Agilebot Robot SDK

Agilebot Robot SDK Manual

Python SDK

High-performance robot control development kit based on Python, featuring elegant and concise API design to help you rapidly build intelligent robotics applications.

Learn More →

C# SDK

Enterprise-grade robot control solution for the .NET ecosystem, providing type-safe strongly-typed APIs for seamless integration into industrial automation systems.

Learn More →

Python SDK

Prologue

Version History

Document Version	SDK Version	Release Date
V2.0	1.7.1.3	2025.07.07
V3.0	2.0.1.0	2026.01.26

Update Notes

Robot Version Compatibility

The SDK supports Agilebot Scara, Puma, and collaborative robot series. It must be used on devices with the robot software installed. Some functions may return different results across versions; see Chapter 4 for details. When the SDK connects to the robotic arm, it checks the motion control software version. If the version is below the minimum requirement, the connection will fail. If it is below the recommended version, a low-version prompt will appear. Please update the robot software version promptly. Some SDK interfaces only support specific controller versions. Please check compatibility for each interface.

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
v2.0.1.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 a Python environment. We recommend using Conda. Install the latest Miniforge or Miniconda for your system.
- Conda download link: Miniforge3
- After installation, launch Miniforge Prompt and enter the following command to initialize conda:

```shell
conda init
```

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

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

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

```shell
conda activate agilesdk
```

  - If setting up a Conda environment is inconvenient, use `venv` to create and activate a virtual environment:

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

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

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

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

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

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:

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

- 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

Term	Description
OAC	Overall Acceleration Control, used to set the overall acceleration multiplier of the 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

```python
from Agilebot import StatusCodeEnum
```

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 50 coordinate systems with IDs in [1, 50]
FILE_NOTEXIST	-25	File does not exist
INVALID_IO_LIST_PARAMETERS	-26	Invalid IO list parameters

Name	Enum Value	Description
INVALID_PARAMETER	-27	Invalid parameter
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

```python
from Agilebot import RobotStatusEnum
```

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
ROBOT_FORCE_DRAG	4	Robot in force-drag mode

Name	Enum Value	Description
ROBOT_IDLE_TO_RUNNING	101	Transition from idle to running
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

```python
from Agilebot import CtrlStatusEnum
```

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

```python
from Agilebot import ServoStatusEnum
```

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

```python
from Agilebot import SoftModeEnum
```

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

```python
from Agilebot import PoseType
```

Fields

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

3.7 TCSType

Description

Coordinate system type.

Import

```python
from Agilebot import TCSType
```

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

```python
from Agilebot import Joint
```

Attributes

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

Attribute	Type	Description
j3	float	Value of joint 3
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

```python
from Agilebot import Position
```

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

Attribute	Type	Description
c	float	C-axis angle

3.10 Posture

Description

Data structure describing the robot posture parameters.

Import

```python
from Agilebot import Posture
```

Attributes

Attribute	Type	Description
turn_cycle	int[]	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 ≥ 180° map to ≥ 1, between -179.99° and 179.99° map to 0, ≤ -180° map to ≤ -1.
wrist_flip	int	Wrist flip posture (-1/1). On 6-axis robots (joint 5), 1 means the wrist flips downward, -1 means it flips upward.
arm_up_down	int	Arm up/down posture (-1/1). 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).
arm_back_front	int	Arm front/back posture (-1/1). 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).
arm_left_right	int	Arm left/right posture (-1/1). 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

```python
from Agilebot import BaseCartData
```

Attributes

Attribute	Type	Description
position	Position	Cartesian pose data
posture	Posture	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

```python
from Agilebot import MotionPose
```

Attributes

Attribute	Type	Description
cartData	BaseCartData	Cartesian data

Attribute	Type	Description
joint	Joint	Joint data
pt	PoseType	Pose type, default is PoseType.JOINT

3.13 DHparam

Description

Robot DH parameter list.

Import

```python
from Agilebot import DHparam
```

Attributes

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

3.14 PayloadInfo

Description

Robot payload information.

Import

```python
from Agilebot import PayloadInfo
```

Attributes

Attribute	Type	Description
id	int	Payload ID
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

```python
from Agilebot import ProgramCartData
```

Attributes

Attribute	Type	Description
baseCart	BaseCartData	Cartesian data
uf	int	User coordinate system ID in use
tf	int	Tool coordinate system ID in use

3.16 ProgramPoseData

Description

Program pose data.

Import

```python
from Agilebot import ProgramPoseData
```

Attributes

Attribute	Type	Description
cartData	ProgramCartData	Program pose data
joint	Joint	Joint data
pt	PoseType	Pose type, default is PoseType.JOINT

3.17 ProgramPose

Description

Program pose.

Import

```python
from Agilebot import ProgramPose
```

Attributes

Attribute	Type	Description
poseData	ProgramPoseData	Program pose data
id	int	Pose ID
name	str	Pose name
comment	str	Pose comment

3.18 PoseRegisterData

Description

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

Import

```python
from Agilebot import PoseRegisterData
```

Attributes

Attribute	Type	Description
cartData	BaseCartData	Cartesian pose data
joint	Joint	Joint data
pt	PoseType	Pose type, default is PoseType.JOINT

3.19 PoseRegister

Description

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

Import

```python
from Agilebot import PoseRegister
```

Attributes

Attribute	Type	Description
poseRegisterData	PoseRegisterData	Register pose data
id	int	Register ID
name	str	Register name
comment	str	Register comment

3.20 TransformStatusEnum

Description

Trajectory transformation status.

Import

```python
from Agilebot import TransformStatusEnum
```

Fields

Name	Enum Value	Description
TRANSFORM_START	0	Transformation started
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

```python
from Agilebot import HWState
```

Fields

Name	Description
TOPIC_JOINT	Publish robot joint feedback
TOPIC_CURRENT_CARTESIAN	Publish current TCP Cartesian pose
TOPIC_UF	Publish current user frame info

Name	Description
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
TOPIC_CTRL_STATUS	Publish controller state
TOPIC_SERVO_STATUS	Publish servo controller status
TOPIC_TRAJECTORY_RECORDS_STATUS	Publish trajectory record status

3.22 CoordinateSystemType

Description

Robot coordinate system type.

Import

```python
from Agilebot import CoordinateSystemType
```

Fields

Name	Enum Value	Description
UserFrame	0	User coordinate system
ToolFrame	1	Tool coordinate system

3.23 Coordinate

Description

Coordinate system data, including tool and user frames.

Import

```python
from Agilebot import Coordinate
```

Attributes

Attribute	Type	Description
id	int	Coordinate system ID
name	str	Coordinate system name
comment	str	Coordinate system comment
data	Position	Coordinate pose data

3.24 DragStatus

Description

Robot drag status.

Import

```python
from Agilebot import DragStatus
```

Attributes

Attribute	Type	Description
cart_status	CartStatus	Cartesian axis drag/lock status
joint_status	JointStatus	Joint axis drag/lock status

Attribute	Type	Description
is_continuous_drag	bool	Continuous drag flag

3.24.1 CartStatus

Description

Cartesian status class representing Cartesian axis drag state.

Import

```python
from Agilebot import CartStatus
```

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

```python
from Agilebot import JointStatus
```

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
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

```python
from Agilebot import ModbusChannel
```

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

```python
from Agilebot import PayloadIdentifyState
```

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

Name	Enum Value	Description
WRONG_DATA	-1	Invalid data

3.27 MoveMode

Description

Jog move mode.

Import

```python
from Agilebot import MoveMode
```

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

```python
from Agilebot import SignalType
```

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
`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

```python
from Agilebot import SignalValue
```

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

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

Constructor

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

Attributes

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

3.31 SubPub Topic Types

3.31.1 RobotTopicType

Description

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

Import

```python
from Agilebot import RobotTopicType
```

Enum Values

Name	Description
JOINT_POSITION	Joint feedback (rad)

Name	Description
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
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

Name	Description
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

```python
from Agilebot import RegTopicType
```

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

```python
from Agilebot import IOTopicType
```

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
TDO	TDO	Tool digital output
TAI	TAI	Tool analog input
AI	AI	Analog input
AO	AO	Analog output

3.32 BasScript Related Types

The BasScript class uses the following enum types to define various parameters and configuration options:

3.32.1 MovePoseType

Description

Coordinate types

Import

```python
from Agilebot import MovePoseType
```

Enum Values

Name	Enum Value	Description
PR	"PR"	PR

3.32.2 SmoothType

Description

Smooth types

Import

```python
from Agilebot import SmoothType
```

Enum Values

Name	Enum Value	Description
FINE	"FINE"	None
SMOOTH_DISTANCE	"SD"	Linear

3.32.3 SpeedType

Description

Speed types

Import

```python
from Agilebot import SpeedType
```

Enum Values

Name	Enum Value	Description
VALUE	0	Absolute
MR	1	Relative

3.32.4 RegisterType

Description

Register types

Import

```python
from Agilebot import RegisterType
```

Enum Values

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

3.32.5 IOType

Description

IO types

Import

```python
from Agilebot import IOType
```

Enum Values

Name	Enum Value	Description
DI	"DI"	DI
DO	"DO"	DO
AI	"AI"	AI
AO	"AO"	AO
RI	"RI"	RI
RO	"RO"	RO
UI	"UI"	UI
UO	"UO"	UO
GI	"GI"	GI
GO	"GO"	GO
TAI	"TAI"	TAI
TAO	"TAO"	TAO
TDI	"TDI"	TDI
TDO	"TDO"	TDO

3.32.6 IOStatus

Description

IO switch status

Import

```python
from Agilebot import IOStatus
```

Enum Values

Name	Enum Value	Description
ON	"ON"	On
OFF	"OFF"	Off
PULSE	"PULSE"	Pulse
POSITIVE_EDGE	"PE"	Positive edge
NEGATIVE_EDGE	"NE"	Negative edge

3.32.7 MathOperator

Description

Math operators

Import

```python
from Agilebot import MathOperator
```

Enum Values

Name	Enum Value	Description
ADD	"+"	Add
SUB	"-"	Subtract
MUL	"*"	Multiply
DIV	"/"	Divide

3.32.8 BooleanOperator

Description

Boolean operators

Import

```python
from Agilebot import BooleanOperator
```

Enum Values

Name	Enum Value	Description
AND	"AND"	And
OR	"OR"	Or
EQ	"="	Equal
NE	"<>"	Not equal
GT	">"	Greater than
GE	">="	Greater than or equal
LT	"<"	Less than
LE	"<="	Less than or equal

3.32.9 AssignType

Description

Assignment types

Import

```python
from Agilebot import AssignType
```

Enum Values

Name	Enum Value	Description
R	"R"	R
MR	"MR"	MR
PR	"PR"	PR
PR_ELEMENT	" PR"	PR element
SR	"SR"	SR
UF	"UF"	UF
TF	"TF"	TF
MH	"MH"	MH
MI	"MI"	MI
DO	"DO"	DO
RO	"RO"	RO
GO	"GO"	GO
AO	"AO"	AO
TAO	"TAO"	TAO
TDO	"TDO"	TDO

3.32.10 OtherType

Description

Other types

Import

```python
from Agilebot import OtherType
```

Enum Values

Name	Enum Value	Description
VALUE	0	Value
STRING	1	String
IO_STATUS	2	Status
CURRENT_POSE	3	Current pose

3.32.11 CurrentPose

Description

Current pose

Import

```python
from Agilebot import CurrentPose
```

Enum Values

Name	Enum Value	Description
J_POS	"J_POS"	Joint values
L_POS	"L_POS"	Cartesian values

3.32.12 LoadType

Description

Load types

Import

```python
from Agilebot import LoadType
```

Enum Values

Name	Enum Value	Description
R	"R"	R
SR	"SR"	SR
STRING	"STRING"	String
VALUE	"VALUE"	Value

3.32.13 StrType

Description

String types

Import

```python
from Agilebot import StrType
```

Enum Values

Name	Enum Value	Description
STRING	"STRING"	String
SR	"SR"	SR register

3.32.14 ValueType

Description

Value types

Import

```python
from Agilebot import ValueType
```

Enum Values

Name	Enum Value	Description
VALUE	"VALUE"	Value
R	"R"	R register

3.32.15 ParamType

Description

Parameter types

Import

```python
from Agilebot import ParamType
```

Enum Values

Name	Enum Value	Description
TF_NO	"TF_NO"	TF number
UF_NO	"UF_NO"	UF number
OVC	"OVC"	Global velocity
OAC	"OAC"	Global acceleration
PAYLOAD_NO	"PAYLOAD_NO"	Payload number

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 completes:

- SDK version checking
- Controller type identification
- Proxy service selection
- Logging of the chosen communication path

Notes

- When the robot software version is lower than 7.7.0, ensure that `local_proxy=True` when instantiating the `Arm` class, and the PC has local communication capability.
- For industrial robots, if you need to keep PC mode active:
  - Call `acquire_access()` after connecting to apply for pendant control authority
  - Promptly call `release_access()` to release authority after operations are completed
  - Avoid long-term occupation of the pendant's control authority

Constructor

Method Name	Arm(`local_proxy` : bool = False)
Description	The Agilebot robot class, encapsulating all available interfaces.
Request Parameters	`local_proxy` : bool Whether to use a local controller proxy (default `False` ; `True` : start the proxy locally, only supports IP address input (not domain name), requires PC with local communication capability; `False` : use the controller/pendant built-in proxy service, robot software must be v7.7 or above and have proxy service installed).
Return Value	StatusCodeEnum: Function execution result
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	connect(arm_controller_ip: str = `COBOT_IP` , teach_panel_ip: Optional[str] = None) -> StatusCodeEnum
Description	Connect to the Agilebot robot. The method auto-detects cooperative/industrial models, processes teach pendant IPs, and starts the appropriate proxy service.
Request Parameters	`arm_controller_ip` : str Controller IP (defaults to `COBOT_IP`). `teach_panel_ip` : Optional[str] Teach pendant IP for industrial robots (when omitted, collaborative robots don't need pendant IP, industrial robots use default pendant IP or controller IP).
Return Value	StatusCodeEnum: Function execution result
Notes	- Local proxy is started only if `local_proxy=True` **and** a valid IP address is provided; Airbot domains only support direct connection and cannot work with the local proxy. - Invalid IP/domain values return `INVALID_IP_ADDRESS` . - A failure to contact the controller returns `CONTROLLER_CONNECTION_ERROR` with details.
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	is_connected() -> bool
Description	Check whether the connection with the robot is valid
Request Parameters	No parameters
Return Value	bool: Connection status, True: Connection is valid, False: Connection is invalid
Notes	The legacy `is_connect()` method is deprecated; use `is_connected()` 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	disconnect()
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 co
nnection
```

```
"""

from Agilebot import Arm, StatusCodeEnum

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

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

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

4.1.4 Get the Current Robot Model

Method Name	get_arm_model_info() -> tuple[str, StatusCodeEnum]
Description	Get the current robot model information
Request Parameters	No parameters
Return Value	str: Robot model, e.g., "GBT-C5A" StatusCodeEnum: 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

py

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

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 获取型号
```

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

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

4.1.5 Get the Robot's Operating Status

Method Name	get_robot_status() -> tuple[RobotStatusEnum, StatusCodeEnum]
Description	Get the operating status of the Agilebot robot
Request Parameters	No parameters
Return Value	RobotStatusEnum: Robot operating status StatusCodeEnum: 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
#!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: {ret.errmsg}"
    )
```

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

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

4.1.6 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	CtrlStatusEnum: Controller operating status StatusCodeEnum: 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

py

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

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
```

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

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

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

4.1.7 Get the Current Servo Controller Status

Method Name	get_servo_status() -> tuple[ServoStatusEnum, StatusCodeEnum]
Description	Get the current status of the Agilebot robot servo controller
Request Parameters	No parameters
Return Value	ServoStatusEnum: Servo controller status StatusCodeEnum: 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.e
rrmsg}"
    )
```

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

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

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

4.1.8 Get the Current Soft Mode of the Robot

Method Name	get_soft_mode() -> tuple[SoftModeEnum, StatusCodeEnum]
Description	Get the current soft mode of the Agilebot robot (PC mode manual/auto state)
Request Parameters	No parameters

Method Name	get_soft_mode() -> tuple[SoftModeEnum, StatusCodeEnum]
Return Value	SoftModeEnum: Soft mode status StatusCodeEnum: 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	set_soft_mode(`soft_mode` : SoftModeEnum) -> StatusCodeEnum
Description	Set the current soft mode of the robot (PC mode manual/auto state)
Request Parameters	`soft_mode` : SoftModeEnum Soft mode value
Return Value	StatusCodeEnum: 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)

# [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 stat
e failed, error code: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

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

4.1.10 Get Robot Operation Mode

Method Name	get_op_mode() -> tuple[SoftModeEnum, StatusCodeEnum]
Description	Get the current robot operation mode (such as manual, auto, etc. operation permission status for robot/virtual controller).
Request Parameters	No parameters
Return Value	SoftModeEnum: Operation mode StatusCodeEnum: Function execution result
Notes	If the controller returns an invalid mode, it will fall back to `SoftModeEnum.UNKNOWN` .
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.11 Set Robot Operation Mode

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

4.1.12 Upper Computer Acquires Control Authority

Method Name	acquire_access()
Description	The PC acquires control authority, putting the robot into PC mode. 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	release_access()
Description	Release the PC control authority, stopping the PC mode heartbeat and handing control back to the teach pendant.

