 120)
반환 값	list[int]: 레지스터 값 StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.7 슬레이브에서 Modbus 입력 레지스터 읽기

메서드 이름	**slave.read_input_regs(address : int, number : int) -> tuple[list[int], StatusCodeEnum]**
설명	슬레이브에서 Modbus 입력 레지스터를 읽습니다.
요청 매개변수	address : int 레지스터 주소 number : int 레지스터 개수(최대 120)
반환 값	list[int]: 레지스터 값 StatusCodeEnum: 함수 실행 결과

메서드 이름	slave.read_input_regs(address : int, number : int) -> tuple[list[int], StatusCodeEnum]
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

예제 코드

modbus.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: modbus使用示例 / Example of modbus usage
"""

from Agilebot import (
    Arm,
    ModbusChannel,
    SerialParams,
    StatusCodeEnum,
)

# [ZH] 初始化Arm类
# [EN] Initialize the robot
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置modbus参数
```

```
# [EN] Set Modbus parameters
params = SerialParams(
    channel=ModbusChannel.CONTROLLER_TCP_TO_485,
    ip="10.27.1.80",
    port=502,
)
id, ret_code = arm.modbus.set_parameter(params)
if ret_code == StatusCodeEnum.OK:
    print(
        "设置Modbus参数成功 / Set Modbus parameters successfully"
    )
else:
    print(
        f"设置Modbus参数失败，错误代码 / Set Modbus parameters failed, error co
de: {ret_code.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 创建从站
# [EN] Create a slave
slave = arm.modbus.get_slave(
    ModbusChannel.CONTROLLER_TCP_TO_485, 1, 1
)

# [ZH] 写入
# [EN] Write to registers
value = [1, 2, 3, 4]
ret = slave.write_coils(0, value)
if ret == StatusCodeEnum.OK:
    print("写入线圈成功 / Write coils successfully")
else:
    print(
        f"写入线圈失败，错误代码 / Write coils failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

ret = slave.write_holding_regs(0, value)
if ret == StatusCodeEnum.OK:
    print(
```

```
            "写入保持寄存器成功 / Write holding registers successfully"
        )
    else:
        print(
            f"写入保持寄存器失败，错误代码 / Write holding registers failed, error c
ode: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

    # [ZH] 读取
    # [EN] Read registers
    res, ret = slave.read_coils(0, 4)
    if ret == StatusCodeEnum.OK:
        print("读取线圈成功 / Read coils successfully")
        print(f"读取的线圈值 / Read coil values：{res}")
    else:
        print(
            f"读取线圈失败，错误代码 / Read coils failed, error code: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

    res, ret = slave.read_holding_regs(0, 4)
    if ret == StatusCodeEnum.OK:
        print(
            "读取保持寄存器成功 / Read holding registers successfully"
        )
        print(f"读取的寄存器值 / Read register values：{res}")
    else:
        print(
            f"读取保持寄存器失败，错误代码 / Read holding registers failed, error co
de: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

    res, ret = slave.read_input_regs(0, 4)
    if ret == StatusCodeEnum.OK:
        print(
            "读取输入寄存器成功 / Read input registers successfully"
        )
```

```
    print(
        f"读取的输入寄存器值 / Read input register values：{res}"
    )
else:
    print(
        f"读取输入寄存器失败，错误代码 / Read input registers failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

res, ret = slave.read_discrete_inputs(0, 4)
if ret == StatusCodeEnum.OK:
    print(
        "读取离散输入成功 / Read discrete inputs successfully"
    )
    print(
        f"读取的离散输入值 / Read discrete input values：{res}"
    )
else:
    print(
        f"读取离散输入失败，错误代码 / Read discrete inputs failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 结束后断开机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.11.8 직렬 통신 매개변수 설정

메서드 이름	**modbus.set_parameter(** `param` **:** SerialParams**) -> tuple[int, StatusCodeEnum]**
설명	직렬 통신 매개변수를 설정합니다.
요청 매개변수	`param` : SerialParams 직렬 통신 매개변수
반환 값	int: 채널 ID StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.11.9 직렬 통신 매개변수 가져오기

메서드 이름	**modbus.get_parameter(** `channel` **: ModbusChannel,** `master_id` **: int = 1) -> tuple[**SerialParams**, StatusCodeEnum]**
설명	직렬 통신 매개변수를 가져옵니다.
요청 매개변수	`channel` : ModbusChannel Modbus 채널 `master_id` : int 마스터 ID(기본값 1)
반환 값	SerialParams: 직렬 통신 매개변수 StatusCodeEnum: 기능 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.6.0.0+ 산업용(Bronze): v7.6.0.0+

4.12 BasScript 프로그램 클래스

개요

BasScript를 사용하면 모션, 로직, 프로그램 호출, 통신, 비전 및 Modbus 명령을 포함하여 호스트 PC에서 구조화된 방식으로 로봇에 대한 BAS 명령 시퀀스를 구성할 수 있습니다.

핵심 기능

- 빌딩 모션 명령 지원(MoveJoint, MoveLine, MoveCircle 등)
- 로직 명령어 구축 지원(If, While, Switch, Goto 등)
- 건물 할당, 대기, 일시 중지 및 중단 지침 지원
- 프로그램 호출 및 관리지시 구축 지원
- 소켓 통신 명령어 구축 지원
- Modbus 레지스터 read/write 명령어 구축 지원
- 빌딩 비전 프로그램 지침 지원
- 추가 매개변수 설정 지원(가속, RTCP, 프레임 오프셋 등)
- 퀵모션 명령어 지원(JUMP 시리즈)

사용 사례

- 복잡한 로봇 프로그램을 자동으로 생성
- 로봇 프로그램 편집 및 수정
- 공통 프로그램 모듈 및 명령어 시퀀스 재사용
- 호스트 PC와 로봇 프로그램 간의 원활한 통합 달성
- 맞춤형 로봇 모션 궤적 구축
- 비전, 통신, Modbus 기능을 로봇 프로그램에 통합

생성자

메서드 이름	BasScript(name : str) -> BasScript
설명	로봇에 대한 BAS 명령 시퀀스 구성
요청 매개변수	name : 스크립트 이름
호환되는 로봇 소프트웨어 버전	협업(Copper): v7.5.2.0+ 산업용(Bronze): v7.6.0.0+
메모	이 클래스의 모든 메서드에는 이 클래스와 동일한 호환성 요구 사항이 있습니다.

서브클래스 구조

BasScript 클래스에는 다양한 유형의 명령을 구성하기 위한 다음 하위 클래스가 포함되어 있습니다.

1. **ExtraParam**: 복잡한 로봇 제어 명령을 구축하기 위한 추가 매개변수 클래스
2. **BasMotion**: 모든 로봇 동작 관련 메서드를 포함하는 동작 명령 하위 클래스
3. **BasLogical**: 모든 논리 제어 관련 메서드를 포함하는 논리 명령어 하위 클래스
4. **BasStructure**: 모든 구조 제어 관련 메서드를 포함하는 구조 명령 서브클래스
5. **BasSocket**: 모든 소켓 통신 관련 메서드를 포함하는 소켓 통신 하위 클래스
6. **BasModbus**: 모든 Modbus 통신 관련 방법을 포함하는 Modbus 통신 하위 클래스
7. **BasVision**: 모든 비전 관련 메서드를 포함하는 비전 지침 하위 클래스

4.12.1 추가 매개변수

메서드 이름	ExtraParam() -> ExtraParam
설명	복잡한 로봇 제어 명령을 구축하기 위한 추가 매개변수 클래스
요청 매개변수	없음
반환 값	ExtraParam 객체

4.12.1.1 가속도 설정

메서드 이름	extra_param.acceleration(value : float) -> None
설명	가속도 설정, 범위는 1~120%
요청 매개변수	value : float 가속도 값, 단위: %(float)
반환 값	없음

4.12.1.2 RTCP 매개변수 설정

메서드 이름	extra_param.rtcp() -> None
설명	RTCP 매개변수 설정, MoveL 및 MoveC 명령어만 지원
요청 매개변수	없음
반환 값	없음

4.12.1.3 프레임 오프셋 설정

메서드 이름	extra_param.offset(index : int) -> None
설명	프레임 오프셋 설정
요청 매개변수	index : int 오프셋 인덱스(정수)
반환 값	없음

4.12.1.4 시간 기반 작업 또는 할당 설정

메서드 이름	extra_param.tb(second : int, type : str, name : str = None, index : int = None, status : str = None) -> None
설명	시간 기반 작업 또는 할당을 설정합니다. 범위는 0.01-30초이며, 0.01보다 작은 값은 성공적으로 저장되지 않습니다.
요청 매개변수	second : int 초, 단위: sec(정수) type : str 실행 유형(문자열) name : str 이름(연산용, 문자열)

메서드 이름	**extra_param.tb(second : int, type : str, name : str = None, index : int = None, status : str = None) -> None**
	index : int 인덱스(할당용, 정수) status : str 상태 (할당용, 문자열)
반환 값	없음

4.12.1.5 세트 점프

메서드 이름	**extra_param.skip(index : int) -> None**
설명	점프 설정, SKIP CONDITION 명령에 의해 설정된 점프 조건이 충족되면 SKIP 명령이 포함된 현재 라인에서 대상 명령 라인 또는 대상 프로그램으로 직접 점프합니다.
요청 매개변수	index : int 대상 라벨의 인덱스(정수)
반환 값	없음

