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

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

3 Data Structures

3.1 StatusCodeEnum

Description

Interface return status codes.

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

Import

```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
INVALID_PARAMETER	-27	Invalid parameter

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

3.2 RobotStatusEnum

Description

Robot operating status.

Import

```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
ROBOT_IDLE_TO_RUNNING	101	Transition from idle to running

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

3.3 CtrlStatusEnum

Description

Controller operating status.

Import

```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
j3	float	Value of joint 3

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

3.9 Position

Description

Data structure describing the robot Cartesian pose.

Import

```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
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 (-1/0/1). 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 (-1/0/1). On 6-axis robots (joint 3), 1 means the arm is above the line between joints 4 and 2 (with J3 < 0), -1 means it is below that line (with J3 > 0).
arm_back_front	int	Arm front/back posture (-1/0/1). On 6-axis robots (joint 1), 1 means the arm is in front (axis 2 on the left of axis 1 when facing forward), -1 means the arm is behind (axis 2 on the right of axis 1).
arm_left_right	int	Arm left/right posture (-1/0/1). On 4-axis SCARA robots (joint 2), 1 means the arm is on the right, -1 means the arm is on the left.

3.11 BaseCartData

Description

Data structure describing the robot Cartesian target pose.

Import

```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
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

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

MassCenter

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

InertiaMoment

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

3.15 ProgramCartData

Description

Program Cartesian data.

Import

```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

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

3.21 HWState

Description

Hardware state enumeration.

Import

```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
TOPIC_TF	Publish current tool frame info
TOPIC_VELOCITY	Publish global velocity ratio
TOPIC_RUNNING_STATUS	Publish controller running status
TOPIC_INTERPRETER_STATUS	Publish interpreter status
TOPIC_ROBOT_STATUS	Publish robot status

Name	Description
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
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

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

3.25 ModbusChannel

Description

Modbus channel type.

Import

```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
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

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

3.29 SignalValue

Description

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

Import

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

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

3.31.2 RegTopicType

Description

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

Import

```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

Name	Value	Description
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

Name	Enum Value	Description
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

Name	Enum Value	Description
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

Name	Enum Value	Description
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 connect
ion
```

```
"""

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

# [ZH] 获取型号
```

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

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

4.1.5 Get the Robot's Operating Status

Method Name	**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.errms
g}"
    )
    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 co
de: {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 status of
the controller
"""

from Agilebot import Arm, StatusCodeEnum

# [ZH] 初始化捷勃特机器人
```

```python
# [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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

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

4.1.7 Get the Current Servo Controller Status

Method Name	**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.errms
g}"
    )
```

```
    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.errms
g}"
    )
    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 successfull
y"
    )
    print(
```

```
            f"机器人当前的软状态 / Robot current soft state: {state.name}"
        )
    else:
        print(
            f"获取机器人当前的软状态失败，错误代码 / Get the robot's current soft state fa
iled, 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 permissions
"""

from Agilebot import Arm, StatusCodeEnum

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

# [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.errms
g}"
    )
    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
"""
```

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

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.errms
g}"
    )
    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 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"
)
```

4.1.15 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 of the
LED
"""

import time

from Agilebot import Arm, StatusCodeEnum

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

```
arm = Arm()
# [ZH] 连接捷勃特机器人
# [EN] Connect to the Agilebot robot
ret = arm.connect("10.27.1.254")
if ret == StatusCodeEnum.OK:
    print("机器人连接成功 / Robot connected successfully")
else:
    print(
        f"机器人连接失败，错误代码 / Robot connection failed, error code: {ret.errms
g}"
    )
    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.e
rrmsg}"
    )
    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
Request Parameters	No parameters
Return Value	StatusCodeEnum: Function execution result

Method Name	**servo_off**() -> StatusCodeEnum
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 example
"""

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

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

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

# [ZH] 机器人伺服上电
# [EN] Robot servo power on
ret = arm.servo_on()
if ret == StatusCodeEnum.OK:
    print("伺服上电成功 / Servo power on successfully")
else:
    print(
```

```
            f"伺服上电失败，错误代码 / Servo power on failed, error code: {ret.errmsg}"
        )
        arm.disconnect()
        exit(1)

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

4.1.19 Emergency Stop

Method Name	**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
"""
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
```

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

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

4.2 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.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 获取机器人各参数并打印
# [EN] Get robot parameters and print
res, ret = arm.motion.get_OVC()
if ret == StatusCodeEnum.OK:
    print(
        "获取全局速度比率成功 / Get global velocity ratio successful"
    )
else:
    print(
        f"获取全局速度比率失败，错误代码 / Get global velocity ratio failed, error co
de: {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 failed, err
or 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 successful"
    )
else:
    print(
        f"获取示教坐标系类型失败，错误代码 / Get teaching coordinate system type faile
d, 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, error cod
e: {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, error cod
e: {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
Request Parameters	`value` : float Acceleration ratio (range 0.01~1.2)

Method Name	**motion.set_OAC(** value : float) -> StatusCodeEnum
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.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置机器人各参数
```

```
# [EN] Set robot parameters
ret = arm.motion.set_OVC(0.7)
if ret == StatusCodeEnum.OK:
    print(
        "设置全局速度比率成功 / Set global velocity ratio successful"
    )
else:
    print(
        f"设置全局速度比率失败，错误代码 / Set global velocity ratio failed, error co
de: {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 failed, err
or 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 successful"
    )
else:
    print(
        f"设置示教坐标系类型失败，错误代码 / Set teaching coordinate system type faile
d, 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, error cod
e: {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, 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"
)
```

4.2.3 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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

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

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

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

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

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

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

# [ZH] 发送运动请求
# [EN] Send motion request
ret = arm.motion.move_joint(motion_pose, vel=0.5, acc=0.5)
if ret == StatusCodeEnum.OK:
    print(
        "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 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"
)
```

4.2.6 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
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

Method Name	**motion.move_line(** `pose` : MotionPose, `vel` : float = 100, `acc` : float = 1) -> StatusCodeEnum
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/move_line.py

py

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

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

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

```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.errms
g}"
    )
    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 successfu
lly"
    )
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, error
code: {ret_code.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

4.2.8 Get 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.errms
g}"
    )
    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.errms
g}"
    )
    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

Method Name	**motion.get_DH_param**() -> tuple[list[DHparam], StatusCodeEnum]
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
#!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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

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

# [ZH] 打印结果
# [EN] Print result
for param in res:
    print(
```

```
        f"DH参数的ID / DH parameter ID: {param.id}\n"
        f"杆件长度 / Link length: {param.a}\n"
        f"杆件扭角 / Link twist angle: {param.alpha}\n"
        f"关节距离 / Joint distance: {param.d}\n"
        f"关节转角 / Joint angle: {param.offset}"
    )

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

4.2.11 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
"""
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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

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

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

4.2.12 Set 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.errms
g}"
    )
    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.errms
g}"
    )
    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.errms
g}"
    )
    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.e
rrmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

# [ZH] 退出拖动示教
# [EN] Exit drag teaching mode
ret = arm.motion.enable_drag(False)
if ret == StatusCodeEnum.OK:
    print(
        "退出拖动示教成功 / Exit drag teaching successful"
    )
else:
    print(
        f"退出拖动示教失败，错误代码 / Exit drag teaching failed, error code: {ret.er
rmsg}"
    )
    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	Collaborative (Copper): v7.5.2.0+

Method Name	**motion.enter_position_control**() -> StatusCodeEnum
Version	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

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); `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_feedback_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 mod
e 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.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置udp反馈参数
# [EN] Set UDP feedback parameters
ret = arm.motion.set_udp_feedback_params(
    True, "10.27.1.254", 20, 1, [0, 1, 2]
)
if ret == StatusCodeEnum.OK:
    print(
        "设置UDP反馈参数成功 / Set UDP feedback parameters successful"
    )
else:
```

```
    print(
        f"设置UDP反馈参数失败，错误代码 / Set UDP feedback parameters failed, error c
ode: {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 f
ailed, 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 successful"
    )
else:
    print(
        f"退出实时位置控制模式失败，错误代码 / Exit real-time position control mode fa
iled, 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"
)
```

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
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

Name	Field	Description
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
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
```

