

# Agilebot Robot SDK

## Agilebot Robot SDK Manual

[Python SDK](#)[C# SDK](#)

### Multi-language Support

Provides both Python and C# SDK versions to meet the needs of different development environments and technical stacks.



### Robot Motion Control

Comprehensive motion control APIs supporting joint motion, linear motion, trajectory planning, and real-time control.



### Real-time Data Interaction

High-performance gRPC-based communication mechanism supporting real-time status monitoring, register read/write, and hardware status retrieval.



### Fly-shoot Vision Functionality

Integrated fly-shoot vision system supporting precise visual recognition and positioning during high-speed motion.



### ROS2 Integration



### Cross-platform Compatibility

Provides ROS2 interface support for seamless integration into existing robotic ecosystems and workflows.

Supports Windows and Ubuntu systems, compatible with X86 and ARM architectures to adapt to various hardware environments.



### Rich Example Code

Offers complete example programs and test cases to help developers quickly get started and integrate SDK features.



### Modular Design

Modular API design including independent functional modules such as motion control, IO operations, status monitoring, and file management.

---

Copyright © 2025-present Agilebot Robotics Co., Ltd.

# Python SDK

# Prologue

# Version History

Document Version	SDK Version	Release Date
V2.0	1.7.1.3	2025.07.07
V3.0	2.0.0.0	2025.11.05

---

## Update Notes

# Robot Version Compatibility

The SDK supports Agilebot Scara, Puma, and collaborative robot series. It must be used with devices that have the robot software installed. It is compatible with the robot software versions. Due to different versions, some functions may return different results. For details, please refer to the function list in Chapter 4. When the SDK connects to the robotic arm, it will check the version of the robotic arm motion control software. If the version is lower than the minimum requirement, the connection will fail. If it is lower than the recommended version, a prompt indicating that the version is too low will appear. Please update the robot software version in time. Some interfaces of the SDK only support the corresponding version of the controller. Please pay attention to the compatibility of specific interfaces.

SDK Version	Robot Software Version	Support Status
v1.7.0.X	Copper v7.6.X.X、 Bronze v7.6.X.X	Supported
v1.7.1.X	Copper v7.6.X.X、 Bronze v7.6.X.X	Supported
v2.0.0.X	Copper v7.7.X.X、 Bronze v7.7.X.X	Supported

# 1 Introduction and Deployment

# 1.1 Robot System

The SDK is installed on the customer's host computer. It provides interfaces for the host computer to operate and detect the robot. This system consists of a host computer, a controller, and a robot body. The robot body refers to the mechanical body, which is composed of motors, reducers, encoders, and transmission mechanisms. The operation panel of the control cabinet is equipped with status indicator lights, control cabinet switches, communication interfaces, etc.

# 1.2 Environment Requirements

Hardware and Operating System:

- Windows 10 or later
  - x86\_64 architecture
- Linux (Ubuntu 18.04 or later is recommended)
  - x86\_64 architecture
  - Arm architecture

Python Version:

- 3.8 or later

## 1.3 Installation

### Environment Setup

- Download the Python environment. It is recommended to use Conda to configure the environment. Download and install the latest Miniforge software or Miniconda software for your current system.
- Conda download link: [Miniforge3](#)
- After installation, launch Miniforge Prompt and enter the following command to initialize conda:

```
conda init
```

shell

- After installation, launch Miniforge Prompt or your system terminal and enter the following command to create a new Python environment:

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

shell

- Enter the following command to activate the newly created environment:

```
conda activate agilesdk
```

shell

- If installing the Conda environment is inconvenient, you can use pip to create and activate a virtual environment:

```
# Create
python3.10 -m venv agilesdk
# Activate the environment on Windows
.\agilesdk\Scripts\activate
# Activate the environment on Linux
source myenv/bin/activate
```

shell

- Enter the `cd` command to navigate to the extracted directory and use the following command to install the required Python packages:

```
cd extracted_package_directory
pip install -r requirements.txt
```

shell

- If pip installation fails, open `requirements.txt` and use conda to install each package:

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

shell

---

## IDE Installation

- It is recommended to use PyCharm as your development environment.
- PyCharm download link: [PyCharm](#)
- After downloading and installing PyCharm Community Edition, launch the software.
- Create a new Python project and select "Pure Python" as the project type.
- Select "Base Conda" as the interpreter type and set the path to the previously created `agilesdk` environment.
- Click "Create" to start writing code.

---

## SDK Installation and Testing

- Enter the following command to install the Python version of the robot SDK. Replace `x.x.x` with the current SDK version number:

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

shell

- Navigate to the `example` folder and use the following command to run tests:

shell

```
cd example  
python example_program_name (e.g., python arm/get_version.py)
```

- When running examples, the host computer must be connected to the robot network.

## 2 Glossary

Term	Description
teach pendant	The pendant attached to the robot, used for teaching and controlling the robot
SDK	Software Development Kit, used for programming and controlling the robot
robot network	The network connection between the robot and the external computer
controller	The control unit of the robot, responsible for executing motion commands, processing sensor data, and managing robot status
robotic arm	The main moving part of the robot, consisting of multiple joints and links
servo system	The motor drive system that controls robot joint motion, providing precise position and speed control
teaching	The process of recording robot motion trajectories and actions through manual operation of the robot or teach pendant
joint	The movable component connecting various links in the robot arm, each joint corresponding to one degree of freedom
Cartesian coordinates	A three-dimensional coordinate system based on three mutually perpendicular X, Y, Z axes, used to describe the robot's position and orientation in space
pose	The combination of the robot's position and orientation in space, including position coordinates and rotation angles
trajectory	The path of the robot's end effector moving in space, usually composed of a series of pose points
payload	The weight and objects carried by the robot's end effector, affecting the robot's motion performance and accuracy
coordinate system	A reference system used to describe robot position and orientation, including base coordinate system, tool coordinate system, user coordinate system, etc.
OVC	Overall Velocity Control, used to set the overall motion speed multiplier of the robot
OAC	Overall Acceleration Control, used to set the overall acceleration multiplier of the

Term	Description
	robot
TF	Tool Frame, a coordinate system with the robot's end tool as the origin
UF	User Frame, a user-defined coordinate system for convenient programming and positioning
TCS	Teach Coordinate System, a coordinate reference system used during teaching
DH parameters	Denavit-Hartenberg parameters, standard parameters used to describe the geometric relationships of robot links
PR register	Pose Register, a register used to store robot pose information
MR register	Motion Register, a register used to store motion-related parameters
SR register	String Register, a register used to store string information
R register	Real Register, a register used to store numerical information
MH register	Modbus Holding Register, a holding register for Modbus communication
MI register	Modbus Input Register, an input register for Modbus communication
BAS	Basic Script, a high-level programming language used to write robot control programs
Scara	Selective Compliance Assembly Robot Arm, a type of four-axis industrial robot
collaborative robot	A robot capable of safe collaboration with humans, usually equipped with force sensing and collision detection capabilities
industrial robot	A robot used for industrial automation production, usually with high precision, high speed, and high load capacity
Copper	The codename for Agilebot's collaborative robot product line
Bronze	The codename for Agilebot's industrial robot product line

## 3 Data Structures

### 3.1 StatusCodeEnum

#### Description

Interface return status codes.

**Note:** If you encounter a return code that is not listed here, consult the robot fault-code table or interpret the value literally.

#### Import

```
from Agilebot import StatusCodeEnum
```

python

#### Fields

Name	Enum Value	Description
OK	0	Success
INCOMPATIBLE_VERSION	-1	Incompatible version
CONNECTION_TIMEOUT	-3	Connection timeout
INTERFACE_NOTIMPLEMENTED	-4	Interface not implemented on the current controller
INDEX_OUT_OF_RANGE	-5	Index out of range
UNSUPPORTED_FILETYPE	-6	Unsupported file type
UNSUPPORTED_PARAMETER	-7	Unsupported robot parameter
UNSUPPORTED_SIGNAL_TYPE	-8	Unsupported IO signal type
PROGRAM_NOT_FOUND	-9	Program not found

Name	Enum Value	Description
PROGRAM_POSE_NOT_FOUND	-10	Program pose not found
WRITE_PROGRAM_POSE_FAILED	-11	Failed to write program pose
GET_ALARM_CODE_FAILED	-12	Failed to retrieve alarm code
WRONG_POSITION_INFO	-13	Incorrect position information from the controller
UNSUPPORTED_TRATYPE	-14	Unsupported motion type
INVALID_DH_LIST	-15	Invalid transformation parameter list. Please contact the developer
INTERVAL_PORTS_MUST_NOTNONE	-16	Interval, port list, and level duration must not be empty
INVALID_IP_ADDRESS	-17	Invalid IP address
INVALID_DH_PARAMETERS	-18	Invalid DH parameters
INVALID_PAYLOAD_INFO	-19	Invalid payload information
INVALID_FLYSHOT_CONFIG	-20	Invalid flyshot configuration parameters
FAILED_TO_DOWNLOAD_SAME_NAME_FILE	-21	Download failed because a file with the same name already exists
FAILED_TO_CONVERT_CART_TO_JOINT	-22	Failed to convert Cartesian coordinates to joint coordinates
FAILED_TO_GET_ALL_INVERSE_KINEMATICS	-23	Failed to obtain the eight solutions within one revolution
COORDINATE_LIMIT_EXCEEDED	-24	Coordinate system limit exceeded. The maximum is 10 coordinate systems with IDs in [1, 10]
FILE_NOTEXIST	-25	File does not exist
INVALID_IO_LIST_PARAMETERS	-26	Invalid IO list parameters
INVALID_PARAMETER	-27	Invalid parameter

Name	Enum Value	Description
NOT_FOUND	-28	Requested data not found
LOCAL_PROXY_UNSUPPORTED	-29	Local proxy is not supported in the current environment
CONTROLLER_ERROR	-254	Controller error. Contact the developer for details
SERVER_ERR	-255	Other reasons

## 3.2 RobotStatusEnum

### Description

Robot operating status.

### Import

```
from Agilebot import RobotStatusEnum
```

python

### Fields

Name	Enum Value	Description
ROBOT_UNKNOWN	-1	Unknown state
ROBOT_IDLE	0	Robot idle
ROBOT_RUNNING	1	Robot running
ROBOT_TEACHING	2	Robot teaching
ROBOT_DRAG	3	Robot in drag mode
ROBOR_FORCE_DRAG	4	Robot in force-drag mode
ROBOT_IDLE_TO_RUNNING	101	Transition from idle to running

Name	Enum Value	Description
ROBOT_IDLE_TO_TEACHING	102	Transition from idle to teaching
ROBOT_RUNNING_TO_IDLE	103	Transition from running to idle
ROBOT_TEACHING_TO_IDLE	104	Transition from teaching to idle

## 3.3 CtrlStatusEnum

### Description

Controller operating status.

### Import

```
from Agilebot import CtrlStatusEnum
```

python

### Fields

Name	Enum Value	Description
CTRL_UNKNOWN	-1	Unknown controller state
CTRL_INIT	0	Controller initializing
CTRL_ENGAGED	1	Controller enabled
CTRL_ESTOP	2	Controller emergency stop
CTRL_TERMINATED	3	Controller terminated
CTRL_ANY_TO_ESTOP	101	Transition from any state to emergency stop
CTRL_ESTOP_TO_ENGAGED	102	Transition from emergency stop to enabled
CTRL_ESTOP_TO_TERMINATED	103	Transition from emergency stop to terminated

## 3.4 ServoStatusEnum

### Description

Servo controller status.

### Import

```
from Agilebot import ServoStatusEnum
```

python

### Fields

Name	Enum Value	Description
SERVO_UNKNOWN	-1	Unknown servo controller state
SERVO_IDLE	1	Servo controller idle
SERVO_RUNNING	2	Servo controller running
SERVO_DISABLE	3	Servo controller disabled
SERVO_WAIT_READY	4	Servo controller waiting for ready
SERVO_WAIT_DOWN	5	Servo controller waiting for shutdown
SERVO_INIT	10	Servo controller initializing

## 3.5 SoftModeEnum

### Description

Robot PC status mode.

### Import

```
from Agilebot import SoftModeEnum
```

python

## Fields

Name	Enum Value	Description
UNKNOWN	0	Unknown
AUTO	1	Automatic mode
MANUAL_LIMIT	2	Manual speed-limited mode
MANUAL	3	Manual mode

## 3.6 PoseType

### Description

Robot pose type.

### Import

```
from Agilebot import PoseType
```

python

## Fields

Name	Enum Value	Description
JOINT	0	Joint space
CART	1	Cartesian coordinates

## 3.7 TCSType

### Description

Coordinate system type.

### Import

```
from Agilebot import TCSType
```

python

## Fields

Name	Enum Value	Description
JOINT	0	Joint space
BASE	1	Base coordinate system
WORLD	2	World coordinate system
USER	3	User coordinate system
TOOL	4	Tool coordinate system
RTCP_USER	5	RTCP user coordinate system
RTCP_TOOL	6	RTCP tool coordinate system

## 3.8 Joint

### Description

Data structure describing the robot joint positions.

### Import

```
from Agilebot import Joint
```

python

### Attributes

Attribute	Type	Description
j1	float	Value of joint 1
j2	float	Value of joint 2
j3	float	Value of joint 3

Attribute	Type	Description
j4	float	Value of joint 4
j5	float	Value of joint 5
j6	float	Value of joint 6
j7	float	Value of joint 7
j8	float	Value of joint 8
j9	float	Value of joint 9

## 3.9 Position

### Description

Data structure describing the robot Cartesian pose.

### Import

```
from Agilebot import Position
```

python

### Attributes

Attribute	Type	Description
x	float	X-axis coordinate
y	float	Y-axis coordinate
z	float	Z-axis coordinate
a	float	A-axis angle
b	float	B-axis angle
c	float	C-axis angle

## 3.10 Posture

### Description

Data structure describing the robot posture parameters.

### Import

```
from Agilebot import Posture
```

python

### Attributes

Attribute	Type	Description
<code>turn_cycle</code>	<code>int[]</code>	Turn count for each axis. Integer range ..., -2, -1, 0, 1, 2, ... with 0 at the 0° posture. During linear/arc motions the target turn count is chosen automatically. Rotations $\geq 180^\circ$ map to $\geq 1$ , between $-179.99^\circ$ and $179.99^\circ$ map to 0, $\leq -180^\circ$ map to $\leq -1$ .
<code>wrist_flip</code>	<code>int</code>	Wrist flip posture (-1/0/1). On 6-axis robots (joint 5), 1 means the wrist flips downward, -1 means it flips upward.
<code>arm_up_down</code>	<code>int</code>	Arm up/down posture (-1/0/1). On 6-axis robots (joint 3), 1 means the arm is above the line between joints 4 and 2 (with $J3 < 0$ ), -1 means it is below that line (with $J3 > 0$ ).
<code>arm_back_front</code>	<code>int</code>	Arm front/back posture (-1/0/1). On 6-axis robots (joint 1), 1 means the arm is in front (axis 2 on the left of axis 1 when facing forward), -1 means the arm is behind (axis 2 on the right of axis 1).
<code>arm_left_right</code>	<code>int</code>	Arm left/right posture (-1/0/1). On 4-axis SCARA robots (joint 2), 1 means the arm is on the right, -1 means the arm is on the left.

## 3.11 BaseCartData

### Description

Data structure describing the robot Cartesian target pose.

## Import

```
from Agilebot import BaseCartData
```

python

## Attributes

Attribute	Type	Description
<code>position</code>	<a href="#">Position</a>	Cartesian pose data
<code>posture</code>	<a href="#">Posture</a>	Robot posture parameters

## 3.12 MotionPose

### Description

Data structure describing the robot pose. Distances in the XYZ directions are measured in millimeters (mm) and angle data is in degrees (deg). In some versions the angle information is returned in radians; refer to the API return description for details.

## Import

```
from Agilebot import MotionPose
```

python

## Attributes

Attribute	Type	Description
<code>cartData</code>	<a href="#">BaseCartData</a>	Cartesian data
<code>joint</code>	<a href="#">Joint</a>	Joint data
<code>pt</code>	<a href="#">PoseType</a>	Pose type, default is <code>PoseType.JOINT</code>

## 3.13 DHparam

## Description

Robot DH parameter list.

## Import

```
from Agilebot import DHparam
```

python

## Attributes

Attribute	Type	Description	Default Value
<code>id</code>	<code>int</code>	ID of the DH parameter	-1
<code>d</code>	<code>float</code>	Joint distance	-1.0
<code>a</code>	<code>float</code>	Link length	-1.0
<code>alpha</code>	<code>float</code>	Link twist angle	-1.0
<code>offset</code>	<code>float</code>	Joint offset	-1.0

## 3.14 PayloadInfo

### Description

Robot payload information.

### Import

```
from Agilebot import PayloadInfo
```

python

### Attributes

Attribute	Type	Description
<code>id</code>	<code>int</code>	Payload ID

Attribute	Type	Description
weight	float	Payload weight (kg)
comment	str	Comment
mass_center	MassCenter	Center of mass
inertia_moment	InertiaMoment	Moment of inertia

MassCenter

Field	Type	Description
x	float	X
y	float	Y
z	float	Z

InertiaMoment

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

3.15 ProgramCartData

Description

Program Cartesian data.