4.12.1.6 출발 거리 및 접근 거리 설정

메서드 이름	**extra_param.approach(departure_dist : float, approaching_dist : float) -> None**
설명	문형 모션의 출발거리와 접근거리를 설정하며 JUMP 명령에만 사용됩니다.
요청 매개변수	departure_dist : float 출발 거리, 단위: mm(float) approaching_dist : float 접근 거리, 단위: mm(float)
반환 값	없음

4.12.2 바스모션

4.12.2.1 MoveJoint 모션-포인트 명령

메서드 이름	**motion.move_joint(** pose_type **:** MovePoseType**,** pose_index **: int,** speed_type **:** SpeedType**,** speed_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	Joint 모션 명령은 지정된 이동 속도와 방법으로 로봇을 작업 공간의 지정된 위치로 이동합니다.
요청 매개변수	pose_type : MovePoseType 포즈 유형 pose_index : int 포즈 인덱스 speed_type : SpeedType 속도 유형 speed_value : float 속도 값, 단위: % smooth_type : SmoothType Smooth type smooth_distance : float Smooth 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.2.2 MoveLine 선형 모션-포인트 명령어

메서드 이름	**motion.move_line(** pose_type **:** MovePoseType**,** pose_index **: int,** speed_type **:** SpeedType**,** speed_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	선형 동작 명령은 로봇을 지정된 이동 속도와 방법으로 작업 공간의 지정된 위치까지 직선으로 이동시킵니다.
요청 매개변수	pose_type : MovePoseType 포즈 유형 pose_index : int 포즈 인덱스 speed_type : SpeedType 속도 유형 speed_value : float 속도 값, 단위: mm/s smooth_type : SmoothType Smooth 유형 smooth_distance : float Smooth 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.2.3 MoveCircle 호 모션 투 포인트 명령어

메서드 이름	**motion.move_circle(** pose_type1 **:** MovePoseType**,** pose_index1 **: int,** pose_type2 **:** MovePoseType**,** pose_index2 **: int,** speed_type **:** SpeedType**,** speed_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	원호 모션 명령은 지정된 이동 속도와 방법으로 로봇을 작업 공간의 지정된 위치로 호 형태로 이동시킵니다.
요청 매개변수	pose_type1 : MovePoseType 첫 번째 포즈 유형 pose_index1 : int 첫 번째 포즈 인덱스 pose_type2 : MovePoseType 두 번째 포즈 유형 pose_index2 : int 두 번째 포즈 인덱스 speed_type : SpeedType 속도 유형 speed_value : 부동 속도 값, 단위: mm/s smooth_type : SmoothType 부드러운 유형 smooth_distance : float 부드러운 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.2.4 JUMP 퀵모션 명령어

메서드 이름	**motion.move_jump(** pose_type **:** MovePoseType**,** pose_index **: int,** speed_value **: float,** speed_ratio **: float,** lim_Z_type **:** SpeedType**,** lim_Z_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	도어형 모션 명령은 로봇이 도어형 모션을 수행하도록 지정합니다(첫 번째 수직 상승, 다음 수평 이동, 마지막으로 수직 하강).
요청 매개변수	pose_type : MovePoseType 대상 포즈 저장 유형 pose_index : int 대상 위치 인덱스 speed_value : float 이동 속도 값, 단위: mm/s speed_ratio : float 이동 속도 비율, 단위: % lim_Z_type : SpeedType Z축 제한 유형 lim_Z_value : 부동 Z축 제한 값 smooth_type : SmoothType Smooth 유형 smooth_distance : 부동 Smooth 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수

메서드 이름	**motion.move_jump(** pose_type **:** MovePoseType**,** pose_index **: int,** speed_value **: float,** speed_ratio **: float,** lim_Z_type **:** SpeedType**,** lim_Z_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.2.5 JUMP3 쿽모션 명령어

메서드 이름	**motion.move_jump3(** pose_type **:** MovePoseType**,** pose_index **: List[int],** speed_value **: float,** speed_ratio **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	도어 유형 모션 명령은 출발, 접근 및 목표의 세 가지 위치로 도어 유형 모션을 수행하도록 로봇을 지정합니다.
요청 매개변수	pose_type : MovePoseType 타겟 포즈 저장 유형 pose_index : List[int] 3개의 타겟 위치 인덱스 목록(출발, 접근, 타겟) speed_value : float 이동 속도 값, 단위: mm/s speed_ratio : float 이동 속도 비율, 단위: % smooth_type : SmoothType Smooth 유형 smooth_distance : float 부드러운 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.2.6 JUMP3CP 쿽모션 명령어

메서드 이름	**motion.move_jump3cp(** pose_type **:** MovePoseType**,** pose_index **: List[int],** speed_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
설명	도어 유형 모션 명령은 출발, 접근 및 목표의 세 가지 위치로 도어 유형 모션을 수행하도록 로봇을 지정합니다.
요청 매개변수	pose_type : MovePoseType 타겟 포즈 저장 유형 pose_index : List[int] 3개의 타겟 위치 인덱스 목록(출발, 접근, 타겟) speed_value : float 이동 속도 값, 단위: mm/s

메서드 이름	**motion.move_jump3cp(** pose_type **:** MovePoseType**,** pose_index **: List[int],** speed_value **: float,** smooth_type **:** SmoothType**,** smooth_distance **: float = 0.0,** extra_param **: ExtraParam = None) -> StatusCodeEnum**
	smooth_type : SmoothType Smooth 유형 smooth_distance : float Smooth 거리, 단위: mm, 범위: 0~1000 extra_param : ExtraParam 추가 매개변수
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3 BasLogical

4.12.3.1 LogiIf 조건부 명령어

메서드 이름	**logical.logi_if(** param1 **: Union[**RegisterType**,** IOType**],** index **: int,** param2 **: Union[**RegisterType**,** IOType**,** OtherType**],** value **: Union[int, float, str,** IOStatus**],** operator **:** BooleanOperator **=** BooleanOperator**.EQ) -> StatusCodeEnum**
설명	조건부 판단 명령, 지정된 조건에 따라 다른 코드 블록을 실행합니다.
요청 매개변수	param1 : Union[RegisterType, IOType] 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[RegisterType, IOType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 유형) value : Union[int, float, str, IOStatus] 매개변수 2의 인덱스 또는 값 operator : BooleanOperator 논리 연산자
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.2 LogiElseIf 조건부 분기 명령어

메서드 이름	**logical.logi_else_if(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum**
설명	조건 분기 명령, If 조건이 만족되지 않을 때 ElseIf 조건 충족 여부를 판단합니다.
요청 매개변수	param1 : Union[RegisterType, IOType] 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[ValueType, IOType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 유형) value : Union[int, float, IOStatus] 매개변수 2의 인덱스 또는 값 operator : BooleanOperator 논리 연산자
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.3 LogiElse Else 명령어

메서드 이름	**logical.logi_else() -> StatusCodeEnum**
설명	Else 명령은 모든 If 및 ElseIf 조건이 충족되지 않을 때 실행됩니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.4 LogiEndIf 조건부 명령어 종료

메서드 이름	**logical.logi_end_if() -> StatusCodeEnum**
설명	조건문 종료 명령은 If 조건문 블록을 종료하는 데 사용됩니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.5 LogiWhile 루프 명령어

메서드 이름	**logical.logi_while(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum**
설명	루프 명령, 조건이 충족되면 루프 본문의 코드를 반복적으로 실행합니다.
요청 매개변수	param1 : Union[RegisterType, IOType] 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[RegisterType, IOType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 유형) value : Union[int, float, str, IOStatus] 매개변수 2의 인덱스 또는 값 operator : BooleanOperator 논리 연산자
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.6 LogiEndWhile 종료 루프 명령어

메서드 이름	**logical.logi_end_while() -> StatusCodeEnum**
설명	루프 종료 명령은 While 루프 문 블록을 종료하는 데 사용됩니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.7 LogiSwitch 다중 브랜치 선택 지침

메서드 이름	**logical.logi_switch(param : Union[RegisterType, IOType], index : int) -> StatusCodeEnum**
설명	다중 분기 선택 명령, 표현식의 값에 따라 실행할 다른 분기를 선택하고 분기를 종료하는 BREAK 문을 지원합니다.
요청 매개변수	param : Union[RegisterType, IOType] 레지스터 또는 IO 신호 index : int 인덱스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.8 LogiCase 분기 지침

메서드 이름	**logical.logi_case(param : Union[RegisterType, IOType, OtherType], value : Union[int, float, str]) -> StatusCodeEnum**
설명	Switch 문에서 분기 조건을 정의하는 데 사용되는 분기 명령어
요청 매개변수	param : Union[RegisterType, IOType, OtherType] 레지스터, IO 신호 또는 기타 유형 value : Union[int, float, str] 인덱스 번호 또는 값
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.9 LogiDefault 기본 분기 명령어

메서드 이름	**logical.logi_default() -> StatusCodeEnum**
설명	기본 분기 명령어는 모든 Case 조건이 충족되지 않을 때 실행됩니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.10 LogiEndSwitch 종료 다중 브랜치 선택 명령어