```
Instruction: 用户软限位设置获取示例 / Example of obtaining the user's soft limit sett
ing
"""

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.errms
g}"
    )
    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: {re
t.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
"""
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.errms
g}"
    )
    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.e
rrmsg}"
    )
    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 based on
ID
"""

from Agilebot import Arm, StatusCodeEnum

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

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

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

# [ZH] 打印结果
# [EN] Print result
print(
    f"负载ID / Payload ID: {res.id}\n"
    f"负载质量 / Payload mass: {res.weight}\n"
    f"负载注释 / Payload comment: {res.comment}\n"
    f"负载质心 / Payload mass center: {res.mass_center.x}, {res.mass_center.y}, {re
s.mass_center.z}\n"
    f"负载转动惯量 / Payload inertia moment: {res.inertia_moment.lx}, {res.inertia_m
oment.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
#!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.errms
g}"
    )
    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.e
rrmsg}"
    )
    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

Method Name	**motion.payload.add_payload(** payload_info : PayloadInfo) -> StatusCodeEnum
Compatible Version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

motion/add_payload.py

py

```python
#!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.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 初始化负载
# [EN] Initialize payload
new_payload = PayloadInfo()
new_payload.id = 6
new_payload.comment = "Test"
```

```
new_payload.weight = 5
new_payload.mass_center.x = -151
new_payload.mass_center.y = 1.0
new_payload.mass_center.z = 75
new_payload.inertia_moment.lx = 0.11
new_payload.inertia_moment.ly = 0.61
new_payload.inertia_moment.lz = 0.54

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

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

4.2.18.5 Delete Payload

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

Method Name	**motion.payload.delete_payload(** payload_id : int) -> StatusCodeEnum
Note	Note: The currently activated payload cannot be deleted. If you want to delete the activated payload, please activate another payload first and then delete the current one.

Example Code

motion/delete_payload.py

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

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 删除指定ID负载
# [EN] Delete payload with specified ID
ret = arm.motion.payload.delete_payload(6)
if ret == StatusCodeEnum.OK:
```

```
    print("删除负载成功 / Delete payload successful")
else:
    print(
        f"删除负载失败，错误代码 / Delete payload failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print("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
"""
```

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

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

# [ZH] 更新负载
# [EN] Update payload
ret = arm.motion.payload.update_payload(payload_info)
if ret == StatusCodeEnum.OK:
```

```
    print("更新负载成功 / Update payload successful")
else:
    print(
        f"更新负载失败，错误代码 / Update payload failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印结果
# [EN] Print result
print(
    f"负载ID / Payload ID: {payload_info.id}\n"
    f"负载质量 / Payload mass: {payload_info.weight}\n"
    f"负载质心 / Payload mass center: {payload_info.mass_center.x}, {payload_info.mass_center.y}, {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
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
#!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.errms
g}"
    )
    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.errm
sg}"
    )
    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
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

Method Name	**motion.payload.payload_identify_result**() -> tuple[PayloadInfo, StatusCodeEnum]
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
#!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.errms
g}"
    )
    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.er
rmsg}"
    )
    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")
```

```python
# [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.errms
g}"
    )
    arm.disconnect()
    exit(1)

program_name = "test"

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

# [ZH] 获取所有正在运行的程序
# [EN] Get all running programs
programs_list, ret = arm.execution.all_running_programs()
if ret == StatusCodeEnum.OK:
    print(
        "获取运行程序列表成功 / Get running programs list successful"
    )
else:
    print(
        f"获取运行程序列表失败，错误代码 / Get running programs list failed, error co
de: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
for program in programs_list:
    print(f"正在运行的程序名：{program}")
```

```
time.sleep(2)

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

time.sleep(2)

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

time.sleep(2)

# [ZH] 停止程序
# [EN] Stop program
ret = arm.execution.stop(program_name)
if ret == StatusCodeEnum.OK:
    print("程序停止成功 / Program stop successful")
else:
    print(
        f"程序停止失败，错误代码 / Program stop failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
```

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

4.3.6 Execute BAS Script Program

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

execution/bas_script_execution.py

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

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


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

4.4 Program 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

Method Name	**program_pose.add(** `program_name` : str, `index` : int, `value` : ProgramPose, `program_type` : str = USER_PROGRAM) -> StatusCodeEnum
	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.

Method Name	**ProgramPoses.convert_pose(** `pose` : ProgramPose, `from_type` : PoseType, `to_type` : PoseType) -> tuple[ProgramPose, StatusCodeEnum]
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

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

# [ZH] 读取单个位姿
# [EN] Read a single pose
pose, ret = arm.program_pose.read(program_name, 1)
if ret == StatusCodeEnum.OK:
    print(
        "读取单个位姿成功 / Read single pose successfully"
    )
    # [ZH] 打印位姿信息
    # [EN] Print pose information
    print(
        f"位姿ID / Pose ID: {pose.id}\n"
        f"位姿名称 / Pose name: {pose.name}\n"
        f"位姿类型 / Pose type: {pose.poseData.pt}\n"
        f"X: {pose.poseData.cartData.baseCart.position.x}\n"
        f"Y: {pose.poseData.cartData.baseCart.position.y}\n"
        f"Z: {pose.poseData.cartData.baseCart.position.z}\n"
        f"J1: {pose.poseData.joint.j1}\n"
        f"J2: {pose.poseData.joint.j2}\n"
        f"J3: {pose.poseData.joint.j3}\n"
    )
else:
```

```
    print(
        f"读取单个位姿失败，错误代码 / Read single pose failed, error code: {ret.errm
sg}"
    )
    arm.disconnect()
    exit(1)

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

# [ZH] 转换位姿
# [EN] Convert pose
converted_pose, ret = arm.program_pose.convert_pose(
    pose, PoseType.CART, PoseType.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.errmsg}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 打印位姿信息
# [EN] Print pose information
print(
    f"位姿ID / Pose ID：{converted_pose.id}\n"
    f"位姿名称 / Pose name：{converted_pose.name}\n"
    f"位姿类型 / Pose type：{converted_pose.poseData.pt}\n"
    f"X：{converted_pose.poseData.cartData.baseCart.position.x}\n"
    f"Y：{converted_pose.poseData.cartData.baseCart.position.y}\n"
```

```
    f"Z: {converted_pose.poseData.cartData.baseCart.position.z}\n"
    f"J1: {converted_pose.poseData.joint.j1}\n"
    f"J2: {converted_pose.poseData.joint.j2}\n"
    f"J3: {converted_pose.poseData.joint.j3}\n"
)

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

4.5 IO Signals

Overview

The Signals module offers a unified read/write interface to the controller's I/O, 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 writing
"""

from Agilebot import (
    Arm,
```

```
    SignalType,
    SignalValue,
    StatusCodeEnum,
)

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

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

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

```
    )
    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) -> tuple[float, StatusCodeEnum]
Description	Reads the value of an R numeric register.
Request Parameters	`index` : int R register number to read

Method Name	**register.read_R(** `index` : int) -> tuple[float, StatusCodeEnum]
Return Value	float: R register numeric value 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
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
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
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: R寄存器读写示例 / Example of reading and writing the R register
"""

from Agilebot import Arm, StatusCodeEnum

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

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

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

4.6.2 MR Motion Register Operations

4.6.2.1 Read MR Register Value

Method Name	**register.read_MR(** index : int) -> tuple[int, StatusCodeEnum]
Description	Reads the value of an MR motion register.
Request Parameters	index : int MR register number to read
Return Value	int: MR register numeric value 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
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
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
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
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: MR寄存器读写示例 / Example of reading and writing MR Registers
"""