Import

```
from Agilebot import ProgramCartData
```

python

## Attributes

Attribute	Type	Description
<code>baseCart</code>	<a href="#">BaseCartData</a>	Cartesian data
<code>uf</code>	<code>int</code>	User coordinate system ID in use
<code>tf</code>	<code>int</code>	Tool coordinate system ID in use

## 3.16 ProgramPoseData

### Description

Program pose data.

### Import

```
from Agilebot import ProgramPoseData
```

python

## Attributes

Attribute	Type	Description
<code>cartData</code>	<a href="#">ProgramCartData</a>	Program pose data
<code>joint</code>	<a href="#">Joint</a>	Joint data
<code>pt</code>	<a href="#">PoseType</a>	Pose type, default is <code>PoseType.JOINT</code>

## 3.17 ProgramPose

### Description

Program pose.

## Import

```
from Agilebot import ProgramPose
```

python

## Attributes

Attribute	Type	Description
<code>poseData</code>	<a href="#">ProgramPoseData</a>	Program pose data
<code>id</code>	<code>int</code>	Pose ID
<code>name</code>	<code>str</code>	Pose name
<code>comment</code>	<code>str</code>	Pose comment

## 3.18 PoseRegisterData

### Description

Register data class, used to represent the data stored in a register.

## Import

```
from Agilebot import PoseRegisterData
```

python

## Attributes

Attribute	Type	Description
<code>cartData</code>	<a href="#">BaseCartData</a>	Cartesian pose data
<code>joint</code>	<a href="#">Joint</a>	Joint data
<code>pt</code>	<a href="#">PoseType</a>	Pose type, default is <code>PoseType.JOINT</code>

## 3.19 PoseRegister

### Description

Register class, used to represent the register data stored on the controller.

### Import

```
from Agilebot import PoseRegister
```

python

### Attributes

Attribute	Type	Description
<code>poseRegisterData</code>	<code>PoseRegisterData</code>	Register pose data
<code>id</code>	<code>int</code>	Register ID
<code>name</code>	<code>str</code>	Register name
<code>comment</code>	<code>str</code>	Register comment

## 3.20 TransformStatusEnum

### Description

Trajectory transformation status.

### Import

```
from Agilebot import TransformStatusEnum
```

python

### Fields

Name	Enum Value	Description
<code>TRANSFORM_START</code>	0	Transformation started

Name	Enum Value	Description
TRANSFORM_RUNNING	1	Transformation in progress
TRANSFORM_SUCCESS	2	Transformation succeeded
TRANSFORM_FAILED	3	Transformation failed
TRANSFORM_NOT_FOUND	4	Transformation not found
TRANSFORM_UNKNOWN	5	Unknown transformation status

## 3.21 HWState

### Description

Hardware state enumeration.

### Import

```
from Agilebot import HWState
```

python

### Fields

Name	Description
TOPIC_JOINT	Publish robot joint feedback
TOPIC_CURRENT_CARTESIAN	Publish current TCP Cartesian pose
TOPIC_UF	Publish current user frame info
TOPIC_TF	Publish current tool frame info
TOPIC_VELOCITY	Publish global velocity ratio
TOPIC_RUNNING_STATUS	Publish controller running status
TOPIC_INTERPRETER_STATUS	Publish interpreter status
TOPIC_ROBOT_STATUS	Publish robot status

Name	Description
<code>TOPIC_CTRL_STATUS</code>	Publish controller state
<code>TOPIC_SERVO_STATUS</code>	Publish servo controller status
<code>TOPIC_TRAJECTORY_RECORDS_STATUS</code>	Publish trajectory record status

## 3.22 CoordinateSystemType

### Description

Robot coordinate system type.

### Import

```
from Agilebot import CoordinateSystemType
```

python

### Fields

Name	Enum Value	Description
<code>UserFrame</code>	0	User coordinate system
<code>ToolFrame</code>	1	Tool coordinate system

## 3.23 Coordinate

### Description

Coordinate system data, including tool and user frames.

### Import

```
from Agilebot import Coordinate
```

python

## Attributes

Attribute	Type	Description
<code>id</code>	<code>int</code>	Coordinate system ID
<code>name</code>	<code>str</code>	Coordinate system name
<code>comment</code>	<code>str</code>	Coordinate system comment
<code>data</code>	<a href="#">Position</a>	Coordinate pose data

## 3.24 DragStatus

### Description

Robot drag status.

### Import

```
from Agilebot import DragStatus
```

python

## Attributes

Attribute	Type	Description
<code>cart_status</code>	<a href="#">CartStatus</a>	Cartesian axis drag/lock status
<code>joint_status</code>	<a href="#">JointStatus</a>	Joint axis drag/lock status
<code>is_continuous_drag</code>	<code>bool</code>	Continuous drag flag

### 3.24.1 CartStatus

### Description

Cartesian status class representing Cartesian axis drag state.

### Import

```
from Agilebot import CartStatus
```

python

## Attributes

Attribute	Type	Description
x	bool	X-axis status
y	bool	Y-axis status
z	bool	Z-axis status
a	bool	A-axis status
b	bool	B-axis status
c	bool	C-axis status

## 3.24.2 JointStatus

### Description

Joint status class representing the status of each robot joint. All joints default to `True` (available).

### Import

```
from Agilebot import JointStatus
```

python

## Attributes

Attribute	Type	Description
j1	bool	Status of J1
j2	bool	Status of J2
j3	bool	Status of J3
j4	bool	Status of J4
j5	bool	Status of J5

Attribute	Type	Description
j6	bool	Status of J6
j7	bool	Status of J7
j8	bool	Status of J8
j9	bool	Status of J9

## 3.25 ModbusChannel

### Description

Modbus channel type.

### Import

```
from Agilebot import ModbusChannel
```

python

### Fields

Name	Enum Value	Description
CONTROLLER_TCP_TO_485	2	TCP-to-RS485 module channel
WRIST_485_0	3	Wrist AI channel
WRIST_485_1	4	Wrist DO channel
CONTROLLER_485	5	Controller 485 channel

## 3.26 PayloadIdentifyState

### Description

Payload identification state.

Import

```
from Agilebot import PayloadIdentifyState
```

python

Fields

Name	Enum Value	Description
PAYLOAD_CHECK_INIT	0	Payload check initialization in progress
PAYLOAD_CHECK_RUNNING	1	Payload check running
PAYLOAD_CHECK_SUCCESS	2	Payload check succeeded
PAYLOAD_CHECK_FAILED	3	Payload check failed
PAYLOAD_IDENTIFY_INIT	4	Payload identification initialization
PAYLOAD_IDENTIFY_RUNNING	5	Payload identification running
PAYLOAD_IDENTIFY_SUCCESS	6	Payload identification succeeded
PAYLOAD_IDENTIFY_FAILED	7	Payload identification failed
WRONG_DATA	-1	Invalid data

3.27 MoveMode

Description

Jog move mode.

Import

```
from Agilebot import MoveMode
```

python

Fields

Name	Enum Value	Description
Continuous	0	Continuous motion
Stepping	1	Step motion

## 3.28 SignalType

### Description

**SignalType** is an enumeration class that distinguishes between digital, user, robot, group, and analog IO signals so the robot control system can correctly identify and manage each signal type.

### Import

```
from Agilebot import SignalType
```

python

### Fields

Name	Enum Value	Description
DI	1	Digital input signal
DO	2	Digital output signal
UI	3	User input signal
UO	4	User output signal
RI	5	Remote-control input signal
RO	6	Remote-control output signal
GI	7	Group input signal
GO	8	Group output signal
TAI	9	Tool analog input signal
TDI	10	Tool digital input signal

Name	Enum Value	Description
TDO	11	Tool digital output signal
AI	12	Analog input signal
AO	13	Analog output signal

## 3.29 SignalValue

### Description

**SignalValue** defines signal on/off values using two constants, **OFF** and **ON**, which represent the off and on states of a signal.

### Import

```
from Agilebot import SignalValue
```

python

### Attributes

Name	Value	Description
OFF	0	Signal off
ON	1	Signal on

## 3.30 SerialParams

### Description

Serial communication configuration. Used when performing Modbus-RTU/TCP to 485 communication. The instance can be passed directly to the lower-level communication interface to describe connection parameters.

### Import

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

python

## Constructor

```
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  
)
```

python

## Attributes

Attribute	Type	Description
id	int	Device station / slave ID
channel	ModbusChannel	Communication channel enumeration, defaults to <code>CONTROLLER_TCP_TO_485</code>
ip	str	Gateway IP address when using TCP-to-485
port	int	Gateway port when using TCP-to-485
baud	int	Baud rate, default <code>9600</code> bps
data_bit	int	Data bits, default 8
stop_bit	int	Stop bits, default 1
parity	ModbusParity	Parity setting, default <code>ModbusParity.NONE</code>
timeout	int	Read/write timeout in milliseconds, default <code>1000</code>

## 3.31 SubPub Topic Types

### 3.31.1 RobotTopicType

#### Description

Robot real-time topic enumeration used for subscribing or publishing robot status.

#### Import

```
from Agilebot import RobotTopicType
```

python

#### Enum Values

Name	Description
JOINT_POSITION	Joint feedback (rad)
CARTESIAN_POSITION	TCP pose under the active user/tool frames
TCP_WORLD_CARTESIAN	TCP pose in the world frame
TCP_BASE_CARTESIAN	TCP pose in the base frame
USER_FRAME	Current user frame
TOOL_FRAME	Current tool frame
VELOCITY_RATIO	Global speed ratio (0-100%)
GLOBAL_ACC_RATIO	Global acceleration ratio (0-100%)
CALIBRATE_STATUS	Axis-group calibration status
TRAJECTORY_RECORD	Trajectory recording status
RUNNING_STATUS	Controller running status
INTERPRETER_STATUS	Interpreter status (running/paused/stopped/error)
ROBOT_STATUS	Robot overall status

Name	Description
CTRL_STATUS	Controller readiness status
SERVO_STATUS	Servo enable status
USER_OP_MODE	User operation mode
SOFT_OP_MODE	Soft operation mode
OPERATION_STATUS	Operation status
PERFORMANCE_GEAR	Performance gear
POWER_STATUS	Power status
POWER_ON_STATUS	Power-on status
SAFETY_ALARM	Safety alarm status
SAFETY_PLANE_STATUS	Safety plane status
TOOL_POSTURE_LIMIT_STATUS	Tool posture limit status
JOINTS_RECORD	Joint record
TP_PROGRAM_STATUS	TP program status

### 3.31.2 RegTopicType

#### Description

Register subscription topic enumeration for accessing internal R / MR / SR / PR registers.

#### Import

```
from Agilebot import RegTopicType
```

python

#### Enum Values

Name	Value	Description
R	R	Generic register
MR	MR	Motion register
SR	SR	String register
PR	PR	Position register

### 3.31.3 IOTopicType

#### Description

IO subscription topic enumeration covering digital, group, analog, robot, and tool IO.

#### Import

```
from Agilebot import IOTopicType
```

python

#### Enum Values

Name	Value	Description
DI	DI	Digital input
DO	DO	Digital output
GI	GI	Group input
GO	GO	Group output
RI	RI	Remote-control input
RO	RO	Remote-control output
UI	UI	User input
UO	UO	User output
TDI	TDI	Tool digital input

Name	Value	Description
TDO	TDO	Tool digital output
TAI	TAI	Tool analog input
AI	AI	Analog input
AO	AO	Analog output

## 4 Methods and Examples

## 4.1 Basic Robot Operations

### Overview

The Arm class encapsulates most high-frequency interfaces related to Jiebote robots, handling connection management, status queries, and control-command delivery. Typical usage:

1. Instantiate `Arm(local_proxy=False)`
2. Call `connect()` to establish communication with the controller/pendant
3. Invoke motion, status, or I/O APIs as required
4. Finally call `disconnect()` or `release_access()` to free resources

No manual configuration is needed; the class automatically checks SDK version, identifies controller type, selects the proxy service, and logs the chosen communication path.

When `local_proxy=True`, ensure the local machine is on the same subnet and the controller IP is reachable. For industrial robots, if you need to keep the PC mode active, call `acquire_access()` after connecting and promptly call `release_access()` afterward to avoid locking the pendant's control authority.

### Constructor

Method Name	<code>Arm( local_proxy : bool = False)</code>
Description	The Agilebot robot class, containing all available interfaces. When instantiated it prints the SDK version and manual URL.
Request Parameters	<code>local_proxy</code> : bool Whether to launch a local controller proxy service. Default <code>False</code> . <ul style="list-style-type: none"><li>- <code>True</code> : start the proxy locally (requires the controller IP, not a domain).</li><li>- <code>False</code> : use the proxy service on the controller/teach pendant (robot software <math>\geq</math> v7.7 with the proxy installed).</li></ul>
Return Value	<a href="#"><code>StatusCodeEnum</code></a> : Function execution result

Method Name	<b>Arm( <code>local_proxy</code> : bool = False)</b>
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.1 Connect to the Robot

Method Name	<b>connect(arm_controller_ip: str = <code>COBOT_IP</code> , teach_panel_ip: Optional[str] = None) -&gt; StatusCodeEnum</b>
Description	Connect to the Agilebot robot. The method auto-detects cooperative/industrial models, fills in teach pendant IPs, and starts the appropriate proxy service.
Request Parameters	<code>arm_controller_ip</code> : str Controller IP (defaults to the <code>COBOT_IP</code> constant). <code>teach_panel_ip</code> : Optional[str] Teach pendant IP for industrial robots. When omitted, the method auto-assigns the default pendant IP or reuses the controller IP.
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result
Notes	<ul style="list-style-type: none"> <li>- Local proxy is started only if <code>local_proxy=True</code> <b>and</b> a valid IP address is provided; Airbot domains only support HTTP and cannot work with the local proxy.</li> <li>- Invalid IP/domain values return <code>INVALID_IP_ADDRESS</code> .</li> <li>- A failure to contact the controller returns <code>CONTROLLER_CONNECTION_ERROR</code> with details.</li> </ul>
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.2 Check Connection Status with the Robot

Method Name	<b>is_connected() -&gt; bool</b>
Description	Check whether the connection with the robot is valid
Request Parameters	No parameters

Method Name	<b>is_connected()</b> -> bool
Return Value	bool: Connection status, True: Connection is valid, False: Connection is invalid
Notes	The legacy <code>is_connect()</code> method is deprecated; use <code>is_connected()</code> instead.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.3 Disconnect from the Robot

Method Name	<b>disconnect()</b>
Description	Disconnect from the Agilebot robot
Request Parameters	No parameters
Return Value	No return
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 arm/connect\_disconnect.py

py

```
#!/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 connection
"""
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] 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 Get the Current Robot Model

Method Name	<code>get_arm_model_info()</code> -> tuple[str, StatusCodeEnum]
Description	Get the current robot model information
Request Parameters	No parameters
Return Value	str: Robot model, e.g., "GBT-C5A" <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 `arm/get_arm_model_info.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.errmsg}")
    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.errmsg}")
    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 Get the Robot's Operating Status

Method Name	<b>get_robot_status()</b> -> tuple[RobotStatusEnum, StatusCodeEnum]
Description	Get the operating status of the Agilebot robot
Request Parameters	No parameters
Return Value	<a href="#">RobotStatusEnum</a> : Robot operating status <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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.errmsg}")
    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, error code")
    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 Get the Current Controller Operating Status

Method Name	get_ctrl_status() -> tuple[CtrlStatusEnum, StatusCodeEnum]
Description	Get the current operating status of the Agilebot robot controller
Request Parameters	No parameters
Return Value	<a href="#">CtrlStatusEnum</a> : Controller operating status <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

📄 arm/get\_ctrl\_status.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 status of the
"""
from Agilebot import 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 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")

```

## 4.1.7 Get the Current Servo Controller Status

Method Name	<code>get_servo_status()</code> -> tuple[ServoStatusEnum, StatusCodeEnum]
Description	Get the current status of the Agilebot robot servo controller

Method Name	<code>get_servo_status()</code> -> tuple[ServoStatusEnum, StatusCodeEnum]
Request Parameters	No parameters
Return Value	<a href="#">ServoStatusEnum</a> : Servo controller status <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

🐍 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.errmsg}")
    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: {state.msg}")
    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 Get the Current Soft Mode of the Robot

Method Name	<b>get_soft_mode()</b> -> tuple[SoftModeEnum, StatusCodeEnum]
Description	Get the current soft mode of the Agilebot robot (PC mode manual/auto state)
Request Parameters	No parameters
Return Value	<a href="#">SoftModeEnum</a> : Soft mode status <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.9 Set the Current Soft Mode of the Robot

Method Name	<b>set_soft_mode( <code>soft_mode</code> : SoftModeEnum)</b> -> StatusCodeEnum
Description	Set the current soft mode of the robot (PC mode manual/auto state)
Request Parameters	<code>soft_mode</code> : <a href="#">SoftModeEnum</a> Soft mode value

Method Name	<code>set_soft_mode( <a href="#">soft_mode</a> : SoftModeEnum) -&gt; StatusCodeEnum</code>
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

🐍 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.errmsg}")
    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 state 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 current soft state to manual limit mode failed, error code: {ret.errmsg}")
    arm.disconnect()
    exit(1)
```

```

        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 current soft
    )
    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 successfully")
    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 state fail
    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 Get Robot Operation Mode