메서드 이름	**logical.logi_end_switch() -> StatusCodeEnum**
설명	Switch 문 블록을 종료하는 데 사용되는 다중 분기 선택 종료 명령
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.11 LogiGoto 점프 지침

메서드 이름	**logical.logi_goto(index : int) -> StatusCodeEnum**
설명	프로그램을 동일한 프로그램 내의 지정된 레이블 위치로 점프하고 해당 위치에서 계속 실행하는 데 사용되는 점프 명령어. LABEL 명령을 먼저 삽입한 다음 GOTO 명령을 사용해야 합니다.
요청 매개변수	index : int 대상 라벨의 인덱스

메서드 이름	**logical.logi_goto(index : int) -> StatusCodeEnum**
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.12 LogiLabel 라벨 지침

메서드 이름	**logical.logi_label(index : int) -> StatusCodeEnum**
설명	프로그램에서 점프 대상 위치를 정의하는 데 사용되는 라벨 명령어, GOTO 명령어는 이 라벨 위치로 점프할 수 있습니다.
요청 매개변수	index : 정수 라벨 인덱스
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.13 LogiSkipCondition 조건 건너뛰기 명령어

메서드 이름	**logical.logi_skip_condition(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum**
설명	건너뛰기 조건 명령은 점프 조건을 설정하는 데 사용됩니다. 조건이 충족되면 SKIP 명령이 포함된 현재 라인에서 대상 명령 라인 또는 대상 프로그램으로 직접 점프합니다.
요청 매개변수	param1 : Union[RegisterType, IOType] 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[RegisterType, IOType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 유형) value : Union[int, float, str, IOStatus] 매개변수 2의 인덱스 또는 값 operator : BooleanOperator 논리 연산자
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.14 LogiBreak 브레이크 루프 명령어

메서드 이름	logical.logi_break() -> StatusCodeEnum
설명	현재 루프 또는 Switch 문을 종료하는 데 사용되는 루프 중단 명령
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.3.15 LogiContinue 루프 건너뛰기 명령어

메서드 이름	logical.logi_continue() -> StatusCodeEnum
설명	현재 루프의 나머지 부분을 건너뛰고 다음 루프로 들어가는 데 사용되는 루프 건너뛰기 명령
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4 기본구조

4.12.4.1 대기 대기 조건 지시

메서드 이름	structure.wait(param1 : Union[RegisterType, IOType], index : int, param2 : Union[ValueType, IOType, OtherType], value : Union[int, float, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
설명	대기 조건 명령, 조건이 충족될 때만 WAIT 명령을 실행하고, 그렇지 않으면 조건이 충족될 때까지 기다립니다.
요청 매개변수	param1 : Union[RegisterType, IOType] 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[ValueType, IOType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 유형) value : Union[int, float, IOStatus] 매개변수 2의 인덱스 또는 값 operator : BooleanOperator 논리 연산자
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.2 WaitTime 대기 시간 지시사항

메서드 이름	**structure.wait_time(param : ValueType, value : Union[int, float]) -> StatusCodeEnum**
설명	대기 시간 명령, WAIT TIME 명령 실행, 로봇은 후속 명령 실행을 계속하기 전에 지정된 시간 동안 기다립니다.
요청 매개변수	param : ValueType 매개변수 유형(R 레지스터 또는 값) value : Union[int, float] 시간 값, 단위: 초
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.3 일시정지 일시정지 안내

메서드 이름	**structure.pause() -> StatusCodeEnum**
설명	일시 정지 명령은 일시 정지 명령을 실행할 때 프로그램 실행을 일시 정지하고 로봇은 즉시 감속 궤도를 계획하고 정지하며 프로그램은 일시 정지 상태로 들어갑니다. 계속 실행하려면 시작 버튼을 다시 눌러야 합니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.4 중단 중단 명령

메서드 이름	**structure.abort() -> StatusCodeEnum**
설명	강제 종료 명령: 프로그램 실행을 종료하고, 로봇 서보가 즉시 제동하고, 브레이크가 적용되고, 서보 버스의 전원이 꺼집니다. 강제 종료 명령을 실행한 후 프로그램은 종료 상태로 들어가고 커서는 현재 행에서 멈춥니다.
요청 매개변수	없음

메서드 이름	structure.abort() -> StatusCodeEnum
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.5 호출 동기 프로그램 호출 명령

메서드 이름	structure.call(`name` : str) -> StatusCodeEnum
설명	동기 프로그램 호출 명령, 지정된 프로그램을 호출하고 현재 프로그램을 계속 실행하기 전에 실행이 완료될 때까지 기다림, 사전 설정된 BAS 프로그램 호출 지원
요청 매개변수	`name` : str 프로그램 이름
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.6 비동기 프로그램 호출 명령 실행

메서드 이름	structure.run(`name` : str) -> StatusCodeEnum
설명	비동기 프로그램 호출 명령, 지정된 프로그램 호출 및 즉시 반환, 현재 프로그램 실행 계속, 사전 설정된 BAS 프로그램 호출 지원
요청 매개변수	`name` : str 프로그램 이름
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.7 로드 로드 프로그램 지침

메서드 이름	structure.load(`name` : str) -> StatusCodeEnum
설명	프로그램 로드 명령, 지정된 프로그램을 메모리에 로드합니다.
요청 매개변수	`name` : str 프로그램 이름

메서드 이름	structure.load(name : str) -> StatusCodeEnum
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.8 언로드 언로드 프로그램 지침

메서드 이름	structure.unload(name : str) -> StatusCodeEnum
설명	프로그램 언로드 명령, 메모리에서 지정된 프로그램을 언로드합니다.
요청 매개변수	name : str 프로그램 이름
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.4.9 실행 프로그램 실행 명령

메서드 이름	structure.exec(name : str) -> StatusCodeEnum
설명	프로그램 명령 실행, 지정된 프로그램을 실행합니다.
요청 매개변수	name : str 프로그램 이름
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.5 BasSocket

4.12.5.1 SocketOpen 오픈 소켓 연결 지침

메서드 이름	socket.socket_open(index : int) -> StatusCodeEnum
설명	소켓 열기 명령을 사용하여 서버를 생성하고 클라이언트 연결을 기다립니다.
요청 매개변수	index : int SK 레지스터 시퀀스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.5.2 SocketClose 닫기 소켓 연결 지침

메서드 이름	socket.socket_close(index : int) -> StatusCodeEnum
설명	지정된 소켓 연결을 닫습니다.
요청 매개변수	index : int SK 레지스터 시퀀스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.5.3 SocketConnect 연결 소켓 명령어

메서드 이름	socket.socket_connect(index : int) -> StatusCodeEnum
설명	소켓 연결 명령을 사용하여 지정된 소켓 서버에 연결
요청 매개변수	index : int SK 레지스터 시퀀스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.5.4 SocketSend 소켓 데이터 전송 명령

메서드 이름	socket.socket_send(index : int, msg_type : StrType, value : Union[int, str]) -> StatusCodeEnum
설명	소켓 전송 명령어를 사용하여 지정된 소켓 연결로 데이터를 보냅니다.
요청 매개변수	index : int SK 레지스터 시퀀스 번호 msg_type : StrType 메시지 유형 value : Union[int, str] 메시지 내용 또는 시퀀스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.5.5 SocketRecv 소켓 데이터 수신 명령어

메서드 이름	socket.socket_recv(index : int, msg_lenth : int, msg_type : StrType, value : Union[int, str]) -> StatusCodeEnum
설명	Socket Recv 명령어를 사용하여 지정된 소켓 연결에서 문자를 읽고 최대 길이를 지정할 수 있습니다.

메서드 이름	**socket.socket_recv(index : int, msg_lenth : int, msg_type : StrType, value : Union[int, str]) -> StatusCodeEnum**
요청 매개변수	index : int SK 레지스터 시퀀스 번호 msg_lenth : int 최대 메시지 길이 msg_type : StrType 메시지 유형 value : Union[int, str] 메시지 내용 또는 시퀀스 번호
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.6 BasModbus

4.12.6.1 ModbusReadMH Modbus 보유 레지스터 읽기 명령 읽기

메서드 이름	**modbus.modbus_read_MH(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum**
설명	Modbus 보유 레지스터 명령 읽기
요청 매개변수	index : int 채널 시퀀스 번호 id : int Modbus ID address : int 레지스터 시작 주소 length : int 레지스터 길이, 즉 읽을 레지스터 수 r_index : int R 레지스터 쓰기를 위한 시작 시퀀스 번호 결과
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.6.2 ModbusReadMI Modbus 입력 레지스터 읽기 명령

메서드 이름	**modbus.modbus_read_MI(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum**
설명	Modbus 입력 레지스터 명령 읽기
요청 매개변수	index : int 채널 시퀀스 번호 id : int Modbus ID address : int 레지스터 시작 주소

메서드 이름	**modbus.modbus_read_MI(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum**
	length : int 레지스터 길이, 즉 읽을 레지스터 수 r_index : int R 레지스터 쓰기를 위한 시작 시퀀스 번호 결과
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.6.3 ModbusWriteMH Modbus 홀딩 레지스터 쓰기 명령어