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 添加MR寄存器
# [EN] Add MR register
ret = arm.register.write_MR(5, 8)
if ret == StatusCodeEnum.OK:
    print("写入MR寄存器成功 / Write MR register successful")
else:
    print(
        f"写入MR寄存器失败，错误代码 / Write MR register failed, error code: {ret.err
msg}"
    )
    arm.disconnect()
    exit(1)
```

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

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

# [ZH] 删除MR寄存器
# [EN] Delete MR register
ret = arm.register.delete_MR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除MR寄存器成功 / Delete MR register successful"
    )
else:
    print(
        f"删除MR寄存器失败，错误代码 / Delete MR register failed, error code: {ret.er
rmsg}"
    )
    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) -> tuple[str, StatusCodeEnum]
Description	Reads the value of an SR string register.
Request Parameters	`index` : int SR register number to read
Return Value	str: SR register string value 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
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
Return Value	StatusCodeEnum: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.3.3 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

Method Name	**register.delete_SR(** index : int) -> StatusCodeEnum
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

registers/SR.py

py

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

from Agilebot import Arm, StatusCodeEnum

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

# [ZH] 添加SR寄存器
# [EN] Add SR register
ret = arm.register.write_SR(5, "pytest")
if ret == StatusCodeEnum.OK:
    print("写入SR寄存器成功 / Write SR register successful")
```

```
else:
    print(
        f"写入SR寄存器失败，错误代码 / Write SR register failed, error code: {ret.err
msg}"
    )
    arm.disconnect()
    exit(1)

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

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

# [ZH] 删除SR寄存器
# [EN] Delete SR register
ret = arm.register.delete_SR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除SR寄存器成功 / Delete SR register successful"
    )
else:
    print(
        f"删除SR寄存器失败，错误代码 / Delete SR register failed, error code: {ret.er
rmsg}"
    )
    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) -> tuple[PoseRegister, StatusCodeEnum]
Description	Reads the value of a PR pose register.
Request Parameters	`index` : int PR register number to read
Return Value	PoseRegister: PR register pose data 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) -> StatusCodeEnum
Description	Writes the value of a PR pose register.
Request Parameters	`value` : PoseRegister PR register pose data to write
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
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: PR寄存器读写示例 / Example of reading and writing to the PR register
"""

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

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

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

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

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

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

# [ZH] 删除PR寄存器
# [EN] Delete PR register
ret = arm.register.delete_PR(5)
if ret == StatusCodeEnum.OK:
    print(
        "删除PR寄存器成功 / Delete PR register successful"
    )
else:
    print(
        f"删除PR寄存器失败，错误代码 / Delete PR register failed, error code: {ret.er
rmsg}"
    )
    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.
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 re
gister and MI register
"""

from Agilebot import Arm, StatusCodeEnum

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

```
    )
    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.errm
sg}"
    )
    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.err
msg}"
    )
    arm.disconnect()
    exit(1)

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

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

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

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

Method Name	**trajectory.execute_offline_trajectory**() -> StatusCodeEnum
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.
Request Parameters	`file_name` : str Trajectory file path (returned by `transform_csv_to_trajectory`).

Method Name	**trajectory.check_transform_status(** file_name : str) -> tuple[TransformStatusEnum, StatusCodeEnum]
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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

# [ZH] 等待控制器到位
# [EN] Wait for the controller to be ready
while True:
    robot_status, ret = arm.get_robot_status()
```

```
    if ret == StatusCodeEnum.OK:
        print(
            "获取机器人状态成功 / Get robot status successful"
        )
    else:
        print(
            f"获取机器人状态失败，错误代码 / Get robot status failed, error code: {re
t.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.
Parameters	`name` : trajectory program name.
Return	StatusCodeEnum: Execution result of the function.

Method Name	**trajectory.trajectory_record_finish(** `name` : str) -> StatusCodeEnum
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.
Parameters	`name` : trajectory program name.
Return	MotionPose: Starting pose of the trajectory. StatusCodeEnum: Execution result of the function.

Method Name	**trajectory.get_trajectory_record_start_pose(** name : str) -> tuple[MotionPose, StatusCodeEnum]
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

trajectory/trajectory_record.py

py

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

import time

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

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

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

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

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

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

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

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

```
    exit(1)

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

# [ZH] 等待控制器到位
# [EN] Wait for the controller to be ready
while True:
    robot_status, ret = arm.get_robot_status()
    if ret == StatusCodeEnum.OK:
        print(
            "获取机器人状态成功 / Get robot status successful"
        )
    else:
        print(
            f"获取机器人状态失败，错误代码 / Get robot status failed, error code: {re
t.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 code: {r
et.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 code: {re
t.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 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"
)
```

4.7.13 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). `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): v7.8.0.0+ 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): v7.8.0.0+ 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): v7.8.0.0+ 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.

Method Name	**trajectory.get_path_state(** `path_list` : list[str]) -> tuple[dict[str, int], StatusCodeEnum]
Return	`dict[str, int]` : mapping of file name → status code. StatusCodeEnum: execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ 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): v7.8.0.0+ 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.

Method Name	**trajectory.get_path_planner_parameter**() -> tuple[float, float, StatusCodeEnum]
	StatusCodeEnum: Execution result of the function.
Compatible robot software version	Collaborative (Copper): v7.8.0.0+ Industrial (Bronze): not supported

4.7.19 Move Along a Trajectory / Path

Method Name	**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): v7.8.0.0+ 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.errms
g}"
    )
    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.err
msg}"
    )
    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.errms
g}"
    )
    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.errmsg}"
    )
    arm.disconnect()
    exit(1)
# 等待3秒，让机械臂持续运动
time.sleep(2)
# 停止机械臂运动
arm.jogging.stop()
time.sleep(2)
ret = arm.jogging.move(-3, MoveMode.Continuous)
if ret == StatusCodeEnum.OK:
    print("点动运动成功 / Jogging movement successful")
else:
    print(
        f"点动运动失败，错误代码 / Jogging movement failed, error code: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)
# 等待3秒，让机械臂持续运动
time.sleep(2)
# 停止机械臂运动
arm.jogging.stop()
time.sleep(2)
```

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

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

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

# [ZH] 获取轨迹规划参数以验证设置是否成功
# [EN] Get trajectory planning parameters to verify if the setting is successful
param1, param2, ret = (
    arm.trajectory.get_path_planner_parameter()
)
if ret == StatusCodeEnum.OK:
    print(
        "获取路径规划参数成功 / Get path planner parameter successful"
```

```
    )
else:
    print(
        f"获取路径规划参数失败，错误代码 / Get path planner parameter failed, error c
ode: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

4.8 Alarm Information

Overview

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

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

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

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

4.9 File Manager

Overview

FileManager is used to upload, download, delete, or search resources such as robot programs, trajectories, and temporary files between the host computer and the robot controller, 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).
Return Value	StatusCodeEnum: Result of the function execution.

Method Name	**file_manager.search(** pattern : str, file_list : list) -> StatusCodeEnum
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 download
"""

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.errmsg}"
    )
    exit(1)

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

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

# [ZH] 搜索文件
# [EN] Search for files
file_list = list()
ret = file_manager.search("test.csv", file_list)
```

```
if ret == StatusCodeEnum.OK:
    print("文件搜索成功 / File search successful")
else:
    print(
        f"文件搜索失败，错误代码 / File search failed, error code: {ret.errmsg}"
    )
    exit(1)
print("搜索文件：", file_list)

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

# [ZH] 删除文件
# [EN] Delete a file
ret = file_manager.delete(
    "test_torque.trajectory", TRAJECTORY
)
if ret == StatusCodeEnum.OK:
    print("文件删除成功 / File delete successful")
else:
    print(
        f"文件删除失败，错误代码 / File delete failed, error code: {ret.errmsg}"
    )
    exit(1)
```

4.10 Coordinate Systems

Overview

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

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.
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
#!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.errms
```

```
g}"
    )
    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, err
or 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 cod
e: {ret.errmsg}"
    )
    arm.disconnect()
    exit(1)