Method Name	<b>get_op_mode()</b> -> tuple[SoftModeEnum, StatusCodeEnum]
Description	Retrieve the robot operation mode (manual/auto permissions in PC mode).
Request Parameters	None
Return Value	<a href="#">SoftModeEnum</a> : Operation mode <a href="#">StatusCodeEnum</a> : Function execution result
Notes	If the controller returns an invalid value, the method falls back to <code>SoftModeEnum.UNKNOWN</code> .
Compatible robot software	Collaborative (Copper): v7.5.0.0+

Method Name	<b>get_op_mode()</b> -> tuple[SoftModeEnum, StatusCodeEnum]
version	Industrial (Bronze): v7.5.0.0+

## 4.1.11 Set Robot Operation Mode

Method Name	<b>set_op_mode</b> ( <code>soft_mode</code> : SoftModeEnum) -> StatusCodeEnum
Description	Set the robot operation mode. Only supported for virtual robots/simulators.
Request Parameters	<code>soft_mode</code> : Target <a href="#">SoftModeEnum</a> , cannot be <code>UNKNOWN</code> .
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Notes	Passing <code>SoftModeEnum.UNKNOWN</code> returns <code>UNSUPPORTED_PARAMETER</code> .
Compatible robot software version	Collaborative (Copper): Simulation only Industrial (Bronze): Simulation only

## 4.1.12 Upper Computer Acquires Control Authority

Method Name	<b>acquire_access()</b>
Description	The PC acquires control authority, putting the robot into <a href="#">PC mode</a> . Internally starts a heartbeat thread to keep the mode alive.
Request Parameters	No parameters
Return Value	No return
Note	Only industrial robots require this call; collaborative robots and P7A do not.
Compatible robot software version	Collaborative (Copper): Not supported Industrial (Bronze): v7.5.0.0+

## 4.1.13 Upper Computer Releases Control Authority

Method Name	<code>release_access()</code>
Description	Release the PC control authority, stopping the <a href="#">PC mode</a> heartbeat and handing control back to the teach pendant.
Request Parameters	No parameters
Return Value	No return
Note	Only industrial robots need this call.
Compatible robot software version	Collaborative (Copper): Not supported Industrial (Bronze): v7.5.0.0+

### Example Code

 `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 permissions
"""
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 Get the Robot Controller Version

Method Name	<b>get_controller_version()</b> -> tuple[str, StatusCodeEnum]
Description	Recommended API to obtain the current controller version.
Request Parameters	No parameters
Return Value	str: Controller version <a href="#">StatusCodeEnum</a> : Function execution result
Notes	If the version is below the recommended level, the SDK prints an upgrade warning during connection.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 arm/get\_version.py

```
#!/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 versi
"""
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: ")
    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 Set the Robot LED Indicator Light

Method Name	<code>switch_led_light( <b>mode</b> : bool) -&gt; StatusCodeEnum</code>
Description	Set the LED indicator light of the Agilebot robot

Method Name	<code>switch_led_light( mode : bool) -&gt; StatusCodeEnum</code>
Request Parameters	<code>mode</code> : bool, True to turn on, False to turn off
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.1.3+ Industrial (Bronze): Not supported

Example Code

🐍 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 of the LE
"""
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.errmsg}")
    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.err}")
    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.err}")
    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 Power On the Robot Servo

Method Name	<code>servo_on()</code> -> <code>StatusCodeEnum</code>
Description	Power on the servo of the Agilebot robot
Request Parameters	No parameters
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.17 Power Off the Robot Servo

Method Name	<b><code>servo_off()</code></b> -> <code>StatusCodeEnum</code>
Description	Power off the servo of the Agilebot robot
Request Parameters	No parameters
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.1.18 Reset the Robot Servo

Method Name	<b><code>servo_reset()</code></b> -> <code>StatusCodeEnum</code>
Description	Reset the servo of the Agilebot robot
Request Parameters	No parameters
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 `arm/servo_operate.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 example
"""
import time

from Agilebot import Arm, StatusCodeEnum
```

py

```

# [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] 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.errmsg}")
    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.errmsg}")
    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.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")
```

### 4.1.19 Emergency Stop

Method name	<b>estop()</b> -> StatusCodeEnum
Description	Immediately triggers an emergency stop on the current Agilebot robot.
Parameters	None
Return value	<a href="#">StatusCodeEnum</a> : execution result of the function
Compatible robot software versions	Cobot (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 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.errmsg}")
    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 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 Robot Motion Control and Status

### Overview

The Motion class is the core object for motion control of Jiebote robots. It provides a unified interface for setting speed/acceleration parameters, coordinate systems, point conversion, trajectory motion, drag-to-teach, real-time control, and payload management.

Typical workflow: after the Arm connection is established, obtain the Motion instance via `arm.motion` .

### 4.2.1 Get Robot Parameters

#### 4.2.1.1 Get OVC Overall Velocity Coefficient

Method Name	<code>motion.get_OVC()</code> -> tuple[float, StatusCodeEnum]
Description	Get the current OVC Overall Velocity Coefficient of the robot, which ranges from 0 to 1
Request Parameters	No parameters
Return Value	float: Velocity ratio, ranging from 0 to 1 <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### 4.2.1.2 Get OAC Overall Acceleration Coefficient

Method Name	<code>motion.get_OAC()</code> -> tuple[float, StatusCodeEnum]
Description	Get the current OAC Overall Acceleration Coefficient of the robot, which ranges from 0 to 1.2

Method Name	<b>motion.get_OAC()</b> -> tuple[float, StatusCodeEnum]
Request Parameters	No parameters
Return Value	float: Acceleration ratio, ranging from 0 to 1.2 <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.2.1.3 Get Current TF Tool Coordinate System Number

Method Name	<b>motion.get_TF()</b> -> tuple[int, StatusCodeEnum]
Description	Get the current TF tool coordinate system number used by the robot, valid range 0~50
Request Parameters	No parameters
Return Value	int: Tool coordinate system number <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.2.1.4 Get Current UF User Coordinate System Number

Method Name	<b>motion.get_UF()</b> -> tuple[int, StatusCodeEnum]
Description	Get the current UF user coordinate system number used by the robot, valid range 0~50
Request Parameters	No parameters
Return Value	int: User coordinate system number <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.2.1.5 Get Current TCS Teaching Coordinate System

Method Name	<b>motion.get_TCS()</b> -> tuple[TCSType, StatusCodeEnum]
Description	Get the current TCS teaching coordinate system used by the robot, refer to <a href="#">TCSType</a> for details
Request Parameters	No parameters
Return Value	<a href="#">TCSType</a> : Teaching coordinate system number <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

🐍 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.errmsg}")
    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, error code")
    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 failed, error code")
    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 successful")
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")
    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, error code")
    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, error code")
    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")
```

## 4.2.2 Set Robot Parameters

### 4.2.2.1 Set OVC Overall Velocity Coefficient

Method Name	<b>motion.set_OVC</b> ( <span style="background-color: #e6f2ff;">value</span> : float) -> StatusCodeEnum
Description	Set the OVC Overall Velocity Coefficient of the robot
Request Parameters	<span style="background-color: #e6f2ff;">value</span> : float, velocity ratio in the range (0, 1], i.e., greater than 0 and no more than 1
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.2.2.2 Set OAC Overall Acceleration Coefficient

Method Name	<b>motion.set_OAC</b> ( <span style="background-color: #e6f2ff;">value</span> : float) -> StatusCodeEnum
Description	Set the OAC Overall Acceleration Coefficient of the robot
Request Parameters	<span style="background-color: #e6f2ff;">value</span> : float, acceleration ratio in the range (0.01, 1.2]
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.2.2.3 Set Current TF Tool Coordinate System

Method Name	<b>motion.set_TF</b> ( <span style="background-color: #e6f2ff;">value</span> : int) -> StatusCodeEnum
Description	Set the current TF tool coordinate system used by the robot

Method Name	<b>motion.set_TF</b> ( <code>value</code> : int) -> StatusCodeEnum
Request Parameters	<code>value</code> : int, tool coordinate system number, valid range 0~50
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### 4.2.2.4 Set Current UF User Coordinate System

Method Name	<b>motion.set_UF</b> ( <code>value</code> : int) -> StatusCodeEnum
Description	Set the current UF user coordinate system used by the robot
Request Parameters	<code>value</code> : int, user coordinate system number, valid range 0~50
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### 4.2.2.5 Set Current TCS Teaching Coordinate System

Method Name	<b>motion.set_TCS</b> ( <code>value</code> : TCSType) -> StatusCodeEnum
Description	Set the current TCS teaching coordinate system used by the robot, refer to <a href="#">TCSType</a> for details
Request Parameters	<code>value</code> : <a href="#">TCSType</a> , TCS teaching coordinate system
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

```
 motion/get_param.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.errmsg}")
    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, error code")
    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 failed, error")
    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 successful")
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}")
    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, error code: {ret}")
    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, error code: {ret}")
    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")

```

## 4.2.3 Convert Cartesian Pose to Joint Pose

Method Name	<code>motion.convert_cart_to_joint( <a href="#">pose</a> : MotionPose, <a href="#">uf_index</a> : int = 0, <a href="#">tf_index</a> : int = 0) -&gt; tuple[MotionPose, StatusCodeEnum]</code>
Description	Convert pose data from Cartesian coordinates to joint coordinates
Request Parameters	<p><a href="#">pose</a> : <a href="#">MotionPose</a> Robot pose in Cartesian coordinates (PoseType.CART). If posture is not provided, the SDK automatically finds a feasible posture</p> <p><a href="#">uf_index</a> : int User coordinate system ID</p> <p><a href="#">tf_index</a> : int Tool coordinate system ID</p>
Return Value	<p><a href="#">MotionPose</a>: Robot pose</p> <p><a href="#">StatusCodeEnum</a>: Function execution result</p>
Compatible robot software version	<p>Collaborative (Copper): v7.5.0.0+</p> <p>Industrial (Bronze): v7.5.0.0+</p>

## 4.2.4 Convert Joint Pose to Cartesian Pose

Method Name	<code>motion.convert_joint_to_cart( <a href="#">pose</a> : MotionPose, <a href="#">uf_index</a> : int = 0, <a href="#">tf_index</a> : int = 0) -&gt; tuple[MotionPose, StatusCodeEnum]</code>
Description	Convert joint coordinates to Cartesian coordinates
Request Parameters	<p><a href="#">pose</a> : <a href="#">MotionPose</a> Robot's pose in joint coordinates</p> <p><a href="#">uf_index</a> : int User coordinate system ID</p> <p><a href="#">tf_index</a> : int Tool coordinate system ID</p>
Return Value	<p><a href="#">MotionPose</a>: Robot pose</p> <p><a href="#">StatusCodeEnum</a>: Function execution result</p>
Compatible robot software version	<p>Collaborative (Copper): v7.5.0.0+</p> <p>Industrial (Bronze): v7.5.0.0+</p>

### Example Code

 `motion/convert_pose.py`

```
#!/python
"""
```

py

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.errmsg}")
    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, 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: {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 failed, error")
    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 Move the Robot End Effector to a Specified Position

Method Name	<code>motion.move_joint( <b>pose</b> : MotionPose, <b>vel</b> : float = 1, <b>acc</b> : float = 1) -&gt; StatusCodeEnum</code>
Description	Move the robot end effector to a specified position using the fastest path
Request Parameters	<b>pose</b> : <a href="#">MotionPose</a> Coordinates of a point in Cartesian or joint coordinate system <b>vel</b> : float Velocity of movement, ranging from 0 to 1, representing a multiple of the maximum velocity <b>acc</b> : float Ranging from 0 to 1.2, representing a multiple of the maximum acceleration
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 motion/move\_joint.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

# [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.errmsg}")
    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_joint(motion_pose, vel=0.5, acc=0.5)
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.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 Move the Robot End Effector in a Straight Line to a Specified Position

Method Name	<b>motion.move_line</b> ( <b>pose</b> : MotionPose, <b>vel</b> : float = 100, <b>acc</b> : float = 1) -> StatusCodeEnum
Description	Move the robot end effector in a straight line to a specified position

Method Name	<code>motion.move_line( pose : MotionPose, vel : float = 100, acc : float = 1) -&gt; StatusCodeEnum</code>
Request Parameters	<code>pose</code> : <a href="#">MotionPose</a> Coordinates of a point in Cartesian or joint coordinate system <code>vel</code> : float Velocity of movement, ranging from 1 to 4000 mm/s, representing the end effector's movement speed <code>acc</code> : float Ranging from 0 to 1.2, representing a multiple of the maximum acceleration
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

 `motion/move_line.py`

py

```
#!/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.errmsg}")
    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("直线运动成功 / Line motion successful")
else:
    print(f"直线运动失败, 错误代码 / Line motion 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 after completion
arm.disconnect()
print("机器人断开连接成功 / Robot disconnected successfully")
```

### 4.2.7 Move the Robot End Effector in a Circular Path to a Specified Position

Method Name	<code>motion.move_circle( <span>pose1</span> : MotionPose, <span>pose2</span> : MotionPose, <span>vel</span> : float = 100, <span>acc</span> : float = 1.0) -&gt; StatusCodeEnum</code>
Description	Move the robot end effector in a circular path to a specified position
Request Parameters	<div><div><span>pose1</span></div><div>: <a href="#">MotionPose</a></div><div>Intermediate point pose of the robot movement</div></div> <div><div><span>pose2</span></div><div>: <a href="#">MotionPose</a></div><div>End point pose of the robot movement</div></div> <div><div><span>vel</span></div><div>: float</div><div>Velocity of movement, ranging from 1 to 4000 mm/s, representing the end effector's movement speed</div></div>

Method Name	<b>motion.move_circle</b> ( <b>pose1</b> : MotionPose, <b>pose2</b> : MotionPose, <b>vel</b> : float = 100, <b>acc</b> : float = 1.0) -> StatusCodeEnum
	<b>acc</b> : float Ranging from 0 to 1.2, representing a multiple of the maximum acceleration
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

🐍 motion/move\_circle.py

py

```
#!/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 multiprocessing.connection import wait

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.errmsg}")
    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("运动到初始点成功 / Move to initial position successful")
else:
    print(f"运动到初始点失败，错误代码 / Move to initial position failed, error code: {ret}")
    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("圆弧运动成功 / Circle motion successful")
else:
    print(f"圆弧运动失败, 错误代码 / Circle motion failed, error 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 Get the Current Pose of the Robot

Method Name	<code>motion.get_current_pose( <span>pose_type</span> : PoseType, <span>uf_index</span> : int = 0, <span>tf_index</span> : int = 0) -&gt; tuple[MotionPose, StatusCodeEnum]</code>
Description	Get the current pose of the robot, which can be in Cartesian or joint coordinate system
Request Parameters	<span>pose_type</span> : <a href="#">PoseType</a> Pose type <span>uf_index</span> : When using PoseType.CART, the user coordinate system ID must be provided, default is 0 <span>tf_index</span> : When using PoseType.CART, the tool coordinate system ID must be provided, default is 0
Return Value	<a href="#">MotionPose</a> : Robot pose <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 `motion/get_current_pose.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.errmsg}")
    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.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"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.err}")
    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 Get the Robot's DH Parameters


Method Name	<b>motion.get_DH_param()</b> -> tuple[list[DHparam], StatusCodeEnum]
Description	Get the robot's DH parameters

Method Name	<b>motion.get_DH_param()</b> -> tuple[list[DHparam], StatusCodeEnum]
Request Parameters	No parameters
Return Value	list( <a href="#">DHparam</a> ): List of DH parameters <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

### 4.2.10 Set the Robot's DH Parameters

Method Name	<b>motion.set_DH_param( <code>dh_list</code> : list[DHparam])</b> -> StatusCodeEnum
Description	Set the robot's DH parameters
Request Parameters	<code>dh_list</code> : list( <a href="#">DHparam</a> ) List of DH parameters
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

#### Example Code

 motion/DH\_param.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()
```

py

```

# [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.errmsg}")
    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: {ret.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: {ret.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}\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")
```

## 4.2.11 Get the Robot Axis Lock Status

Method Name	<code>motion.get_drag_set()</code> -> tuple[DragStatus, StatusCodeEnum]
Description	Get the current robot axis lock status, which only applies to teaching movements
Request Parameters	No parameters
Return Value	<a href="#">DragStatus</a> : Axis lock status, True indicates the axis is movable, False indicates it is locked <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

### Example Code

 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.errmsg}")
    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.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")

```

## 4.2.12 Set the Robot Axis Lock Status

Method Name	<b>motion.set_drag_set</b> ( <b>drag_status</b> : DragStatus) -> StatusCodeEnum
Description	Set the current robot axis lock status, which only applies to teaching movements
Request Parameters	<b>drag_status</b> : <a href="#">DragStatus</a> Lock status of each axis, default is all True: unlocked state
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

🐍 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.errmsg}")
    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 Enable Dragging for the Robot

Method Name	<code>motion.enable_drag( drag_state : bool) -&gt; StatusCodeEnum</code>
Description	Enable or disable dragging for the robot
Request Parameters	<code>drag_state</code> : bool Whether the robot is allowed to drag, True indicates entering drag mode, False indicates exiting drag mode
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

### Example Code

 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.errmsg}")
    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.errr
    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: {ret.errr
    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 Enter Real-Time Position Control Mode

Method Name	<code>motion.enter_position_control()</code> -> <code>StatusCodeEnum</code>
Description	Enter real-time position control mode to allow precise position control of the robot
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Note	After entering real-time control mode, control commands must be sent via UDP

Method Name	<b>motion.enter_position_control()</b> -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