메서드 이름	**modbus.modbus_write_MH(index : int, id : int, address : int, length : int, value_type : ValueType, value : int) -> StatusCodeEnum**
설명	Modbus 보유 레지스터 명령어 작성
요청 매개변수	index : int 채널 시퀀스 번호 id : int Modbus ID address : int 레지스터 시작 주소 length : int 레지스터 길이, 즉 쓸 레지스터 수 value_type : ValueType 값 유형 value : int 값 또는 R 레지스터 시작 인덱스
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.7 바스비전

4.12.7.1 VisionFind 비전 찾기 프로그램 안내

메서드 이름	**vision.vision_find(name : str) -> StatusCodeEnum**
설명	비전찾기 프로그램 실행
요청 매개변수	name : str 비전 프로그램 이름
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.7.2 VisionGetOffset 비전 프로그램 오프셋 명령 가져오기

메서드 이름	**vision.vision_get_offset(name : str, index : int, label_index : int) -> StatusCodeEnum**
설명	비전 프로그램 실행 후 오프셋 가져오기
요청 매개변수	name : str 비전 프로그램 이름 index : int 비전 레지스터 인덱스 label_index : int 라벨 인덱스
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.7.3 VisionGetQuantity 비전 프로그램 결과 지시사항 가져오기

메서드 이름	**vision.vision_get_quantity(name : str, index : int) -> StatusCodeEnum**
설명	비전 프로그램 실행 후 결과 수량 가져오기
요청 매개변수	name : str 비전 프로그램 이름 index : int R 결과 저장을 위한 레지스터 인덱스
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.8 직접 방법

4.12.8.1 SetParam 매개변수 설정 명령

메서드 이름	**bas_script.set_param(type : ParamType, value_type : ValueType, value : Union[int, float]) -> StatusCodeEnum**
설명	로봇의 다양한 매개변수를 설정하는 데 사용되는 매개변수 설정 명령
요청 매개변수	type : ParamType 매개변수 유형 value_type : ValueType 값 유형 value : Union[int, float] 값
반환 값	StatusCodeEnum: 함수 실행 결과

4.12.8.2 AssignValue 할당 지침

메서드 이름	**bas_script.assign_value(** param1 **:** AssignType**,** index **: int,** param2 **:** **Union[**AssignType**,** OtherType**],** value **: Union[int, float, str,** IOStatus**,** CurrentPose**],** opt_index **: int = 0,** opt_value **= 0) -> StatusCodeEnum**
설명	레지스터, IO 신호 등에 값을 할당하는 데 사용되는 할당 명령입니다.
요청 매개변수	param1 : AssignType 첫 번째 매개변수(레지스터 또는 IO 신호) index : int 매개변수 1의 인덱스 param2 : Union[AssignType, OtherType] 두 번째 매개변수(레지스터, IO 신호 또는 기타 type) value : Union[int, float, str, IOStatus, CurrentPose] 매개변수 2의 인덱스 또는 값 opt_index : int 매개변수 1이 PR_ELEMENT인 경우 추가 인덱스 opt_value : int 파라미터 2가 PR_ELEMENT일 때 추가 인덱스, 값이 IOStatus.PULSE일 때 펄스 값
반환 값	StatusCodeEnum: 함수 실행 결과

4.13 로봇티칭모션

개요

조깅 모듈은 단일 축 스테핑, 연속 조깅, 다축 조정 모션 및 비상 정지를 지원하는 티치 모드에서 로봇에 대한 조그 제어 인터페이스를 제공합니다.

핵심 기능

- 로봇 단축 스텝 티칭 모션 지원
- 로봇 연속 조깅 티칭 동작 지원
- 로봇 다축 연속 티칭 모션 지원
- 비상 정지 로봇 연속 동작 지원
- 스텝 길이 및 회전 스텝 각도 설정 지원
- 다양한 좌표계에서의 조깅 제어 지원

사용 사례

- 로봇 디버깅 및 교육 프로세스
- Position 미세 조정 시나리오
- 호스트 PC 원격 수동 자세 조정
- 로봇 설치 및 교정
- 복잡한 궤적의 정확한 조정

4.13.1 단축 스텝 티칭 모션

메서드 이름	**jogging.step_move(** `aj_num` **: int,** `step_length` **: float = 0,** `step_angle` **: float = 0) -> StatusCodeEnum**
설명	구성된 스텝 크기에 따라 단축 스텝 티칭 동작을 수행합니다.

메서드 이름	**jogging.step_move(** `aj_num` **: int,** `step_length` **: float = 0,** `step_angle` **: float = 0) -> StatusCodeEnum**
요청 매개변수	`aj_num` : int 값 1-6은 현재 좌표계의 축에 해당합니다. 데카르트 좌표에서 x, y, z, rx, ry, rz는 1-6에 매핑됩니다. 기호는 동작 방향을 나타냅니다. `step_length` : float 선형 스텝 길이(mm); 현재 단계 크기를 유지하기 위해 생략되었습니다. `step_angle` : float 회전 단계 각도(도); 현재 회전 단계를 유지하기 위해 생략되었습니다.
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

jogging/step_move.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 示教运动功能-步进关节点动运动使用示例 / Teaching motion function - Joint step motion usage example
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1
# [EN] Jog joint 1 under the joint coordinate system
ret = arm.jogging.step_move(1, 2, 2)
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.13.2 단축 연속 조깅 티칭 모션

메서드 이름	**jogging.continuous_move(aj_num : int) -> StatusCodeEnum**
설명	단축 연속 조깅 티칭 동작을 수행합니다.
요청 매개변수	aj_num : int 축 인덱스(현재 좌표계의 경우 1~6; 데카르트의 경우 x/y/z/rx/ry/rz는 1~6으로 매핑되며 부호는 방향을 나타냄).
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

jogging/continuous_move.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 示教运动功能-单关节点动运动使用示例 / Example of Teaching motion fun
ction - Single-joint motion usage
"""

import time

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1
# [EN] Jog joint 1 under the joint coordinate system
ret = arm.jogging.continuous_move(1)
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
```

```
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 运动3s
# [EN] Move for 3 seconds
time.sleep(3)

# [ZH] 停止
# [EN] Stop
arm.jogging.stop()

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.13.3 다축 연속 티칭 모션

메서드 이름	**jogging.multi_move(`aj_num` : List[int]) -> StatusCodeEnum**
설명	다축 연속 티칭 동작을 수행합니다.
요청 매개변수	`aj_num` : List[int] 축 인덱스 목록(현재 좌표계의 경우 1~6; 데카르트에서는 x/y/z/rx/ry/rz가 1~6으로 매핑되며 기호는 방향을 나타냄).
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

예제 코드

jogging/multi_move.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 示教运动功能-多关节点动运动使用示例 / / Teaching motion function - M
ulti-joint motion usage example
"""

import time

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1,2,3
# [EN] Jog joint 1,2,3 under the joint coordinate system
ret = arm.jogging.multi_move([1, 2, 3])
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error cod
```

```
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 运动3s
# [EN] Move for 3 seconds
time.sleep(3)