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

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

```python
# [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
Return Value	StatusCodeEnum: Result of the function execution

Method Name	**slave.write_holding_regs(** `address` : int, `value` : list[int]) -> StatusCodeEnum
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
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.errms
g}"
    )
    arm.disconnect()
    exit(1)

# [ZH] 设置modbus参数
# [EN] Set Modbus parameters
params = SerialParams(
    channel=ModbusChannel.CONTROLLER_TCP_TO_485,
    ip="10.27.1.80",
    port=502,
)
id, ret_code = arm.modbus.set_parameter(params)
```

```
if ret_code == StatusCodeEnum.OK:
    print(
        "设置Modbus参数成功 / Set Modbus parameters successfully"
    )
else:
    print(
        f"设置Modbus参数失败，错误代码 / Set Modbus parameters failed, error code: {r
et_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.errmsg}"
    )
    arm.disconnect()
    exit(1)

ret = slave.write_holding_regs(0, value)
if ret == StatusCodeEnum.OK:
    print(
        "写入保持寄存器成功 / Write holding registers successfully"
    )
else:
    print(
        f"写入保持寄存器失败，错误代码 / Write holding registers failed, error code:
{ret.errmsg}"
    )
    arm.disconnect()
```

```
    exit(1)

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

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

res, ret = slave.read_input_regs(0, 4)
if ret == StatusCodeEnum.OK:
    print(
        "读取输入寄存器成功 / Read input registers successfully"
    )
    print(
        f"读取的输入寄存器值 / Read input register values: {res}"
    )
else:
    print(
        f"读取输入寄存器失败，错误代码 / Read input registers failed, error code: {re
t.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

Method Name	**modbus.set_parameter(** `param` : SerialParams) -> tuple[int, StatusCodeEnum]
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
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 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
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

Method Name	**logical.logi_default**() -> StatusCodeEnum
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
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
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

Method Name	**structure.exec(** `name` : str) -> StatusCodeEnum
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
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

Method Name	**modbus.modbus_read_MH(** index : int, id : int, address : int, length : int, r_index : int) -> StatusCodeEnum
	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
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 - Joint step motion usage example
"""

from Agilebot import Arm, StatusCodeEnum

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

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

# [ZH] 仅支持手动模式下进行jogging
# [EN] Only manual mode is supported for jogging
# [ZH] 点动关节坐标系下的关节1
# [EN] Jog joint 1 under the joint coordinate system
ret = arm.jogging.step_move(1, 2, 2)
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error code: {re
t.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
#!python
"""
Copyright © 2016 Agilebot Robotics Ltd. All rights reserved.
Instruction: 示教运动功能-单关节点动运动使用示例 / Example of Teaching motion function
- 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.errms
g}"
    )
    arm.disconnect()
    exit(1)

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

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

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

```
if ret == StatusCodeEnum.OK:
    print(
        "点动运动开始成功 / Jogging movement started successfully"
    )
else:
    print(
        f"点动运动开始失败，错误代码 / Jogging movement start failed, error code: {re
t.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: {ret}"
        )
        return

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

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

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

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

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

    logger.info(
        "IO订阅成功 / IO subscription successful"
```

```
        )

        # 单次接收消息示例 / Single message receive example
        logger.info(
            "尝试单次接收消息 / Attempting single message receive"
        )
        try:
            # 设置超时 / Set timeout
            message, ret = await asyncio.wait_for(
                arm.sub_pub.receive(), timeout=5.0
            )
            if ret == StatusCodeEnum.OK:
                logger.info(
                    f"单次接收消息成功 / Single message receive successful: {messag
e}"
                )
            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 calling extension service commands
- 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

Method Name	**extension.get_robot_ip**() -> str
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 system.
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 configu
ration 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 environment
local_robot_ip = "10.27.1.254"  # 机器人控制器IP / Robot controller IP (modify to 19
2.168.110.2 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 Non
e）/ Teach pendant IP or AP IP (modify to 192.168.110.102 or None for industrial ro
bot)

# ========== 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 above
    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 robot I
P: {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.errms
g}"
    )
    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.

Method Name	**extension.get(** `name` : str) → tuple[ExtensionInfo, StatusCodeEnum]
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).
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
#!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.errms
g}"
    )
    arm.disconnect()
    exit(1)

res, ret = arm.extension.get("MathService")
if ret == StatusCodeEnum.OK:
    print(
        "获取插件信息成功 / Get extension information successfully"
    )
    print(f"插件信息 / Extension information: {res}")
else:
    print(
        f"获取插件信息失败，错误代码 / Get extension information failed, error code:
{ret.errmsg}"
    )
```

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

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

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: 自然语言基础控制示例 / Natural Language Basic Control 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.errms
g}"
    )
    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"
    )

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

```
        )
        print("=" * 50)