Method Name	release_access()
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 permiss
ions
"""

from Agilebot import Arm, StatusCodeEnum

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

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

```
    exit(1)

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

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

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

4.1.14 Get the Robot Controller Version

Method Name	get_controller_version() -> tuple[str, StatusCodeEnum]
Description	Get the current controller version of the Agilebot robot.
Request Parameters	No parameters
Return Value	str: Controller version StatusCodeEnum: 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

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

4.1.15 Set the Robot LED Indicator Light

Method Name	switch_led_light(`mode` : bool) -> StatusCodeEnum
Description	Control the LED indicator light switch of the Agilebot robot
Request Parameters	`mode` : bool LED status (True on, False off)
Return Value	StatusCodeEnum: 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 o
f the LED
"""

import time

from Agilebot import Arm, StatusCodeEnum
```

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

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

time.sleep(5)

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

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

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

4.1.16 Power On the Robot Servo

Method Name	servo_on() -> StatusCodeEnum
Description	Power on the servo of the Agilebot robot
Request Parameters	No parameters
Return Value	StatusCodeEnum: 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	servo_off() -> StatusCodeEnum
Description	Power off the servo of the Agilebot robot

Method Name	servo_off() -> StatusCodeEnum
Request Parameters	No parameters
Return Value	StatusCodeEnum: 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	servo_reset() -> StatusCodeEnum
Description	Reset the servo of the Agilebot robot
Request Parameters	No parameters
Return Value	StatusCodeEnum: 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

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

import time

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

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

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

4.1.19 Emergency Stop

Method Name	estop() -> StatusCodeEnum
Description	Make the Agilebot robot perform an emergency stop immediately.
Request Parameters	No parameters
Return Value	StatusCodeEnum: Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

arm/estop.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

4.2 Robot Motion Control and Status

Overview

The Motion class is the core object for motion control of Jiebote robots, responsible for encapsulating the following core functionalities:

- Speed/acceleration parameter management
- Coordinate system management
- Point conversion
- Trajectory motion control
- Drag-to-teach
- Real-time control
- Payload management

Typical Workflow

After the Arm connection is established, obtain the Motion instance via `arm.motion` , no separate initialization required.

4.2.1 Get Robot Parameters

4.2.1.1 Get OVC Overall Velocity Coefficient

Method Name	motion.get_OVC() -> tuple[float, StatusCodeEnum]
Description	Get the current OVC Overall Velocity Coefficient of the robot, which ranges from 0 to 1
Request Parameters	No parameters

Method Name	motion.get_OVC() -> tuple[float, StatusCodeEnum]
Return Value	float: Velocity ratio, ranging from 0 to 1 StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.1.2 Get OAC Overall Acceleration Coefficient

Method Name	motion.get_OAC() -> tuple[float, StatusCodeEnum]
Description	Get the current OAC Overall Acceleration Coefficient of the robot, which ranges from 0 to 1.2
Request Parameters	No parameters
Return Value	float: Acceleration ratio, ranging from 0 to 1.2 StatusCodeEnum: Function execution result
Compatible 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	motion.get_TF() -> 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 StatusCodeEnum: Function execution result
Compatible 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	motion.get_UF() -> 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 StatusCodeEnum: Function execution result
Compatible 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	motion.get_TCS() -> tuple[TCSType, StatusCodeEnum]
Description	Get the current TCS teaching coordinate system used by the robot, refer to TCSType for details
Request Parameters	No parameters
Return Value	TCSType: Teaching coordinate system number StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/get_param.py

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

from Agilebot import Arm, StatusCodeEnum, TCSType
```

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

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

res, ret = arm.motion.get_OAC()
if ret == StatusCodeEnum.OK:
    print(
        "获取全局加速度比率成功 / Get global acceleration ratio successful"
    )
else:
    print(
```

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

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

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

res, ret = arm.motion.get_TF()
if ret == StatusCodeEnum.OK:
    print(
        "获取工具坐标系成功 / Get tool coordinate system successful"
```

```
    )
else:
    print(
        f"获取工具坐标系失败，错误代码 / Get tool coordinate system failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
print(f"工具坐标系 / Tool coordinate system: {res}")

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

4.2.2 Set Robot Parameters

4.2.2.1 Set OVC Overall Velocity Coefficient

Method Name	motion.set_OVC(`value` : float) -> StatusCodeEnum
Description	Set the OVC Overall Velocity Coefficient of the robot
Request Parameters	`value` : float Velocity ratio (range 0~1, > 0)
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.2.2 Set OAC Overall Acceleration Coefficient

Method Name	motion.set_OAC(`value` : float) -> StatusCodeEnum
Description	Set the OAC Overall Acceleration Coefficient of the robot

Method Name	motion.set_OAC(`value` : float) -> StatusCodeEnum
Request Parameters	`value` : float Acceleration ratio (range 0.01~1.2)
Return Value	StatusCodeEnum: Function execution result
Compatible 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	motion.set_TF(`value` : int) -> StatusCodeEnum
Description	Set the current TF tool coordinate system used by the robot
Request Parameters	`value` : int Tool coordinate system number (range 0~50)
Return Value	StatusCodeEnum: Function execution result
Compatible 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	motion.set_UF(`value` : int) -> StatusCodeEnum
Description	Set the current UF user coordinate system used by the robot
Request Parameters	`value` : int User coordinate system number (range 0~50)
Return Value	StatusCodeEnum: Function execution result
Compatible 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	motion.set_TCS(`value` : TCSType) -> StatusCodeEnum
Description	Set the current TCS teaching coordinate system used by the robot, refer to TCSType for details

Method Name	motion.set_TCS(value : TCSType) -> StatusCodeEnum
Request Parameters	value : TCSType TCS teaching coordinate system
Return Value	StatusCodeEnum: Function execution result
Compatible 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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置机器人各参数
```

```
# [EN] Set robot parameters
ret = arm.motion.set_OVC(0.7)
if ret == StatusCodeEnum.OK:
    print(
        "设置全局速度比率成功 / Set global velocity ratio successful"
    )
else:
    print(
        f"设置全局速度比率失败，错误代码 / Set global velocity ratio failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

ret = arm.motion.set_UF(0)
```

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

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

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

4.2.3 Convert Cartesian Pose to Joint Pose

Method Name	motion.convert_cart_to_joint(pose : MotionPose, uf_index : int = 0, tf_index : int = 0) -> tuple[MotionPose, StatusCodeEnum]
Description	Convert pose data from Cartesian coordinates to joint coordinates

Method Name	motion.convert_cart_to_joint(`pose` : MotionPose, `uf_index` : int = 0, `tf_index` : int = 0) -> tuple[MotionPose, StatusCodeEnum]
Request Parameters	`pose` : MotionPose Cartesian pose (PoseType.CART; if posture is omitted, the SDK auto-solves a feasible posture) `uf_index` : int User coordinate system ID (default 0) `tf_index` : int Tool coordinate system ID (default 0)
Return Value	MotionPose: Robot pose StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.4 Convert Joint Pose to Cartesian Pose

Method Name	motion.convert_joint_to_cart(`pose` : MotionPose, `uf_index` : int = 0, `tf_index` : int = 0) -> tuple[MotionPose, StatusCodeEnum]
Description	Convert joint coordinates to Cartesian coordinates
Request Parameters	`pose` : MotionPose Joint pose `uf_index` : int User coordinate system ID (default 0) `tf_index` : int Tool coordinate system ID (default 0)
Return Value	MotionPose: Robot pose StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/convert_pose.py

py

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

```
"""

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

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

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

# [ZH] 转换关节值坐标
# [EN] Convert to joint coordinates
joint_pose, ret = arm.motion.convert_cart_to_joint(
    motion_pose
```

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

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

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

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

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

4.2.5 Joint Space Motion

Method Name	motion.move_joint(`pose` : MotionPose, `vel` : float = 1, `acc` : float = 1) -> StatusCodeEnum
Description	Control the robot end effector to move to the specified position using the fastest path in joint space
Request Parameters	`pose` : MotionPose Point in Cartesian or joint space `vel` : float Velocity ratio (range 0~1) `acc` : float Acceleration ratio (range 0~1.2)
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/move_joint.py

py

```python
#!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,
    Posture,
    StatusCodeEnum,
)

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

# [ZH] 初始化位姿
# [EN] Initialize pose
motion_pose = MotionPose()
motion_pose.pt = PoseType.CART
motion_pose.cartData.position.x = 121.3
motion_pose.cartData.position.y = 416.386
motion_pose.cartData.position.z = 74.104
```

```
motion_pose.cartData.position.a = -180
motion_pose.cartData.position.b = 0
motion_pose.cartData.position.c = 0
motion_pose.cartData.posture = Posture()
motion_pose.cartData.posture.arm_back_front = -1
motion_pose.cartData.posture.arm_left_right = -1
motion_pose.cartData.posture.arm_up_down = -1
motion_pose.cartData.posture.wrist_flip = -1


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

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

4.2.6 Linear Motion

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

Method Name	motion.move_line(`pose` : MotionPose, `vel` : float = 100, `acc` : float = 1) -> StatusCodeEnum
Request Parameters	`pose` : MotionPose Point in Cartesian or joint space `vel` : float TCP speed (range 1~4000 mm/s) `acc` : float Acceleration ratio (range 0~1.2)
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/move_line.py

py

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

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

# [ZH] 初始化Arm类
# [EN] Initialize the Arm class
arm = Arm()
# [ZH] 连接控制器
# [EN] Connect to the controller
ret = arm.connect("10.27.1.254")
# [ZH] 检查是否连接成功
# [EN] Check if the connection is successful
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
```

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

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

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

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

4.2.7 Circular Motion

Method Name	motion.move_circle(`pose1` : MotionPose, `pose2` : MotionPose, `vel` : float = 100, `acc` : float = 1.0) -> StatusCodeEnum
Description	Control the robot end effector to move along a circular trajectory to the specified position
Request Parameters	`pose1` : MotionPose Intermediate pose `pose2` : MotionPose Target pose `vel` : float TCP speed (range 1~4000 mm/s) `acc` : float Acceleration ratio (range 0~1.2)
Return Value	StatusCodeEnum: Function execution result
Compatible 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 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(
        "运动到初始点指令下发成功 / Move to initial position command issued successfully"
    )
else:
    print(
        f"运动到初始点指令下发失败，错误代码 / Move to initial position command issued failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
```

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

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

# [ZH] 等待运动结束
# [EN] Wait for motion to complete
time.sleep(10)

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

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

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

4.2.8 Get Current Pose

Method Name	motion.get_current_pose(pose_type : PoseType, uf_index : int = 0, tf_index : int = 0) -> tuple[MotionPose, StatusCodeEnum]
Description	Get the current pose of the robot, which can be in Cartesian or joint coordinate system
Request Parameters	pose_type :PoseType Pose type uf_index : int User coordinate system ID (only for PoseType.CART; default 0) tf_index : int Tool coordinate system ID (only for PoseType.CART; default 0)
Return Value	MotionPose: Robot pose StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/get_current_pose.py

py

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

from Agilebot import Arm, PoseType, StatusCodeEnum

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

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

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

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

# [ZH] 获取当前位姿
```

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

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

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

4.2.9 Get DH Parameters

Method Name	motion.get_DH_param() -> tuple[list[DHparam], StatusCodeEnum]
Description	Get the robot's DH parameters
Request Parameters	No parameters
Return Value	list(DHparam): List of DH parameters StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

4.2.10 Set DH Parameters

Method Name	motion.set_DH_param(`dh_list` : list[DHparam]) -> StatusCodeEnum
Description	Set the robot's DH parameters
Request Parameters	`dh_list` : list(DHparam) DH parameter list
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

motion/DH_param.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

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

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

4.2.11 Get Axis Lock Status

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

Example Code

motion/get_drag_set.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

# [ZH] 打印结果
# [EN] Print result
```

```
print(
    f"当前X轴拖动状态 / Current X axis drag status: {drag_status.cart_status.
x}\n"
    f"当前Y轴拖动状态 / Current Y axis drag status: {drag_status.cart_status.
y}\n"
    f"当前Z轴拖动状态 / Current Z axis drag status: {drag_status.cart_status.
z}"
)

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

4.2.12 Set Robot Axis Lock Status

Method Name	motion.set_drag_set(`drag_status` : DragStatus) -> StatusCodeEnum
Description	Set the current robot axis lock status, which only applies to teaching movements
Request Parameters	`drag_status` : DragStatus Axis lock status (default all True: unlocked)
Return Value	StatusCodeEnum: Function execution result
Compatible 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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

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

# [ZH] 设置要锁定的轴
# [EN] Set axes to be locked
```

```
drag_status = DragStatus()
drag_status.cart_status.x = False
drag_status.cart_status.y = False
# [ZH] 设置连续拖动开关
# [EN] Set continuous drag switch
drag_status.is_continuous_drag = True

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

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

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

4.2.13 Enable Robot Drag

Method Name	motion.enable_drag(`drag_state` : bool) -> StatusCodeEnum
Description	Enable or disable dragging for the robot
Request Parameters	`drag_state` : bool Drag mode switch (True enter drag mode, False exit drag mode)
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

motion/enable_drag.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 拖动示教使用示例 / example of drag teaching usage
"""

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

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

4.2.14 Enter Real-Time Position Control Mode

Method Name	motion.enter_position_control() -> StatusCodeEnum
Description	Enter real-time position control mode to allow precise position control of the robot
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result
Note	After entering real-time control mode, control commands must be sent via UDP
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.15 Exit Real-Time Position Control Mode

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

Example Code

motion/position_control.py

py

```python
#!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 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: {ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

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

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

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

4.2.16 Set UDP Feedback Parameters

Method Name	motion.set_udp_feedback_params(`flag` : bool, `ip` : str, `interval` : int, `feedback_type` : int, `DO_list` : list[int] = None) -> StatusCodeEnum
Description	Configure the UDP feedback parameters for pushing data to a specified IP address
Request Parameters	`flag` : bool Enable UDP data push; `ip` : str Receiver IP address; `interval` : int Send interval (ms);

Method Name	motion.set_udp_feedback_params(`flag` : bool, `ip` : str, `interval` : int, `feedback_type` : int, `DO_list` : list[int] = None) -> StatusCodeEnum
	`feedback_type` : int Feedback format (0: XML, 1: JSON, 2: PROTO); `DO_list` : list[int] DO signal list (up to 10, optional)
Return Value	StatusCodeEnum: Function execution result
Note	Parameter settings are only effective when the UDP data pushing function is enabled
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Example Code

motion/UDP_pose_feedback.py

py

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

import json
import socket

from Agilebot.IR.A.arm import Arm
from Agilebot.IR.A.status_code import StatusCodeEnum

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

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

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

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


# [ZH] 接收UDP反馈数据示例
```

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

# [ZH] 断开捷勃特机器人连接
# [EN] Disconnect from Agilebot robot
udp_sock.close()
ret = arm.motion.set_udp_feedback_params(
    False,
    udp_local_ip,
    udp_feedback_interval,
    udp_feedback_type,
    udp_do_list,
)
if ret == StatusCodeEnum.OK:
    print(
        "设置UDP反馈参数成功 / Set UDP feedback parameters successful"
    )
```

```
else:
    print(
        f"设置UDP反馈参数失败，错误代码 / Set UDP feedback parameters failed, er
ror code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

Data Pushing Description

Name	Field	Description
RIst: Cartesian Position	X	X-direction value in the tool coordinate system, in millimeters
	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
AIPos: Joint Position	A1-A6	Values of the six joints, in degrees
EIPos: Additional Axis Data	EIPos	Additional axis data

Name	Field	Description
WristBtnState: Wrist Button State	Button State	1 = Button pressed, 0 = Button released
	DragModel	Drag button state
	RecordJoint	Teaching record button state
	PauseResume	Pause/Resume button state
Digout: DO Output	Digout	State of digital output (DO)
ProgramStatus: Program Status	ProgId	Program ID
	Status	Interpreter execution status: 0 = INTERPRETER_IDLE 1 = INTERPRETER_EXECUTE 2 = INTERPRETER_PAUSED
	XPath	Program fragment return value, in the format `program_name:line_number`
IPOC: Timestamp	IPOC	Timestamp

4.2.17 Get Robot Soft Limits

Method Name	motion.get_user_soft_limit() -> 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 StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/get_user_soft_limit.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

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

4.2.18 Payload-Related Interfaces

4.2.18.1 Get the Current Activated Payload ID

Method Name	motion.payload.get_current_payload() -> tuple[int, StatusCodeEnum]
Description	Get the current activated payload ID
Request Parameters	None
Return Value	int: Payload ID StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/get_current_payload.py

py