## 4.2.15 Exit Real-Time Position Control Mode

Method Name	<b>motion.exit_position_control()</b> -> StatusCodeEnum
Description	Exit real-time position control mode and return to the default robot control state
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Note	After exiting, the robot will no longer accept real-time control commands
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

## 4.2.16 Set UDP Feedback Parameters

Method Name	<b>motion.set_udp_feedback_params( <a href="#">flag</a> : bool, <a href="#">ip</a> : str, <a href="#">interval</a> : int, <a href="#">feedback_type</a> : int, <a href="#">DO_list</a> : list[int] = None)</b> -> StatusCodeEnum
Description	Configure the UDP feedback parameters for pushing data to a specified IP address
Request Parameters	<a href="#">flag</a> : bool Whether to enable UDP data pushing; <a href="#">ip</a> : str IP address of the receiver; <a href="#">interval</a> : int Interval for sending data (unit: milliseconds); <a href="#">feedback_type</a> : int Feedback data format (0: XML, 1: JSON, 2: PROTO); <a href="#">DO_list</a> : list[int] List of DO signals to be obtained (up to ten, optional)
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

Method Name	<code>motion.set_udp_feedback_params( <b>flag</b> : bool, <b>ip</b> : str, <b>interval</b> : int, <b>feedback_type</b> : int, <b>DO_list</b> : list[int] = None) -&gt; StatusCodeEnum</code>
Note	Parameter settings are only effective when the UDP data pushing function is enabled
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Example Code

🐍 motion/UDP\_feedback\_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 control mode u
"""
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.errmsg}")
    arm.disconnect()
    exit(1)

# [ZH] 设置udp反馈参数
# [EN] Set UDP feedback parameters
ret = arm.motion.set_udp_feedback_params(True, "10.27.1.254", 20, 1, [0, 1, 2])
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, error code")
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 successful")
else:
    print(f"进入实时位置控制模式失败, 错误代码 / Enter real-time position control mode failed, error code")
    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 successful")
else:
    print(f"退出实时位置控制模式失败, 错误代码 / Exit real-time position control mode failed, error code")
    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")
```

## Data Pushing Description

Name	Field	Description
<b>Rlst: Cartesian Position</b>	X	X-direction value in the tool coordinate system, in millimeters

Name	Field	Description
	Y	Y-direction value in the tool coordinate system, in millimeters
	Z	Z-direction value in the tool coordinate system, in millimeters
	A	Rotation around the X-axis in the tool coordinate system, in degrees
	B	Rotation around the Y-axis in the tool coordinate system, in degrees
	C	Rotation around the Z-axis in the tool coordinate system, in degrees
<b>AIPos: Joint Position</b>	A1-A6	Values of the six joints, in degrees
<b>EIPos: Additional Axis Data</b>	EIPos	Additional axis data
<b>WristBtnState: Wrist Button State</b>	Button State	1 = Button pressed, 0 = Button released
	DragModel	Drag button state
	RecordJoint	Teaching record button state
	PauseResume	Pause/Resume button state
<b>Digout: DO Output</b>	Digout	State of digital output (DO)
<b>ProgramStatus: Program Status</b>	ProgId	Program ID
	Status	Interpreter execution status: 0 = INTERPRETER_IDLE 1 = INTERPRETER_EXECUTE 2 = INTERPRETER_PAUSED
	XPath	Program fragment return value, in the format <code>program_name:line_number</code>
<b>IPOC: Timestamp</b>	IPOC	Timestamp

## 4.2.17 Get the Robot's Soft Limits

Method Name	<b>motion.get_user_soft_limit()</b> -> tuple[list[list[float]], StatusCodeEnum]
Description	Get the current soft limits of the robot
Request Parameters	None
Return Value	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 <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 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 limit 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.errmsg}")
    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.e
    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 Payload-Related Interfaces

### 4.2.18.1 Get the Current Activated Payload ID

Method Name	<b>motion.payload.get_current_payload()</b> -> tuple[int, StatusCodeEnum]
Description	Get the current activated payload ID
Request Parameters	None
Return Value	int: Payload ID <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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.errmsg}")
    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 Get Payload Information by Specified ID

Method Name	<code>motion.payload.get_payload_by_id( <a href="#">payload_id</a> : int) -&gt; tuple[PayloadInfo, StatusCodeEnum]</code>
Description	Get the payload information by the specified ID
Request Parameters	<code><a href="#">payload_id</a> : int</code> The specified payload ID
Return Value	<a href="#">PayloadInfo</a> : Payload information <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 `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 based 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.errmsg}")
    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.errmsg}")
    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.ma"
    f"负载转动惯量 / Payload inertia moment: {res.inertia_moment.lx}, {res.inertia_momer"
)

# [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 Activate Payload by Specified ID

Method Name	<code>motion.payload.set_current_payload( <a href="#">payload_id</a> : int) -&gt; StatusCodeEnum</code>
Description	Activate the payload by the specified ID
Request Parameters	<code><a href="#">payload_id</a></code> : int The specified payload ID
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

Method Name	<b>motion.payload.set_current_payload</b> ( <b>payload_id</b> : int) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

🐍 motion/set\_current\_payload.py

py

```
#!/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.errmsg}")
    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 Add a User-Defined Payload to the Robot Controller

Method Name	<code>motion.payload.add_payload( <a href="#">payload_info</a> : PayloadInfo) -&gt; StatusCodeEnum</code>
Description	Add a user-defined payload to the robot controller
Request Parameters	<code>payload_info</code> : <a href="#">PayloadInfo</a> User-defined payload information
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 motion/add\_payload.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:
```

py

```

    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] 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.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")

```

#### 4.2.18.5 Delete the Payload Information by Specified ID

Method Name	<code>motion.payload.delete_payload(payload_id : int) -&gt; StatusCodeEnum</code>
Description	Delete a user-defined payload from the controller

Method Name	<b>motion.payload.delete_payload( <code>payload_id</code> : int) -&gt; StatusCodeEnum</b>
Request Parameters	<code>payload_id</code> : int The specified payload ID
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	Note: The currently activated payload cannot be deleted. If you want to delete the activated payload, please activate another payload first and then delete the current one.

## Example Code

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

# [ZH] 打印结果
# [EN] Print result
print(f"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 Update an Existing Payload Information

Method Name	<code>motion.payload.update_payload( <code>payload_info</code> : PayloadInfo) -&gt; StatusCodeEnum</code>
Description	Update the information of an existing user-defined payload
Request Parameters	<code>payload_info</code> : <a href="#">PayloadInfo</a> User-defined updated payload information
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

 `motion/update_payload.py`

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

py

```

"""
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.errmsg}")
    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.errmsg}")
    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}\n"
    f"负载转动惯量 / Payload inertia moment: {payload_info.inertia_moment.lx}, {payload_info.inertia_moment.ly}\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 Get All Payload Information

Method Name	<code>motion.payload.get_all_payload()</code> -> tuple[list, StatusCodeEnum]
Description	Get all payload information
Request Parameters	None
Return Value	list : List of all payload information <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 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.errmsg}")
    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: {ret.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" f"负载注释 / Payload comment: {payload}")

# [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 Check if Axis 3 is Horizontal

Method Name	<code>motion.payload.check_axis_three_horizontal()</code> -> tuple[float, StatusCodeEnum]
Description	Check if Axis 3 is horizontal

Method Name	<b>motion.payload.check_axis_three_horizontal()</b> -> tuple[float, StatusCodeEnum]
Request Parameters	None
Return Value	float: The horizontal angle of Axis 3, in degrees <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	The horizontal angle must be between -1 and 1 to proceed with payload identification

#### 4.2.18.9 Get Payload Identification State

Method Name	<b>motion.payload.get_payload_identify_state()</b> -> tuple[PayloadIdentifyState, StatusCodeEnum]
Description	Get the state of payload identification
Request Parameters	None
Return Value	<a href="#">PayloadIdentifyState</a> The state of payload identification <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

#### 4.2.18.10 Start Payload Identification

Method Name	<b>motion.payload.start_payload_identify( <a href="#">weight</a> : float, <a href="#">angle</a> : float) -&gt; StatusCodeEnum</b>
Description	Start payload identification
Request Parameters	<a href="#">weight</a> : float Payload weight (use -1 for unknown weight); <a href="#">angle</a> : float Allowed rotation angle of Axis 6 (30-90 degrees)
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

Method Name	<b>motion.payload.start_payload_identify</b> ( <b>weight</b> : float, <b>angle</b> : float) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	The robot must enter the payload identification state before starting payload identification

#### 4.2.18.11 Get Payload Identification Result

Method Name	<b>motion.payload.payload_identify_result</b> () -> tuple[PayloadInfo, StatusCodeEnum]
Description	Get the result of payload identification
Request Parameters	None
Return Value	<a href="#">PayloadInfo</a> : The result of payload identification <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

#### 4.2.18.12 Start Interference Check for Payload Identification

Method Name	<b>motion.payload.interference_check_for_payload_identify</b> ( <b>weight</b> : float, <b>angle</b> : float) -> StatusCodeEnum
Description	Start interference check for payload identification
Request Parameters	<b>weight</b> : float Payload weight; <b>angle</b> : float Rotation angle of Axis 6 (30-90 degrees)
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

#### 4.2.18.13 Enter Payload Identification State

Method Name	<b>motion.payload.payload_identify_start()</b> -> StatusCodeEnum
Description	Enter payload identification state
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

#### 4.2.18.14 Exit Payload Identification State

Method Name	<b>motion.payload.payload_identify_done()</b> -> StatusCodeEnum
Description	Exit payload identification state
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

#### 4.2.18.15 Complete Payload Identification Process

Method Name	<b>motion.payload.payload_identify(</b> <a href="#">weight</a> : float, <a href="#">angle</a> : float) -> tuple[PayloadInfo, StatusCodeEnum]
Description	Complete payload identification process, including all the interfaces mentioned above. For general payload identification without special requirements, this interface is sufficient.
Request Parameters	<a href="#">weight</a> : float Payload weight (use -1 for unknown weight); <a href="#">angle</a> : float Rotation angle of Axis 6 (30-90 degrees)
Return Value	<a href="#">PayloadInfo</a> : Payload identification result <a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Method Name	<b>motion.payload.payload_identify</b> ( <b>weight</b> : float, <b>angle</b> : float) -> tuple[PayloadInfo, StatusCodeEnum]
Note	<p>The returned payload can be added to the robot or written to an existing payload in the robot.</p> <p>The complete process steps are:</p> <ol style="list-style-type: none"><li>1. Move to the specified horizontal position and check if it is horizontal</li><li>2. Enter payload identification state</li><li>3. Start payload identification</li><li>4. Get the payload identification result</li><li>5. Exit payload identification state</li></ol>

Example Code

🔗 motion/payload\_identify.py

```
#!/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.errmsg}")
    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.error_code}")
    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.error_code}")
        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.error_code}")

```

```
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 Robot Program Operation Class

### Overview

The Execution class provides a unified scheduling interface for robot programs and motion tasks. It can start/stop/pause/resume teach-pendant programs, manage the list of concurrently running tasks, and execute BAS scripts or other custom workflows. Leveraging the connection and motion capabilities of Arm and Motion, Execution is responsible for triggering and controlling program flow in the controller from the host side.

### 4.3.1 Execute a Specified Program

Method Name	<code>execution.start( <a href="#">program_name</a> : str) -&gt; StatusCodeEnum</code>
Description	Execute a specified program.
Request Parameters	<a href="#">program_name</a> : str The name of the program to be executed.
Return Value	<a href="#">StatusCodeEnum</a> : The result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.3.2 Stop the Currently Executing Program

Method Name	<code>execution.stop( <a href="#">program_name</a> : str = '') -&gt; StatusCodeEnum</code>
Description	Stop the currently executing program or stop the robot's current motion.
Request Parameters	<a href="#">program_name</a> : str The name of the program to be stopped. Default is an empty string, indicating stopping the currently running program or motion command.

Method Name	<b>execution.stop( <code>program_name</code> : str = '') -&gt; StatusCodeEnum</b>
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : The result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.3.3 Return Detailed Information of All Running Programs

Method Name	<b>execution.all_running_programs() -&gt; tuple[list, StatusCodeEnum]</b>
Description	Return detailed information of all running programs, including thread ID, program name, xpath, program status, and program type.
Request Parameters	None
Return Value	list: A list of running program details, where each element contains <code>thread_id</code> , <code>program_name</code> , <code>xpath</code> , <code>program_status</code> , <code>program_type</code> , etc. <a href="#"><u>StatusCodeEnum</u></a> : The result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.3.4 Pause Program Execution

Method Name	<b>execution.pause( <code>program_name</code> : str = '') -&gt; StatusCodeEnum</b>
Description	Pause the currently executing program or the robot's current motion.
Request Parameters	<code>program_name</code> : str The name of the program to be paused. Default is an empty string, indicating pausing the currently running program or motion command.
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : The result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.3.5 Resume Program Execution

Method Name	<code>execution.resume( <a href="#">program_name</a> : str = '') -&gt; StatusCodeEnum</code>
Description	Continue running a program that is in a paused state.
Request Parameters	<a href="#">program_name</a> : str The name of the program to be resumed. Default is an empty string, indicating resuming the currently paused program or motion command.
Return Value	<a href="#">StatusCodeEnum</a> : The result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 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
# [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.errmsg}")
    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.errmsg}")
    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.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")

```

## 4.3.6 Execute BAS Script Program

| Method Name | **execution.execute\_bas\_script**( **bas\_script** : BasScript | list[str]) -> StatusCodeEnum | |-----|-----| | Description | Execute a user-defined BAS script program. | | Request Parameters | **bas\_script** : [BasScript](#) or list[str] The user-defined BAS script program. | | Return Value | [StatusCodeEnum](#): The result of the function execution. | | Note | The pause, resume, and stop methods for BAS script programs are the same as for regular programs. | | Compatible robot software version | Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): v7.6.0.0+ |