        result.print_code()
        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute this cod
e:"
        )
        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"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: 自然语言顺序执行示例 / Natural Language Sequential Execution 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.errms
g}"
    )
    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 sequent
ially
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"
    )
    if result.needs_approval:
```

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

        result.print_code()
        print(
            "\n请确认是否执行此代码 / Please confirm if you want to execute this cod
e:"
        )
        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"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.

C# SDK

Prologue

Version History

Document Version	SDK Version Number	Version Date
V3.2	2.0.3.*	2025.12.17

Update Notes

Robot Version Compatibility

The SDK supports Agilebot Scara, Puma, and collaborative robot series. It must be used with devices that have the robot software installed and is compatible with the robot software versions. Some functions may return different results due to version differences.

When the SDK connects to the robotic arm, it will check the version of the robotic arm motion control software. If the version is lower than the minimum requirement, the connection will fail. If it is lower than the recommended version, a prompt indicating that the version is too low will appear. Please update the robot software version in a timely manner.

Some interfaces of the SDK only support the corresponding version of the controller. Please check the compatibility of specific interfaces.

SDK Version	Compatible Robot Software Versions	Support Status
0.1.1.X	Copper v7.5.X.X, Bronze v7.4.X.X	Discontinued
0.1.2.X	Copper v7.5.X.X, Bronze v7.4.X.X	Discontinued
0.2.0.X	Copper v7.5.X.X, Bronze v7.4.X.X	Discontinued
1.0.0.X	Copper v7.6.X.X, Bronze v7.5.X.X	Supported
2.0.X.X	Copper v7.7.X.X, Bronze v7.7.X.X	Supported

1 Introduction and Deployment

1.1 Environment Requirements

System:

- Windows 10 or later
  - x86_64 architecture
- .NET Version
  - 6.0 or higher
- .NET Framework Version
  - 4.7 or higher

1.2 Installation

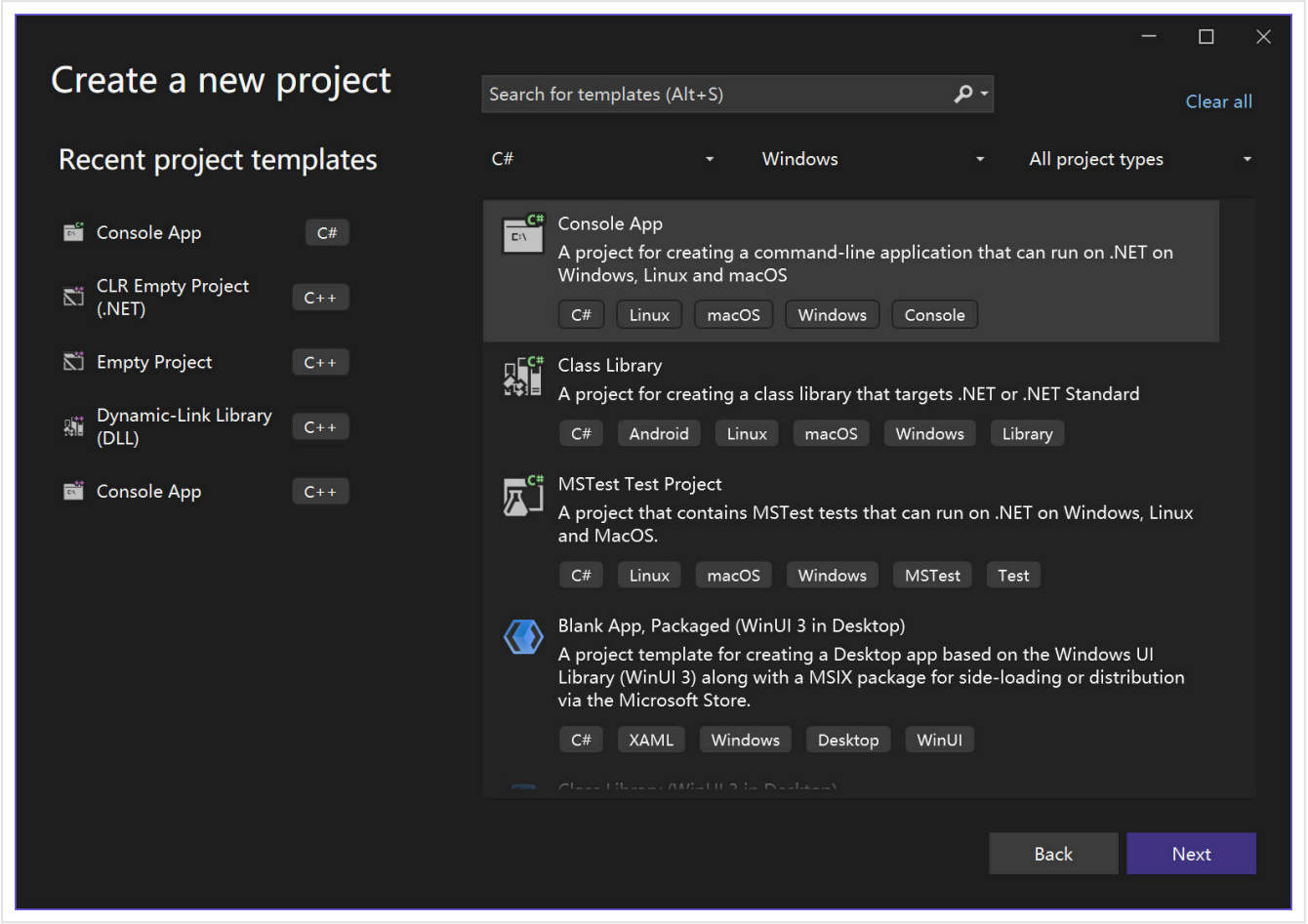
This section walks through IDE preparation, SDK installation, and the most common runtime caveats so you can start experimenting with the Agilebot SDK right away.

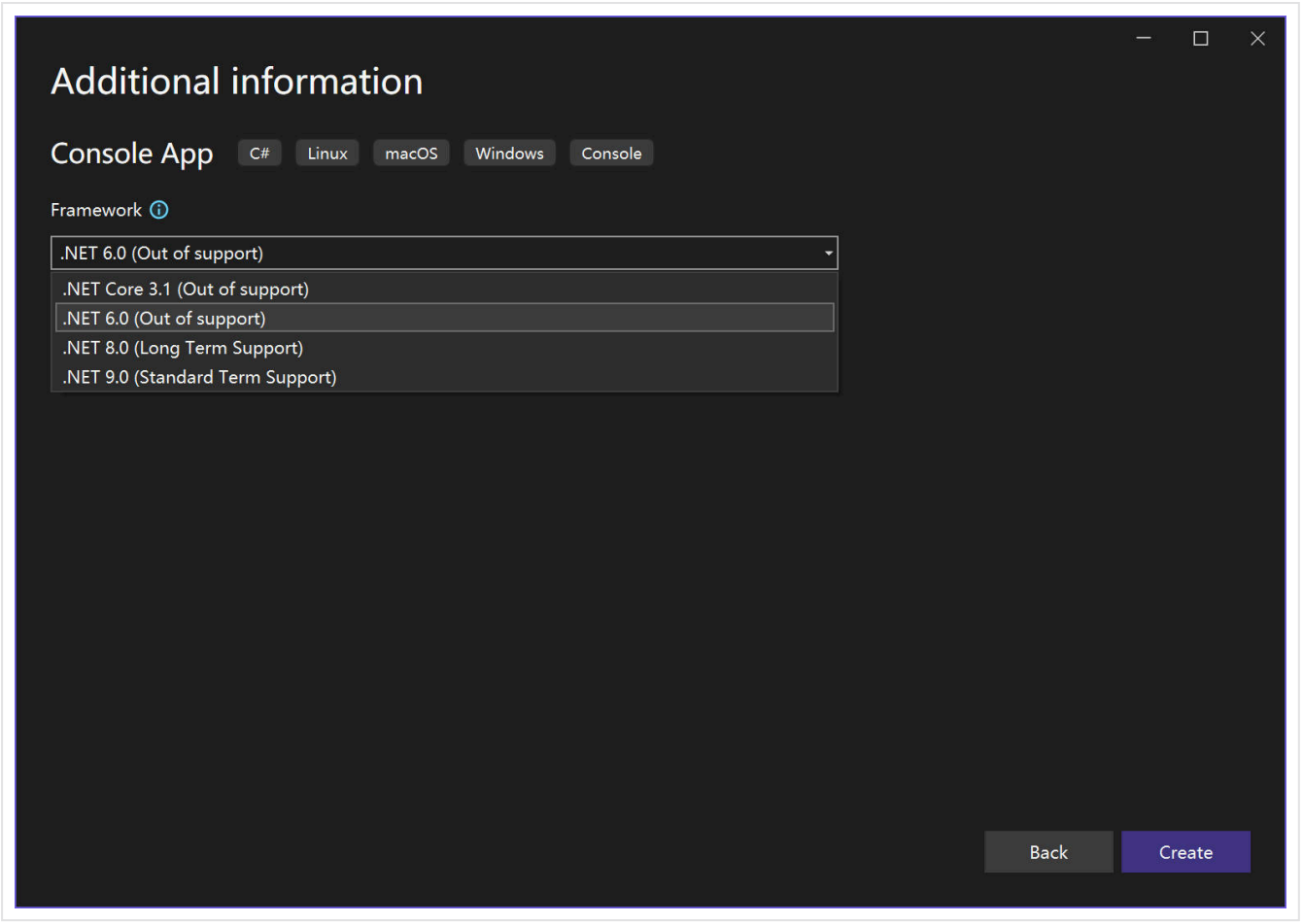
IDE Setup

1. Visual Studio is the recommended IDE for C# development. Download it from [Download Visual Studio Tools - Free Install for Windows, Mac, Linux](#).
2. After installation, launch Visual Studio and finish the initial setup (sign in, install required workloads, etc.).