```python
#!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 ID

Method Name	motion.payload.get_payload_by_id(payload_id : int) -> tuple[PayloadInfo, StatusCodeEnum]
Description	Get the payload information by the specified ID
Request Parameters	payload_id : int Payload ID
Return Value	PayloadInfo: Payload information StatusCodeEnum: Function execution result
Compatible 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 base
d on ID
```

```
"""

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 获取负载
# [EN] Get payload
res, ret = arm.motion.payload.get_payload_by_id(6)
if ret == StatusCodeEnum.OK:
    print("获取负载成功 / Get payload successful")
else:
    print(
        f"获取负载失败，错误代码 / Get payload failed, error code: {ret.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"负载ID / Payload ID: {res.id}\n"
    f"负载质量 / Payload mass: {res.weight}\n"
    f"负载注释 / Payload comment: {res.comment}\n"
    f"负载质心 / Payload mass center: {res.mass_center.x}, {res.mass_center.
```

```
y}, {res.mass_center.z}\n"
    f"负载转动惯量 / Payload inertia moment: {res.inertia_moment.lx}, {res.ine
rtia_moment.ly}, {res.inertia_moment.lz}\n"
)

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

4.2.18.3 Activate Specified Payload

Method Name	motion.payload.set_current_payload(`payload_id` : int) -> StatusCodeEnum
Description	Activate the payload by the specified ID
Request Parameters	`payload_id` : int Payload ID
Return Value	StatusCodeEnum: Function execution result
Compatible 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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

4.2.18.4 Add Custom Payload

Method Name	motion.payload.add_payload(`payload_info` : PayloadInfo) -> StatusCodeEnum
Description	Add a user-defined payload to the robot controller
Request Parameters	`payload_info` : PayloadInfo User-defined payload information
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/add_payload.py

py

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

from Agilebot import Arm, PayloadInfo, StatusCodeEnum

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

```
    exit(1)

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

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

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

4.2.18.5 Delete Payload

Method Name	motion.payload.delete_payload(payload_id : int) -> StatusCodeEnum
Description	Delete a user-defined payload from the controller

Method Name	motion.payload.delete_payload(`payload_id` : int) -> StatusCodeEnum
Request Parameters	`payload_id` : int Payload ID
Return Value	StatusCodeEnum: Function execution result
Compatible 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
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 删除负载使用示例 / Example of delete the load usage
"""

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

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

4.2.18.6 Update Payload

Method Name	motion.payload.update_payload(`payload_info` : PayloadInfo) -> StatusCodeEnum
Description	Update the information of an existing user-defined payload
Request Parameters	`payload_info` : PayloadInfo User-defined updated payload information
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/update_payload.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

```
payload_info.inertia_moment.ly = 10
payload_info.inertia_moment.lz = 10

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

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

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

4.2.18.7 Get All Payloads

Method Name	motion.payload.get_all_payload() -> tuple[list, StatusCodeEnum]
Description	Get all payload information

Method Name	motion.payload.get_all_payload() -> tuple[list, StatusCodeEnum]
Request Parameters	None
Return Value	list : List of all payload information StatusCodeEnum: Function execution result
Compatible 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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取所有负载
```

```
# [EN] Get all payloads
res, ret = arm.motion.payload.get_all_payload()
if ret == StatusCodeEnum.OK:
    print("获取所有负载成功 / Get all payloads successful")
else:
    print(
        f"获取所有负载失败，错误代码 / Get all payloads failed, error code: {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负载注释 / Payload comment: {payload[1]}\n"
    )

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

4.2.18.8 Check Axis 3 Horizontal Level

Method Name	motion.payload.check_axis_three_horizontal() -> tuple[float, StatusCodeEnum]
Description	Check if Axis 3 is horizontal
Request Parameters	None
Return Value	float: The horizontal angle of Axis 3, in degrees StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Method Name	motion.payload.check_axis_three_horizontal() -> tuple[float, StatusCodeEnum]
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	motion.payload.get_payload_identify_state() -> tuple[PayloadIdentifyState, StatusCodeEnum]
Description	Get the state of payload identification
Request Parameters	None
Return Value	PayloadIdentifyState The state of payload identification StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.18.10 Start Payload Identification

Method Name	motion.payload.start_payload_identify(`weight` : float, `angle` : float) -> StatusCodeEnum
Description	Start the payload identification process
Request Parameters	`weight` : float Payload weight (use -1 if unknown); `angle` : float Axis 6 allowable rotation angle (30~90 degrees)
Return Value	StatusCodeEnum: Function execution result
Compatible 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	motion.payload.payload_identify_result() -> tuple[PayloadInfo, StatusCodeEnum]
Description	Get the result of payload identification
Request Parameters	None
Return Value	PayloadInfo: The result of payload identification StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.18.12 Start Interference Check

Method Name	motion.payload.interference_check_for_payload_identify(`weight` : float, `angle` : float) -> StatusCodeEnum
Description	Start interference check for payload identification
Request Parameters	`weight` : float Payload weight; `angle` : float Axis 6 rotation angle (30~90 degrees)
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.18.13 Enter Payload Identification State

Method Name	motion.payload.payload_identify_start() -> StatusCodeEnum
Description	Enter payload identification state
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.18.14 Exit Payload Identification State

Method Name	motion.payload.payload_identify_done() -> StatusCodeEnum
Description	Exit payload identification state
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.18.15 Complete Payload Identification Process

Method Name	motion.payload.payload_identify(`weight` : float, `angle` : 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	`weight` : float Payload weight (use -1 if unknown); `angle` : float Axis 6 rotation angle (30~90 degrees)
Return Value	PayloadInfo: Payload identification result StatusCodeEnum: Function execution result
Compatible Version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	The returned payload can be added to the robot or written to an existing payload in the robot. The complete process steps are: 1. Move to the specified horizontal position and check if it is horizontal 2. Enter payload identification state 3. Start payload identification 4. Get the payload identification result 5. Exit payload identification state

Example Code

motion/payload_identify.py

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

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

motion_pose.joint.j1 = 0
motion_pose.joint.j2 = 0
```

```
motion_pose.joint.j3 = 0
motion_pose.joint.j4 = 0
motion_pose.joint.j5 = 0
motion_pose.joint.j6 = 0

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

while True:
    # [ZH] 获取伺服状态
    # [EN] Get servo status
    state, ret = arm.get_servo_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取伺服状态成功 / Get servo status successful"
        )
    else:
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

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

# [ZH] 开始负载测定并获取结果
```

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

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

4.3 Robot Program Operation Class

Overview

The Execution class provides a unified scheduling interface for robot programs and motion tasks, responsible for the following core functionalities:

- Start/stop/pause/resume teach-pendant programs
- Manage the list of concurrently running tasks
- Execute BAS scripts and other custom workflows

Combining 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	execution.start(`program_name` : str) -> StatusCodeEnum
Description	Execute a specified program.
Request Parameters	`program_name` : str The name of the program to be executed.
Return Value	StatusCodeEnum: 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	execution.stop(`program_name` : str = '') -> StatusCodeEnum
Description	Stop the currently executing program or stop the robot's current motion.

Method Name	execution.stop(program_name : str = '') -> StatusCodeEnum
Request Parameters	program_name : str Program name (default empty string, stops the current running program or motion command).
Return Value	StatusCodeEnum: 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	execution.all_running_programs() -> tuple[list, StatusCodeEnum]
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 thread_id , program_name , xpath , program_status , program_type , etc. StatusCodeEnum: 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	execution.pause(program_name : str = '') -> StatusCodeEnum
Description	Pause the currently executing program or the robot's current motion.
Request Parameters	program_name : str Program name (default empty string, pauses the current running program or motion command).
Return Value	StatusCodeEnum: The result of the function execution.

Method Name	execution.pause(program_name : str = '') -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.3.5 Resume Program Execution

Method Name	execution.resume(program_name : str = '') -> StatusCodeEnum
Description	Continue running a program that is in a paused state.
Request Parameters	program_name : str Program name (default empty string, resumes the current paused program or motion command).
Return Value	StatusCodeEnum: 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.e
rrmsg}"
    )
    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.errms
g}"
    )
    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, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
for program in programs_list:
```

```
    print(f"正在运行的程序名：{program}")

time.sleep(2)

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

time.sleep(2)

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

time.sleep(2)

# [ZH] 停止程序
# [EN] Stop program
ret = arm.execution.stop(program_name)
if ret == StatusCodeEnum.OK:
    print("程序停止成功 / Program stop successful")
else:
    print(
        f"程序停止失败，错误代码 / Program stop failed, error code: {ret.errms
```

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

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

4.3.6 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

py

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

from Agilebot import *
```

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

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

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


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

4.4 Program Pose Read and Write

Overview

The ProgramPose interfaces are used to read, write, and convert pose points in robot teach-pendant programs. Via the `program_pose` module, you can:

- Locate a specific program and point index
- Perform CRUD operations
- Switch between Cartesian and joint representations

This facilitates batch maintenance of program points or offline editing in host PC scenarios.

4.4.1 Get the Value of a Specified Pose in a Program

Method Name	program_pose.read(`program_name` : str, `index` : int, `program_type` : str = USER_PROGRAM) -> tuple[ProgramPose, StatusCodeEnum]
Description	Get the value of a Pose with a specified index in a program.
Request Parameters	`program_name` : str Program name. `index` : int Pose index. `program_type` : str Program type (default USER_PROGRAM).
Return Value	ProgramPose: Pose information. StatusCodeEnum: 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	program_pose.write(program_name : str, index : int, value : ProgramPose, program_type : str = USER_PROGRAM) -> StatusCodeEnum
Description	Modify the value of a Pose with a specified index in a specified program.
Request Parameters	program_name : str Program name. index : int Pose index. value : ProgramPose Pose value to update. program_type : str Program type (default USER_PROGRAM).
Return Value	StatusCodeEnum: 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	program_pose.read_all_poses(program_name : str, program_type : str = USER_PROGRAM) -> tuple[list[ProgramPose], StatusCodeEnum]
Description	Get all Pose information in a specified program.
Request Parameters	program_name : str Program name. program_type : str Program type (default USER_PROGRAM).
Return Value	list[ProgramPose]: List of Pose information. StatusCodeEnum: 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	program_pose.add(program_name : str, index : int, value : ProgramPose, program_type : str = USER_PROGRAM) -> StatusCodeEnum
Description	Add a new Pose at the specified index in a specified program. Returns an error if the index already exists.
Request Parameters	program_name : str Program name. index : int Pose index. value : ProgramPose Pose value to add. program_type : str Program type (default USER_PROGRAM).
Return Value	StatusCodeEnum: 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	program_pose.write_poses(program_name : str, poses_to_update : 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	program_name : str Program name. poses_to_update : list[ProgramPose] List of poses to be updated.
Return Value	StatusCodeEnum: 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	ProgramPoses.convert_pose(`pose` : ProgramPose, `from_type` : PoseType, `to_type` : PoseType) -> tuple[ProgramPose, StatusCodeEnum]
Description	Convert the robot Pose values between joint coordinates and Cartesian space coordinates in a program.
Request Parameters	`pose` : ProgramPose The Pose value to be converted. `from_type` : PoseType The type before conversion. `to_type` : PoseType The desired type after conversion.
Return Value	ProgramPose: Pose information. StatusCodeEnum: 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.e
rrmsg}"
    )
```

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

program_name = "test_prog"

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

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

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

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

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

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

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

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

4.5 IO Signals

Overview

The Signals module offers a unified read/write interface to the controller's I/O, encapsulating the following core functionalities:

- Digital/analog input and output control
- Multi-channel batch operations
- Timed triggering capability

Via `signals` you can:

- Query current signal states
- Batch-write DO/RO/GO ports
- Generate pulses at specified intervals

This enables coordination with external grippers, sensors, or production-line equipment.

4.5.1 Read IO Value of Specified Type and Port

Method Name	signals.read(`signal_type` : SignalType, `index` : int) -> tuple[float, StatusCodeEnum]
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	`signal_type` : SignalType IO type. `index` : int IO index (starts 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). StatusCodeEnum: Result of the function execution.

Method Name	signals.read(`signal_type` : SignalType, `index` : int) -> tuple[float, StatusCodeEnum]
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 IO Value to Specified Type and Port

Method Name	signals.write(`signal_type` : SignalType, `index` : int, `value` : float) -> StatusCodeEnum
Description	Write the value of an IO of a specified type and port. Currently only DO/RO/GO/TDO/AO are supported.
Request Parameters	`signal_type` : SignalType IO type. `index` : int IO index (starts from 1). `value` : float IO value (DO/RO/TDO accept 0/1; GO expects integer; AO accepts float analog).
Return Value	StatusCodeEnum: 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 writi
ng
"""

from Agilebot import (
```

```
    Arm,
    SignalType,
    SignalValue,
    StatusCodeEnum,
)

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

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

# [ZH] 写入IO
# [EN] Write IO
ret = arm.signals.write(SignalType.DO, 1, SignalValue.ON)
if ret == StatusCodeEnum.OK:
    print("写入IO成功 / Write IO successfully")
else:
    print(
```

```
            f"写入IO失败，错误代码 / Write IO failed, error code: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

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

4.5.3 Batch Read DO (Digital Output) Port Values

Method Name	signals.multi_read(`signal_type` : SignalType, `io_list` : list) -> tuple[list, StatusCodeEnum]
Description	Batch read DO (digital output) port values.
Request Parameters	`signal_type` : SignalType IO type (DO only). `io_list` : list Port numbers (must not be empty).
Return Value	list: Controller response in the form [port1, state1, port2, state2, ...] StatusCodeEnum: 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.

4.5.4 Batch Write DO (Digital Output) Signals

Method Name	signals.multi_write(`signal_type` : SignalType, `io_list` : list) -> StatusCodeEnum
Description	Batch write DO (digital output) signals.

Method Name	signals.multi_write(`signal_type` : SignalType, `io_list` : list) -> StatusCodeEnum
Request Parameters	`signal_type` : SignalType IO type (DO only). `io_list` : list Port/value pairs (e.g., [port1, state1, port2, state2]; length even and > 0).
Return Value	StatusCodeEnum: 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 writin
g
"""

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

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

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

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

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

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

4.5.5 Trigger IO Channels with Time Intervals

Method Name	signals.trigger_io_with_intervals(`in_port` : int = -1, `intervals` : list, `out_ports` : list, `pulse_duration` : int) -> StatusCodeEnum
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	`in_port` : int Input port (default -1, no input trigger). `intervals` : list Time intervals (ms). `out_ports` : list Output ports. `pulse_duration` : int Pulse duration (ms).
Return Value	StatusCodeEnum: 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, supporting operations on multiple register types.

Core Features

- Support for reading and writing numeric registers (R)
- Support for reading and writing motion registers (MR)
- Support for reading and writing string registers (SR)
- Support for reading and writing pose registers (PR)
- Support for reading and writing Modbus registers (MH holding registers, MI input registers)

Use Cases

- Pass parameters during program runtime
- Synchronize robot status information
- Share configuration data with external systems
- Implement interactive control between robots and external devices

4.6.1 R Numeric Register Operations

4.6.1.1 Read R Register Value

Method Name	register.read_R(index : int, with_meta : bool = False) -> tuple[float | Register, StatusCodeEnum]
Description	Reads the value of an R numeric register.