 `execution/bas_script_execution.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.errmsg}")
    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, error code")
    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.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")
```

# 4.4 Program Information Read and Write

## Overview

The ProgramPose interfaces are used to read, write, and convert pose points stored in the robot’s teach-pendant programs.

Via the `program_pose` module you can locate a specific program and point index, perform CRUD operations, and switch between Cartesian and joint representations—making it easy to batch-maintain program points or perform off-line editing from the host PC.

### 4.4.1 Get the Value of a Specified Pose in a Program

Method Name	<code>program_pose.read( <code>program_name</code> : str, <code>index</code> : int, <code>program_type</code> : str = USER_PROGRAM) -&gt; tuple[ProgramPose, StatusCodeEnum]</code>
Description	Get the value of a Pose with a specified index in a program.
Request Parameters	<code>program_name</code> : str The name of the specified program. <code>index</code> : int The index of the specified Pose. <code>program_type</code> : str The type of the program, default is USER_PROGRAM.
Return Value	<a href="#">ProgramPose</a> : Pose information. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.4.2 Modify the Value of a Specified Pose in a Program

Method Name	<b>program_pose.write</b> ( <code>program_name</code> : str, <code>index</code> : int, <code>value</code> : ProgramPose, <code>program_type</code> : str = USER_PROGRAM) -> StatusCodeEnum
Description	Modify the value of a Pose with a specified index in a user-defined program.
Request Parameters	<code>program_name</code> : str The name of the specified program. <code>index</code> : int The index of the specified Pose. <code>value</code> : <a href="#">ProgramPose</a> The new value of the Pose to be updated. <code>program_type</code> : str The type of the program, default is USER_PROGRAM.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.4.3 Get All Poses in a Specified Program

Method Name	<b>program_pose.read_all_poses</b> ( <code>program_name</code> : str, <code>program_type</code> : str = USER_PROGRAM) -> tuple[list[ProgramPose], StatusCodeEnum]
Description	Get all Pose information in a specified program.
Request Parameters	<code>program_name</code> : str The name of the specified program. <code>program_type</code> : str The type of the program, default is USER_PROGRAM.
Return Value	list[ <a href="#">ProgramPose</a> ]: List of Pose information. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.4.4 Add a Pose Entry to a Program

Method Name	<b>program_pose.add</b> ( <code>program_name</code> : str, <code>index</code> : int, <code>value</code> : ProgramPose, <code>program_type</code> : str = USER_PROGRAM) -> StatusCodeEnum
Description	Add a new Pose at the specified index in a program. Returns an error if the

Method Name	<b>program_pose.add</b> ( <code>program_name</code> : str, <code>index</code> : int, <code>value</code> : ProgramPose, <code>program_type</code> : str = USER_PROGRAM) -> StatusCodeEnum
	index already exists.
Request Parameters	<code>program_name</code> : str The name of the specified program. <code>index</code> : int The index of the Pose to add. <code>value</code> : <a href="#">ProgramPose</a> The Pose data to insert. <code>program_type</code> : str The program type, default is USER_PROGRAM.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.4.5 Write One or More Robot Pose Values in a Program


Method Name	<b>program_pose.write_poses</b> ( <code>program_name</code> : str, <code>poses_to_update</code> : list[ProgramPose]) -> StatusCodeEnum
Description	Write one or more robot Pose values in a program. Note: Only existing points can be updated; new points cannot be added.
Request Parameters	<code>program_name</code> : str The name of the specified program. <code>poses_to_update</code> : list[ <a href="#">ProgramPose</a> ] List of poses to be updated.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.4.6 Robot Program Pose Type Conversion

Method Name	<b>ProgramPoses.convert_pose</b> ( <code>pose</code> : ProgramPose, <code>from_type</code> : PoseType, <code>to_type</code> : PoseType) -> tuple[ProgramPose, StatusCodeEnum]
Description	Convert the robot Pose values between joint coordinates and Cartesian space coordinates in a program.

Method Name	<b>ProgramPoses.convert_pose</b> ( <b>pose</b> : ProgramPose, <b>from_type</b> : PoseType, <b>to_type</b> : PoseType) -> tuple[ProgramPose, StatusCodeEnum]
Request Parameters	<b>pose</b> : <a href="#">ProgramPose</a> The Pose value to be converted. <b>from_type</b> : <a href="#">PoseType</a> The type before conversion. <b>to_type</b> : <a href="#">PoseType</a> The desired type after conversion.
Return Value	<a href="#">ProgramPose</a> : Pose information. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

 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.errmsg}")
    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" f"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.errmsg}")
    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: {ret.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.errmsg}")
arm.disconnect()
exit(1)

# [ZH] 转换位姿
# [EN] Convert pose
converted_pose, ret = arm.program_pose.convert_pose(pose, PoseType.CART, PoseType.JOIN)
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.errmsg}")
    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 Signals

### Overview

The Signals module offers a unified read/write interface to the controller's I/O, covering digital/analog inputs and outputs, multi-channel batch operations, and timed triggering. Via `signals` you can query current signal states, batch-write DO/RO/GO ports, or generate pulses at specified intervals, enabling coordination with external grippers, sensors, or production-line equipment.

### 4.5.1 Read the Value of a Specified Type and Port IO

Method Name	<code>signals.read( <a href="#">signal_type</a> : SignalType, <a href="#">index</a> : int) -&gt; tuple[float, StatusCodeEnum]</code>
Description	Read the value of an IO of a specified type and port (supports DI/DO/UI/UO/RI/RO/GI/GO/TAI/TDI/TDO/AI/AO).
Request Parameters	<code><a href="#">signal_type</a></code> : <a href="#">SignalType</a> The type of IO to read. <code><a href="#">index</a></code> : int The index of the IO to read, starting from 1.
Return Value	float: IO value. DI/DO/RI/RO/TAI/TDI/TDO/AI/AO return 0 or 1, and GI/GO return integer values (negative for Off). <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	UI/UO can only be read, not written.

### 4.5.2 Write the Value of a Specified Type and Port IO

Method Name	<b>signals.write</b> ( <code>signal_type</code> : <code>SignalType</code> , <code>index</code> : <code>int</code> , <code>value</code> : <code>float</code> ) -> <code>StatusCodeEnum</code>
Description	Write the value of an IO of a specified type and port. Currently only DO/RO/GO/TDO/AO are supported.
Request Parameters	<code>signal_type</code> : <code>SignalType</code> The type of IO to write. <code>index</code> : <code>int</code> The index of the IO to write, starting from 1. <code>value</code> : <code>float</code> DO/RO/TDO accept 0 or 1, GO expects an integer value, and AO accepts floating-point analog values.
Return Value	<code>StatusCodeEnum</code> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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 writing
"""
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.errmsg}")
    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 Batch-read DO (Digital Output) Port Values

Method Name	<code>signals.multi_read( <a href="#">signal_type</a> : SignalType, <a href="#">io_list</a> : list) -&gt; tuple[list, StatusCodeEnum]</code>
Description	Batch-read DO (digital output) port values.
Request Parameters	<p><a href="#">signal_type</a> : <a href="#">SignalType</a> The IO type to read; currently only DO is supported.</p> <p><a href="#">io_list</a> : list of port numbers to read and must not be empty.</p>
Return Value	list: Controller response in the form [port1, state1, port2, state2, ...] <a href="#">StatusCodeEnum</a> : Result of the function execution.

Method Name	<b>signals.multi_read</b> ( <code>signal_type</code> : <code>SignalType</code> , <code>io_list</code> : list) -> tuple[list, <code>StatusCodeEnum</code> ]
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	UI/UO can only be read, not written.

## 4.5.4 Batch-write DO (Digital Output) Signals

Method Name	<b>signals.multi_write</b> ( <code>signal_type</code> : <code>SignalType</code> , <code>io_list</code> : list) -> <code>StatusCodeEnum</code>
Description	Batch-write DO (digital output) signals.
Request Parameters	<code>signal_type</code> : <a href="#">SignalType</a> The IO type to write; currently only DO is supported. <code>io_list</code> : list of port numbers and values in the form [port1, state1, port2, state2]; the list length must be even and greater than zero.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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 writing
"""
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.errmsg}")
    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 Trigger IO Channels with Time Intervals

Method Name	<code>signals.trigger_io_with_intervals( in_port : int = -1, intervals : list, out_ports : list, pulse_duration : int) -&gt; StatusCodeEnum</code>
Description	Trigger multiple output ports according to a set of time intervals, optionally waiting for a rising edge on a DI port before starting.
Request Parameters	<code>in_port</code> : int Input port number, default -1 (no input trigger). <code>intervals</code> : list Time intervals in milliseconds, counted after the input trigger. <code>out_ports</code> : list Output port numbers to toggle. <code>pulse_duration</code> : int Pulse duration in milliseconds.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

# 4.6 Register Information

## Overview

The Register module provides a unified gateway for the host computer to read from and write to controller registers, covering numeric registers (R), motion registers (MR), string registers (SR), pose registers (PR), and Modbus registers.

Through the `register` interface you can batch-read, write, or delete data in any of these register types, making it easy to pass parameters, synchronize states, or share configuration with external systems while the program is running.

## 4.6.1 R Numeric Register Operations

### 4.6.1.1 Read the Value of an R Register

Method Name	<code>register.read_R( <code>index</code> : int) -&gt; tuple[float, StatusCodeEnum]</code>
Description	Reads the value of an R numeric register.
Request Parameters	<code>index</code> : int R register number to read
Return Value	float: R register numeric value <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.1.2 Write the Value of an R Register

Method Name	<code>register.write_R( <code>index</code> : int, <code>value</code> : float) -&gt; StatusCodeEnum</code>
Description	Writes the value of an R numeric register.

Method Name	<b>register.write_R</b> ( <code>index</code> : int, <code>value</code> : float) -> StatusCodeEnum
Request Parameters	<code>index</code> : int R register number to write <code>value</code> : float R register numeric value to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.1.3 Delete an R Register

Method Name	<b>register.delete_R</b> ( <code>index</code> : int) -> StatusCodeEnum
Description	Deletes the specified R numeric register.
Request Parameters	<code>index</code> : int R register number to delete
Return Value	<a href="#">StatusCodeEnum</a> : Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 registers/R.py

py

```
#!/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, 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.errmsg}")
    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.errmsg}")
    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] 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: {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")
```

## 4.6.2 MR Motion Register Operations

### 4.6.2.1 Read the Value of an MR Register

Method Name	<b>register.read_MR</b> ( <code>index</code> : int) -> tuple[int, StatusCodeEnum]
Description	Reads the value of an MR motion register.
Request Parameters	<code>index</code> : int MR register number to read
Return Value	int: MR register numeric value <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.2.2 Write the Value of an MR Register

Method Name	<b>register.write_MR</b> ( <code>index</code> : int, <code>value</code> : int) -> StatusCodeEnum
Description	Writes the value of an MR motion register.
Request Parameters	<code>index</code> : int MR register number to write <code>value</code> : int MR register numeric value to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.2.3 Delete an MR Register

Method Name	<b>register.delete_MR</b> ( <a href="#">index</a> : int) -> StatusCodeEnum
Description	Deletes the specified MR motion register.
Request Parameters	<a href="#">index</a> : int MR register number to delete
Return Value	<a href="#">StatusCodeEnum</a> : Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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, 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.errmsg}")
    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: {ret.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: {ret.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] 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: {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")

```

## 4.6.3 SR String Register Operations

### 4.6.3.1 Read the Value of an SR Register

Method Name	<b>register.read_SR</b> ( <code>index</code> : int) -> tuple[str, StatusCodeEnum]
Description	Reads the value of an SR string register.
Request Parameters	<code>index</code> : int SR register number to read
Return Value	str: SR register string value <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.3.2 Write the Value of an SR Register

Method Name	<b>register.write_SR</b> ( <code>index</code> : int, <code>value</code> : str) -> StatusCodeEnum
Description	Writes the value of an SR string register.
Request Parameters	<code>index</code> : int SR register number to write <code>value</code> : str SR register string value to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.3.3 Delete an SR Register

Method Name	<b>register.delete_SR</b> ( <code>index</code> : int) -> StatusCodeEnum
Description	Deletes the specified SR string register.
Request Parameters	<code>index</code> : int SR register number to delete
Return Value	<a href="#">StatusCodeEnum</a> : Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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

# [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.errmsg}")
    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: {ret.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: {ret.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] 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.errno}")
    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 Pose Register Operations

### 4.6.4.1 Read the Value of a PR Register

Method Name	<code>register.read_PR(index : int) -&gt; tuple[PoseRegister, StatusCodeEnum]</code>
Description	Reads the value of a PR pose register.
Request Parameters	<code>index</code> : int PR register number to read
Return Value	<a href="#">PoseRegister</a> : PR register pose data <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.4.2 Write the Value of a PR Register

Method Name	<b>register.write_PR</b> ( <code>value</code> : PoseRegister) -> StatusCodeEnum
Description	Writes the value of a PR pose register.
Request Parameters	<code>value</code> : <a href="#">PoseRegister</a> PR register pose data to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

### 4.6.4.3 Delete a PR Register

Method Name	<b>register.delete_PR</b> ( <code>index</code> : int) -> StatusCodeEnum
Description	Deletes the specified PR pose register.
Request Parameters	<code>index</code> : int PR register number to delete
Return Value	<a href="#">StatusCodeEnum</a> : Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 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 register
"""
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.errmsg}")
    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: {ret.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: {ret.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] 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: {ret.errno}")
    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 Registers (MH Holding Registers, MI Input Registers)

### 4.6.5.1 Read the Value of an MH Register

Method Name	<code>register.read_MH( <a href="#">index</a> : int) -&gt; tuple[int, StatusCodeEnum]</code>
Description	Reads the value of an MH holding register.
Request Parameters	<a href="#">index</a> : int MH register number to read

Method Name	<b>register.read_MH</b> ( <code>index</code> : int) -> tuple[int, StatusCodeEnum]
Return Value	int: MH register numeric value <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

#### 4.6.5.2 Write the Value of an MH Register

Method Name	<b>register.write_MH</b> ( <code>index</code> : int, <code>value</code> : int) -> StatusCodeEnum
Description	Writes the value of an MH holding register.
Request Parameters	<code>index</code> : int MH register number to write <code>value</code> : int MH register numeric value to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

#### 4.6.5.3 Read the Value of an MI Register

Method Name	<b>register.read_MI</b> ( <code>index</code> : int) -> tuple[int, StatusCodeEnum]
Description	Reads the value of an MI input register.
Request Parameters	<code>index</code> : int MI register number to read
Return Value	int: MI register numeric value <a href="#">StatusCodeEnum</a> : Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

#### 4.6.5.4 Write the Value of an MI Register

Method Name	<b>register.write_MI</b> ( <code>index</code> : int, <code>value</code> : int) -> StatusCodeEnum
Description	Writes the value of an MI input register.
Request Parameters	<code>index</code> : int MI register number to write <code>value</code> : int MI register numeric value to write
Return Value	<a href="#">StatusCodeEnum</a> : Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

## Example Code

 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
"""
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.errmsg}")
    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: {ret.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: {ret.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: {ret.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: {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")
```

## 4.7 Trajectory Control

### Overview

The Trajectory module offers a full-cycle interface for offline trajectory execution, trajectory/path recording, conversion, and replay. It lets you designate a trajectory file, start/stop the robot along that trajectory, perform CSV-to-trajectory format conversion inside the controller, and supports host-side trajectory logging, status queries, and path-planner parameter configuration—making it easy to reproduce complex trajectories and create custom motions.

#### 4.7.1 Set the Offline Trajectory File to Execute

Method Name	<b>trajectory.set_offline_trajectory_file</b> ( <code>path</code> : str) -> StatusCodeEnum
Description	Sets the offline trajectory file to execute.
Request Parameters	<p><code>path</code> : str Name of the offline trajectory file (for example: <code>A.trajectory</code> ).</p> <p>File format is a plain text file:</p> <ul style="list-style-type: none"> <li>- Line 1: <code>6</code> → number of axes; <code>0.001</code> → time interval between points (s); <code>8093</code> → total number of trajectory points.</li> <li>- Line 2: Initial positions of the 6 axes.</li> <li>- Lines 3–8095: Trajectory points including position, velocity, acceleration, torque feedforward, <code>do_port</code> , and <code>do_state</code> for the 6 axes.</li> <li>- <code>do_port</code> : DO port used (1–24).</li> <li>- <code>do_port = -1</code> : no IO signal is triggered at this point.</li> <li>- <code>do_port = 1</code> and <code>do_state = 1</code> : <code>do1</code> is set ON at this point.</li> <li>- <code>do_port = 1</code> and <code>do_state = 0</code> : <code>do1</code> is set OFF at this point.</li> </ul>
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.2 Move to the Starting Point at a Safe Speed

Method Name	<b>trajectory.prepare_offline_trajectory()</b> -> StatusCodeEnum
Description	Moves the robot to the starting point of the offline trajectory at a safe speed.
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.3 Start Executing the Offline Trajectory

Method Name	<b>trajectory.execute_offline_trajectory()</b> -> StatusCodeEnum
Description	Starts executing the offline trajectory program.
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.4 Trajectory File Conversion

Method Name	<b>trajectory.transform_csv_to_trajectory()</b> ( <a href="#">file_name</a> : str, <a href="#">separator</a> : str = " ", <a href="#">io_flag</a> : str = "2")
Description	Converts a trajectory CSV file to the controller trajectory format and stores it in the controller's trajectory directory.
Request Parameters	<a href="#">file_name</a> : str CSV trajectory file name, without the <a href="#">.csv</a> suffix. <a href="#">separator</a> : " " (space) or "," (comma). When space, the converted file uses the

Method Name	<code>trajectory.transform_csv_to_trajectory( file_name : str, separator : str = " ", io_flag : str = "2")</code>
	<code>.trajectory</code> suffix; when comma, the converted file uses the <code>.csv</code> suffix. <code>io_flag</code> : If <code>1</code> , generated IO info uses defaults ( <code>do_port = -1</code> , <code>do_state = 0</code> ); if <code>2</code> , use user-specified IO info.
Return Value	str: Path of the converted trajectory file <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.7.5 Query Trajectory File Conversion Status

Method Name	<code>trajectory.check_transform_status( file_name : str) -&gt; tuple[TransformStatusEnum, StatusCodeEnum]</code>
Description	Queries the current status of the trajectory file conversion.
Request Parameters	<code>file_name</code> : str Path returned by <code>transform_csv_to_trajectory</code> .
Return Value	<a href="#">TransformStatusEnum</a> : Conversion status <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

trajectory/offline\_trajectory.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
```

py

```

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.errmsg}")
    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, error co
    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 successful")
else:
    print(f"准备离线轨迹运行失败, 错误代码 / Prepare offline trajectory execution failed,
    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 code: {ret}")
        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}")
        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}")
    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 Start Recording Trajectory

Method Name	<b>trajectory.trajectory_record_begin</b> ( <code>name</code> : str) -> StatusCodeEnum
Description	Instructs the robot to start recording a trajectory.
Parameters	<code>name</code> : trajectory program name.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.7 Stop Recording Trajectory

Method Name	<b>trajectory.trajectory_record_finish</b> ( <code>name</code> : str) -> StatusCodeEnum
Description	Instructs the robot to stop recording a trajectory.
Parameters	<code>name</code> : trajectory program name.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.8 Start Replaying Trajectory

Method Name	<b>trajectory.trajectory_replay_start</b> ( <code>name</code> : str) -> StatusCodeEnum
Description	Instructs the robot to start replaying a trajectory.
Parameters	<code>name</code> : trajectory program name.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+

Method Name	<b>trajectory.trajectory_replay_start</b> ( <a href="#">name</a> : str) -> <b>StatusCodeEnum</b>
version	Industrial (Bronze): v7.5.0.0+

## 4.7.9 Stop Replaying Trajectory

Method Name	<b>trajectory.trajectory_replay_stop</b> ( <a href="#">name</a> : str) -> <b>StatusCodeEnum</b>
Description	Instructs the robot to stop replaying a trajectory.
Parameters	<a href="#">name</a> : trajectory program name.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.10 Delete Trajectory

Method Name	<b>trajectory.trajectory_record_delete</b> ( <a href="#">name</a> : str) -> <b>StatusCodeEnum</b>
Description	Deletes the specified trajectory stored in the robot.
Parameters	<a href="#">name</a> : trajectory program name.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.7.11 Get List of Recorded Trajectories

Method Name	<b>trajectory.get_trajectory_record_list()</b> -> tuple[list[str], StatusCodeEnum]
Description	Retrieves the list of recorded trajectories available for replay.
Parameters	None
Return	[list[str]]: list of trajectory program names. <a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.7.12 Get Starting Pose of a Recorded Trajectory

Method Name	<b>trajectory.get_trajectory_record_start_pose( <code>name</code> : str)</b> -> tuple[MotionPose, StatusCodeEnum]
Description	Retrieves the starting pose of a recorded trajectory.
Parameters	<code>name</code> : trajectory program name.
Return	<a href="#">MotionPose</a> : Starting pose of the trajectory. <a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

🐍 trajectory/trajectory\_record.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
```