# [ZH] 停止
# [EN] Stop
arm.jogging.stop()

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from the robot
arm.disconnect()
print(
    "机器人断开连接成功 / Robot disconnected successfully"
)
```

4.13.4 로봇 연속 이동 중지

메서드 이름	jogging.stop()
설명	로봇 조그(JOG) 동작을 종료합니다.
요청 매개변수	매개변수 없음
반환 값	StatusCodeEnum: 함수 실행 결과
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

4.14 로봇 구독 및 게시 인터페이스

개요

SubPub 모듈은 로봇 컨트롤러의 WebSocket 기반 게시/subscribe 채널을 처리합니다. 연결을 설정하고 로봇 상태, 레지스터 및 I/O,와 같은 주제를 구독하며 콜백 또는 호출 차단을 통해 실시간 데이터를 전달합니다.

핵심 기능

- WebSocket 서버 연결 및 연결 끊기 지원
- 로봇 상태 데이터 구독 지원
- 데이터 등록 구독 지원
- IO 신호 데이터 구독 지원
- 콜백 기능을 통한 메시지 수신 지원
- 다음 메시지 수신 차단 지원
- 구독 빈도 설정 지원
- 메시지 수신 오류 처리 지원

사용 사례

- 호스트 PC에서 로봇 동작 상태 실시간 모니터링
- 로봇 데이터 시각화 구현
- 외부 시스템과의 데이터 연계
- 레지스터 및 IO 상태 실시간 모니터링
- 로봇 모니터링 및 제어 시스템 구축
- 로봇 상태에 대한 실시간 알람 및 알림 구현

4.14.1 WebSocket 서버에 연결

메서드 이름	**sub_pub.connect() -> StatusCodeEnum**
설명	로봇 컨트롤러 WebSocket 서버에 연결
요청 매개변수	없음
반환 값	StatusCodeEnum: 연결 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.2 WebSocket 연결 끊기

메서드 이름	**sub_pub.disconnect() -> StatusCodeEnum**
설명	로봇 컨트롤러 WebSocket 서버 연결 끊기
요청 매개변수	없음
반환 값	StatusCodeEnum: 연결 끊김 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.3 로봇 상태 구독 추가

메서드 이름	**sub_pub.subscribe_status(** `topics` **: List[RobotTopicType],** `frequency` **: int = 200) -> StatusCodeEnum**
설명	로봇 상태 데이터 구독 추가
요청 매개변수	`topics` : List[RobotTopicType] 로봇 토픽 유형 목록 `frequency` : int 구독 빈도(Hz, 기본값 200)
반환 값	StatusCodeEnum: 구독 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.4 등록 구독 추가

메서드 이름	**sub_pub.subscribe_register(reg_type : RegTopicType, reg_ids : List[int], frequency : int = 200) -> StatusCodeEnum**
설명	등록 데이터 구독 추가
요청 매개변수	reg_type : RegTopicType 레지스터 유형 reg_ids : List[int] 레지스터 ID 목록 frequency : int 구독 주파수(Hz, 기본값 200)
반환 값	StatusCodeEnum: 구독 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.5 IO 구독 추가

메서드 이름	**sub_pub.subscribe_io(io_list : List[tuple[IOTopicType, int]], frequency : int = 200) -> StatusCodeEnum**
설명	디지털 입력을 포함한 IO 신호 데이터 구독/outputs
요청 매개변수	io_list : List[tuple[IOTopicType, int]] IO 목록(각 요소는 (IO type, IO ID) 입니다) frequency : int 구독 빈도(Hz, 기본값 200)
반환 값	StatusCodeEnum: 구독 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.6 메시지 수신 시작

메서드 이름	**sub_pub.start_receiving(on_message_received : Callable[[Dict[str, Any]], None]) -> StatusCodeEnum**
설명	콜백 함수를 통해 구독 메시지 수신을 시작하고 수신된 데이터를 처리합니다.
요청 매개변수	on_message_received : Callable[[Dict[str, Any]], None] 메시지 수신 콜백 함수
반환 값	StatusCodeEnum: 상태 코드 수신 중
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.7 수신 메시지 오류 처리

메서드 이름	**sub_pub.handle_receive_error() -> StatusCodeEnum**
설명	구독 메시지 수신 시 오류를 처리합니다. 수신 콜백에서 예외가 발생하면 루프를 종료합니다.
요청 매개변수	없음
반환 값	StatusCodeEnum: 상태 코드 수신 중
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.14.8 다음 메시지 수신

메서드 이름	**sub_pub.receive() -> tuple[Dict[str, Any], StatusCodeEnum]**
설명	다음 메시지를 수신하고 반환합니다.
요청 매개변수	없음
반환 값	tuple[Dict[str, Any], StatusCodeEnum]: 수신된 메시지 사전 및 상태 코드
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

예제 코드

sub_pub.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: SubPub使用示例 / SubPub usage example
"""

import asyncio
import logging

from Agilebot import (
    Arm,
    IOTopicType,
    RegTopicType,
    RobotTopicType,
    StatusCodeEnum,
)

# 配置日志 / Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


async def message_handler(message):
    """
    消息处理函数 / Message handler function

    :param message: 接收到的消息字典 / Received message dictionary
    """
    logger.info(f"收到消息 / Received message: {message}")


async def main():
    """
    主函数，演示SubPub的使用 / Main function demonstrating SubPub usage
    """
    # 创建Arm实例 / Create Arm instance
    arm = Arm()
```

```python
# 连接到控制器 / Connect to controller
ret = arm.connect("10.27.1.254")
if ret != StatusCodeEnum.OK:
    logger.error(f"连接失败 / Connection failed: {ret}")
    return

logger.info("Arm连接成功 / Arm connected successfully")

try:
    # 连接WebSocket / Connect WebSocket
    ret = await arm.sub_pub.connect()
    if ret != StatusCodeEnum.OK:
        logger.error(
            f"WebSocket连接失败 / WebSocket connection failed: {ret}"
        )
        return

    logger.info(
        "WebSocket连接成功 / WebSocket connected successfully"
    )

    # 订阅机器人状态 / Subscribe to robot status
    topic_types = [
        RobotTopicType.JOINT_POSITION,
        RobotTopicType.CARTESIAN_POSITION,
        RobotTopicType.ROBOT_STATUS,
        RobotTopicType.CTRL_STATUS,
        RobotTopicType.SERVO_STATUS,
        RobotTopicType.INTERPRETER_STATUS,
        RobotTopicType.TRAJECTORY_RECORD,
        RobotTopicType.USER_OP_MODE,
        RobotTopicType.USER_FRAME,
        RobotTopicType.TOOL_FRAME,
        RobotTopicType.VELOCITY_RATIO,
        RobotTopicType.TRAJECTORY_RECORD,
    ]

    ret = await arm.sub_pub.subscribe_status(
        topic_types, frequency=200
    )
    if ret != StatusCodeEnum.OK:
```

```
            logger.error(
                f"订阅机器人状态失败 / Failed to subscribe to robot status: {re
t}"
            )
            return

        logger.info(
            "机器人状态订阅成功 / Robot status subscription successful"
        )

        # 订阅寄存器 / Subscribe to registers
        ret = await arm.sub_pub.subscribe_register(
            RegTopicType.R, [1, 2, 3], frequency=200
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"订阅寄存器失败 / Failed to subscribe to registers: {ret}"
            )
            return

        logger.info(
            "寄存器订阅成功 / Register subscription successful"
        )

        # 订阅IO信号 / Subscribe to IO signals
        io_list = [(IOTopicType.DI, 1), (IOTopicType.DO, 1)]

        ret = await arm.sub_pub.subscribe_io(
            io_list, frequency=200
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"订阅IO失败 / Failed to subscribe to IO: {ret}"
            )
            return

        logger.info(
            "IO订阅成功 / IO subscription successful"
        )

        # 单次接收消息示例 / Single message receive example
        logger.info(
```

```
            "尝试单次接收消息 / Attempting single message receive"
        )
        try:
            # 设置超时 / Set timeout
            message, ret = await asyncio.wait_for(
                arm.sub_pub.receive(), timeout=5.0
            )
            if ret == StatusCodeEnum.OK:
                logger.info(
                    f"单次接收消息成功 / Single message receive successful: {m
essage}"
                )
            else:
                logger.error(
                    f"单次接收消息失败 / Single message receive failed: {ret}"
                )
        except asyncio.TimeoutError:
            logger.warning(
                "接收消息超时 / Message receive timeout"
            )

        # 启动消息接收 / Start message receiving
        ret = await arm.sub_pub.start_receiving(
            message_handler
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"启动消息接收失败 / Failed to start message receiving: {ret}"
            )
            return

        logger.info(
            "开始接收消息 / Started receiving messages"
        )

        # 运行一段时间 / Run for a while
        logger.info(
            "运行10秒钟... / Running for 10 seconds..."
        )
        await asyncio.sleep(10)

    except Exception as e:
```

```
        logger.error(
            f"运行过程中发生错误 / Error occurred during execution: {str(e)}"
        )

    finally:
        # 断开连接 / Disconnect
        await arm.sub_pub.disconnect()
        arm.disconnect()
        logger.info("连接已断开 / Connection disconnected")


if __name__ == "__main__":
    # 运行异步主函数 / Run async main function
    asyncio.run(main())
```

4.15 확장 관리

개요

확장 모듈은 로봇 컨트롤러의 타사 플러그인을 관리합니다. 컨트롤러의 IP를 검색하고, 설치된 플러그인을 /view 나열하고, /disable를 활성화하고, 해당 서비스를 호출할 수 있습니다.

핵심 기능

- 로봇 IP 주소 얻기 지원
- 확장자 목록 가져오기 지원
- 단일 확장 세부정보 가져오기 지원
- 확장 활성화/disable 상태 전환 지원
- 확장 서비스 명령 호출 지원
- 연장 수명주기의 중앙 집중식 관리 지원

사용 사례

- 호스트 PC에서 로봇 컨트롤러의 타사 플러그인 관리
- 확장에서 제공하는 서비스를 호출하여 로봇 기능 확장
- 확장 활성화 상태 모니터링 및 제어
- 확장 또는 티칭 펜던트 환경에서 로봇 컨트롤러에 연결
- 타사 시스템과 로봇 통합

4.15.1 로봇의 IP 주소 얻기

메서드 이름	extension.get_robot_ip() -> str
설명	SDK가 실행되는 환경을 기준으로 해당 로봇 IP 주소를 반환합니다.
요청 매개변수	없음

메서드 이름	extension.get_robot_ip() -> str
반환 값	str: IP 주소(없음일 수 있음)
호환 로봇 소프트웨어 버전	협업(Copper): v7.5.0.0+ 산업용(Bronze): v7.5.0.0+

IP 획득 논리:

이 메서드는 SDK가 실행되는 환경을 기반으로 적절한 IP 주소를 자동으로 결정하고 반환합니다.

1. **산업용 교육 펜던트 환경**: 컨트롤러의 IP 주소를 반환합니다.
2. **협업 AP 보드 환경**:
   - DHCP 모드: 라우터의 IP 주소(기본 게이트웨이)를 반환합니다.
   - 고정 IP 모드: 구성된 고정 IP 주소를 반환합니다.
3. **클라우드 환경**: 클라우드의 내부 IP 주소를 반환합니다.
4. **기타 환경**: `None` 를 반환합니다.