Method Name	register.read_R(`index` : int, `with_meta` : bool = False) -> tuple[float | Register, StatusCodeEnum]
Request Parameters	`index` : int R register number to read `with_meta` : bool whether to return register metadata (name/comment)
Return Value	float: R register numeric value (`with_meta=False`) Register: register object (`with_meta=True`) StatusCodeEnum: 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 R Register Value

Method Name	register.write_R(`index` : int, `value` : float) -> StatusCodeEnum register.write_R(`register` : Register) -> StatusCodeEnum
Description	Writes the value of an R numeric register.
Request Parameters	`index` : int R register number to write `value` : float R register numeric value to write `register` : Register object (id/name/comment/value)
Return Value	StatusCodeEnum: 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 R Register

Method Name	register.delete_R(`index` : int) -> StatusCodeEnum
Description	Deletes the specified R numeric register.
Request Parameters	`index` : int R register number to delete
Return Value	StatusCodeEnum: 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, Register, StatusCodeEnum

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

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

```
    exit(1)

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

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

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

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

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

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

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

4.6.2 MR Motion Register Operations

4.6.2.1 Read MR Register Value

Method Name	register.read_MR(`index` : int, `with_meta` : bool = False) -> tuple[int | MotionRegister, StatusCodeEnum]
Description	Reads the value of an MR motion register.
Request Parameters	`index` : int MR register number to read `with_meta` : bool whether to return register metadata (name/comment)
Return Value	int: MR register numeric value (`with_meta=False`) MotionRegister: register object (`with_meta=True`) StatusCodeEnum: 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 MR Register Value

Method Name	register.write_MR(`index` : int, `value` : int) -> StatusCodeEnum register.write_MR(`register` : MotionRegister) -> StatusCodeEnum
Description	Writes the value of an MR motion register.
Request Parameters	`index` : int MR register number to write `value` : int MR register numeric value to write `register` : MotionRegister object (id/name/comment/value)
Return Value	StatusCodeEnum: 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 MR Register

Method Name	register.delete_MR(`index` : int) -> StatusCodeEnum
Description	Deletes the specified MR motion register.
Request Parameters	`index` : int MR register number to delete

Method Name	register.delete_MR(index : int) -> StatusCodeEnum
Return Value	StatusCodeEnum: 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
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: MR寄存器读写示例 / Example of reading and writing MR Registers
"""

from Agilebot import Arm, MotionRegister, StatusCodeEnum

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

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

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

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

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

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

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

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

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

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

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

4.6.3 SR String Register Operations

4.6.3.1 Read SR Register Value

Method Name	register.read_SR(index : int, with_meta : bool = False) -> tuple[str | StringRegister, StatusCodeEnum]
Description	Reads the value of an SR string register.
Request Parameters	index : int SR register number to read with_meta : bool whether to return register metadata (name/comment)
Return Value	str: SR register string value (with_meta=False) StringRegister: register object (with_meta=True) StatusCodeEnum: 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 SR Register Value

Method Name	register.write_SR(index : int, value : str) -> StatusCodeEnum register.write_SR(register : StringRegister) -> StatusCodeEnum
Description	Writes the value of an SR string register.
Request Parameters	index : int SR register number to write value : str SR register string value to write register : StringRegister object (id/name/comment/value)
Return Value	StatusCodeEnum: Write operation execution result

Method Name	register.write_SR(index : int, value : str) -> StatusCodeEnum register.write_SR(register : StringRegister) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.3.3 Delete SR Register

Method Name	register.delete_SR(index : int) -> StatusCodeEnum
Description	Deletes the specified SR string register.
Request Parameters	index : int SR register number to delete
Return Value	StatusCodeEnum: 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, StringRegister

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

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

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

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

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

# [ZH] 使用StringRegister对象写入SR寄存器
# [EN] Write SR register with StringRegister object
```

```
reg = StringRegister()
reg.id = 5
reg.value = "pytest"
reg.name = "SR5"
reg.comment = "example"
ret = arm.register.write_SR(reg)
if ret == StatusCodeEnum.OK:
    print(
        "使用StringRegister写入SR寄存器成功 / Write SR register with StringRegi
ster successful"
    )
else:
    print(
        f"使用StringRegister写入SR寄存器失败，错误代码 / Write SR register with
StringRegister failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

# [ZH] 删除SR寄存器
# [EN] Delete SR register
ret = arm.register.delete_SR(5)
if ret == StatusCodeEnum.OK:
    print(
```

```
            "删除SR寄存器成功 / Delete SR register successful"
        )
    else:
        print(
            f"删除SR寄存器失败，错误代码 / Delete SR register failed, error code: {r
et.errmsg}"
        )
        arm.disconnect()
        exit(1)

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

4.6.4 PR Pose Register Operations

4.6.4.1 Read PR Register Value

Method Name	register.read_PR(`index` : int, `with_meta` : bool = False) -> tuple[PoseRegister, StatusCodeEnum]
Description	Reads the value of a PR pose register.
Request Parameters	`index` : int PR register number to read `with_meta` : bool whether to return register metadata (name/comment)
Return Value	PoseRegister: PR register pose data (includes name/comment when `with_meta=True`) StatusCodeEnum: 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 PR Register Value

Method Name	register.write_PR(value : PoseRegister, with_meta : bool = False) -> StatusCodeEnum
Description	Writes the value of a PR pose register.
Request Parameters	value : PoseRegister PR register pose data to write with_meta : bool whether to write name/comment
Return Value	StatusCodeEnum: 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 PR Register

Method Name	register.delete_PR(index : int) -> StatusCodeEnum
Description	Deletes the specified PR pose register.
Request Parameters	index : int PR register number to delete
Return Value	StatusCodeEnum: 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 regis
ter
"""

from Agilebot import (
    Arm,
    PoseRegister,
    PoseType,
```

```
    Posture,
    StatusCodeEnum,
)

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

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

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

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

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

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

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

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

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

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

4.6.5 Modbus Registers (MH Holding Registers, MI Input Registers)

4.6.5.1 Read MH Register Value

Method Name	register.read_MH(`index` : int) -> tuple[int, StatusCodeEnum]
Description	Reads the value of an MH holding register.
Request Parameters	`index` : int MH register number to read
Return Value	int: MH register numeric value StatusCodeEnum: 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 MH Register Value

Method Name	register.write_MH(`index` : int, `value` : int) -> StatusCodeEnum
Description	Writes the value of an MH holding register.
Request Parameters	`index` : int MH register number to write `value` : int MH register numeric value to write
Return Value	StatusCodeEnum: 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 MI Register Value

Method Name	register.read_MI(`index` : int) -> tuple[int, StatusCodeEnum]
Description	Reads the value of an MI input register.

Method Name	register.read_MI(index : int) -> tuple[int, StatusCodeEnum]
Request Parameters	index : int MI register number to read
Return Value	int: MI register numeric value StatusCodeEnum: 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 MI Register Value

Method Name	register.write_MI(index : int, value : int) -> StatusCodeEnum
Description	Writes the value of an MI input register.
Request Parameters	index : int MI register number to write value : int MI register numeric value to write
Return Value	StatusCodeEnum: 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 and MI register
"""

from Agilebot import Arm, StatusCodeEnum

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

```python
# [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: {re
t.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

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

4.7 Trajectory Control

Overview

The Trajectory module offers a full-cycle interface for offline trajectory execution, trajectory/path recording, conversion, and replay, supporting complex trajectory customization and reproduction.

Core Features

- Support for setting and executing offline trajectory files
- Support for moving the robot to the offline trajectory starting point at a safe speed
- Support for CSV to trajectory format file conversion
- Support for trajectory/path recording, playback, deletion, and management
- Support for obtaining trajectory lists and starting positions
- Support for path planner parameter configuration and query
- Support for motion along trajectories/paths

Use Cases

- Implement precise reproduction of complex trajectories
- Record and playback robot motion trajectories
- Convert external trajectory data to robot-executable format
- Customize path planning parameters to optimize motion performance
- Batch manage and query trajectory/path files

4.7.1 Set the Offline Trajectory File to Execute

Method Name	trajectory.set_offline_trajectory_file(`path` : str) -> StatusCodeEnum
Description	Sets the offline trajectory file to execute.
Request Parameters	`path` : str Offline trajectory file name (see Notes for format).
Return Value	StatusCodeEnum: Result of the function execution.
Notes	Trajectory file is plain text: - Line 1: `6` (axes), `0.001` (time interval, s), `8093` (point count). - Line 2: initial positions of 6 axes. - Lines 3-8095: trajectory points incl. position/velocity/acceleration/torque feedforward/ `do_port` / `do_state` . - `do_port` range 1-24; `-1` means no IO trigger; `do_port = 1` with `do_state = 1` sets `do1` ON, `do_state = 0` sets `do1` OFF.
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	trajectory.prepare_offline_trajectory() -> StatusCodeEnum
Description	Moves the robot to the starting point of the offline trajectory at a safe speed.
Request Parameters	None
Return Value	StatusCodeEnum: 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	trajectory.execute_offline_trajectory() -> StatusCodeEnum
Description	Starts executing the offline trajectory program.
Request Parameters	None
Return Value	StatusCodeEnum: 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	trajectory.transform_csv_to_trajectory(`file_name` : str, `separator` : str = " ", `io_flag` : str = "2")
Description	Converts a trajectory CSV file to the controller trajectory format and stores it in the controller's trajectory directory.
Request Parameters	`file_name` : str CSV trajectory file name (without `.csv`). `separator` : str Delimiter (space or comma; space -> `.trajectory` , comma -> `.csv`). `io_flag` : str IO source (1: defaults `do_port = -1` , `do_state = 0` ; 2: user IO).
Return Value	str: Path of the converted trajectory file StatusCodeEnum: 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	trajectory.check_transform_status(`file_name` : str) -> tuple[TransformStatusEnum, StatusCodeEnum]
Description	Queries the current status of the trajectory file conversion.

Method Name	trajectory.check_transform_status(`file_name` : str) -> tuple[TransformStatusEnum, StatusCodeEnum]
Request Parameters	`file_name` : str Trajectory file path (returned by `transform_csv_to_trajectory`).
Return Value	TransformStatusEnum: Conversion status StatusCodeEnum: 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

py

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

import time

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

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

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

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

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

# [ZH] 等待控制器到位
```

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

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

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

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

4.7.6 Start Recording Trajectory

Method Name	trajectory.trajectory_record_begin(`name` : str) -> StatusCodeEnum
Description	Instructs the robot to start recording a trajectory.
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: 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	trajectory.trajectory_record_finish(`name` : str) -> StatusCodeEnum
Description	Instructs the robot to stop recording a trajectory.

Method Name	trajectory.trajectory_record_finish(`name` : str) -> StatusCodeEnum
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: 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	trajectory.trajectory_replay_start(`name` : str) -> StatusCodeEnum
Description	Instructs the robot to start replaying a trajectory.
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.7.9 Stop Replaying Trajectory

Method Name	trajectory.trajectory_replay_stop(`name` : str) -> StatusCodeEnum
Description	Instructs the robot to stop replaying a trajectory.
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: 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	trajectory.trajectory_record_delete(`name` : str) -> StatusCodeEnum
Description	Deletes the specified trajectory stored in the robot.
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: 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 Recorded Trajectory List

Method Name	trajectory.get_trajectory_record_list() -> tuple[list[str], StatusCodeEnum]
Description	Retrieves the list of recorded trajectories.
Parameters	None
Return	[list[str]]: list of trajectory program names. StatusCodeEnum: 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 Recorded Trajectory Starting Pose

Method Name	trajectory.get_trajectory_record_start_pose(`name` : str) -> tuple[MotionPose, StatusCodeEnum]
Description	Retrieves the starting pose of a recorded trajectory.

Method Name	trajectory.get_trajectory_record_start_pose(`name` : str) -> tuple[MotionPose, StatusCodeEnum]
Parameters	`name` : trajectory program name.
Return	MotionPose: Starting pose of the trajectory. StatusCodeEnum: 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

py

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

import time

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

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

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

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

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

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

# [ZH] 结束记录轨迹
# [EN] End recording trajectory
ret = arm.trajectory.trajectory_record_finish("test")
if ret == StatusCodeEnum.OK:
```

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

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

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

# [ZH] 获取轨迹起始位姿
# [EN] Get trajectory start pose
pose, ret = arm.trajectory.get_trajectory_record_start_pose(
    "test"
)
if ret == StatusCodeEnum.OK:
    print(
        "获取轨迹起始位姿成功 / Get trajectory start pose successful"
    )
else:
```

```
    print(
        f"获取轨迹起始位姿失败，错误代码 / Get trajectory start pose failed, erro
r code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

```
        print(
            f"获取伺服状态失败，错误代码 / Get servo status failed, error code:
{ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
    print(f"伺服状态 / Servo status: {servo_status}")
    if (
        robot_status == RobotStatusEnum.ROBOT_IDLE
        and servo_status == ServoStatusEnum.SERVO_IDLE
    ):
        break
    time.sleep(2)

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

time.sleep(5)

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

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

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

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

4.7.13 Start Recording Trajectory / Path

Method Name	trajectory.path_record_begin(`name` : str, `comment` : str, `param` : float, `angle` : float = 1) -> StatusCodeEnum
Description	Starts recording a trajectory table (*.traj*) or a path table (*.path*).
Parameters	`name` : trajectory / path file name, must end with *.traj* or *.path*. `comment` : description for the trajectory / path. `param` : - for *.path* tables: linear-motion distance threshold (mm). - for *.traj* tables: recording interval (ms).

Method Name	trajectory.path_record_begin(`name` : str, `comment` : str, `param` : float, `angle` : float = 1) -> StatusCodeEnum
	`angle` : rotational-angle threshold (°), only effective when `name` is a *.path* file.
Return	StatusCodeEnum: execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.14 Stop Recording Trajectory / Path

Method Name	trajectory.path_record_finish() -> StatusCodeEnum
Description	Stops the current trajectory / path table recording.
Parameters	none
Return	StatusCodeEnum: execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.15 Get Starting Pose of a Trajectory / Path

Method Name	trajectory.get_path_start_pose(`name` : str) -> tuple[MotionPose, StatusCodeEnum]
Description	Retrieves the starting pose of the specified trajectory / path file.
Parameters	`name` : trajectory / path file name.
Return	`MotionPose` : Starting pose. StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.16 Get Status of Trajectories / Paths

Method Name	trajectory.get_path_state(`path_list` : list[str]) -> tuple[dict[str, int], StatusCodeEnum]
Description	Batch-query the current status of trajectory / path files.
Parameters	`path_list` : list of trajectory / path file names to query.
Return	`dict[str, int]` : mapping of file name → status code. StatusCodeEnum: execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.17 Set Path-Planner Parameters

Method Name	trajectory.set_path_planner_parameter(`transition_time` : float, `scaling_factor` : float) -> StatusCodeEnum
Description	Configures path-planner parameters affecting motion smoothness and speed / acceleration distribution.
Parameters	`transition_time` : blending time (0~1; larger = smoother accel/decel). `scaling_factor` : path parameter *s* weight (0~1; 0: uniform time scaling, 1: max-displacement scaling, 0.5: equal-accel, 0.33: equal-jerk).
Return	StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.18 Get Path-Planner Parameters

Method Name	trajectory.get_path_planner_parameter() -> tuple[float, float, StatusCodeEnum]
Description	Reads the current path-planner parameters.
Parameters	none
Return	`transition_time` : blending time, 0–1. `scaling_factor` : redistribution weight, 0–1. StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

4.7.19 Move Along a Trajectory / Path

Method Name	trajectory.move_path(`name` : str, `vel` : float = 100, `acc` : float = 1) -> StatusCodeEnum
Description	Commands the robot end-effector to move along the specified trajectory / path file.
Parameters	`name` : trajectory/path file name. `vel` : TCP speed (0~5000 mm/s). `acc` : acceleration multiplier (0~1.2).
Return	StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.I.0+ Industrial (Bronze): not supported

Example Code

trajectory/path_record.py

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

```
"""

import time

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

# [ZH] 初始化机械臂并连接到指定IP地址
# [EN] Initialize the robotic arm and connect to the specified IP address
arm = Arm()
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.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 and overwrite option
ret = arm.trajectory.path_record_begin(
    "test_path.path", "Path Test", 10, 1
)
if ret == StatusCodeEnum.OK:
    print("开始路径记录成功 / Start path record successful")
else:
    print(
        f"开始路径记录失败，错误代码 / Start path record failed, error code: {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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)
if state["test_path.path"] == 0:
    print("路径表记录中 / Path table recording in progress")

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

```
    arm.disconnect()
    exit(1)
# 等待3秒，让机械臂持续运动
time.sleep(2)
# 停止机械臂运动
arm.jogging.stop()
time.sleep(2)

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

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

# [ZH] 获取记录轨迹的起始位置姿态
# [EN] Get the starting position pose of the recorded trajectory
```