py

```

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.errmsg}")
    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 code: {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, error 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, error 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.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 code: {ret}")
    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}")
    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 code: {ret}")
    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 code: {ret}")
    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 code: {ret}")
    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 Start Recording Trajectory / Path

Method Name	<code>trajectory.path_record_begin( name : str, comment : str, param : float, angle : float = 1) -&gt; StatusCodeEnum</code>
Description	Starts recording a trajectory table (.traj) or a path table (.path).
Parameters	<code>name</code> : trajectory / path file name, must end with .traj or .path. <code>comment</code> : description for the trajectory / path. <code>param</code> : - for .path tables: linear-motion distance threshold (mm). - for .traj tables: recording interval (ms). <code>angle</code> : rotational-angle threshold (°), only effective when <code>name</code> is a .path file.
Return	<a href="#">StatusCodeEnum</a> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.14 Stop Recording Trajectory / Path

Method Name	<b>trajectory.path_record_finish()</b> -> StatusCodeEnum
Description	Stops the current trajectory / path table recording.
Parameters	none
Return	<a href="#">StatusCodeEnum</a> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.15 Get Starting Pose of a Trajectory / Path

Method Name	<b>trajectory.get_path_start_pose( <a href="#">name</a> : str)</b> -> tuple[MotionPose, StatusCodeEnum]
Description	Retrieves the starting pose of the specified trajectory / path file.
Parameters	<a href="#">name</a> : trajectory / path file name.
Return	<a href="#">MotionPose</a> : Starting pose. <a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.16 Get Status of Trajectories / Paths

Method Name	<b>trajectory.get_path_state( <a href="#">path_list</a> : list[str])</b> -> tuple[dict[str, int], StatusCodeEnum]
Description	Batch-query the current status of trajectory / path files.
Parameters	<a href="#">path_list</a> : list of trajectory / path file names to query.
Return	<a href="#">dict[str, int]</a> : mapping of file name → status code. <a href="#">StatusCodeEnum</a> : execution result of the function.

Method Name	<b>trajectory.get_path_state</b> ( <code>path_list</code> : list[str]) -> tuple[dict[str, int], StatusCodeEnum]
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.17 Set Path-Planner Parameters

Method Name	<b>trajectory.set_path_planner_parameter</b> ( <code>transition_time</code> : float, <code>scaling_factor</code> : float) -> StatusCodeEnum
Description	Configures path-planner parameters affecting motion smoothness and speed / acceleration distribution.
Parameters	<p><code>transition_time</code> : blending time, 0–1. Larger values yield smoother acceleration / deceleration.</p> <p><code>scaling_factor</code> : redistribution weight of path parameter <math>s</math>, 0–1:</p> <ul style="list-style-type: none"> <li>- 0: uniform time scaling (equal-length segments).</li> <li>- 1: linear scaling by maximum displacement (equal-velocity segments).</li> <li>- 0.5: equal-acceleration scaling.</li> <li>- 0.33: equal-jerk scaling (balanced).</li> </ul>
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.18 Get Path-Planner Parameters

Method Name	<b>trajectory.get_path_planner_parameter</b> () -> tuple[float, float, StatusCodeEnum]
Description	Reads the current path-planner parameters.
Parameters	none

Method Name	<b>trajectory.get_path_planner_parameter()</b> -> tuple[float, float, StatusCodeEnum]
Return	<code>transition_time</code> : blending time, 0–1. <code>scaling_factor</code> : redistribution weight, 0–1. <a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

### 4.7.19 Move Along a Trajectory / Path

Method Name	<b>trajectory.move_path()</b> ( <code>name</code> : str, <code>vel</code> : float = 100, <code>acc</code> : float = 1) -> StatusCodeEnum
Description	Commands the robot end-effector to move along the specified trajectory / path file.
Parameters	<code>name</code> : trajectory / path file name. <code>vel</code> : end-effector speed, 0–5000 mm/s. <code>acc</code> : acceleration multiplier, 0–1.2.
Return	<a href="#">StatusCodeEnum</a> : Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

#### Example Code

🐍 trajectory/path\_record.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
```

py

```

# [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.errmsg}")
    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, recording mode
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: {ret.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.errmsg}")
    arm.disconnect()
    exit(1)
if state["test_path.path"] == 0:
    print(f"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.errmsg}")
    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.errmsg}")
    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: {ret.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.errmsg}")
    arm.disconnect()
    exit(1)
if state["test_path.path"] == 1:
    print(f"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 code: {ret}")
    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.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()
    # [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 code: {ret}")
        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 ratio)
ret = arm.trajectory.set_path_planner_parameter(0.5, 0.333333)
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, error 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 successful
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, error 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, speed 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.errmsg}")
    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 Alarm Information

## Overview

The Alarm module provides functions to read, reset, and query robot alarm information, allowing you to monitor and handle abnormalities that may occur during operation.

Via the `alarm` interface you can check whether any alarm is currently active, retrieve the full list of alarms, and reset them—helping developers detect and resolve robot faults in real time.

### 4.8.1 Reset Alarms

Method Name	<code>alarm.reset()</code> -> <code>StatusCodeEnum</code>
Description	Resets current errors/alarms.
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.8.2 Get All Active Alarms

Method Name	<code>alarm.get_all_active_alarms( <code>language</code> : <code>LanguageType</code>) -&gt; tuple[list, <code>StatusCodeEnum</code>]</code>
Description	Gets all currently active alarms.
Request Parameters	<code>language</code> : <code>LanguageType</code> Output language. Options: <code>LanguageType.English</code> , <code>LanguageType.Chinese</code> .

Method Name	<b>alarm.get_all_active_alarms</b> ( <code>language</code> : LanguageType) -> tuple[list, StatusCodeEnum]
Return Value	list: Active alarm entries. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.8.3 Get the Highest-Priority Alarm

Method Name	<b>alarm.get_top_alarm</b> () -> tuple[ROBOT_ALARM, StatusCodeEnum]
Description	Gets the alarm with the highest priority.
Request Parameters	None
Return Value	<code>ROBOT_ALARM</code> : Highest-priority alarm. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### Example Code

 alarm.py

py

```
#!/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.errmsg}")
    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 code: {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 code: {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.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")
```

## 4.9 File Manager

### Overview

FileManager is used to upload, download, delete, or search resources such as robot programs, trajectories, and temporary files between the host computer and the robot controller. Through the `file_manager` interface you can batch-manage files of types like `USER_PROGRAM`, `BLOCK_PROGRAM`, `TRAJECTORY`, and `ROBOT_TMP`, with support for overwrite control and pattern-matching queries—making it easy to deploy/back up programs or synchronize debug data from the production line.

### 4.9.1 Upload a Local File to the Robot

Method Name	<code>file_manager.upload( file_path : str, file_type : str, overwriting : bool = False) -&gt; StatusCodeEnum</code>
Description	Uploads a local file to the robot controller.
Request Parameters	<p><code>file_path</code> : str Absolute path to the local file.</p> <p><code>file_type</code> : str File type:</p> <ul style="list-style-type: none"> <li><code>USER_PROGRAM</code> : <code>file_path</code> should point to the program base name (e.g., <code>/root/robot_data/test_prog</code>); uploads <code>test_prog.json</code> and <code>test_prog.xml</code> from the same directory.</li> <li><code>BLOCK_PROGRAM</code> : <code>file_path</code> should point to the block program base name (e.g., <code>/root/robot_data/test_block_prog</code>); uploads <code>test_block_prog.block</code>, <code>test_block_prog.json</code>, and <code>test_block_prog.xml</code>.</li> <li><code>TRAJECTORY</code> : <code>file_path</code> should be the full path to a <code>.trajectory</code> file (e.g., <code>/root/robot_data/test_torque.trajectory</code>).</li> <li><code>ROBOT_TMP</code> : <code>file_path</code> should be the full path to the temporary file (e.g., <code>/root/robot_data/test.csv</code>).</li> </ul> <p><code>overwriting</code> : bool Whether to overwrite an existing file on the robot. Default: <code>False</code>.</p>
Return Value	<a href="#"><code>StatusCodeEnum</code></a> : Result of the function execution.

Method Name	<b>file_manager.upload</b> ( <code>file_path</code> : str, <code>file_type</code> : str, <code>overwriting</code> : bool = False) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	For <code>USER_PROGRAM</code> and <code>BLOCK_PROGRAM</code> , specify only the base name (no suffix).

## 4.9.2 Download a Robot File to Local

Method Name	<b>file_manager.download</b> ( <code>file_name</code> : str, <code>file_path</code> : str, <code>file_type</code> : str, <code>overwriting</code> : bool=False) -> StatusCodeEnum
Description	Downloads a file from the robot to a local directory.
Request Parameters	<p><code>file_name</code> : str File name. Requirements by type:</p> <ul style="list-style-type: none"> <li><code>USER_PROGRAM</code> : provide the program base name (no suffix); downloads both <code>.json</code> and <code>.xml</code> .</li> <li><code>TRAJECTORY</code> : provide the full file name including the <code>.trajectory</code> suffix.</li> <li><code>ROBOT_TMP</code> : provide the full file name including suffix (e.g., <code>.csv</code> ).</li> </ul> <p><code>file_path</code> : str Local save directory.</p> <p><code>file_type</code> : str File type ( <code>BLOCK_PROGRAM</code> is not supported for download).</p> <p><code>overwriting</code> : bool Whether to overwrite an existing file with the same name in the target path.</p>
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.9.3 Delete a File on the Robot

Method Name	<b>file_manager.delete</b> ( <code>file_name</code> : str, <code>file_type</code> : str) -> StatusCodeEnum
Description	Deletes a file from the robot controller.

Method Name	<b>file_manager.delete</b> ( <code>file_name</code> : str, <code>file_type</code> : str) -> <b>StatusCodeEnum</b>
Request Parameters	<p><code>file_name</code> : str File name to delete. Requirements by type:</p> <ul style="list-style-type: none"> <li><code>USER_PROGRAM</code> / <code>BLOCK_PROGRAM</code> : provide the program base name (no suffix).</li> <li><code>TRAJECTORY</code> / <code>ROBOT_TMP</code> : provide the full file name including suffix.</li> </ul> <p><code>file_type</code> : str File type.</p>
Return Value	<b>StatusCodeEnum</b> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.9.4 Search for Files Matching a Pattern

Method Name	<b>file_manager.search</b> ( <code>pattern</code> : str, <code>file_list</code> : list) -> <b>StatusCodeEnum</b>
Description	Searches for files on the controller that match the specified pattern.
Request Parameters	<p><code>pattern</code> : str Filename match pattern.</p> <p><code>file_list</code> : list Output list to receive matching file names.</p>
Return Value	<b>StatusCodeEnum</b> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 file\_manager/file\_manager.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 download
"""
```

py

```

from pathlib import Path

from Agilebot import ROBOT_TMP, TRAJECTORY, TRAJECTORY_CSV, USER_PROGRAM, Arm, FileManag

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.errmsg}")
    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.errmsg}")
    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.errmsg}")
    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.errmsg}")
    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.errmsg}")
    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.errmsg}")
    exit(1)
```

## 4.10 Coordinate Systems

### Overview

The `CoordinateSystem` module manages the robot's User Frames (UF) and Tool Frames (TF), offering a unified API to add, delete, update, query, and calculate frames.

Via `coordinate_system.UF/TF` you can maintain the controller's frame list or quickly solve a new frame from taught points, ensuring a consistent spatial reference when switching tools or debugging multi-station setups.

#### 4.10.1 Get User/Tool Coordinate Summary List

Method Name	<code>coordinate_system.UF/TF.get_coordinate_list()</code> -> <code>tuple[List[Coordinate], StatusCodeEnum]</code>
Description	Get the summary list of user/tool coordinate systems.
Request Parameters	None
Return Value	UserCoordSummaryList: A list of coordinate system summary information. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### 4.10.2 Add a User/Tool Coordinate System

Method Name	<code>coordinate_system.UF/TF.add( <code>coordinate</code> : Coordinate)</code> -> <code>StatusCodeEnum</code>
Description	Add a coordinate system to the current user/tool coordinate set.

Method Name	<b>coordinate_system.UF/TF.add( <a href="#">coordinate</a> : Coordinate) -&gt; StatusCodeEnum</b>
Request Parameters	<a href="#">coordinate</a> : <a href="#">Coordinate</a> The coordinate system information to add.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.10.3 Delete a User/Tool Coordinate System from the Current System

Method Name	<b>coordinate_system.UF/TF.delete( <a href="#">index</a> : int) -&gt; StatusCodeEnum</b>
Description	Delete a coordinate system from the current user/tool coordinate set.
Request Parameters	<a href="#">index</a> : int The coordinate system index.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

### 4.10.4 Update a User/Tool Coordinate System in the Current System

Method Name	<b>coordinate_system.UF/TF.update( <a href="#">coordinate</a> : Coordinate) -&gt; StatusCodeEnum</b>
Description	Update a coordinate system in the current user/tool coordinate set.
Request Parameters	<a href="#">coordinate</a> : <a href="#">Coordinate</a> The updated coordinate system information.
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution.

Method Name	<b>coordinate_system.UF/TF.update(</b> <code>coordinate</code> : Coordinate) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.10.5 Get a User/Tool Coordinate System from the Current System

Method Name	<b>coordinate_system.UF/TF.get(</b> <code>index</code> : int) -> tuple[Coordinate, StatusCodeEnum]
Description	Get a specified coordinate system from the current user/tool coordinate set.
Request Parameters	<code>index</code> : int The coordinate system index.
Return Value	<a href="#">Coordinate</a> : Coordinate system information. <a href="#">StatusCodeEnum</a> : Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.10.6 Calculate User/Tool Coordinate Information

Method Name	<b>coordinate_system.UF/TF.calculate(pose: List[<a href="#">Position</a>])</b> -> tuple[ <a href="#">Position</a> , StatusCodeEnum]
Description	Calculate user/tool coordinate information based on the input poses and return the calculated pose.
Request Parameters	<code>pose</code> : List[ <a href="#">Position</a> ] List of input poses.
Return Value	<a href="#">Position</a> : Calculated pose information. <a href="#">StatusCodeEnum</a> : Result of the function execution.

Method Name	<code>coordinate_system.UF/TF.calculate(pose: List[<a href="#">Position</a>]) -&gt; tuple[<a href="#">Position</a>, StatusCodeEnum]</code>
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