::: 팁 참고

- 이 방법은 로봇 컨트롤러와의 연결이 필요한 확장 환경에서 주로 사용됩니다. :::

예제 코드

extension/connect_operation.py

```py
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.

Instruction: 插件系统连接机器人示例
本示例展示如何在插件系统中连接机器人。插件系统会自动识别运行环境：
- 在插件环境中运行时：自动获取机器人IP地址，无需手动配置
- 在本地开发环境中运行时：使用下方的 local_robot_ip 和 local_ap_ip 配置

Example of extension connection:
This example demonstrates how to connect to the robot in the extension syste
m.
The extension system automatically identifies the running environment:
- In extension environment: Automatically obtains the robot IP address
- In local development environment: Uses the local_robot_ip and local_ap_ip
```

```
configuration below
"""

from Agilebot import Arm, Extension, StatusCodeEnum

extension = Extension()

# ========== 本地开发配置 / Local Development Configuration ==========
# 以下配置仅在本地开发环境中生效，插件环境会自动获取IP
# The following configuration only takes effect in local development environ
ment
local_robot_ip = "10.27.1.254"  # 机器人控制器IP / Robot controller IP (modify
to 192.168.110.2 for industrial robot)
local_ap_ip = "10.27.1.254"  # 示教器或AP IP（工业机器人改为 192.168.110.102 或
None）/ Teach pendant IP or AP IP (modify to 192.168.110.102 or None for ind
ustrial robot)

# ========== 自动识别运行环境 / Automatically Identify Running Environment ===
=======
robot_ip = extension.get_robot_ip()
if robot_ip is None:
    # 本地开发环境：使用上方配置的IP / Local development: use configured IP abov
e
    print("本地开发环境 / Local development environment")
    robot_ip = local_robot_ip
    ap_ip = local_ap_ip
else:
    # 插件环境：自动获取机器人IP / Extension environment: auto-obtain robot IP
    print(
        f"插件环境，自动获取到机器人IP / Extension environment, auto-obtained ro
bot IP: {robot_ip}"
    )
    ap_ip = "127.0.0.1"

print(f"robot_ip: {robot_ip}")
print(f"ap_ip: {ap_ip}")

arm = Arm()
ret = arm.connect(
    arm_controller_ip=robot_ip, teach_panel_ip=ap_ip
)
if ret == StatusCodeEnum.OK:
```

```
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
```

4.15.2 확장 목록 가져오기

메서드 이름	extension.get_list() → tuple[list[ExtensionInfo], StatusCodeEnum]
설명	로봇에 설치된 모든 확장에 대한 정보를 검색합니다.
요청 매개변수	없음
반환 값	`list[ExtensionInfo]` : 확장 정보 목록. `StatusCodeEnum` : 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): 지원되지 않음

4.15.3 확장 세부정보 가져오기

메서드 이름	extension.get(`name` : str) → tuple[ExtensionInfo, StatusCodeEnum]
설명	이름으로 단일 확장의 세부 정보를 쿼리합니다.
요청 매개변수	`name` : 확장명.
반환 값	`ExtensionInfo` : 자세한 확장 정보. `StatusCodeEnum` : 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): 지원되지 않음

4.15.4 확장 활성화 상태 전환

메서드 이름	**extension.toggle(`name` : str) → StatusCodeEnum**
설명	지정된 확장의 활성화된 /disabled 상태를 전환합니다.
요청 매개변수	`name` : 확장명.
반환 값	`StatusCodeEnum` : 함수 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): 지원되지 않음

4.15.5 단순 서비스 확장 명령 호출

메서드 이름	**extension.call_service(`name` : str, `command` : str, `params` : dict = None) → tuple[Any, StatusCodeEnum]**
설명	지정된 확장에서 제공하는 단순 서비스 명령을 호출하고 실행 결과를 반환합니다.
요청 매개변수	`name` : 확장 이름. `command` : 확장 명령 이름. `params` : 명령 매개변수(선택 사항, 기본값 없음).
반환 값	`Any` : 확장 명령어의 실행 결과. `StatusCodeEnum` : 함수의 실행 결과.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): 지원되지 않음

예제 코드

extension/extension_operation.py

```py
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 插件使用示例 / Example of extension usage
```

```
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connection successful")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

res, ret = arm.extension.get("MathService")
if ret == StatusCodeEnum.OK:
    print(
        "获取插件信息成功 / Get extension information successfully"
    )
    print(f"插件信息 / Extension information: {res}")
else:
    print(
        f"获取插件信息失败，错误代码 / Get extension information failed, error c
ode: {ret.errmsg}"
    )

ret = arm.extension.toggle("MathService")
if ret == StatusCodeEnum.OK:
    print(
        "切换插件状态成功 / Toggle extension status successfully"
    )
else:
    print(
        f"切换插件状态失败，错误代码 / Toggle extension status failed, error cod
```

```
e: {ret.errmsg}"
    )

res, ret = arm.extension.call_service(
    "MathService", "add", dict([["a", 1], ["b", 2]])
)
if ret == StatusCodeEnum.OK:
    print(
        "调用插件服务成功 / Call extension service successfully"
    )
    print(
        f"调用插件服务结果 / Call extension service result: {res}"
    )
else:
    print(
        f"调用插件服务失败，错误代码 / Call extension service failed, error cod
e: {ret.errmsg}"
    )
```

4.16 자연어 제어

개요

NLUControl 모듈은 자연어 명령을 통해 로봇을 제어하는 기능을 제공하여 안전 점검 후 실행할 수 있도록 일상 언어를 실행 가능한 코드로 변환하는 기능을 지원합니다.

핵심 기능

- 자연어 명령을 통한 로봇 제어 지원
- 자연어를 실행 가능한 코드로 변환 지원
- 안전한 실행을 보장하기 위한 코드 승인 메커니즘 제공
- 다단계 명령 지원(순차 실행, 조건 판단 등)
- 코드 검토 및 인쇄 기능 제공
- 코드 실행 예외를 자동으로 캡처하고 처리합니다.

사용 사례

- 자연어를 통해 간단한 작업을 수행하는 로봇을 빠르게 제어
- 복잡한 다단계 작업을 위한 자연어 설명 및 실행 구현
- 개발 및 디버깅 중 로봇 기능의 신속한 검증
- 로봇 프로그래밍의 문턱을 낮추고 비전문 작업자 지원
- 로봇과 인간 간의 자연어 상호작용 구현

생성자

메서드 이름	**NLUControl(** `controller_ip` **: str)**
설명	자연어를 통해 로봇의 동작을 제어하는 자연어 제어 수업입니다. 일반적으로 `arm.nlu` 를 통해 액세스하며 수동 인스턴스화가 필요하지 않습니다.

메서드 이름	**NLUControl(** `controller_ip` **: str)**
요청 매개변수	`controller_ip` : str 컨트롤러 IP 주소
반환 값	반환 값 없음
메모	이 클래스는 Arm 클래스가 인스턴스화될 때 자동으로 생성됩니다. 사용자는 `arm.nlu` 를 통해 직접 액세스할 수 있습니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.16.1 자연어 수업 실행

메서드 이름	**execute(** `natural_language` **: str) -> tuple[NLUGeneratedCode, StatusCodeEnum]**
설명	자연어 명령을 통해 로봇을 제어합니다. 시스템은 자연어를 실행 가능한 코드로 변환하고 이를 반환합니다.
요청 매개변수	`natural_language` : str 자연어 명령(예: "로봇을 영점으로 이동하고 현재 위치를 얻습니다", 순서 지정 및 조건과 같은 다단계 명령 지원).
반환 값	NLUGeneratedCode: 실행 코드와 승인 상태를 포함하는 생성된 코드 객체 StatusCodeEnum: 함수 실행 결과
메모	- The returned NLUGeneratedCode object contains a `needs_approval` property indicating whether user approval is required - If `needs_approval=True` , you must call the `approve()` method before execution - You can view the generated code content via `result.code`
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

NLUGeneratedCode 클래스

개요

NLUGeneratedCode 클래스는 NLU 서비스에서 생성된 Python 코드 조각을 캡슐화, 관리 및 안전하게 실행하는 데 사용됩니다. 이 클래스는 코드 승인 및 실행을 위한 보안 메커니즘을 제공하여 승인이 필요한 코드가 실행 전에 명시적으로 승인되어야 함을 보장합니다.