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

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

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

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

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

```
    exit(1)

# [ZH] 获取轨迹规划参数以验证设置是否成功
# [EN] Get trajectory planning parameters to verify if the setting is succes
sful
param1, param2, ret = (
    arm.trajectory.get_path_planner_parameter()
)
if ret == StatusCodeEnum.OK:
    print(
        "获取路径规划参数成功 / Get path planner parameter successful"
    )
else:
    print(
        f"获取路径规划参数失败，错误代码 / Get path planner parameter failed, err
or code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

4.8 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.

Core Features

- Support for alarm reset
- Support for getting all active alarms
- Support for getting the highest-priority alarm
- Support for specifying alarm output language

Use Cases

- Monitor robot operation status and detect abnormalities in time
- Handle errors and alarms during robot operation
- Record and analyze robot alarm history
- Implement automated alarm handling processes
- Integrate into robot monitoring systems

4.8.1 Reset Alarms

Method Name	alarm.reset() -> StatusCodeEnum
Description	Resets current errors/alarms.
Request Parameters	None
Return Value	StatusCodeEnum: Result of the function execution.

Method Name	alarm.reset() -> StatusCodeEnum
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	alarm.get_all_active_alarms(`language` : LanguageType) -> tuple[list, StatusCodeEnum]
Description	Gets all currently active alarms.
Request Parameters	`language` : LanguageType Output language (`LanguageType.English` or `LanguageType.Chinese`).
Return Value	list: Active alarm entries. StatusCodeEnum: 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	alarm.get_top_alarm() -> tuple[ROBOT_ALARM, StatusCodeEnum]
Description	Gets the alarm with the highest priority.
Request Parameters	None
Return Value	`ROBOT_ALARM` : Highest-priority alarm. StatusCodeEnum: 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
#!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 co
de: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

4.9 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, supporting management and operations for multiple file types.

Core Features

- Support for uploading local files to the robot controller
- Support for downloading robot files to local directories
- Support for deleting files from the robot controller
- Support searching files by filename pattern
- Support for managing multiple file types (USER_PROGRAM, BLOCK_PROGRAM, TRAJECTORY, ROBOT_TMP)
- Support for overwrite control during upload/download

Use Cases

- Deploy programs to the robot controller
- Back up programs and trajectory files on the robot
- Synchronize debug data from production lines
- Batch manage and query robot files
- Upload temporary data files to the robot
- Download robot logs or output files to local

4.9.1 Upload a Local File to the Robot

Method Name	file_manager.upload(`file_path` : str, `file_type` : str, `overwriting` : bool = False) -> StatusCodeEnum
Description	Uploads a local file to the robot controller.
Request Parameters	`file_path` : str Local file absolute path. `file_type` : str File type (USER_PROGRAM: `file_path` is the program name path; uploads `json` / `xml` from the same directory; BLOCK_PROGRAM: `file_path` is the block program name path; uploads `block` / `json` / `xml` from the same directory; TRAJECTORY: `file_path` is full trajectory file path; ROBOT_TMP: `file_path` is full temp file path). `overwriting` : bool Overwrite same-name file (default `False`).
Return Value	StatusCodeEnum: Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	For `USER_PROGRAM` and `BLOCK_PROGRAM` , specify only the program name (i.e., filename without extension).

4.9.2 Download a Robot File to Local

Method Name	file_manager.download(`file_name` : str, `file_path` : str, `file_type` : str, `overwriting` : bool=False) -> StatusCodeEnum
Description	Downloads a file from the robot to a local directory.
Request Parameters	`file_name` : str File name. `file_path` : str Local save directory. `file_type` : str File type (`BLOCK_PROGRAM` not supported for download). `overwriting` : bool Overwrite same-name file.
Return Value	StatusCodeEnum: Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Method Name	file_manager.download(`file_name` : str, `file_path` : str, `file_type` : str, `overwriting` : bool=False) -> StatusCodeEnum
Note	For `USER_PROGRAM` , specify only the program name (i.e., filename without extension); the system downloads the matching `.json` / `.xml` . For `TRAJECTORY` , provide the full filename with `.trajectory` . For `ROBOT_TMP` , provide the full filename with suffix (e.g., `.csv`).

4.9.3 Delete a File on the Robot

Method Name	file_manager.delete(`file_name` : str, `file_type` : str) -> StatusCodeEnum
Description	Deletes a file from the robot controller.
Request Parameters	`file_name` : str File name. `file_type` : str File type.
Return Value	StatusCodeEnum: Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	For `USER_PROGRAM` / `BLOCK_PROGRAM` , specify only the program name (i.e., filename without extension). For `TRAJECTORY` / `ROBOT_TMP` , provide the full filename with suffix.

4.9.4 Search Files by Filename Pattern

Method Name	file_manager.search(`pattern` : str, `file_list` : list) -> StatusCodeEnum
Description	Searches for files on the controller that match the filename pattern.
Request Parameters	`pattern` : str Filename match pattern. `file_list` : list Output list (receives matching filenames).

Method Name	file_manager.search(pattern : str, file_list : list) -> StatusCodeEnum
Return Value	StatusCodeEnum: 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

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

from pathlib import Path

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

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

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

# [ZH] 上传其他文件
# [EN] Upload other files
tmp_file_path = path.replace("file_manager.py", "test.csv")
ret = file_manager.upload(tmp_file_path, ROBOT_TMP, True)
```

```
if ret == StatusCodeEnum.OK:
    print("文件上传成功 / File upload successful")
else:
    print(
        f"文件上传失败，错误代码 / File upload failed, error code: {ret.errms
g}"
    )
    exit(1)

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

# [ZH] 上传轨迹
# [EN] Upload a trajectory
trajectory_file_path = path.replace(
    "file_manager.py", "test_torque.trajectory"
)
ret = file_manager.upload(
    trajectory_file_path, TRAJECTORY, True
)
if ret == StatusCodeEnum.OK:
    print("轨迹上传成功 / Trajectory upload successful")
else:
    print(
        f"轨迹上传失败，错误代码 / Trajectory upload failed, error code: {ret.e
rrmsg}"
    )
    exit(1)
```

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

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

# [ZH] 删除文件
# [EN] Delete a file
ret = file_manager.delete(
    "test_torque.trajectory", TRAJECTORY
)
if ret == StatusCodeEnum.OK:
    print("文件删除成功 / File delete successful")
else:
    print(
        f"文件删除失败，错误代码 / File delete failed, error code: {ret.errms
```

```
g}"
    )
    exit(1)
```

4.10 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.

Core Features

- Support for obtaining user/tool coordinate system summary information lists
- Support for adding, deleting, updating, and querying user/tool coordinate systems
- Support for calculating user/tool coordinate systems based on multiple input poses
- Provides a unified coordinate system management interface for maintaining the controller's frame list
- Support for quickly solving new coordinate systems from taught points

Use Cases

- Maintain consistent spatial reference when debugging multi-station setups
- Manage coordinate systems for different tools when switching tools
- Batch manage and query robot coordinate systems
- Automatically calculate new coordinate systems from taught points
- Implement coordinate system switching in complex tasks

4.10.1 Get User/Tool Coordinate List

Method Name	coordinate_system.UF/TF.get_coordinate_list() -> tuple[List[Coordinate], StatusCodeEnum]
Description	Get the summary list of user/tool coordinate systems.
Request Parameters	None

Method Name	coordinate_system.UF/TF.get_coordinate_list() -> tuple[List[Coordinate], StatusCodeEnum]
Return Value	UserCoordSummaryList: A list of coordinate system summary information. StatusCodeEnum: 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 User/Tool Coordinate System

Method Name	coordinate_system.UF/TF.add(`coordinate` : Coordinate) -> StatusCodeEnum
Description	Add a coordinate system to the current user/tool coordinate set.
Request Parameters	`coordinate` : Coordinate The coordinate system information to add.
Return Value	StatusCodeEnum: 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 User/Tool Coordinate System

Method Name	coordinate_system.UF/TF.delete(`index` : int) -> StatusCodeEnum
Description	Delete a coordinate system from the current user/tool coordinate set.
Request Parameters	`index` : int The coordinate system index.
Return Value	StatusCodeEnum: Result of the function execution.

Method Name	coordinate_system.UF/TF.delete(`index` : int) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.10.4 Update User/Tool Coordinate System

Method Name	coordinate_system.UF/TF.update(`coordinate` : Coordinate) -> StatusCodeEnum
Description	Update a coordinate system in the current user/tool coordinate set.
Request Parameters	`coordinate` : Coordinate The updated coordinate system information.
Return Value	StatusCodeEnum: Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.10.5 Get User/Tool Coordinate System

Method Name	coordinate_system.UF/TF.get(`index` : int) -> tuple[Coordinate, StatusCodeEnum]
Description	Get a specified coordinate system from the current user/tool coordinate set.
Request Parameters	`index` : int The coordinate system index.
Return Value	Coordinate: Coordinate system information. StatusCodeEnum: 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 System

Method Name	coordinate_system.UF/TF.calculate(pose: List[Position]) -> tuple[Position, StatusCodeEnum]
Description	Calculate user/tool coordinate information based on the input poses and return the calculated pose.
Request Parameters	`pose` : List[Position] List of input poses.
Return Value	Position: Calculated pose information. StatusCodeEnum: Result of the function execution.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

coordinate_system.py

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

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

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

# [ZH] 连接捷勃特机器人
# [EN] Connect to the robot
ret = arm.connect("10.27.1.254")
```

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

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

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

```
        69.714,
        133.597,
    ),
]

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

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

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

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

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

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

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

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

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

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

```
print(f"  A: {tf.data.a}, B: {tf.data.b}, C: {tf.data.c}")

# [ZH] 更新工具坐标系的名称
# [EN] Update tool coordinate system name
tf.name = "updated_test_tf"
ret = arm.coordinate_system.TF.update(tf)
if ret != StatusCodeEnum.OK:
    print(
        f"更新工具坐标系失败，错误代码 / Update tool coordinate system failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

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

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

4.11 Modbus 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, facilitating linkage with PLCs, I/O expansion modules, and other Modbus devices.

Core Features

- Support for obtaining Modbus slave instances by channel and slave ID
- Support for reading and writing Modbus coil registers from slaves
- Support for reading and writing Modbus holding registers from slaves
- Support for reading Modbus discrete input registers from slaves
- Support for reading Modbus input registers from slaves
- Support for setting and querying serial communication parameters

Use Cases

- Data exchange with PLC devices
- Control of I/O expansion modules
- Implementation of linkage between robots and other Modbus devices
- Configuration and management of Modbus serial communication parameters

Notes

Modbus functions conflict with the bus configuration on the robot software and cannot be used simultaneously.

4.11.1 Get Modbus Slave Instance

Method Name	modbus.get_slave(`channel` : ModbusChannel, `slave_id` : int, `master_id` : int = 0) -> 'Modbus.Slave'
Description	Get a Modbus slave instance for the specified channel and slave ID.
Request Parameters	`channel` : ModbusChannel Modbus channel `slave_id` : int Slave ID `master_id` : int Master ID (default 0)
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	slave.read_coils(`address` : int, `number` : int) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus coil registers from a slave.
Request Parameters	`address` : int Register address `number` : int Register count (max 120)
Return Value	list[int]: Register values StatusCodeEnum: 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	slave.write_coils(`address` : int, `value` : list[int]) -> StatusCodeEnum
Description	Write to Modbus coil registers of a slave.

Method Name	slave.write_coils(`address` : int, `value` : list[int]) -> StatusCodeEnum
Request Parameters	`address` : int Register address `value` : list[int] Register values
Return Value	StatusCodeEnum: 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	slave.read_holding_regs(`address` : int, `number` : int) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus holding registers from a slave.
Request Parameters	`address` : int Register address `number` : int Register count (max 120)
Return Value	list[int]: Register values StatusCodeEnum: 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	slave.write_holding_regs(`address` : int, `value` : list[int]) -> StatusCodeEnum
Description	Write to Modbus holding registers of a slave.
Request Parameters	`address` : int Register address `value` : list[int] Register values

Method Name	slave.write_holding_regs(`address` : int, `value` : list[int]) -> StatusCodeEnum
Return Value	StatusCodeEnum: 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	slave.read_discrete_inputs(`address` : int, `number` : int) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus discrete input registers from a slave.
Request Parameters	`address` : int Register address `number` : int Register count (max 120)
Return Value	list[int]: Register values StatusCodeEnum: Result of the function execution
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	slave.read_input_regs(`address` : int, `number` : int) -> tuple[list[int], StatusCodeEnum]
Description	Read Modbus input registers from a slave.
Request Parameters	`address` : int Register address `number` : int Register count (max 120)
Return Value	list[int]: Register values StatusCodeEnum: Result of the function execution