coordinate\_system.py

```
#!/python
"""
    Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
    Instruction: 坐标系统使用示例 / 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.errmsg}")
    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, -0.001),
    Position(365.4597874970455, 77.95089759481547, 441.39040857936857, -7.343, 12.620, -0.001),
    Position(410.64702354574865, 10.394172666192766, 468.26089261578807, 18.719, 29.15, -0.001),
    Position(483.2519847999948, 112.71925218513972, 448.39071038067624, 33.947, 69.714, -0.001)
]
```

```

# [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 failed, error code: {ret}")
    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}")

# [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, error code: {ret}")
    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 failed, error code: {ret}")
    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, error code:
    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 co
    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 co
    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 Modbus-Related Functions

### Overview

The Modbus module encapsulates the robot's capabilities when acting as a Modbus master: slave management, register read/write, and serial-port parameter configuration.

Through the `modbus` interface you can obtain a Slave object by channel and slave ID, then perform batch reads/writes on coils, holding/input registers, and set/query serial-port communication parameters—making it easy to integrate with PLCs, I/O expansion modules, and other Modbus devices.

**Note:** Modbus-related functions conflict with the bus configuration on the teaching pendant and cannot be used simultaneously.

### 4.11.1 Get a Modbus Slave Instance for a Specified Channel and Slave ID

Method Name	<code>modbus.get_slave(channel : ModbusChannel, slave_id : int, master_id : int = 0) -&gt; 'Modbus.Slave'</code>
Description	Get a Modbus slave instance for a specified channel and slave ID.
Request Parameters	<code>channel</code> : <a href="#">ModbusChannel</a> Modbus channel <code>slave_id</code> : Slave ID <code>master_id</code> : Master ID
Return Value	Modbus.Slave: Modbus slave instance
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

### 4.11.2 Read Modbus Coil Registers from a Slave

Method Name	<b>slave.read_coils</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus coil registers from a slave.
Request Parameters	<code>address</code> : int Register address <code>number</code> : int Number of registers (max 120)
Return Value	list[int]: Register values <a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

### 4.11.3 Write to Modbus Coil Registers of a Slave

Method Name	<b>slave.write_coils</b> ( <code>address</code> : int, <code>value</code> : list[int]) -> StatusCodeEnum
Description	Write to Modbus coil registers of a slave.
Request Parameters	<code>address</code> : int Register address <code>value</code> : list[int] Register values
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

### 4.11.4 Read Modbus Holding Registers from a Slave

Method Name	<b>slave.read_holding_regs</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus holding registers from a slave.
Request Parameters	<code>address</code> : int Register address <code>number</code> : int Number of registers (max 120)

Method Name	<b>slave.read_holding_regs</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Return Value	list[int]: Register values <a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

### 4.11.5 Write to Modbus Holding Registers of a Slave

Method Name	<b>slave.write_holding_regs</b> ( <code>address</code> : int, <code>value</code> : list[int]) -> StatusCodeEnum
Description	Write to Modbus holding registers of a slave.
Request Parameters	<code>address</code> : int Register address <code>value</code> : list[int] Register values
Return Value	<a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

### 4.11.6 Read Modbus Discrete Input Registers from a Slave

Method Name	<b>slave.read_discrete_inputs</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus discrete input registers from a slave.
Request Parameters	<code>address</code> : int Register address <code>number</code> : int Number of registers (max 120)
Return Value	list[int]: Register values <a href="#">StatusCodeEnum</a> : Result of the function execution

Method Name	<b>slave.read_discrete_inputs</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

## 4.11.7 Read Modbus Input Registers from a Slave

Method Name	<b>slave.read_input_regs</b> ( <code>address</code> : int, <code>number</code> : int ) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus input registers from a slave.
Request Parameters	<code>address</code> : int Register address <code>number</code> : int Number of registers (max 120)
Return Value	list[int]: Register values <a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

## Example Code

 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 code: {ret.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.errmsg}")
    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 code: {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 code: {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 code: {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 Set Serial Communication Parameters

Method Name	<code>modbus.set_parameter( param : <a href="#">SerialParams</a>) -&gt; tuple[int, StatusCodeEnum]</code>
Description	Set serial communication parameters.
Request Parameters	<code>param</code> : <a href="#">SerialParams</a> Serial communication parameters
Return Value	int: Channel ID <a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

## 4.11.9 Get Serial Communication Parameters

Method Name	<code>modbus.get_parameter( channel : ModbusChannel, master_id : int = 1) -&gt; tuple[<a href="#">SerialParams</a>, StatusCodeEnum]</code>
Description	Get serial communication parameters.
Request Parameters	<code>channel</code> : <a href="#">ModbusChannel</a> Modbus channel <code>master_id</code> : int Master ID
Return Value	<a href="#">SerialParams</a> : Serial communication parameters <a href="#">StatusCodeEnum</a> : Result of the function execution
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

## 4.12 BasScript Program Class

### Overview

BasScript lets you construct BAS instruction sequences for the teach pendant in a structured way from the host PC, covering motion, logic, program calls, communication, vision, and Modbus commands.

By chaining `BasScript.*` method calls in Python you generate a script flow identical to that on the teach pendant, making it easy to automatically create, edit, or reuse complex programs.

### Constructor

Method Name	<b>BasScript</b> ( <code>name</code> )
Description	Corresponds to the instructions in the teaching pendant program editor
Request Parameters	<code>name</code> : Script name
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): v7.6.0.0+
Notes	All methods under BasScript follow the same compatible robot software versions as this class

### 4.12.1 Motion to Point Instruction

Method Name	<b>BasScript.move_joint</b> ( <code>pose_type</code> , <code>pose_index</code> , <code>speed_type</code> , <code>speed_value</code> , <code>smooth_type</code> , <code>smooth_distance</code> , <code>extra_param</code> )
Description	Corresponds to the MoveJoint instruction in the teaching pendant program writing

Method Name	<b>BasScript.move_joint</b> ( <code>pose_type</code> , <code>pose_index</code> , <code>speed_type</code> , <code>speed_value</code> , <code>smooth_type</code> , <code>smooth_distance</code> , <code>extra_param</code> )
Request Parameters	<code>pose_type</code> : Pose type <code>pose_index</code> : Pose index <code>speed_type</code> : Speed type <code>speed_value</code> : Speed value <code>smooth_type</code> : Smooth type <code>smooth_distance</code> : Smooth distance <code>extra_param</code> : Extra parameter
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.2 Linear Motion to Point Instruction

Method Name	<b>BasScript.move_line</b> ( <code>pose_type</code> , <code>pose_index</code> , <code>speed_type</code> , <code>speed_value</code> , <code>smooth_type</code> , <code>smooth_distance</code> , <code>extra_param</code> )
Description	Corresponds to the MoveLine instruction in the teaching pendant program writing
Request Parameters	<code>pose_type</code> : Pose type <code>pose_index</code> : Pose index <code>speed_type</code> : Speed type <code>speed_value</code> : Speed value <code>smooth_type</code> : Smooth type <code>smooth_distance</code> : Smooth distance <code>extra_param</code> : Extra parameter
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.3 Arc Motion to Point Instruction

Method Name	<b>BasScript.move_circle</b> ( <code>pose_type1</code> , <code>pose_index1</code> , <code>pose_type2</code> , <code>pose_index2</code> , <code>speed_type</code> , <code>speed_value</code> , <code>smooth_type</code> , <code>smooth_distance</code> , <code>extra_param</code> )
Description	Corresponds to the MoveCircle instruction in the teaching pendant program writing

Method Name	<b>BasScript.move_circle</b> ( <code>pose_type1</code> , <code>pose_index1</code> , <code>pose_type2</code> , <code>pose_index2</code> , <code>speed_type</code> , <code>speed_value</code> , <code>smooth_type</code> , <code>smooth_distance</code> , <code>extra_param</code> )
Request Parameters	<code>pose_type1</code> : First pose type <code>pose_index1</code> : First pose index <code>pose_type2</code> : Second pose type <code>pose_index2</code> : Second pose index <code>speed_type</code> : Speed type <code>speed_value</code> : Speed value <code>smooth_type</code> : Smooth type <code>smooth_distance</code> : Smooth distance <code>extra_param</code> : Extra parameter
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.4 If Conditional Instruction

Method Name	<b>BasScript.logi_if</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )
Description	Corresponds to the IF logical statement in the teaching pendant program writing
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>operator</code> : Logical operator
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.5 Elself Conditional Branch Instruction

Method Name	<b>BasScript.logi_else_if</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )
Description	Corresponds to the ELIF logical statement in the teaching pendant program writing

Method Name	<b>BasScript.logi_else_if( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )</b>
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>operator</code> : Logical operator
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.6 Else Instruction

Method Name	<b>BasScript.logi_else()</b>
Description	Corresponds to the ELSE logical statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.7 End Conditional Instruction

Method Name	<b>BasScript.logi_end_if()</b>
Description	Corresponds to the ENDIF logical statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.8 While Loop Instruction

Method Name	<b>BasScript.logi_while( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )</b>
Description	Corresponds to the WHILE logical statement in the teaching pendant program writing
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>operator</code> : Logical operator
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.9 End Loop Instruction

Method Name	<b>BasScript.logi_end_while()</b>
Description	Corresponds to the ENDWHILE logical statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.10 Switch Multi-branch Selection Instruction

Method Name	<b>BasScript.logi_switch( <code>param</code> , <code>index</code> )</b>
Description	Corresponds to the SWITCH logical statement in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>index</code> : Index
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.11 Case Branch Instruction

Method Name	<b>BasScript.logi_case( <code>param</code> , <code>value</code> )</b>
Description	Corresponds to the CASE logical statement in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>value</code> : Value
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.12 Default Branch Instruction

Method Name	<b>BasScript.logi_default()</b>
Description	Corresponds to the DEFAULT logical statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.13 End Multi-branch Selection Instruction

Method Name	<b>BasScript.logi_end_switch()</b>
Description	Corresponds to the ENDSWITCH logical statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.14 Goto Jump Instruction

Method Name	<b>BasScript.logi_goto( <code>index</code> )</b>
Description	Corresponds to the GOTO jump statement in the teaching pendant program writing
Request Parameters	<code>index</code> : Index of the target label
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.15 Label Instruction

Method Name	<b>BasScript.logi_label( <code>index</code> )</b>
Description	Corresponds to the LABEL label statement in the teaching pendant program writing
Request Parameters	<code>index</code> : Label index
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.16 Skip Condition Instruction

Method Name	<b>BasScript.logi_skip_condition( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )</b>
Description	Corresponds to the SKIP CONDITION jump statement in the teaching pendant program writing
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>operator</code> : Logical operator

Method Name	<b>BasScript.logi_skip_condition</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.17 Break Loop Instruction

Method Name	<b>BasScript.logi_break</b> ()
Description	Corresponds to the BREAK statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.18 Skip Loop Instruction

Method Name	<b>BasScript.logi_continue</b> ()
Description	Corresponds to the CONTINUE statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.19 Assignment Instruction

Method Name	<b>BasScript.assign_value</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>opt_index</code> , <code>opt_value</code> )
Description	Corresponds to the ASSIGN assignment statement in the teaching pendant program writing

Method Name	<b>BasScript.assign_value</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>opt_index</code> , <code>opt_value</code> )
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>opt_index</code> : Optional index <code>opt_value</code> : Optional value
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.20 Wait Condition Instruction

Method Name	<b>BasScript.wait</b> ( <code>param1</code> , <code>index</code> , <code>param2</code> , <code>value</code> , <code>operator</code> )
Description	Corresponds to the WAIT COND wait condition statement in the teaching pendant program writing
Request Parameters	<code>param1</code> : First parameter <code>index</code> : Index <code>param2</code> : Second parameter <code>value</code> : Value <code>operator</code> : Logical operator
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.21 Wait Time Instruction

Method Name	<b>BasScript.wait_time</b> ( <code>param</code> , <code>value</code> )
Description	Corresponds to the WAIT TIME wait time statement in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>value</code> : Time value

Method Name	<b>BasScript.wait_time</b> ( <code>param</code> , <code>value</code> )
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.22 Pause Instruction

Method Name	<b>BasScript.pause</b> ()
Description	Corresponds to the PAUSE pause statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.23 Abort Instruction

Method Name	<b>BasScript.abort</b> ()
Description	Corresponds to the ABORT terminate statement in the teaching pendant program writing
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.24 Call Synchronous Program Call Instruction

Method Name	<b>BasScript.call</b> ( <code>name</code> )
Description	Corresponds to the CALL synchronous program call in the teaching pendant program writing

Method Name	<b>BasScript.call( <code>name</code> )</b>
Request Parameters	<code>name</code> : Program name
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.25 Run Asynchronous Program Call Instruction

Method Name	<b>BasScript.run( <code>name</code> )</b>
Description	Corresponds to the RUN asynchronous program call in the teaching pendant program writing
Request Parameters	<code>name</code> : Program name
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.26 Load Program Instruction

Method Name	<b>BasScript.load( <code>param</code> , <code>value</code> )</b>
Description	Corresponds to the LOAD load program in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>value</code> : Value
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.27 Unload Program Instruction

Method Name	<b>BasScript.unload( <code>param</code> , <code>value</code> )</b>
Description	Corresponds to the UNLOAD unload program in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>value</code> : Value
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.28 Execute Program Instruction

Method Name	<b>BasScript.exec( <code>param</code> , <code>value</code> )</b>
Description	Corresponds to the EXEC execute program in the teaching pendant program writing
Request Parameters	<code>param</code> : Parameter <code>value</code> : Value
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.29 Open Socket Connection Instruction

Method Name	<b>BasScript.socket_open( <code>index</code> )</b>
Description	Corresponds to the SOCKET OPEN open socket connection in the teaching pendant program writing
Request Parameters	<code>index</code> : SK register sequence number
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.30 Close Socket Connection Instruction

Method Name	<b>BasScript.socket_close( <a href="#">index</a> )</b>
Description	Corresponds to the SOCKET CLOSE close socket connection in the teaching pendant program writing
Request Parameters	<a href="#">index</a> : SK register sequence number
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.31 Connect Socket Instruction

Method Name	<b>BasScript.socket_connect( <a href="#">index</a> )</b>
Description	Corresponds to the SOCKET CONNECT connect socket in the teaching pendant program writing
Request Parameters	<a href="#">index</a> : SK register sequence number
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.32 Send Socket Data Instruction

Method Name	<b>BasScript.socket_send( <a href="#">index</a> , <a href="#">msg_type</a> , <a href="#">value</a> )</b>
Description	Corresponds to the SOCKET SEND send data in the teaching pendant program writing
Request Parameters	<a href="#">index</a> : SK register sequence number <a href="#">msg_type</a> : Message type <a href="#">value</a> : Message content or sequence number
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.33 Receive Socket Data Instruction

Method Name	<b>BasScript.socket_recv</b> ( <code>index</code> , <code>msg_length</code> , <code>msg_type</code> , <code>value</code> )
Description	Corresponds to the SOCKET_RECV receive data in the teaching pendant program writing
Request Parameters	<code>index</code> : SK register sequence number <code>msg_length</code> : Message length <code>msg_type</code> : Message type <code>value</code> : Message content or sequence number
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.34 Read Modbus Holding Register Instruction

Method Name	<b>BasScript.modbus_read_MH</b> ( <code>index</code> , <code>id</code> , <code>address</code> , <code>length</code> , <code>r_index</code> )
Description	Corresponds to the READ_MH read Modbus holding register in the teaching pendant program writing
Request Parameters	<code>index</code> : Channel sequence number <code>id</code> : Modbus ID <code>address</code> : Register address <code>length</code> : Register length <code>r_index</code> : R register sequence number
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.12.35 Read Modbus Input Register Instruction

Method Name	<b>BasScript.modbus_read_MI</b> ( <code>index</code> , <code>id</code> , <code>address</code> , <code>length</code> , <code>r_index</code> )
Description	Corresponds to the READ_MI read Modbus input register in the teaching pendant program writing

Method Name	<b>BasScript.modbus_read_MI</b> ( <code>index</code> , <code>id</code> , <code>address</code> , <code>length</code> , <code>r_index</code> )
Request Parameters	<code>index</code> : Channel sequence number <code>id</code> : Modbus ID <code>address</code> : Register address <code>length</code> : Register length <code>r_index</code> : R register sequence number
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.36 Write Modbus Holding Register Instruction

Method Name	<b>BasScript.modbus_write_MH</b> ( <code>index</code> , <code>id</code> , <code>address</code> , <code>length</code> , <code>value_type</code> , <code>value</code> )
Description	Corresponds to the WRITE_MH write Modbus holding register in the teaching pendant program writing
Request Parameters	<code>index</code> : Channel sequence number <code>id</code> : Modbus ID <code>address</code> : Register address <code>length</code> : Register length <code>value_type</code> : Value type <code>value</code> : Value or index
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.37 Find Vision Program Instruction

Method Name	<b>BasScript.vision_find</b> ( <code>name</code> )
Description	Corresponds to the VISION FIND find vision program in the teaching pendant program writing
Request Parameters	<code>name</code> : Vision program name

Method Name	<b>BasScript.vision_find( <code>name</code> )</b>
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.38 Get Vision Program Offset Instruction

Method Name	<b>BasScript.vision_get_offset( <code>name</code> , <code>index</code> , <code>label_index</code> )</b>
Description	Corresponds to the VISION GET OFFSET get vision program offset in the teaching pendant program writing
Request Parameters	<code>name</code> : Vision program name <code>index</code> : Vision register index <code>label_index</code> : Label index
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.39 Get Vision Program Result Instruction

Method Name	<b>BasScript.vision_get_quantity( <code>name</code> , <code>index</code> )</b>
Description	Corresponds to the VISION GET QUANTITY get vision program result in the teaching pendant program writing
Request Parameters	<code>name</code> : Vision program name <code>index</code> : R register index
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.40 Set Parameter Instruction

Method Name	<b>BasScript.set_param( <code>type</code> , <code>value_type</code> , <code>value</code> )</b>
Description	Corresponds to the SET PARAM set parameter in the teaching pendant program writing
Request Parameters	<code>type</code> : Parameter type <code>value_type</code> : Value type <code>value</code> : Value
Return Value	<a href="#"><u>StatusCodeEnum</u></a> : Function execution result

## 4.12.41 Extra Parameter Class

Method Name	<b>ExtraParam()</b>
Description	Extra parameter class, used to construct complex robot control commands
Request Parameters	None
Return Value	None

### 4.12.41.1 Set Acceleration

Method Name	<b>ExtraParam.acceleration( <code>value</code> )</b>
Description	Set acceleration, range is 1~120
Request Parameters	<code>value</code> : Acceleration value (float)
Return Value	None

### 4.12.41.2 Set RTCP Parameter

Method Name	<b>ExtraParam.rctp()</b>
Description	Set RTCP parameter
Request Parameters	None

Method Name	<b>ExtraParam.rctp()</b>
Return Value	None

#### 4.12.41.3 Set Frame Offset

Method Name	<b>ExtraParam.offset( <code>index</code> )</b>
Description	Set frame offset
Request Parameters	<code>index</code> : Offset index (integer)
Return Value	None

#### 4.12.41.4 Set Time-Based Operation or Assignment

Method Name	<b>ExtraParam.tb( <code>second</code> , <code>type</code> , <code>name</code> =None, <code>index</code> =None, <code>status</code> =None)</b>
Description	Set time-based operation or assignment
Request Parameters	<code>second</code> : Seconds (integer) <code>type</code> : Execution type (string) <code>name</code> : Name (for operation, string) <code>index</code> : Index (for assignment, integer) <code>status</code> : Status (for assignment, string)
Return Value	None

#### 4.12.41.5 Set Jump

Method Name	<b>ExtraParam.skip( <code>index</code> )</b>
Description	Set jump
Request Parameters	<code>index</code> : Label index (integer)
Return Value	None

## 4.12.42 JUMP Quick Motion Instruction

### 4.12.42.1 JUMP Quick Motion Instruction

Method Name	<b>move_jump</b> ( <a href="#">pose_type</a> , <a href="#">pose_index</a> , <a href="#">speed_value</a> , <a href="#">speed_ratio</a> , <a href="#">lim_Z_type</a> , <a href="#">lim_Z_value</a> , <a href="#">smooth_type</a> , <a href="#">smooth_distance</a> =0.0, <a href="#">extra_param</a> =None)
Description	Robot point-to-point move to the specified position
Request Parameters	<p><a href="#">pose_type</a> : Target pose storage type ( <a href="#">MovePoseType</a> , currently supports PR register)</p> <p><a href="#">pose_index</a> : Index of the target position (integer, indicating the position in the PR register on the controller)</p> <p><a href="#">speed_value</a> : Movement speed value (float, unit is mm/s)</p> <p><a href="#">speed_ratio</a> : Movement speed ratio (float, unit is %, range is 0~100)</p> <p><a href="#">lim_Z_type</a> : Z-axis limit type ( <a href="#">SpeedType</a> , can be MR register or numerical VALUE)</p> <p><a href="#">lim_Z_value</a> : Z-axis limit value (if <a href="#">lim_Z_type</a> is MR register, it indicates the MR register sequence number; if it is numerical VALUE, it indicates the Z-axis limit value, unit is mm/s, range is 0~100)</p> <p><a href="#">smooth_type</a> : Smooth type ( <a href="#">SmoothType</a> , can be FINE or SMOOTH_DISTANCE)</p> <p><a href="#">smooth_distance</a> : Smooth distance (float, only effective when <a href="#">smooth_type</a> is SMOOTH_DISTANCE, range is 0~1000)</p> <p><a href="#">extra_param</a> : Extra parameter ( <a href="#">ExtraParam</a> , used to set the extra parameters of the MOVE instruction, optional)</p>
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.42.2 JUMP3 Quick Motion Instruction

Method Name	<b>move_jump3</b> ( <a href="#">pose_type</a> , <a href="#">pose_index</a> , <a href="#">speed_value</a> , <a href="#">speed_ratio</a> , <a href="#">smooth_type</a> , <a href="#">smooth_distance</a> =0.0, <a href="#">extra_param</a> =None)
Description	Robot point-to-point move to the specified position
Request Parameters	<p><a href="#">pose_type</a> : Target pose storage type ( <a href="#">MovePoseType</a> , currently supports PR register)</p> <p><a href="#">pose_index</a> : Index of the 3 target positions (list, containing 3 integers, indicating the positions in the PR register on the controller)</p> <p><a href="#">speed_value</a> : Movement speed value (float, unit is mm/s)</p> <p><a href="#">speed_ratio</a> : Movement speed ratio (float, unit is %, range is 0~100)</p>

Method Name	<b>move_jump3</b> ( <a href="#">pose_type</a> , <a href="#">pose_index</a> , <a href="#">speed_value</a> , <a href="#">speed_ratio</a> , <a href="#">smooth_type</a> , <a href="#">smooth_distance</a> =0.0, <a href="#">extra_param</a> =None)
	<a href="#">smooth_type</a> : Smooth type ( <a href="#">SmoothType</a> , can be FINE or SMOOTH_DISTANCE) <a href="#">smooth_distance</a> : Smooth distance (float, only effective when <a href="#">smooth_type</a> is SMOOTH_DISTANCE, range is 0~1000) <a href="#">extra_param</a> : Extra parameter ( <a href="#">ExtraParam</a> , used to set the extra parameters of the MOVE instruction, optional)
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

### 4.12.42.3 JUMP3CP Quick Motion Instruction

Method Name	<b>move_jump3cp</b> ( <a href="#">pose_type</a> , <a href="#">pose_index</a> , <a href="#">speed_value</a> , <a href="#">smooth_type</a> , <a href="#">smooth_distance</a> =0.0, <a href="#">extra_param</a> =None)
Description	Robot point-to-point move to the specified position
Request Parameters	<a href="#">pose_type</a> : Target pose storage type ( <a href="#">MovePoseType</a> , currently supports PR register) <a href="#">pose_index</a> : Index of the 3 target positions (list, containing 3 integers, indicating the positions in the PR register on the controller) <a href="#">speed_value</a> : Movement speed value (float, unit is mm/s) <a href="#">smooth_type</a> : Smooth type ( <a href="#">SmoothType</a> , can be FINE or SMOOTH_DISTANCE) <a href="#">smooth_distance</a> : Smooth distance (float, only effective when <a href="#">smooth_type</a> is SMOOTH_DISTANCE, range is 0~1000) <a href="#">extra_param</a> : Extra parameter ( <a href="#">ExtraParam</a> , used to set the extra parameters of the MOVE instruction, optional)
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result

## 4.13 Robot Teaching Motion

### Overview

The Jogging module provides jog-control interfaces for the robot in teach mode, supporting single-axis stepping, continuous jogging, multi-axis coordinated motion, and emergency stop. Using the `jogging` API you can remotely perform manual adjustments from the host PC that are identical to those on the teach pendant—ideal for debugging, teaching, or fine-tuning positions.

#### 4.13.1 Robot Jogging Step Move

Method Name	<code>jogging.step_move( <code>aj_num</code> : int, <code>step_length</code> : float = 0, <code>step_angle</code> : float = 0) -&gt; StatusCodeEnum</code>
Description	Perform single-axis step jogging based on the configured step size.
Request Parameters	<code>aj_num</code> : int Values 1–6 correspond to axes in the current coordinate system. In Cartesian coordinates, x, y, z, rx, ry, rz map to 1–6. The sign indicates motion direction. <code>step_length</code> : float Linear step size in mm; omitted to keep the current step size. <code>step_angle</code> : float Rotational step size in degrees; omitted to keep the current rotation step.
Return Value	<a href="#"><code>StatusCodeEnum</code></a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