속성

속성 이름	유형	설명
`needs_approval`	부울	코드의 위험도에 따라 실행 전 승인이 필요한지 여부
`code`	str	생성된 실행 가능 Python 코드 문자열

4.16.2 코드 실행 승인

메서드 이름	**approve()**
설명	Approve code execution. After calling this method, the code is marked as approved and can be executed via the `execute_code()` method.
요청 매개변수	매개변수 없음
반환 값	반품 불가
메모	- This method only needs to be called when `needs_approval=True` - Executing code that requires approval without calling `approve()` will return an error
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.16.3 생성된 코드 실행

메서드 이름	**execute_code() -> tuple[Any, StatusCodeEnum]**
설명	NLU 생성 코드를 실행합니다. 코드에 승인이 필요하지만 승인되지 않은 경우 오류가 반환됩니다.
요청 매개변수	매개변수 없음

메서드 이름	execute_code() -> tuple[Any, StatusCodeEnum]
반환 값	Any: 코드 실행 결과, 유형은 생성된 코드에 따라 다름 StatusCodeEnum: 함수 실행 결과
메모	- If `needs_approval=True` but `approve()` was not called, returns `NLU_APPROVAL_REQUIRED` error - Any exceptions during code execution are caught and returned as corresponding error status codes
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

4.16.4 생성된 코드 인쇄

메서드 이름	print_code()
설명	사용자 검토를 위해 생성된 코드를 인쇄합니다. 출력 형식에는 구문 강조가 포함됩니다.
요청 매개변수	매개변수 없음
반환 값	반품 불가
메모	승인이 필요한 경우 실행 승인 여부를 결정하기 전에 이 메서드를 호출하여 코드 내용을 확인하는 것이 좋습니다.
호환 로봇 소프트웨어 버전	협업(Copper): v7.7.0.0+ 산업용(Bronze): v7.7.0.0+

사용 예

예 1: 기본 쿼리 작업

nlu_control/get_controller_version.py

```py
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
```

```
Instruction: 自然语言基础控制与安全评级示例 / NLU Basic Control with Safety Ratin
g Example
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")

# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)


# [ZH] 示例：使用自然语言获取控制器版本信息
# [EN] Example: Use natural language to get controller version information
result, ret = arm.nlu.execute("获取控制器版本信息")

if ret == StatusCodeEnum.OK:
    print(
        "自然语言指令执行成功 / Natural language command executed successfully"
    )
    danger_level = getattr(
        result, "danger_level", "unknown"
    )
    warnings = getattr(result, "warnings", [])

    result.print_code()
    print(f"安全评级 / Danger level: {danger_level}")
    if warnings:
```

```
        print("安全提示 / Safety warnings:")
        for idx, warning in enumerate(warnings, 1):
            print(f"  {idx}. {warning}")

    if result.needs_approval:
        print("\n" + "=" * 50)
        print(
            "警告：需要用户确认 / Warning: User Approval Required"
        )
        print("=" * 50)

        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute thi
s code:"
        )
        user_input = input("\n(yes/no): ").strip().lower()
        if user_input not in ["yes", "y"]:
            print(
                "用户拒绝执行代码 / User rejected code execution"
            )
            arm.disconnect()
            exit(1)
        else:
            result.approve()

    exec_result, ret = result.execute_code()
    if ret == StatusCodeEnum.OK:
        print(f"执行结果 / Execution result: {exec_result}")
    else:
        print(
            f"代码执行失败 / Code execution failed: {ret.errmsg}"
        )
else:
    print(
        f"自然语言指令执行失败 / Natural language command failed: {ret.errmsg}"
    )

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
```

```
    "\n机器人断开连接成功 / Robot disconnected successfully"
)
```

예 2: 여러 작업의 순차적 실행

nlu_control/serial_execution.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 自然语言顺序执行与安全评级示例 / NLU Sequential Execution with Safet
y Rating Example
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
# [EN] Initialize the Agilebot robot
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")

# [ZH] 检查是否连接成功
# [EN] Check if connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)


# [ZH] 示例：使用自然语言让机器人按顺序执行任务
# [EN] Example: Use natural language to instruct the robot to execute tasks
sequentially
result, ret = arm.nlu.execute(
```

```
    """
按顺序执行：
1. 清除报警；
2. 控制机械臂所有关节运动至零点位置；
3. 启动程序 'Put_into_box'，等待下发完成，并等待其执行结束；
4. 获取当前位姿。
"""
)

if ret == StatusCodeEnum.OK:
    print(
        "自然语言指令执行成功 / Natural language command executed successfully"
    )
    danger_level = getattr(
        result, "danger_level", "unknown"
    )
    warnings = getattr(result, "warnings", [])

    result.print_code()
    print(f"安全评级 / Danger level: {danger_level}")
    if warnings:
        print("安全提示 / Safety warnings:")
        for idx, warning in enumerate(warnings, 1):
            print(f"  {idx}. {warning}")

    if result.needs_approval:
        print("\n" + "=" * 50)
        print(
            "警告：需要用户确认 / Warning: User Approval Required"
        )
        print("=" * 50)

        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute thi
s code:"
        )
        user_input = input("\n(yes/no): ").strip().lower()
        if user_input not in ["yes", "y"]:
            print(
                "用户拒绝执行代码 / User rejected code execution"
            )
            arm.disconnect()
```

```
            exit(1)
        else:
            result.approve()

    exec_result, ret = result.execute_code()
    if ret == StatusCodeEnum.OK:
        print(f"执行结果 / Execution result: {exec_result}")
    else:
        print(
            f"代码执行失败 / Code execution failed: {ret.errmsg}"
        )
else:
    print(
        f"自然语言指令执行失败 / Natural language command failed: {ret.errmsg}"
    )

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
arm.disconnect()
print(
    "\n机器人断开连接成功 / Robot disconnected successfully"
)
```

Agilebot Python SDK 업데이트 노트

2.0.1.0 업데이트(2026/1/26)

1. 기본 통신 방법을 변경했습니다.
2. 로컬 프록시 서비스에 대한 지원이 추가되었습니다.
3. `Jogging` 클래스 아래에 `step_move` 및 `continuous_move` 인터페이스가 추가되었습니다.
4. 플러그인 관련 인터페이스가 추가되었습니다.
5. 가장 심각한 알람을 가져오기 위해 `get_top_alarm` 인터페이스를 추가했습니다.
6. `BasScript` 클래스 아래에 JUMP 관련 명령을 추가했습니다.
7. 궤적 재생과 관련된 인터페이스가 추가되었습니다.
8. 좌표계 정보 클래스를 최적화했습니다.
9. 페이로드 정보 클래스를 업데이트했습니다.
10. `get_all_active_alarms` 인터페이스는 이제 언어 선택을 지원합니다.
11. `get_version` API의 이름이 `get_controller_version` 로 변경되었습니다.
12. `is_connect` API의 이름이 `is_connected` 로 변경되었습니다.
13. 매니퓰레이터의 작동 모드를 쿼리하기 위해 `get_op_mode` API를 추가했습니다.
14. 로봇 상태, 등록 데이터 및 I/O 정보를 구독하기 위한 구독 클래스 `SubPub` 를 추가했습니다.
15. `CoordinateSystem` 클래스는 이제 TF/UF. 계산을 위한 인터페이스를 제공합니다.
16. `CoordinateSystem` 클래스 아래에 두 개의 하위 클래스 `TF` 및 `UF` 를 추가했습니다.
17. 플러그인 관리 클래스는 이제 플러그인 관련 API를 노출합니다.
18. `Trajectory` 클래스에는 궤적 재생을 위한 인터페이스가 추가되었습니다.
19. Python 종속성을 최적화했습니다. 이제 Python 3.8 이상이 지원됩니다.
20. `BasScript` 클래스 아래의 `load` , `unload` 및 `exec` 관련 명령어의 오류를 수정했습니다.

1.7.1.3 업데이트(2025/7/3)

1. `get_all_active_alarms` 인터페이스가 실패할 수 있는 문제를 수정했습니다.

2. `move_circle` 의 내부 매개변수 설정 오류를 수정했습니다.
3. 예제 프로그램을 업데이트했습니다.
4. `extension.get_robot_ip` 인터페이스를 제공하는 플러그인 관리 클래스를 추가했습니다.
5. 궤적 재현과 관련된 인터페이스가 추가되었습니다.

1.7.0.0 업데이트(2025/5/30)

1. 변경사항 등록
    1. 이제 모든 레지스터는 `Registers` 클래스 아래에 배치되며 다른 `Registers` 는 향후 더 이상 사용되지 않습니다.
    2. 인터페이스 이름이 `read_R` , `write_R` , `delete_R` 등으로 변경되었습니다.
    3. `add` 방법이 제거되었습니다. 대신 `write_XX` 를 사용하세요.
    4. 레지스터 읽기 및 쓰기를 위한 RPC 인터페이스가 값만 읽는 새로운 RPC로 변경되어 속도가 향상되었습니다.
    5. `read_R` , `read_MR` , `read_SR` 는 이제 해당 값과 실행 결과를 직접 반환합니다.
2. `arm` 클래스 인터페이스 변경 사항
    1. 인터페이스 `get_arm_model_info` , `get_ctrl_status` , `get_robot_status` , `get_servo_status` , `get_soft_mode` , `get_version` 가 '결과 + 실행 상태'를 반환하도록 변경되었습니다.
    2. 인터페이스 `servo_on` , `servo_off` , `servo_reset` 는 `arm` 클래스로 이동되었으며 향후 `execution` 에서는 더 이상 사용되지 않습니다.
3. `motion` 클래스 변경 사항
    1. 페이로드 관련 작업 인터페이스가 `motion.payload` 로 이동되었습니다.
    2. 페이로드 측정을 위한 새로운 인터페이스가 추가되었습니다.
    3. 새로운 인터페이스 `get_OVC` , `set_OVC` , `get_OAC` , `set_OAC` , `get_TF` , `set_TF` , `get_UF` , `set_UF` , `get_TCS` , `set_TCS` 가 추가되었습니다.
    4. 이제 `convert_cart_to_joint_simple` 인터페이스에는 `uf_index` 및 `tf_index` 에 대한 설정이 포함됩니다.
    5. 새로운 인터페이스 `move_joint` , `move_line` , `move_circle` 가 추가되었습니다.
    6. 새로운 인터페이스 `convert_joint_to_cart` , `convert_cart_to_joint` 가 추가되었습니다.

7. 새로운 인터페이스 `enter_position_control` , `exit_position_control` , `set_udp_feedback_params` 가 추가되었습니다.

4. 이름을 단순화하기 위해 `digital_signals` 클래스의 이름이 `signals` 로 변경되었습니다.

5. `execution` 클래스에는 스크립트 실행을 위한 `execute_bas_script` 인터페이스가 추가되었습니다.

6. `jogging.move` 인터페이스가 단계 값 수정: `move(*self*, *aj_num*: *int*, *move_mode*: MoveMode = None, *step_length*: *float* = 0)` 를 포함하도록 업데이트되었습니다.

7. 스크립트 생성을 위해 새로운 `bas_script` 클래스가 추가되었습니다.

8. Modbus 장치를 직접 제어하기 위해 새로운 `modbus` 클래스가 추가되었습니다.

돕다

에이전트 스킬

에이전트 스킬은 산업 및 개발 시나리오를 위한 표준 실행 워크플로와 도구를 AI 에이전트(예: Codex, Claude Code)에 확장해 주는 기능입니다.

지원되는 에이전트 유형

에이전트 스킬은 현재 다음 AI 도구와 CLI에 대한 자동 감지 및 설치를 지원합니다.

- **IDE / 편집기**: Cursor, Windsurf, Trae, Trae CN, VS Code (Copilot / Continue), Neovate, Pochi
- **명령줄(CLI)**: Claude Code, Kimi Code CLI, Gemini CLI, iFlow CLI, Kiro CLI, Mistral Vibe
- **프레임워크 / 플랫폼**: OpenHands, Replit, Cline, Roo Code, Codex, Amp, Antigravity, Augment, OpenClaw
- **기타**: CodeBuddy, Command Code, Droid, Junie, Kilo Code, Kode, MCPJam, Mux, OpenCode, Pi, Qoder, Qwen Code, Zencoder, AdaL

빠른 시작

1. 환경 준비

설치 전에 로컬 환경에 Node.js가 설치되어 있는지 확인하세요. 안내 문서: Node.js Download and Installation

2. 설치 명령 실행

터미널에서 다음 명령을 실행하면 대화형 설치 프로그램이 시작됩니다.