Method Name	slave.read_input_regs(address : int, number : int) -> tuple[list[int], StatusCodeEnum]
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.e
rrmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置modbus参数
```

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

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

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

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

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

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

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

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

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

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

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

4.11.8 Set Serial Communication Parameters

Method Name	modbus.set_parameter(`param` : SerialParams) -> tuple[int, StatusCodeEnum]
Description	Set serial communication parameters.
Request Parameters	`param` : SerialParams Serial communication parameters
Return Value	int: Channel ID StatusCodeEnum: 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	modbus.get_parameter(`channel` : ModbusChannel, `master_id` : int = 1) -> tuple[SerialParams, StatusCodeEnum]
Description	Get serial communication parameters.
Request Parameters	`channel` : ModbusChannel Modbus channel `master_id` : int Master ID (default 1)
Return Value	SerialParams: Serial communication parameters StatusCodeEnum: 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 robot in a structured way from the host PC, covering motion, logic, program calls, communication, vision, and Modbus commands.

Core Features

- Support for building motion instructions (MoveJoint, MoveLine, MoveCircle, etc.)
- Support for building logic instructions (If, While, Switch, Goto, etc.)
- Support for building assignment, wait, pause, and abort instructions
- Support for building program call and management instructions
- Support for building Socket communication instructions
- Support for building Modbus register read/write instructions
- Support for building vision program instructions
- Support for setting extra parameters (acceleration, RTCP, frame offset, etc.)
- Support for quick motion instructions (JUMP series)

Use Cases

- Automatically generate complex robot programs
- Edit and modify robot programs
- Reuse common program modules and instruction sequences
- Achieve seamless integration between host PC and robot programs
- Build customized robot motion trajectories
- Integrate vision, communication, and Modbus functions into robot programs

Constructor

Method Name	BasScript(`name` : str) -> BasScript
Description	Construct BAS instruction sequences for the robot
Request Parameters	`name` : Script name
Compatible Robot Software Versions	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): v7.6.0.0+
Notes	All methods under this class have the same compatibility requirements as this class

Subclass Structure

BasScript class contains the following subclasses to organize different types of instructions:

1. **ExtraParam**: Extra parameter class for building complex robot control commands
2. **BasMotion**: Motion instruction subclass containing all robot motion-related methods
3. **BasLogical**: Logic instruction subclass containing all logic control-related methods
4. **BasStructure**: Structure instruction subclass containing all structure control-related methods
5. **BasSocket**: Socket communication subclass containing all Socket communication-related methods
6. **BasModbus**: Modbus communication subclass containing all Modbus communication-related methods
7. **BasVision**: Vision instruction subclass containing all vision-related methods

4.12.1 ExtraParam

Method Name	ExtraParam() -> ExtraParam
Description	Extra parameter class for building complex robot control commands
Request Parameters	None
Return Value	ExtraParam object

4.12.1.1 Set Acceleration

Method Name	extra_param.acceleration(value : float) -> None
Description	Set acceleration, range is 1~120%
Request Parameters	value : float Acceleration value, unit: % (float)
Return Value	None

4.12.1.2 Set RTCP Parameter

Method Name	extra_param.rtcp() -> None
Description	Set RTCP parameter, only supports MoveL and MoveC instructions
Request Parameters	None
Return Value	None

4.12.1.3 Set Frame Offset

Method Name	extra_param.offset(index : int) -> None
Description	Set frame offset
Request Parameters	index : int Offset index (integer)
Return Value	None

4.12.1.4 Set Time-Based Operation or Assignment

Method Name	extra_param.tb(second : int, type : str, name : str = None, index : int = None, status : str = None) -> None
Description	Set time-based operation or assignment, range is 0.01-30s, values less than 0.01 will not be saved successfully
Request Parameters	second : int Seconds, unit: sec (integer) type : str Execution type (string) name : str Name (for operation, string)

Method Name	extra_param.tb(second : int, type : str, name : str = None, index : int = None, status : str = None) -> None
	index : int Index (for assignment, integer) status : str Status (for assignment, string)
Return Value	None

4.12.1.5 Set Jump

Method Name	extra_param.skip(index : int) -> None
Description	Set jump, when the jump condition set by the SKIP CONDITION instruction is met, jump directly from the current line containing the SKIP instruction to the target instruction line or target program
Request Parameters	index : int Index of the target label (integer)
Return Value	None

4.12.1.6 Set Departure Distance and Approaching Distance

Method Name	extra_param.approach(departure_dist : float, approaching_dist : float) -> None
Description	Set departure distance and approaching distance for door-type motion, only used for JUMP instructions
Request Parameters	departure_dist : float Departure distance, unit: mm (float) approaching_dist : float Approaching distance, unit: mm (float)
Return Value	None

4.12.2 BasMotion

4.12.2.1 MoveJoint Motion to Point Instruction

Method Name	motion.move_joint(pose_type : MovePoseType, pose_index : int, speed_type : SpeedType, speed_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Joint motion instruction, moves the robot to a specified position in the workspace with the specified movement speed and method
Request Parameters	pose_type : MovePoseType Pose type pose_index : int Pose index speed_type : SpeedType Speed type speed_value : float Speed value, unit: % smooth_type : SmoothType Smooth type smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters
Return Value	StatusCodeEnum: Function execution result

4.12.2.2 MoveLine Linear Motion to Point Instruction

Method Name	motion.move_line(pose_type : MovePoseType, pose_index : int, speed_type : SpeedType, speed_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Linear motion instruction, moves the robot in a straight line to a specified position in the workspace with the specified movement speed and method
Request Parameters	pose_type : MovePoseType Pose type pose_index : int Pose index speed_type : SpeedType Speed type speed_value : float Speed value, unit: mm/s smooth_type : SmoothType Smooth type smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters
Return Value	StatusCodeEnum: Function execution result

4.12.2.3 MoveCircle Arc Motion to Point Instruction

Method Name	motion.move_circle(pose_type1 : MovePoseType, pose_index1 : int, pose_type2 : MovePoseType, pose_index2 : int, speed_type : SpeedType, speed_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Arc motion instruction, moves the robot in an arc to a specified position in the workspace with the specified movement speed and method
Request Parameters	pose_type1 : MovePoseType First pose type pose_index1 : int First pose index pose_type2 : MovePoseType Second pose type pose_index2 : int Second pose index speed_type : SpeedType Speed type speed_value : float Speed value, unit: mm/s smooth_type : SmoothType Smooth type smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters
Return Value	StatusCodeEnum: Function execution result

4.12.2.4 JUMP Quick Motion Instruction

Method Name	motion.move_jump(pose_type : MovePoseType, pose_index : int, speed_value : float, speed_ratio : float, lim_Z_type : SpeedType, lim_Z_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Door-type motion instruction, specifies the robot to perform door-type motion (first vertical ascent, then horizontal movement, finally vertical descent)
Request Parameters	pose_type : MovePoseType Target pose storage type pose_index : int Index of the target position speed_value : float Movement speed value, unit: mm/s speed_ratio : float Movement speed ratio, unit: % lim_Z_type : SpeedType Z-axis limit type lim_Z_value : float Z-axis limit value smooth_type : SmoothType Smooth type smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters

Method Name	motion.move_jump(pose_type : MovePoseType, pose_index : int, speed_value : float, speed_ratio : float, lim_Z_type : SpeedType, lim_Z_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Return Value	StatusCodeEnum: Function execution result

4.12.2.5 JUMP3 Quick Motion Instruction

Method Name	motion.move_jump3(pose_type : MovePoseType, pose_index : List[int], speed_value : float, speed_ratio : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Door-type motion instruction, specifies the robot to perform door-type motion with three positions: departure, approach, and target
Request Parameters	pose_type : MovePoseType Target pose storage type pose_index : List[int] List of 3 target position indices (departure, approach, target) speed_value : float Movement speed value, unit: mm/s speed_ratio : float Movement speed ratio, unit: % smooth_type : SmoothType Smooth type smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters
Return Value	StatusCodeEnum: Function execution result

4.12.2.6 JUMP3CP Quick Motion Instruction

Method Name	motion.move_jump3cp(pose_type : MovePoseType, pose_index : List[int], speed_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
Description	Door-type motion instruction, specifies the robot to perform door-type motion with three positions: departure, approach, and target
Request Parameters	pose_type : MovePoseType Target pose storage type pose_index : List[int] List of 3 target position indices (departure, approach, target) speed_value : float Movement speed value, unit: mm/s smooth_type : SmoothType Smooth type

Method Name	motion.move_jump3cp(pose_type : MovePoseType, pose_index : List[int], speed_value : float, smooth_type : SmoothType, smooth_distance : float = 0.0, extra_param : ExtraParam = None) -> StatusCodeEnum
	smooth_distance : float Smooth distance, unit: mm, range: 0~1000 extra_param : ExtraParam Additional parameters
Return Value	StatusCodeEnum: Function execution result

4.12.3 BasLogical

4.12.3.1 LogiIf Conditional Instruction

Method Name	logical.logi_if(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Description	Conditional judgment instruction, executes different code blocks according to the specified condition
Request Parameters	param1 : Union[RegisterType, IOType] First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[RegisterType, IOType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, str, IOStatus] Index or value of parameter 2 operator : BooleanOperator Logical operator
Return Value	StatusCodeEnum: Function execution result

4.12.3.2 LogiElseIf Conditional Branch Instruction

Method Name	logical.logi_else_if(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Description	Conditional branch instruction, judges whether the ElseIf condition is met when the If condition is not met

Method Name	logical.logi_else_if(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Request Parameters	param1 : Union[RegisterType, IOType] First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[ValueType, IOType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, IOStatus] Index or value of parameter 2 operator : BooleanOperator Logical operator
Return Value	StatusCodeEnum: Function execution result

4.12.3.3 LogiElse Else Instruction

Method Name	logical.logi_else() -> StatusCodeEnum
Description	Else instruction, executes when all If and ElseIf conditions are not met
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.4 LogiEndIf End Conditional Instruction

Method Name	logical.logi_end_if() -> StatusCodeEnum
Description	End conditional instruction, used to end the If condition statement block
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.5 LogiWhile Loop Instruction

Method Name	logical.logi_while(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Description	Loop instruction, repeatedly executes the code in the loop body when the condition is met
Request Parameters	param1 : Union[RegisterType, IOType] First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[RegisterType, IOType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, str, IOStatus] Index or value of parameter 2 operator : BooleanOperator Logical operator
Return Value	StatusCodeEnum: Function execution result

4.12.3.6 LogiEndWhile End Loop Instruction

Method Name	logical.logi_end_while() -> StatusCodeEnum
Description	End loop instruction, used to end the While loop statement block
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.7 LogiSwitch Multi-branch Selection Instruction

Method Name	logical.logi_switch(param : Union[RegisterType, IOType], index : int) -> StatusCodeEnum
Description	Multi-branch selection instruction, selects different branches to execute according to the value of the expression, supports BREAK statement to exit the branch
Request Parameters	param : Union[RegisterType, IOType] Register or IO signal index : int Index number
Return Value	StatusCodeEnum: Function execution result

4.12.3.8 LogiCase Branch Instruction

Method Name	logical.logi_case(param : Union[RegisterType, IOType, OtherType], value : Union[int, float, str]) -> StatusCodeEnum
Description	Branch instruction, used to define branch conditions in Switch statements
Request Parameters	param : Union[RegisterType, IOType, OtherType] Register, IO signal, or other type value : Union[int, float, str] Index number or value
Return Value	StatusCodeEnum: Function execution result

4.12.3.9 LogiDefault Default Branch Instruction

Method Name	logical.logi_default() -> StatusCodeEnum
Description	Default branch instruction, executes when all Case conditions are not met
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.10 LogiEndSwitch End Multi-branch Selection Instruction

Method Name	logical.logi_end_switch() -> StatusCodeEnum
Description	End multi-branch selection instruction, used to end the Switch statement block
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.11 LogiGoto Jump Instruction

Method Name	logical.logi_goto(index : int) -> StatusCodeEnum
Description	Jump instruction, used to jump the program to a specified label position within the same program and continue executing from that position; must insert the LABEL instruction first, then use the GOTO instruction

Method Name	logical.logi_goto(index : int) -> StatusCodeEnum
Request Parameters	index : int Index of the target label
Return Value	StatusCodeEnum: Function execution result

4.12.3.12 LogiLabel Label Instruction

Method Name	logical.logi_label(index : int) -> StatusCodeEnum
Description	Label instruction, used to define jump target positions in the program, GOTO instructions can jump to this label position
Request Parameters	index : int Label index
Return Value	StatusCodeEnum: Function execution result

4.12.3.13 LogiSkipCondition Skip Condition Instruction

Method Name	logical.logi_skip_condition(param1 : Union[RegisterType, IOType], index : int, param2 : Union[RegisterType, IOType, OtherType], value : Union[int, float, str, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Description	Skip condition instruction, used to set jump conditions, when the condition is met, jump directly from the current line containing the SKIP instruction to the target instruction line or target program
Request Parameters	param1 : Union[RegisterType, IOType] First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[RegisterType, IOType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, str, IOStatus] Index or value of parameter 2 operator : BooleanOperator Logical operator
Return Value	StatusCodeEnum: Function execution result

4.12.3.14 LogiBreak Break Loop Instruction

Method Name	logical.logi_break() -> StatusCodeEnum
Description	Break loop instruction, used to exit the current loop or Switch statement
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.3.15 LogiContinue Skip Loop Instruction

Method Name	logical.logi_continue() -> StatusCodeEnum
Description	Skip loop instruction, used to skip the remaining part of the current loop and enter the next loop
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.4 BasStructure

4.12.4.1 Wait Wait Condition Instruction

Method Name	structure.wait(param1 : Union[RegisterType, IOType], index : int, param2 : Union[ValueType, IOType, OtherType], value : Union[int, float, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Description	Wait condition instruction, executes the WAIT instruction only when the condition is met, otherwise waits until the condition is met
Request Parameters	param1 : Union[RegisterType, IOType] First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[ValueType, IOType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, IOStatus] Index or value of parameter 2 operator : BooleanOperator Logical operator

Method Name	structure.wait(param1 : Union[RegisterType, IOType], index : int, param2 : Union[ValueType, IOType, OtherType], value : Union[int, float, IOStatus], operator : BooleanOperator = BooleanOperator.EQ) -> StatusCodeEnum
Return Value	StatusCodeEnum: Function execution result

4.12.4.2 WaitTime Wait Time Instruction

Method Name	structure.wait_time(param : ValueType, value : Union[int, float]) -> StatusCodeEnum
Description	Wait time instruction, executes the WAIT TIME instruction, the robot waits for the specified time before continuing to execute subsequent instructions
Request Parameters	param : ValueType Parameter type (R register or value) value : Union[int, float] Time value, unit: sec
Return Value	StatusCodeEnum: Function execution result

4.12.4.3 Pause Pause Instruction

Method Name	structure.pause() -> StatusCodeEnum
Description	Pause instruction, when executing to the Pause instruction, pauses the program execution, the robot immediately plans a deceleration trajectory and stops, the program enters the pause state. To continue execution, you need to press the start button again
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.4.4 Abort Abort Instruction

Method Name	structure.abort() -> StatusCodeEnum
Description	Force end instruction: Ends program execution, the robot servo immediately brakes, the brake is applied, and the servo bus is powered off. After executing the force end instruction, the program enters the terminated state, and the cursor stops at the current line
Request Parameters	None
Return Value	StatusCodeEnum: Function execution result

4.12.4.5 Call Synchronous Program Call Instruction

Method Name	structure.call(`name` : str) -> StatusCodeEnum
Description	Synchronous program call instruction, calls the specified program and waits for its execution to complete before continuing to execute the current program, supports calling preset BAS programs
Request Parameters	`name` : str Program name
Return Value	StatusCodeEnum: Function execution result

4.12.4.6 Run Asynchronous Program Call Instruction

Method Name	structure.run(`name` : str) -> StatusCodeEnum
Description	Asynchronous program call instruction, calls the specified program and returns immediately, continuing to execute the current program, supports calling preset BAS programs
Request Parameters	`name` : str Program name
Return Value	StatusCodeEnum: Function execution result

4.12.4.7 Load Load Program Instruction

Method Name	structure.load(`name` : str) -> StatusCodeEnum
Description	Load program instruction, loads the specified program into memory
Request Parameters	`name` : str Program name
Return Value	StatusCodeEnum: Function execution result

4.12.4.8 Unload Unload Program Instruction

Method Name	structure.unload(`name` : str) -> StatusCodeEnum
Description	Unload program instruction, unloads the specified program from memory
Request Parameters	`name` : str Program name
Return Value	StatusCodeEnum: Function execution result

4.12.4.9 Exec Execute Program Instruction

Method Name	structure.exec(`name` : str) -> StatusCodeEnum
Description	Execute program instruction, executes the specified program
Request Parameters	`name` : str Program name
Return Value	StatusCodeEnum: Function execution result

4.12.5 BasSocket

4.12.5.1 SocketOpen Open Socket Connection Instruction

Method Name	socket.socket_open(`index` : int) -> StatusCodeEnum
Description	Use Socket Open instruction to create a Server and wait for Client connection
Request Parameters	`index` : int SK register sequence number

Method Name	socket.socket_open(index : int) -> StatusCodeEnum
Return Value	StatusCodeEnum: Function execution result

4.12.5.2 SocketClose Close Socket Connection Instruction

Method Name	socket.socket_close(index : int) -> StatusCodeEnum
Description	Close the specified socket connection
Request Parameters	index : int SK register sequence number
Return Value	StatusCodeEnum: Function execution result

4.12.5.3 SocketConnect Connect Socket Instruction

Method Name	socket.socket_connect(index : int) -> StatusCodeEnum
Description	Use Socket Connect instruction to connect to the specified socket server
Request Parameters	index : int SK register sequence number
Return Value	StatusCodeEnum: Function execution result

4.12.5.4 SocketSend Send Socket Data Instruction

Method Name	socket.socket_send(index : int, msg_type : StrType, value : Union[int, str]) -> StatusCodeEnum
Description	Use Socket Send instruction to send data to the specified socket connection
Request Parameters	index : int SK register sequence number msg_type : StrType Message type value : Union[int, str] Message content or sequence number
Return Value	StatusCodeEnum: Function execution result

4.12.5.5 SocketRecv Receive Socket Data Instruction

Method Name	socket.socket_recv(index : int, msg_lenth : int, msg_type : StrType, value : Union[int, str]) -> StatusCodeEnum
Description	Use Socket Recv instruction to read characters from the specified socket connection, can specify maximum length
Request Parameters	index : int SK register sequence number msg_lenth : int Maximum message length msg_type : StrType Message type value : Union[int, str] Message content or sequence number
Return Value	StatusCodeEnum: Function execution result

4.12.6 BasModbus

4.12.6.1 ModbusReadMH Read Modbus Holding Register Instruction

Method Name	modbus.modbus_read_MH(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum
Description	Read Modbus holding register instruction
Request Parameters	index : int Channel sequence number id : int Modbus ID address : int Register starting address length : int Register length, i.e., the number of registers to read r_index : int R register starting sequence number for writing results
Return Value	StatusCodeEnum: Function execution result

4.12.6.2 ModbusReadMI Read Modbus Input Register Instruction

Method Name	modbus.modbus_read_MI(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum
Description	Read Modbus input register instruction

Method Name	modbus.modbus_read_MI(index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum
Request Parameters	index : int Channel sequence number id : int Modbus ID address : int Register starting address length : int Register length, i.e., the number of registers to read r_index : int R register starting sequence number for writing results
Return Value	StatusCodeEnum: Function execution result

4.12.6.3 ModbusWriteMH Write Modbus Holding Register Instruction

Method Name	modbus.modbus_write_MH(index : int, id : int, address : int, length : int, value_type : ValueType, value : int) -> StatusCodeEnum
Description	Write Modbus holding register instruction
Request Parameters	index : int Channel sequence number id : int Modbus ID address : int Register starting address length : int Register length, i.e., the number of registers to write value_type : ValueType Value type value : int Value or R register starting index
Return Value	StatusCodeEnum: Function execution result

4.12.7 BasVision

4.12.7.1 VisionFind Find Vision Program Instruction

Method Name	vision.vision_find(name : str) -> StatusCodeEnum
Description	Execute vision find program
Request Parameters	name : str Vision program name
Return Value	StatusCodeEnum: Function execution result

4.12.7.2 VisionGetOffset Get Vision Program Offset Instruction

Method Name	vision.vision_get_offset(name : str, index : int, label_index : int) -> StatusCodeEnum
Description	Get the offset after executing the vision program
Request Parameters	name : str Vision program name index : int Vision register index label_index : int Label index
Return Value	StatusCodeEnum: Function execution result

4.12.7.3 VisionGetQuantity Get Vision Program Result Instruction

Method Name	vision.vision_get_quantity(name : str, index : int) -> StatusCodeEnum
Description	Get the result quantity after executing the vision program
Request Parameters	name : str Vision program name index : int R register index for storing results
Return Value	StatusCodeEnum: Function execution result

4.12.8 Direct Methods

4.12.8.1 SetParam Set Parameter Instruction

Method Name	bas_script.set_param(type : ParamType, value_type : ValueType, value : Union[int, float]) -> StatusCodeEnum
Description	Set parameter instruction, used to set various parameters of the robot
Request Parameters	type : ParamType Parameter type value_type : ValueType Value type value : Union[int, float] Value
Return Value	StatusCodeEnum: Function execution result

4.12.8.2 AssignValue Assignment Instruction

Method Name	bas_script.assign_value(param1 : AssignType, index : int, param2 : Union[AssignType, OtherType], value : Union[int, float, str, IOStatus, CurrentPose], opt_index : int = 0, opt_value = 0) -> StatusCodeEnum
Description	Assignment instruction, used to assign values to registers, IO signals, etc.
Request Parameters	param1 : AssignType First parameter (register or IO signal) index : int Index of parameter 1 param2 : Union[AssignType, OtherType] Second parameter (register, IO signal, or other type) value : Union[int, float, str, IOStatus, CurrentPose] Index or value of parameter 2 opt_index : int Additional index when parameter 1 is PR_ELEMENT opt_value : int Additional index when parameter 2 is PR_ELEMENT or pulse value when value is IOStatus.PULSE
Return Value	StatusCodeEnum: 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.

Core Features

- Support for robot single-axis step teaching motion
- Support for robot continuous jogging teaching motion
- Support for robot multi-axis continuous teaching motion
- Support for emergency stopping robot continuous motion
- Support for setting step length and rotation step angle
- Support for jogging control in different coordinate systems

Use Cases

- Robot debugging and teaching processes
- Position fine-tuning scenarios
- Host PC remote manual pose adjustment
- Robot installation and calibration
- Precise adjustment of complex trajectories

4.13.1 Single-axis Step Teaching Motion

Method Name	jogging.step_move(`aj_num` : int, `step_length` : float = 0, `step_angle` : float = 0) -> StatusCodeEnum
Description	Perform single-axis step teaching motion based on the configured step size.

Method Name	jogging.step_move(`aj_num` : int, `step_length` : float = 0, `step_angle` : float = 0) -> StatusCodeEnum
Request Parameters	`aj_num` : 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. `step_length` : float Linear step length in mm; omitted to keep the current step size. `step_angle` : float Rotational step angle in degrees; omitted to keep the current rotation step.
Return Value	StatusCodeEnum: 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

py

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

from Agilebot import Arm, StatusCodeEnum

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

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

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

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

4.13.2 Single-axis Continuous Jogging Teaching Motion

Method Name	jogging.continuous_move(`aj_num` : int) -> StatusCodeEnum
Description	Perform single-axis continuous jogging teaching motion.
Request Parameters	`aj_num` : int Axis index (1~6 for current coordinate system; in Cartesian, x/y/z/rx/ry/rz map to 1~6; sign indicates direction).
Return Value	StatusCodeEnum: Function execution result

Method Name	jogging.continuous_move(aj_num : int) -> StatusCodeEnum
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 fun
ction - Single-joint motion usage
"""

import time

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1
# [EN] Jog joint 1 under the joint coordinate system
ret = arm.jogging.continuous_move(1)
```

```
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 运动3s
# [EN] Move for 3 seconds
time.sleep(3)

# [ZH] 停止
# [EN] Stop
arm.jogging.stop()

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

4.13.3 Multi-axis Continuous Teaching Motion

Method Name	jogging.multi_move(`aj_num` : List[int]) -> StatusCodeEnum
Description	Perform multi-axis continuous teaching motion.
Request Parameters	`aj_num` : List[int] Axis index list (1~6 for current coordinate system; in Cartesian, x/y/z/rx/ry/rz map to 1~6; sign indicates direction).
Return Value	StatusCodeEnum: Function execution result

Method Name	jogging.multi_move(aj_num : List[int]) -> StatusCodeEnum
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-多关节点动运动使用示例 / / Teaching motion function - M
ulti-joint motion usage example
"""

import time

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1,2,3
# [EN] Jog joint 1,2,3 under the joint coordinate system
ret = arm.jogging.multi_move([1, 2, 3])
```

```
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 运动3s
# [EN] Move for 3 seconds
time.sleep(3)

# [ZH] 停止
# [EN] Stop
arm.jogging.stop()

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

4.13.4 Stop the Robot Continuous Move

Method Name	jogging.stop()
Description	Terminate the robot jog (JOG) motion.
Request Parameters	No parameters
Return Value	StatusCodeEnum: 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.

Core Features

- Support for connecting to and disconnecting from WebSocket servers
- Support for subscribing to robot status data
- Support for subscribing to register data
- Support for subscribing to IO signal data
- Support for receiving messages via callback functions
- Support for blocking receive of the next message
- Support for setting subscription frequency
- Support for handling receive message errors

Use Cases

- Real-time monitoring of robot operation status from host PC
- Implementation of robot data visualization
- Data linkage with external systems
- Real-time monitoring of register and IO status
- Building robot monitoring and control systems
- Implementation of real-time alarm and notification for robot status

4.14.1 Connect to WebSocket Server

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

4.14.2 Disconnect WebSocket Connection

Method Name	sub_pub.disconnect() -> StatusCodeEnum
Description	Disconnect from the robot controller WebSocket server
Request Parameters	None
Return Value	StatusCodeEnum: Disconnection status code
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	sub_pub.subscribe_status(`topics` : List[RobotTopicType], `frequency` : int = 200) -> StatusCodeEnum
Description	Add robot status data subscriptions
Request Parameters	`topics` : List[RobotTopicType] Robot topic type list `frequency` : int Subscription frequency (Hz, default 200)
Return Value	StatusCodeEnum: 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	sub_pub.subscribe_register(`reg_type` : RegTopicType, `reg_ids` : List[int], `frequency` : int = 200) -> StatusCodeEnum
Description	Add register data subscriptions
Request Parameters	`reg_type` : RegTopicType Register type `reg_ids` : List[int] Register ID list `frequency` : int Subscription frequency (Hz, default 200)
Return Value	StatusCodeEnum: 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	sub_pub.subscribe_io(`io_list` : List[tuple[IOTopicType, int]], `frequency` : int = 200) -> StatusCodeEnum
Description	Subscribe to IO signal data, including digital inputs/outputs
Request Parameters	`io_list` : List[tuple[IOTopicType, int]] IO list (each element is `(IO type, IO ID)`) `frequency` : int Subscription frequency (Hz, default 200)
Return Value	StatusCodeEnum: 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	sub_pub.start_receiving(`on_message_received` : Callable[[Dict[str, Any]], None]) -> StatusCodeEnum
Description	Start receiving subscription messages and process received data via the callback function
Request Parameters	`on_message_received` : Callable[[Dict[str, Any]], None] Message receive callback function
Return Value	StatusCodeEnum: Receiving status code
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.14.7 Handle Receive Message Error

Method Name	sub_pub.handle_receive_error() -> StatusCodeEnum
Description	Handle errors when receiving subscription messages; exits the loop if the receive callback raises an exception.
Request Parameters	None
Return Value	StatusCodeEnum: 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	sub_pub.receive() -> tuple[Dict[str, Any], StatusCodeEnum]
Description	Receive the next message and return it
Request Parameters	None
Return Value	tuple[Dict[str, Any], StatusCodeEnum]: Received message dictionary and status code

Method Name	sub_pub.receive() -> tuple[Dict[str, Any], StatusCodeEnum]
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

Example Code

sub_pub.py

py

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

import asyncio
import logging

from Agilebot import (
    Arm,
    IOTopicType,
    RegTopicType,
    RobotTopicType,
    StatusCodeEnum,
)

# 配置日志 / Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


async def message_handler(message):
    """
    消息处理函数 / Message handler function

    :param message: 接收到的消息字典 / Received message dictionary
    """
    logger.info(f"收到消息 / Received message: {message}")


async def main():
```

```
    """
    主函数，演示SubPub的使用 / Main function demonstrating SubPub usage
    """
    # 创建Arm实例 / Create Arm instance
    arm = Arm()

    # 连接到控制器 / Connect to controller
    ret = arm.connect("10.27.1.254")
    if ret != StatusCodeEnum.OK:
        logger.error(f"连接失败 / Connection failed: {ret}")
        return

    logger.info("Arm连接成功 / Arm connected successfully")

    try:
        # 连接WebSocket / Connect WebSocket
        ret = await arm.sub_pub.connect()
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"WebSocket连接失败 / WebSocket connection failed: {ret}"
            )
            return

        logger.info(
            "WebSocket连接成功 / WebSocket connected successfully"
        )

        # 订阅机器人状态 / Subscribe to robot status
        topic_types = [
            RobotTopicType.JOINT_POSITION,
            RobotTopicType.CARTESIAN_POSITION,
            RobotTopicType.ROBOT_STATUS,
            RobotTopicType.CTRL_STATUS,
            RobotTopicType.SERVO_STATUS,
            RobotTopicType.INTERPRETER_STATUS,
            RobotTopicType.TRAJECTORY_RECORD,
            RobotTopicType.USER_OP_MODE,
            RobotTopicType.USER_FRAME,
            RobotTopicType.TOOL_FRAME,
            RobotTopicType.VELOCITY_RATIO,
            RobotTopicType.TRAJECTORY_RECORD,
        ]
```

```
        ret = await arm.sub_pub.subscribe_status(
            topic_types, frequency=200
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"订阅机器人状态失败 / Failed to subscribe to robot status: {re
t}"
            )
            return

        logger.info(
            "机器人状态订阅成功 / Robot status subscription successful"
        )

        # 订阅寄存器 / Subscribe to registers
        ret = await arm.sub_pub.subscribe_register(
            RegTopicType.R, [1, 2, 3], frequency=200
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"订阅寄存器失败 / Failed to subscribe to registers: {ret}"
            )
            return

        logger.info(
            "寄存器订阅成功 / Register subscription successful"
        )

        # 订阅IO信号 / Subscribe to IO signals
        io_list = [(IOTopicType.DI, 1), (IOTopicType.DO, 1)]

        ret = await arm.sub_pub.subscribe_io(
            io_list, frequency=200
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"订阅IO失败 / Failed to subscribe to IO: {ret}"
            )
            return

        logger.info(
```

```
            "IO订阅成功 / IO subscription successful"
        )

        # 单次接收消息示例 / Single message receive example
        logger.info(
            "尝试单次接收消息 / Attempting single message receive"
        )
        try:
            # 设置超时 / Set timeout
            message, ret = await asyncio.wait_for(
                arm.sub_pub.receive(), timeout=5.0
            )
            if ret == StatusCodeEnum.OK:
                logger.info(
                    f"单次接收消息成功 / Single message receive successful: {m
essage}"
                )
            else:
                logger.error(
                    f"单次接收消息失败 / Single message receive failed: {ret}"
                )
        except asyncio.TimeoutError:
            logger.warning(
                "接收消息超时 / Message receive timeout"
            )

        # 启动消息接收 / Start message receiving
        ret = await arm.sub_pub.start_receiving(
            message_handler
        )
        if ret != StatusCodeEnum.OK:
            logger.error(
                f"启动消息接收失败 / Failed to start message receiving: {ret}"
            )
            return

        logger.info(
            "开始接收消息 / Started receiving messages"
        )

        # 运行一段时间 / Run for a while
        logger.info(
```

```
                "运行10秒钟... / Running for 10 seconds..."
            )
            await asyncio.sleep(10)

        except Exception as e:
            logger.error(
                f"运行过程中发生错误 / Error occurred during execution: {str(e)}"
            )

        finally:
            # 断开连接 / Disconnect
            await arm.sub_pub.disconnect()
            arm.disconnect()
            logger.info("连接已断开 / Connection disconnected")


if __name__ == "__main__":
    # 运行异步主函数 / Run async main function
    asyncio.run(main())
```