```
 jogging/step_move.py
```

```

#!/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
"""

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 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")

```

## 4.13.2 Robot Jogging Continuous Move

Method Name	<b>jogging.continuous_move</b> ( <b>aj_num</b> : int) -> StatusCodeEnum
Description	Perform single-axis continuous jogging.
Request Parameters	<b>aj_num</b> : int Values 1–6 correspond to axes in the current coordinate system. In Cartesian coordinates, x, y, z, rx, ry, rz map to 1–6. The sign indicates motion direction.
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## Example Code

 jogging/continuous\_move.py

py

```
#!/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 function -
"""
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.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.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 code: {ret}.")
    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 Robot Multi-axis Jogging Continuous Move

Method Name	<code>jogging.multi_move( <a href="#">aj_num</a> : List[int]) -&gt; StatusCodeEnum</code>
Description	Perform multi-axis continuous jogging.
Request Parameters	<code>aj_num</code> : List[int] Values 1–6 correspond to axes in the current coordinate system; list only the axes to move. In Cartesian coordinates, x, y, z, rx, ry, rz map to 1–6. The sign of each value indicates motion direction.
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### Example Code

 jogging/multi\_move.py

py

```

#!/python
"""
    Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
    Instruction: 示教运动功能-多关节点运动使用示例 / / Teaching motion function - Multi-joi
"""

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.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1,2,3
# [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 code: {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 Stop the Robot Continuous Move

Method Name	<b>jogging.stop()</b>
Description	Terminate the robot jog (JOG) motion.
Request Parameters	No parameters
Return Value	<a href="#">StatusCodeEnum</a> : Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

## 4.14 Robot Subscription & Publish Interface

### Overview

The SubPub module handles the robot controller's WebSocket-based publish/subscribe channels: it establishes the connection, subscribes to topics such as robot state, registers, and I/O, and delivers real-time data via callback or blocking calls.

With this module the host PC can listen to live robot information, enable data visualization, and synchronize with external systems.

#### 4.14.1 Connect to WebSocket Server

Method Name	<b>sub_pub.connect()</b> -> StatusCodeEnum
Description	Connect to the robot controller WebSocket server
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Connection status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

#### 4.14.2 Disconnect from WebSocket Server

Method Name	<b>sub_pub.disconnect()</b> -> StatusCodeEnum
Description	Disconnect from the robot controller WebSocket server
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Disconnection status code

Method Name	<b>sub_pub.disconnect()</b> -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

### 4.14.3 Add Robot Status Subscription

Method Name	<b>sub_pub.subscribe_status</b> ( <b>topics</b> : List[ <a href="#">RobotTopicType</a> ], <b>frequency</b> : int = 200) -> StatusCodeEnum
Description	Add robot status data subscriptions
Request Parameters	<b>topics</b> : List[ <a href="#">RobotTopicType</a> ] List of robot topic types to subscribe <b>frequency</b> : int Subscription frequency in Hz, default 200
Return Value	<a href="#">StatusCodeEnum</a> : Subscription status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

### 4.14.4 Add Register Subscription

Method Name	<b>sub_pub.subscribe_register</b> ( <b>reg_type</b> : <a href="#">RegTopicType</a> , <b>reg_ids</b> : List[int], <b>frequency</b> : int = 200) -> StatusCodeEnum
Description	Add register data subscriptions
Request Parameters	<b>reg_type</b> : <a href="#">RegTopicType</a> Register type <b>reg_ids</b> : List[int] Register ID list to subscribe <b>frequency</b> : int Subscription frequency in Hz, default 200
Return Value	<a href="#">StatusCodeEnum</a> : Subscription status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

## 4.14.5 Add IO Subscription

Method Name	<code>sub_pub.subscribe_io( <b>io_list</b> : List[tuple[<a href="#">IOTopicType</a>, int]], <b>frequency</b> : int = 200) -&gt; StatusCodeEnum</code>
Description	Subscribe to IO signal data, including digital inputs/outputs
Request Parameters	<b>io_list</b> : List[tuple[ <a href="#">IOTopicType</a> , int]] IO list, each element is ( <a href="#">IO type</a> , <a href="#">IO ID</a> ) <b>frequency</b> : int Subscription frequency in Hz, default 200
Return Value	<a href="#">StatusCodeEnum</a> : Subscription status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

## 4.14.6 Start Receiving Messages

Method Name	<code>sub_pub.start_receiving( <b>on_message_received</b> : Callable[[Dict[str, Any]], None]) -&gt; StatusCodeEnum</code>
Description	Start receiving subscription messages and process received data via the callback function
Request Parameters	<b>on_message_received</b> : Callable[[Dict[str, Any]], None] Message receive callback function
Return Value	<a href="#">StatusCodeEnum</a> : Receiving status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

## 4.14.7 Handle Receive Error

Method Name	<b>sub_pub.handle_receive_error() -&gt; StatusCodeEnum</b>
Description	Handle errors when receiving subscription messages; exits the loop if the receive callback raises an exception.
Request Parameters	None
Return Value	<a href="#">StatusCodeEnum</a> : Receiving status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

## 4.14.8 Receive Next Message

Method Name	<b>sub_pub.receive() -&gt; tuple[Dict[str, Any], StatusCodeEnum]</b>
Description	Receive the next message and return it
Request Parameters	None
Return Value	tuple[Dict[str, Any], StatusCodeEnum]: Received message dictionary and status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

### Example Code

 sub\_pub.py

py

```
#!/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 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()

    # 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: {ret}")
    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: {mes
        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 Extension Management

### Overview

The Extension module manages third-party plug-ins on the robot controller. It can retrieve the controller's IP, list/view installed plug-ins, enable/disable them, and invoke their services.

Using the `extension` interface you can centrally control plug-in life-cycle from the host PC and send commands to extend robot functionality.

### 4.15.1 Get the Robot's IP Address

Method Name	<code>extension.get_robot_ip()</code> -> str
Description	Returns the corresponding robot IP address based on the environment where the SDK is running
Request Parameters	None
Return Value	str: IP address (may be None)
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

#### IP Acquisition Logic:

This method automatically determines and returns the appropriate IP address based on the environment where the SDK is running:

- Industrial Teaching Pendant Environment:** Returns the controller's IP address
- Collaborative AP Board Environment:**
  - DHCP mode: Returns the router's IP address (default gateway)
  - Static IP mode: Returns the configured static IP address
- Cloud Environment:** Returns the cloud's internal IP address

#### 4. Other Environments: Returns **None**

##### Note

- This method is primarily used in extension or teaching pendant environments where connection to the robot controller is required

### 4.15.2 Get Extension List

Method Name	<b>extension.get_list()</b> → tuple[list[ExtensionInfo], StatusCodeEnum]
Description	Retrieves information for all extensions currently installed on the robot.
Request Parameters	None
Return Value	<b>list[ExtensionInfo]</b> : list of extension information. <b>StatusCodeEnum</b> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

### 4.15.3 Get Extension Details

Method Name	<b>extension.get( <b>name</b> : str)</b> → tuple[ExtensionInfo, StatusCodeEnum]
Description	Queries detailed information of a single extension by its name.
Request Parameters	<b>name</b> : extension name.
Return Value	<b>ExtensionInfo</b> : detailed extension information. <b>StatusCodeEnum</b> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

## 4.15.4 Toggle Extension Enable Status

Method Name	<b>extension.toggle</b> ( <code>name</code> : str) → <code>StatusCodeEnum</code>
Description	Toggles the enabled/disabled state of the specified extension.
Request Parameters	<code>name</code> : extension name.
Return Value	<code>StatusCodeEnum</code> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

## 4.15.5 Call Simple-Service Extension Command

Method Name	<b>extension.call_service</b> ( <code>name</code> : str, <code>command</code> : str, <code>params</code> : dict = None) → tuple[Any, <code>StatusCodeEnum</code> ]
Description	Invokes a simple-service command provided by the specified extension and returns its execution result.
Request Parameters	<code>name</code> : extension name. <code>command</code> : extension command name. <code>params</code> : command parameters (optional, defaults to None).
Return Value	<code>Any</code> : execution result of the extension command. <code>StatusCodeEnum</code> : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

### Example Code

 extension/extension\_operation.py

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

py

## Instruction: 插件使用示例 / Example of extension usage

```
"""
```

```
from Agilebot import Arm, Extension, 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.errmsg}")
    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 code: {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 code: {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 code: {ret.errmsg}")
```



# Agilebot Python SDK Update Notes

---

## 2.0.0.0 Update (2025/12/5)

1. Changed the underlying communication method.
2. Added support for a local proxy service.
3. Added `step_move` and `continue_move` interfaces under the Jogging class.
4. Added plugin-related interfaces.
5. Added `get_top_alarm` interface to fetch the most severe alarm.
6. Added JUMP-related instructions under the `bas_script` class.
7. Added interfaces related to trajectory replay.
8. Optimized the coordinate system information class.
9. Updated the payload information class.
10. The `get_all_active_alarms` interface now supports language selection.
11. The `get_version` API has been renamed to `get_controller_version` .
12. The `is_connect` API has been renamed to `is_connected` .
13. Added the `get_op_mode` API to query the manipulator's operation mode.
14. Added a subscription class `SubPub` for subscribing to robot status, register data, and I/O information.
15. The `CoordinateSystem` class now provides interfaces for calculating TF/UF.
16. Added two subclasses `TF` and `UF` under the `CoordinateSystem` class.
17. The plugin-management class now exposes plugin-related APIs.
18. The `Trajectory` class added interfaces for trajectory replay.
19. Python dependencies have been optimized; Python 3.8 and above is now supported.

---

## 1.7.0.0 Update (2025/5/30)

1. Register Changes

1. All registers are now placed under the `Registers` class, and other `Registers` will be deprecated in the future.
2. Interface names have been changed to `read_R` , `write_R` , `delete_R` , etc.
3. The `add` method has been removed; use `write_XX` instead.
4. The RPC interface for reading and writing registers has been changed to a new RPC that only reads values, improving speed.
5. `read_R` , `read_MR` , `read_SR` now directly return the corresponding values and execution results.

## 2. Changes to `arm` Class Interfaces

1. The interfaces `get_arm_model_info` , `get_ctrl_status` , `get_robot_status` , `get_servo_status` , `get_soft_mode` , `get_version` have been changed to return 'result + execution status'.
2. The interfaces `servo_on` , `servo_off` , `servo_reset` have been moved to the `arm` class, and will be deprecated under `execution` in the future.

## 3. Changes to `motion` Class

1. Payload-related operation interfaces have been moved to `motion.payload` .
  2. New interfaces for payload measurement have been added.
  3. New interfaces `get_OVC` , `set_OVC` , `get_OAC` , `set_OAC` , `get_TF` , `set_TF` , `get_UF` , `set_UF` , `get_TCS` , `set_TCS` have been added.
  4. The `convert_cart_to_joint_simple` interface now includes settings for `uf_index` and `tf_index` .
  5. New interfaces `move_joint` , `move_line` , `move_circle` have been added.
  6. New interfaces `convert_joint_to_cart` , `convert_cart_to_joint` have been added.
  7. New interfaces `enter_position_control` , `exit_position_control` , `set_udp_feedback_params` have been added.
- ## 4. The `digital_signals` class has been renamed to `signals` to simplify the name.
5. The `execution` class has added the `execute_bas_script` interface for executing scripts.
  6. The `jogging.move` interface has been updated to include step value modification: `move(*self*, *aj_num*: *int*, *move_mode*: MoveMode = None, *step_length*: *float* = 0)` .
  7. A new `bas_script` class has been added for generating scripts.
  8. A new `modbus` class has been added for direct control of Modbus devices.

### 1.7.1.3 Update (2025/7/3)

1. Fixed the issue where the `get_all_active_alarms` interface might fail.
2. Fixed the internal parameter setting error in `move_circle` .
3. Updated the example programs.
4. Added the plugin management class, providing the `extension.get_robot_ip` interface.
5. Added interfaces related to trajectory reproduction.