```bash
npx skills add https://dev.sh-agilebot.com -g
```

TIP

명령을 실행하면 도구가 로컬에 설치된 에이전트를 스캔합니다. 안내에 따라 스킬을 설치할 에이전트(예: Cursor 또는 Claude Code)를 선택하세요.

사용 가능한 스킬

agilebot-nlu-control (Agilebot 자연어 제어)

이 스킬은 Agilebot 로봇을 위해 설계되었습니다. NLU(Natural Language Understanding)를 사용하여 에이전트가 제어 명령을 직접 해석하고 실행할 수 있도록 합니다.

- 환경 요구 사항: 시스템 Python을 사용할 수 있어야 하며 Conda가 설치되어 있어야 합니다. 스킬은 필요한 의존성을 점검하고, 누락된 항목이 있으면 설치를 안내합니다.
- 트리거: 대화에서 "Agilebot"이 언급되고 구체적인 제어 의도가 함께 주어질 때
- 핵심 기능: 자연어 명령 해석 및 실행, 로봇 상태 조회, 기본 동작 제어, 매개변수화된 동작 명령(예: distance/angle/speed)

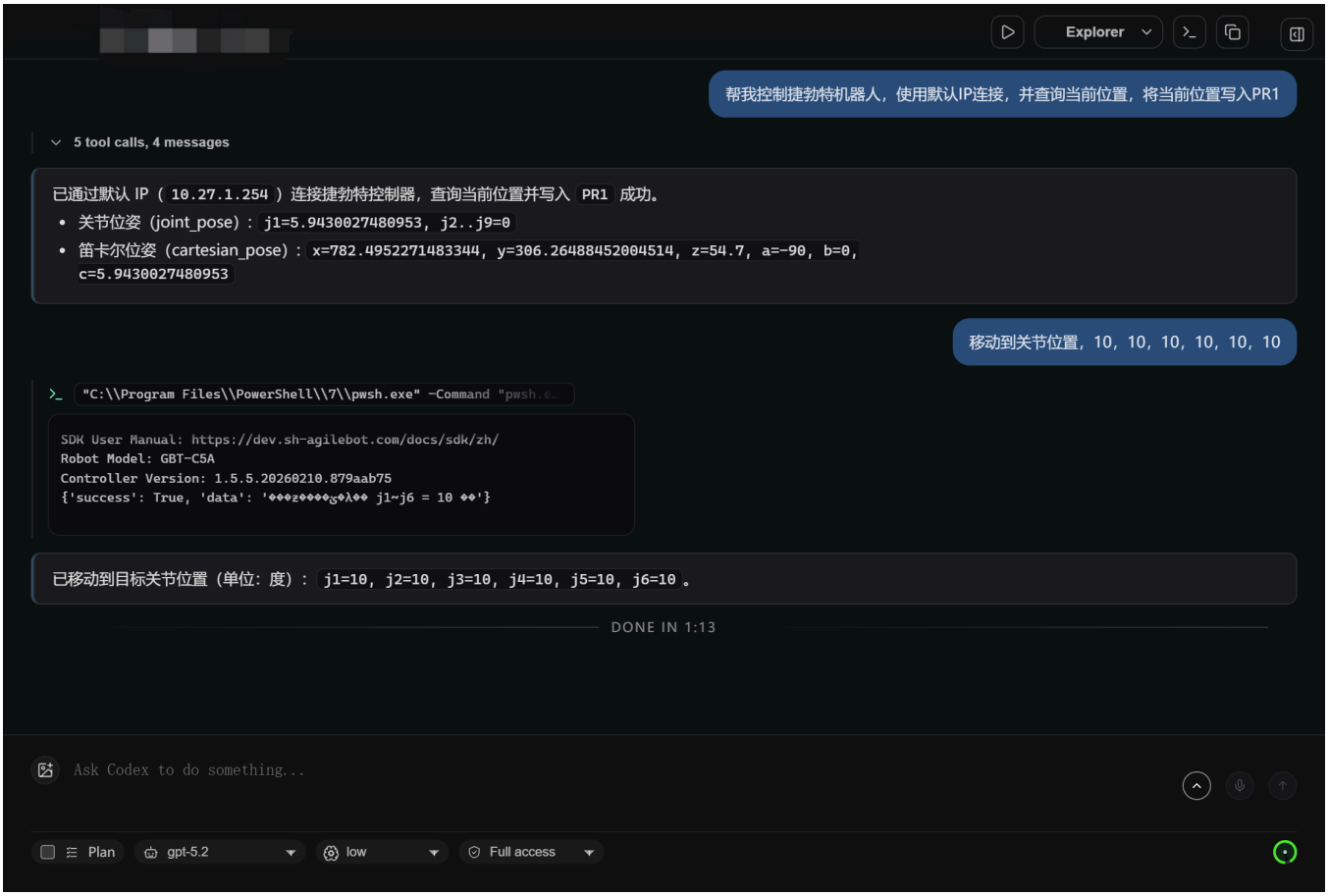
경고

설치된 스킬은 에이전트의 전체 권한으로 실행됩니다. 반드시 신뢰할 수 있는 환경에서 사용하세요.

로봇 동작은 위험할 수 있으며, 인명 피해나 장비 손상을 초래할 수 있습니다. 안전이 확보된 환경에서 사용해야 하며, 사용으로 인해 발생한 위험과 손실에 대해서는 책임지지 않습니다.

사용 예

Codex를 예로 들면, 설치 후 채팅에 직접 지시를 입력해 사용할 수 있습니다.



AI 코딩 지원

이 문서에서는 AI 지원 도구(예: CodeBuddy, Codex, Cursor 등)를 사용하여 로봇 프로그램을 빠르게 개발하는 방법을 설명합니다.

준비

AI 코딩을 사용하기 전에 참조 문서를 준비해야 합니다.

- **SDK 문서**: https://dev.sh-agilebot.com/docs/sdk/knowledge/docs.txt

> **팁**: AI 에이전트가 URL을 잘 읽을 수 없는 경우 위의 txt 문서를 로컬 프로젝트 디렉터리에 다운로드하고 프롬프트에서 로컬 파일 경로를 참조하세요.

예제 프롬프트

다음은 로봇 상태를 읽는 Python 프로그램을 생성하는 완전한 예입니다.

```
Read the following documentation and write a Python program that reads the r
obot's current position, coordinate system number, servo status, and other i
nformation.

Reference materials:
SDK Documentation: https://dev.sh-agilebot.com/docs/sdk/knowledge/docs.txt
```

로컬 문서에 의존하는 경우 다음과 같이 수정할 수 있습니다.

```
Read the following documentation and write a Python program that reads the r
obot's current position, coordinate system number, servo status, and other i
nformation.

Reference materials:
SDK Documentation: ./docs/sdk_docs.txt
```

사용 팁

1. **명확한 요구사항**: 구현하려는 기능을 명확하게 설명하세요.
2. **컨텍스트 제공**: 관련 문서 및 예시 참조
3. **단계적 구현**: AI를 사용하여 복잡한 기능을 단계별로 생성할 수 있습니다.

메모

1. AI가 생성한 코드는 검증과 테스트가 필요함
2. 코드가 프로젝트 코딩 표준을 준수하는지 확인하세요.
3. 로봇 제어와 관련된 코드는 보안 검토를 받아야 합니다.