4.15 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.

Core Features

- Support for getting robot IP address
- Support for getting extension list
- Support for getting single extension details
- Support for toggling extension enable/disable status
- Support for deleting extensions
- Support for calling extension service commands
- Support for installing extension packages
- Support for installing wheel packages
- Support for centralized management of extension life-cycle

Use Cases

- Management of third-party plug-ins on robot controllers from host PC
- Extending robot functionality by calling services provided by extensions
- Monitoring and controlling extension enable status
- Connecting to robot controllers in extension or teaching pendant environments
- Integration of robots with third-party systems

4.15.1 Get the Robot's IP Address

Method Name	extension.get_robot_ip() -> 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:

1. **Industrial Teaching Pendant Environment**: Returns the controller's IP address
2. **Collaborative AP Board Environment**:
   - DHCP mode: Returns the router's IP address (default gateway)
   - Static IP mode: Returns the configured static IP address
3. **Cloud Environment**: Returns the cloud's internal IP address
4. **Other Environments**: Returns `None`

Note

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

Example Code

extension/connect_operation.py

```py
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.

Instruction: 插件系统连接机器人示例
本示例展示如何在插件系统中连接机器人。插件系统会自动识别运行环境：
- 在插件环境中运行时：自动获取机器人IP地址，无需手动配置
```

```
- 在本地开发环境中运行时：使用下方的 local_robot_ip 和 local_ap_ip 配置

Example of extension connection:
This example demonstrates how to connect to the robot in the extension syste
m.
The extension system automatically identifies the running environment:
- In extension environment: Automatically obtains the robot IP address
- In local development environment: Uses the local_robot_ip and local_ap_ip
configuration below
"""

from Agilebot import Arm, Extension, StatusCodeEnum

extension = Extension()

# ========== 本地开发配置 / Local Development Configuration ==========
# 以下配置仅在本地开发环境中生效，插件环境会自动获取IP
# The following configuration only takes effect in local development environ
ment
local_robot_ip = "10.27.1.254"  # 机器人控制器IP / Robot controller IP (modify
to 192.168.110.2 for industrial robot)
local_ap_ip = "10.27.1.254"  # 示教器或AP IP (工业机器人改为 192.168.110.102 或
None) / Teach pendant IP or AP IP (modify to 192.168.110.102 or None for ind
ustrial robot)

# ========== 自动识别运行环境 / Automatically Identify Running Environment ===
=======
robot_ip = extension.get_robot_ip()
if robot_ip is None:
    # 本地开发环境：使用上方配置的IP / Local development: use configured IP abov
e
    print("本地开发环境 / Local development environment")
    robot_ip = local_robot_ip
    ap_ip = local_ap_ip
else:
    # 插件环境：自动获取机器人IP / Extension environment: auto-obtain robot IP
    print(
        f"插件环境，自动获取到机器人IP / Extension environment, auto-obtained ro
bot IP: {robot_ip}"
    )
    ap_ip = "127.0.0.1"
```

```
print(f"robot_ip: {robot_ip}")
print(f"ap_ip: {ap_ip}")

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

4.15.2 Get Extension List

Method Name	extension.get_list() → tuple[list[ExtensionInfo], StatusCodeEnum]
Description	Retrieves information for all extensions installed on the robot
Request Parameters	None
Return Value	list[ExtensionInfo] : list of extension information. StatusCodeEnum : 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	extension.get(`name` : str) → tuple[ExtensionInfo, StatusCodeEnum]
Description	Queries details of a single extension by its name
Request Parameters	`name` : extension name.
Return Value	`ExtensionInfo` : detailed extension information. `StatusCodeEnum` : 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	extension.toggle(`name` : str) → StatusCodeEnum
Description	Toggles the enabled/disabled state of the specified extension.
Request Parameters	`name` : extension name.
Return Value	`StatusCodeEnum` : 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	extension.call_service(`name` : str, `command` : str, `params` : dict = None) → tuple[Any, StatusCodeEnum]
Description	Invokes a simple-service command provided by the specified extension and returns its execution result
Request Parameters	`name` : extension name. `command` : extension command name. `params` : command parameters (optional, default None).

Method Name	extension.call_service(`name` : str, `command` : str, `params` : dict = None) → tuple[Any, StatusCodeEnum]
Return Value	`Any` : execution result of the extension command. `StatusCodeEnum` : 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

py

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

from Agilebot import Arm, StatusCodeEnum

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

res, ret = arm.extension.get("MathService")
if ret == StatusCodeEnum.OK:
```

```
    print(
        "获取插件信息成功 / Get extension information successfully"
    )
    print(f"插件信息 / Extension information: {res}")
else:
    print(
        f"获取插件信息失败，错误代码 / Get extension information failed, error c
ode: {ret.errmsg}"
    )

ret = arm.extension.toggle("MathService")
if ret == StatusCodeEnum.OK:
    print(
        "切换插件状态成功 / Toggle extension status successfully"
    )
else:
    print(
        f"切换插件状态失败，错误代码 / Toggle extension status failed, error cod
e: {ret.errmsg}"
    )

ret = arm.extension.delete("MathService")
if ret == StatusCodeEnum.OK:
    print("删除插件成功 / Delete extension successfully")
else:
    print(
        f"删除插件失败，错误代码 / Delete extension failed, error code: {ret.er
rmsg}"
    )

res, ret = arm.extension.call_service(
    "MathService", "add", dict([["a", 1], ["b", 2]])
)
if ret == StatusCodeEnum.OK:
    print(
        "调用插件服务成功 / Call extension service successfully"
    )
    print(
        f"调用插件服务结果 / Call extension service result: {res}"
    )
else:
    print(
```

```
        f"调用插件服务失败，错误代码 / Call extension service failed, error cod
e: {ret.errmsg}"
    )
```

4.15.6 Delete Extension

Method Name	extension.delete(`name` : str) → StatusCodeEnum
Description	Deletes the specified extension
Request Parameters	`name` : extension name.
Return Value	`StatusCodeEnum` : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

4.15.7 Install Extension Package

Method Name	extension.install_extension(`path` : str) → tuple[ExtensionInfo, StatusCodeEnum]
Description	Uploads and installs an extension package, returning the extension base information
Request Parameters	`path` : extension package file path.
Return Value	`ExtensionInfo` : installed extension information. `StatusCodeEnum` : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

4.15.8 Install Wheel Package

Method Name	extension.install_wheel(`path` : str) → tuple[bool, StatusCodeEnum]
Description	Uploads and installs a Python wheel package
Request Parameters	`path` : wheel package file path.
Return Value	`bool` : installation result. `StatusCodeEnum` : execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): not supported

Example Code

extension/install_operation.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 插件安装示例 / Example of extension installation
"""

from pathlib import Path

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
local_robot_ip = (
    "10.27.1.254"  # 机器人控制器IP / Robot controller IP
)
local_ap_ip = "10.27.1.254"  # 示教器或AP IP / Teach pendant IP or AP IP
ret = arm.connect(
    arm_controller_ip=local_robot_ip,
    teach_panel_ip=local_ap_ip,
)
```

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

current_path = Path(__file__).resolve()
example_dir = current_path.parent / "files"
extension_package_path = (
    example_dir / "HelloAgilebot.gbtapp"
)
wheel_package_path = (
    example_dir / "six-1.17.0-py2.py3-none-any.whl"
)

# [ZH] 安装插件包
# [EN] Install the extension package
if extension_package_path.is_file():
    ext_info, install_ret = arm.extension.install_extension(
        str(extension_package_path)
    )
    if install_ret == StatusCodeEnum.OK:
        print(
            "插件安装成功 / Install extension successfully"
        )
        print(
            f"插件信息 / Extension information: {ext_info}"
        )
    else:
        print(
            f"插件安装失败，错误代码 / Install extension failed, error code: {i
nstall_ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
else:
    print(
        f"插件包文件不存在 / Extension package not found: {extension_package_p
ath}"
    )
```

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

# [ZH] 安装 wheel 包
# [EN] Install the wheel package
if wheel_package_path.is_file():
    wheel_ok, install_ret = arm.extension.install_wheel(
        str(wheel_package_path)
    )
    if install_ret == StatusCodeEnum.OK and wheel_ok:
        print(
            "wheel 安装成功 / Wheel package installed successfully"
        )
    else:
        print(
            f"wheel 安装失败，错误代码 / Wheel installation failed, error code:
{install_ret.errmsg}"
        )
        arm.disconnect()
        exit(1)
else:
    print(
        f"wheel 包文件不存在 / Wheel package not found: {wheel_package_path}"
    )
    arm.disconnect()
    exit(1)

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

4.16 Natural Language Control

Overview

The NLUControl module provides the ability to control robots through natural language instructions, supporting conversion of everyday language into executable code for execution after safety checks.

Core Features

- Support for controlling robots through natural language instructions
- Support for translating natural language into executable code
- Provide code approval mechanism to ensure safe execution
- Support for multi-step instructions (sequential execution, conditional judgment, etc.)
- Provide code review and printing functionality
- Automatically capture and handle code execution exceptions

Use Cases

- Quick control of robots to perform simple tasks through natural language
- Implementation of natural language description and execution for complex multi-step tasks
- Rapid verification of robot functions during development and debugging
- Lowering the threshold for robot programming, supporting non-professional operators
- Implementation of natural language interaction between robots and humans

Constructor

Method Name	NLUControl(`controller_ip` : str)
Description	Natural language control class for controlling robot motion through natural language. Usually accessed via `arm.nlu` , no manual instantiation needed.

Method Name	NLUControl(`controller_ip` : str)
Request Parameters	`controller_ip` : str Controller IP address
Return Value	No return value
Notes	This class is automatically created when the Arm class is instantiated. Users can access it directly through `arm.nlu` .
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.16.1 Execute Natural Language Instruction

Method Name	execute(`natural_language` : str) -> tuple[NLUGeneratedCode, StatusCodeEnum]
Description	Control the robot through natural language instructions. The system translates the natural language into executable code and returns it.
Request Parameters	`natural_language` : str Natural language instruction (e.g., "Move the robot to zero point and get the current position"; supports multi-step instructions like sequencing and conditionals).
Return Value	NLUGeneratedCode: Generated code object containing executable code and approval status StatusCodeEnum: Function execution result
Notes	- The returned NLUGeneratedCode object contains a `needs_approval` property indicating whether user approval is required - If `needs_approval=True` , you must call the `approve()` method before execution - You can view the generated code content via `result.code`
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

NLUGeneratedCode Class

Overview

The NLUGeneratedCode class is used to encapsulate, manage, and safely execute Python code snippets generated by the NLU service. This class provides a security mechanism for code approval and execution, ensuring that code requiring approval must be explicitly approved before execution.

Properties

Property Name	Type	Description
`needs_approval`	bool	Whether approval is required before execution, determined by the code's danger level
`code`	str	The generated executable Python code string

4.16.2 Approve Code Execution

Method Name	approve()
Description	Approve code execution. After calling this method, the code is marked as approved and can be executed via the `execute_code()` method.
Request Parameters	No parameters
Return Value	No return
Notes	- This method only needs to be called when `needs_approval=True` - Executing code that requires approval without calling `approve()` will return an error
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.16.3 Execute Generated Code

Method Name	execute_code() -> tuple[Any, StatusCodeEnum]
Description	Execute the NLU-generated code. If the code requires approval and has not been approved, an error is returned.

Method Name	execute_code() -> tuple[Any, StatusCodeEnum]
Request Parameters	No parameters
Return Value	Any: Code execution result, type depends on the generated code StatusCodeEnum: Function execution result
Notes	- If `needs_approval=True` but `approve()` was not called, returns `NLU_APPROVAL_REQUIRED` error - Any exceptions during code execution are caught and returned as corresponding error status codes
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.16.4 Print Generated Code

Method Name	print_code()
Description	Print the generated code for user review. Output format includes syntax highlighting.
Request Parameters	No parameters
Return Value	No return
Notes	It is recommended to call this method to view the code content before deciding whether to approve execution when approval is needed
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

Usage Examples

Example 1: Basic Query Operation

nlu_control/get_controller_version.py

py

```python
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 自然语言基础控制与安全评级示例 / NLU Basic Control with Safety Rating Example
"""

from Agilebot import Arm, StatusCodeEnum

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

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


# [ZH] 示例：使用自然语言获取控制器版本信息
# [EN] Example: Use natural language to get controller version information
result, ret = arm.nlu.execute("获取控制器版本信息")

if ret == StatusCodeEnum.OK:
    print(
        "自然语言指令执行成功 / Natural language command executed successfully"
    )
    danger_level = getattr(
        result, "danger_level", "unknown"
    )
    warnings = getattr(result, "warnings", [])
```

```
    result.print_code()
    print(f"安全评级 / Danger level: {danger_level}")
    if warnings:
        print("安全提示 / Safety warnings:")
        for idx, warning in enumerate(warnings, 1):
            print(f"  {idx}. {warning}")

    if result.needs_approval:
        print("\n" + "=" * 50)
        print(
            "警告：需要用户确认 / Warning: User Approval Required"
        )
        print("=" * 50)

        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute thi
s code:"
        )
        user_input = input("\n(yes/no): ").strip().lower()
        if user_input not in ["yes", "y"]:
            print(
                "用户拒绝执行代码 / User rejected code execution"
            )
            arm.disconnect()
            exit(1)
        else:
            result.approve()

    exec_result, ret = result.execute_code()
    if ret == StatusCodeEnum.OK:
        print(f"执行结果 / Execution result: {exec_result}")
    else:
        print(
            f"代码执行失败 / Code execution failed: {ret.errmsg}"
        )
else:
    print(
        f"自然语言指令执行失败 / Natural language command failed: {ret.errmsg}"
    )

# [ZH] 断开捷勃特机器人连接
```

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

Example 2: Sequential Execution of Multiple Tasks

nlu_control/serial_execution.py

py

```
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 自然语言顺序执行与安全评级示例 / NLU Sequential Execution with Safety Rating Example
"""

from Agilebot import Arm, StatusCodeEnum

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

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


# [ZH] 示例：使用自然语言让机器人按顺序执行任务
```

```
# [EN] Example: Use natural language to instruct the robot to execute tasks sequentially
result, ret = arm.nlu.execute(
    """
按顺序执行：
1. 清除报警；
2. 控制机械臂所有关节运动至零点位置；
3. 启动程序 'Put_into_box'，等待下发完成，并等待其执行结束；
4. 获取当前位姿。
"""
)

if ret == StatusCodeEnum.OK:
    print(
        "自然语言指令执行成功 / Natural language command executed successfully"
    )
    danger_level = getattr(
        result, "danger_level", "unknown"
    )
    warnings = getattr(result, "warnings", [])

    result.print_code()
    print(f"安全评级 / Danger level: {danger_level}")
    if warnings:
        print("安全提示 / Safety warnings:")
        for idx, warning in enumerate(warnings, 1):
            print(f"  {idx}. {warning}")

    if result.needs_approval:
        print("\n" + "=" * 50)
        print(
            "警告：需要用户确认 / Warning: User Approval Required"
        )
        print("=" * 50)

        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute this code:"
        )
        user_input = input("\n(yes/no): ").strip().lower()
        if user_input not in ["yes", "y"]:
            print(
```

```
                    "用户拒绝执行代码 / User rejected code execution"
                )
                arm.disconnect()
                exit(1)
            else:
                result.approve()

        exec_result, ret = result.execute_code()
        if ret == StatusCodeEnum.OK:
            print(f"执行结果 / Execution result: {exec_result}")
        else:
            print(
                f"代码执行失败 / Code execution failed: {ret.errmsg}"
            )
    else:
        print(
            f"自然语言指令执行失败 / Natural language command failed: {ret.errmsg}"
        )

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

Agilebot Python SDK Update Notes

2.0.1.0 Update (2026/1/26)

1. Changed the underlying communication method.
2. Added support for a local proxy service.
3. Added `step_move` and `continuous_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 `BasScript` 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. Optimized Python dependencies; Python 3.8 and above is now supported.
20. Fixed errors in `load` , `unload` , and `exec` related instructions under the `BasScript` class.

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.

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.

Help

Agent Skills

Agent Skills extend AI agents (such as Codex and Claude Code) with standardized execution workflows and tooling for industrial or development scenarios.

Compatible Agent Types

Agent Skills currently support automatic detection and installation for the following AI tools and CLIs:

- **IDE / Editors**: Cursor, Windsurf, Trae, Trae CN, VS Code (via Copilot / Continue), Neovate, Pochi
- **Command-Line (CLI)**: Claude Code, Kimi Code CLI, Gemini CLI, iFlow CLI, Kiro CLI, Mistral Vibe
- **Frameworks / Platforms**: OpenHands, Replit, Cline, Roo Code, Codex, Amp, Antigravity, Augment, OpenClaw
- **Others**: CodeBuddy, Command Code, Droid, Junie, Kilo Code, Kode, MCPJam, Mux, OpenCode, Pi, Qoder, Qwen Code, Zencoder, AdaL

Quick Start

1. Environment Preparation

Before installing, make sure Node.js is installed locally. Guide: Node.js Download and Installation

2. Run the Install Command

Run the following command in your terminal to enter the interactive installer:

```bash
npx skills add https://dev.svfactory.com -g
```

TIP

After running the command, the tool scans locally installed agents. Follow the prompts to select which agents to install the skill into (for example, Cursor or Claude Code).

Available Skills

agilebot-nlu-control (Natural Language Control for Agilebot)

This skill is designed for Agilebot robots. It uses NLU (Natural Language Understanding) so the agent can parse and execute control instructions directly.

- Environment requirements: system Python must be available and Conda must be installed. The skill checks required dependencies and guides installation if anything is missing.
- Trigger: when the conversation mentions "Agilebot" and provides a concrete control intent.
- Core features: natural-language command parsing and execution, robot status queries, basic motion control, parameterized motion commands (e.g., distance/angle/speed).

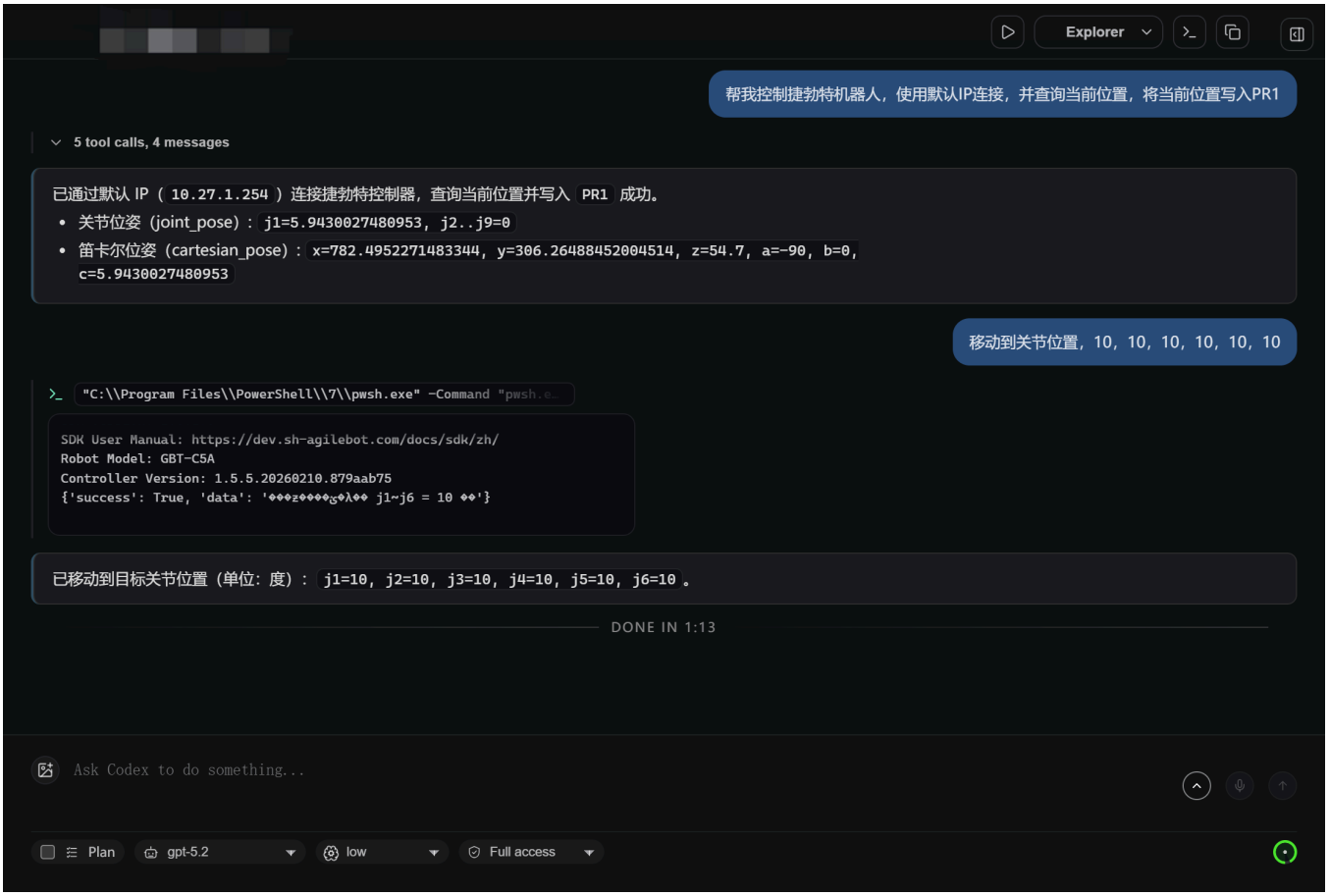
Warning

Installed skills run with full agent permissions. Make sure you are operating in a trusted environment before use.

Robot motion can be hazardous and may cause injury or equipment damage. Use in a safe environment. We are not responsible for any risks or losses arising from usage.

Usage Example

Using Codex as an example, after installation you can enter instructions directly in the chat:



AI Coding Support

This document describes how to use AI assistance tools (such as CodeBuddy, Codex, Cursor, etc.) to quickly develop robot programs.

Preparation

Before using AI coding, you need to prepare the reference documentation:

- **SDK Documentation**: https://dev.svfactory.com/docs/sdk/knowledge/docs.txt

> **Tip**: If your AI agent cannot read URLs well, download the txt document above to your local project directory and reference the local file path in your prompt.

Example Prompt

Here is a complete example for creating a Python program that reads robot status:

```
Read the following documentation and write a Python program that reads the r
obot's current position, coordinate system number, servo status, and other i
nformation.

Reference materials:
SDK Documentation: https://dev.svfactory.com/docs/sdk/knowledge/docs.txt
```

If you rely on local documentation, you can modify it to:

```
Read the following documentation and write a Python program that reads the r
obot's current position, coordinate system number, servo status, and other i
nformation.

Reference materials:
SDK Documentation: ./docs/sdk_docs.txt
```

Usage Tips

1. **Clear Requirements**: Clearly describe the functionality you want to implement
2. **Provide Context**: Reference relevant documentation and examples
3. **Stepwise Implementation**: Complex features can be generated step by step with AI

Notes

1. Code generated by AI needs to be verified and tested
2. Ensure code complies with project coding standards
3. Code involving robot control must undergo security review