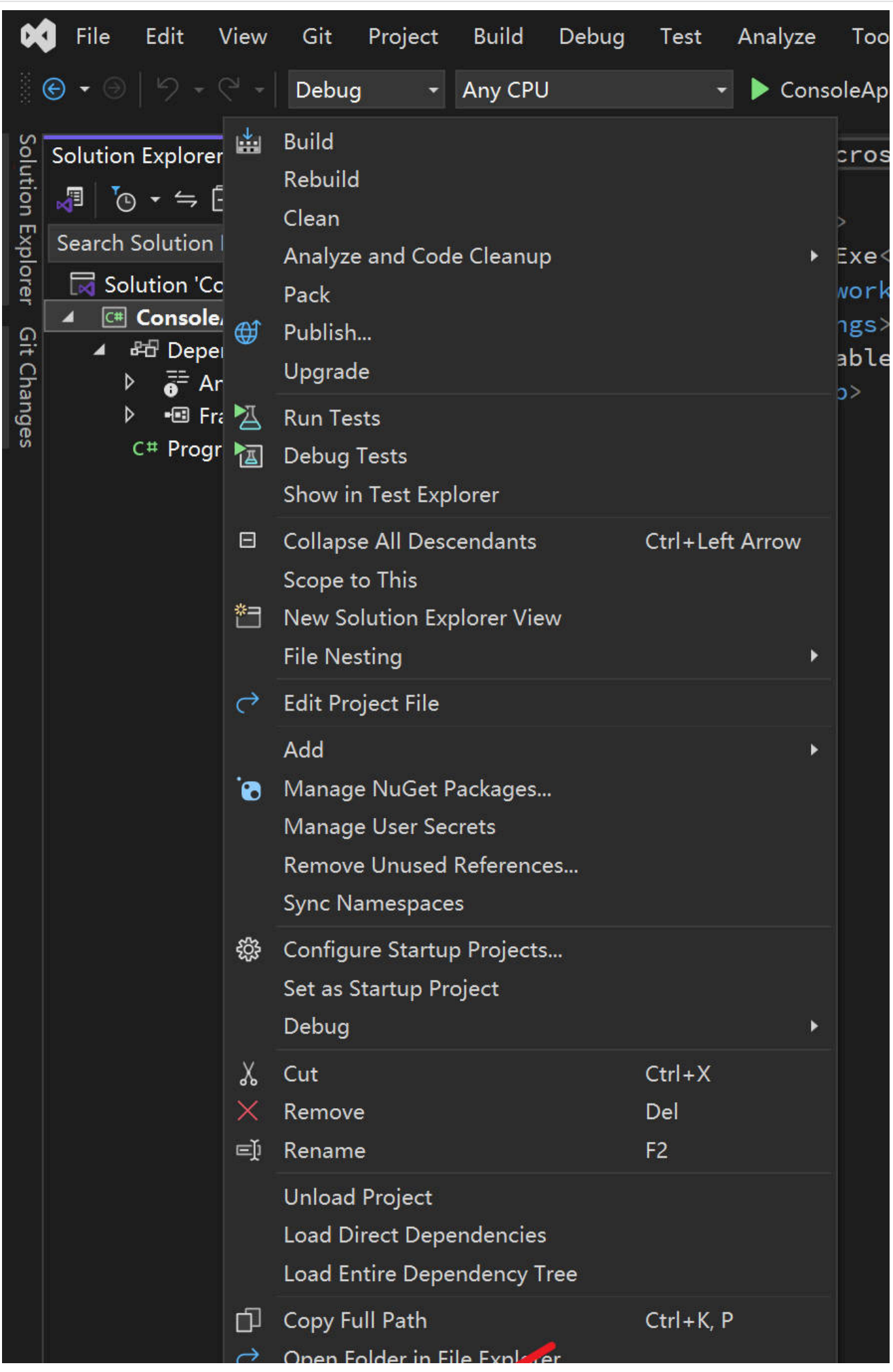
Get the SDK and Create a Project

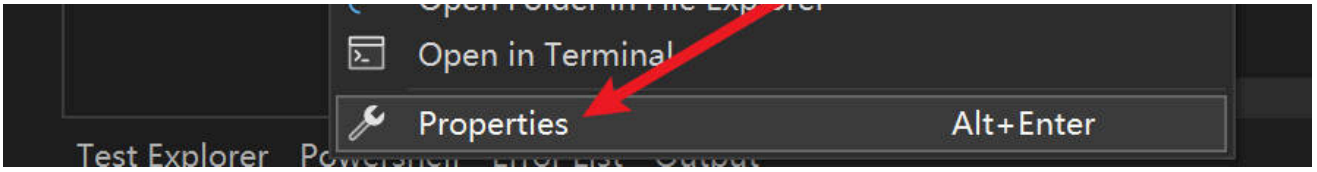
1. Create a new **C# Console App** and choose `.NET 6.0` or later as the target framework.

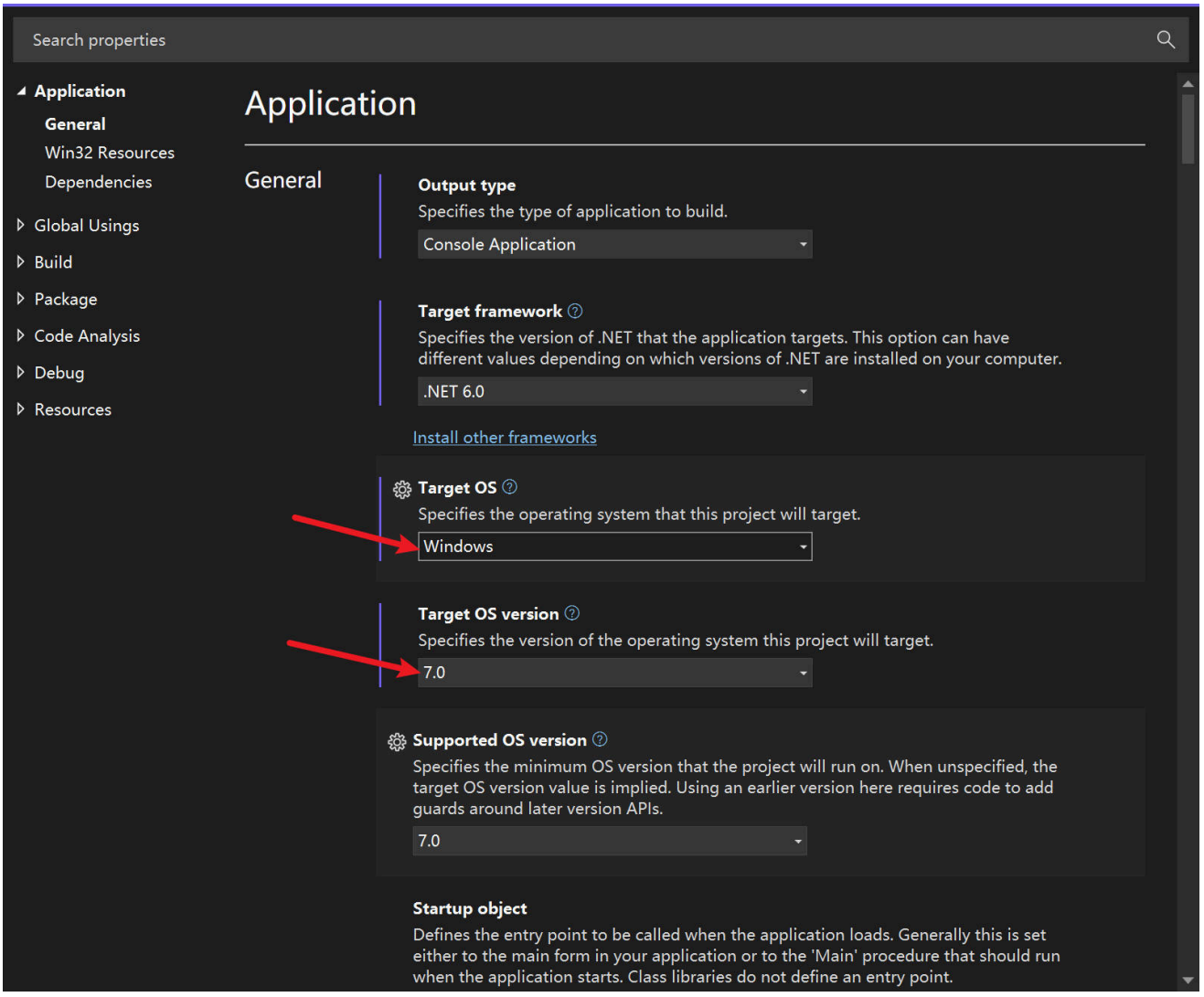




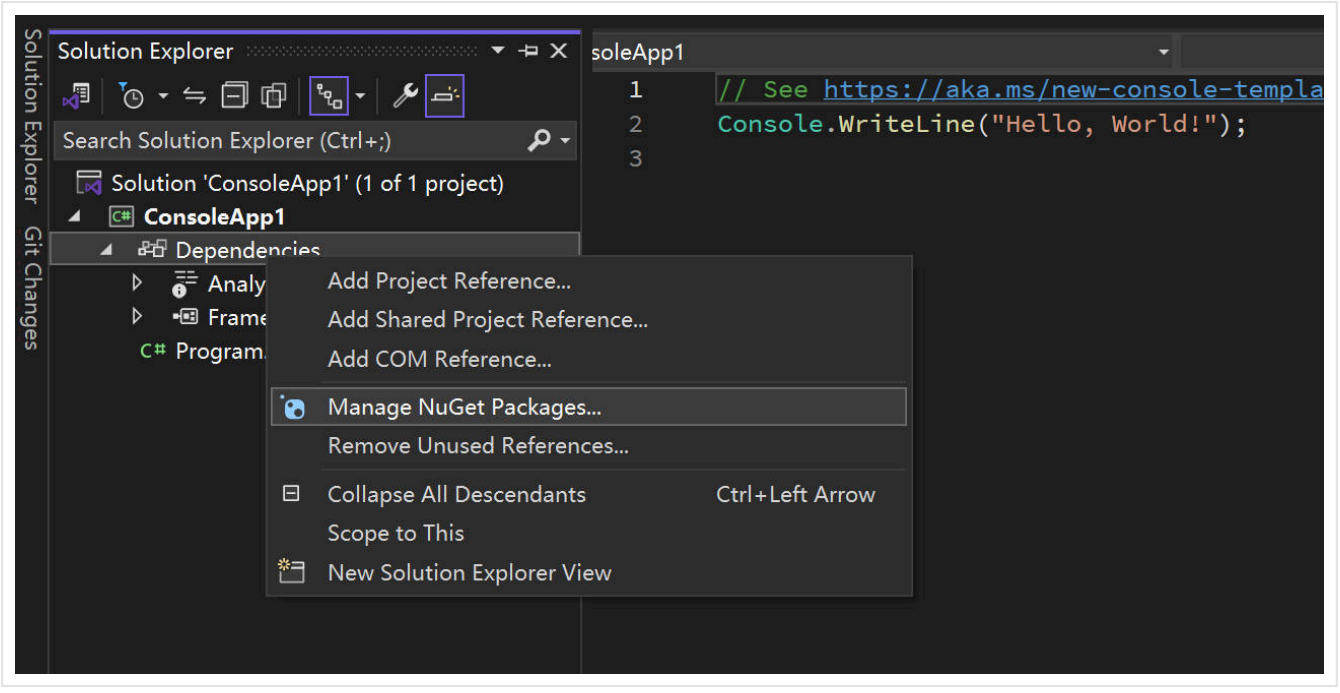
2. Open the project properties, set the target OS to **Windows**, and pick **version 7.0 or higher** to leverage the latest WinApp SDK features.

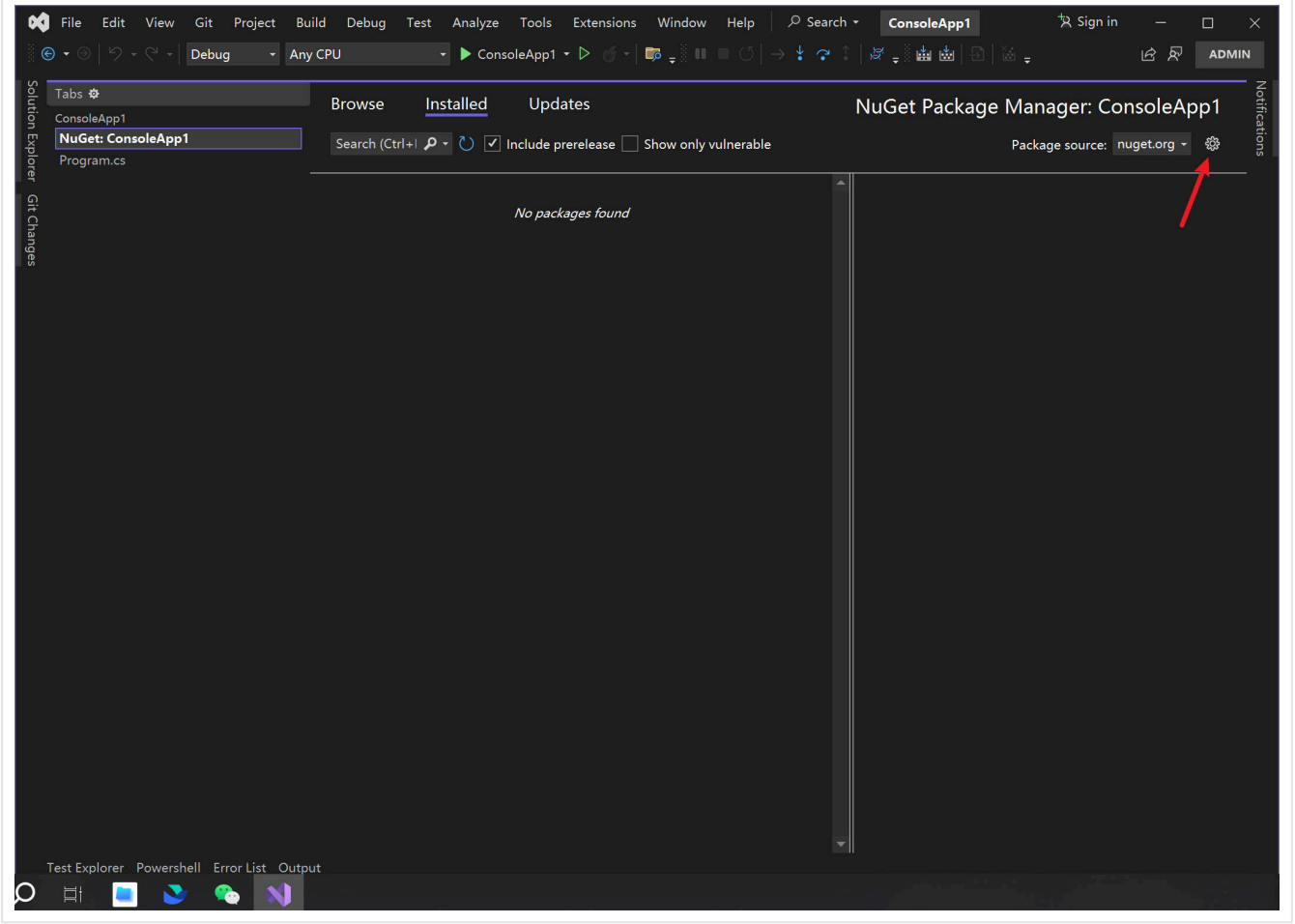


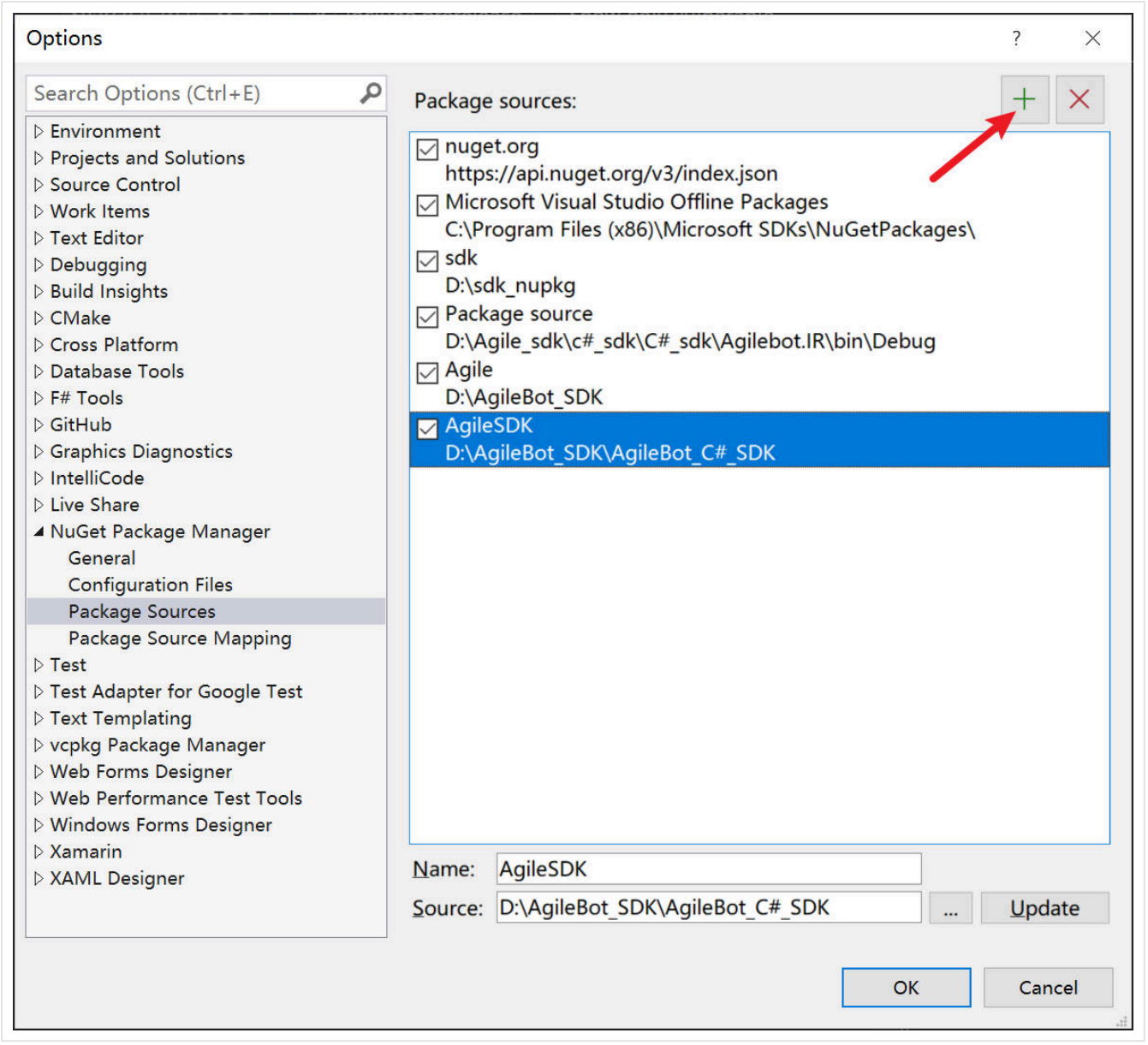


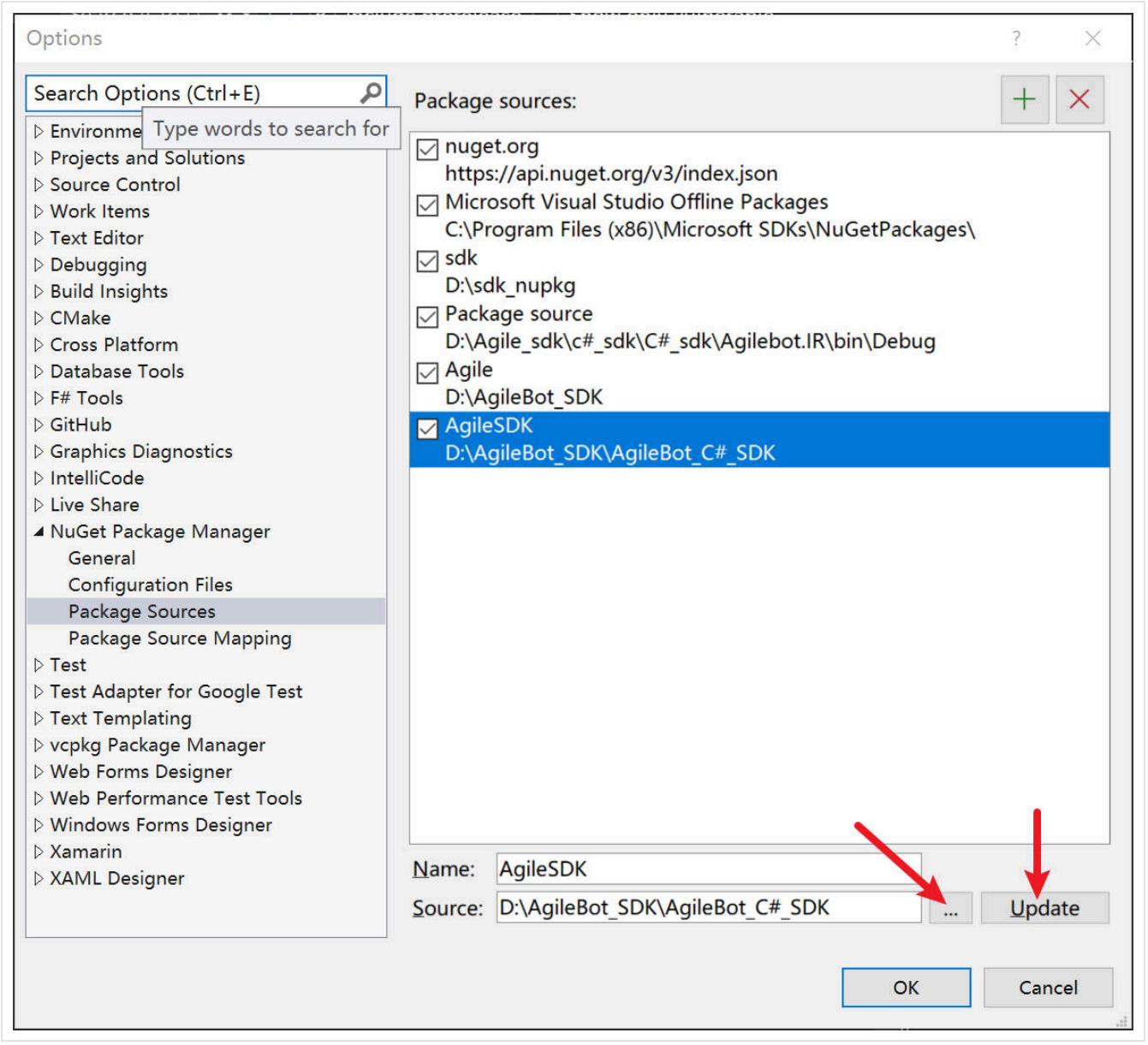


3. Navigate to `Tools > NuGet Package Manager > Package Manager Settings`, then add the directory containing the SDK package as a new package source.

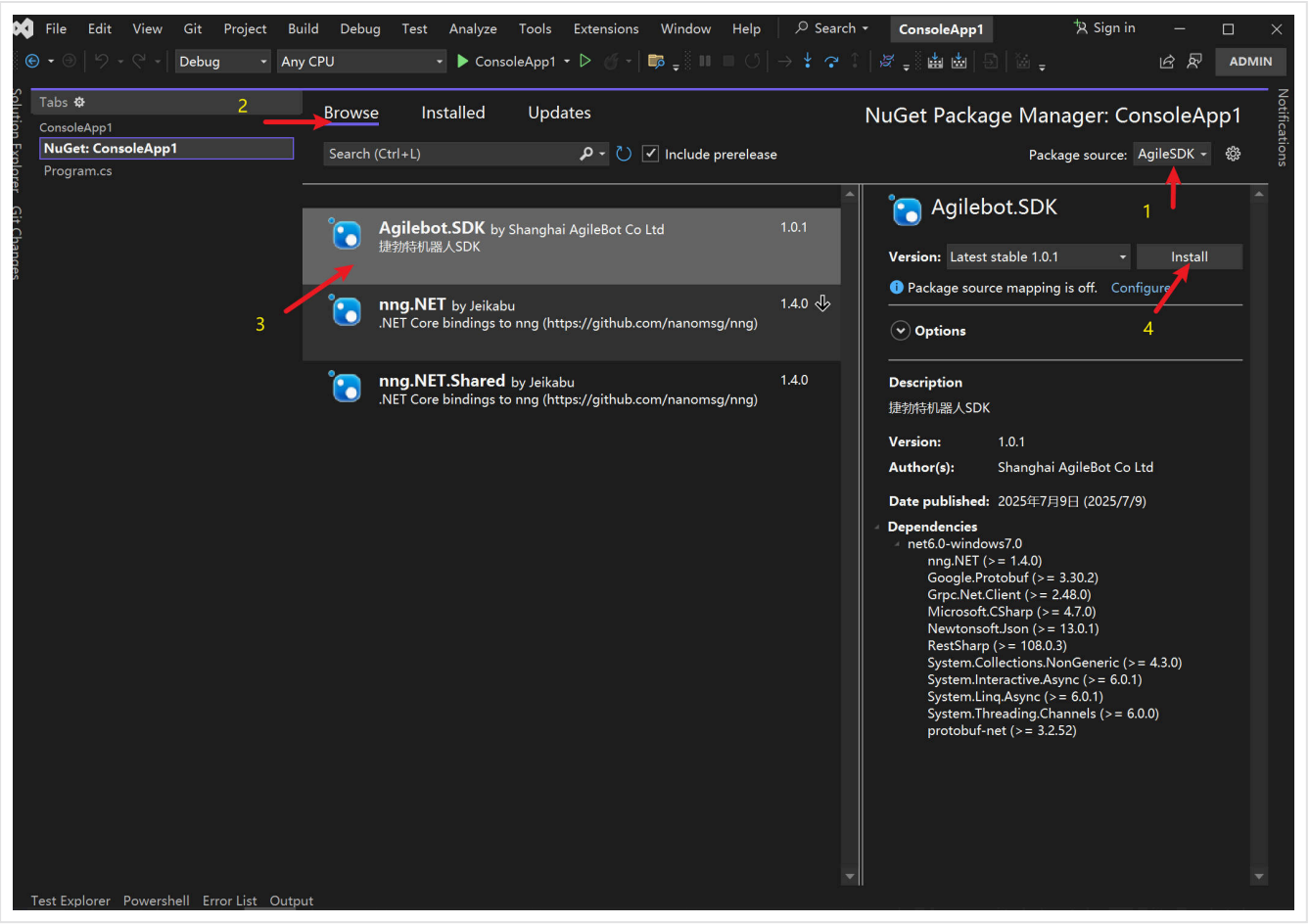








4. Switch the NuGet package source to the newly added entry and install the `Agilebot.SDK` package.



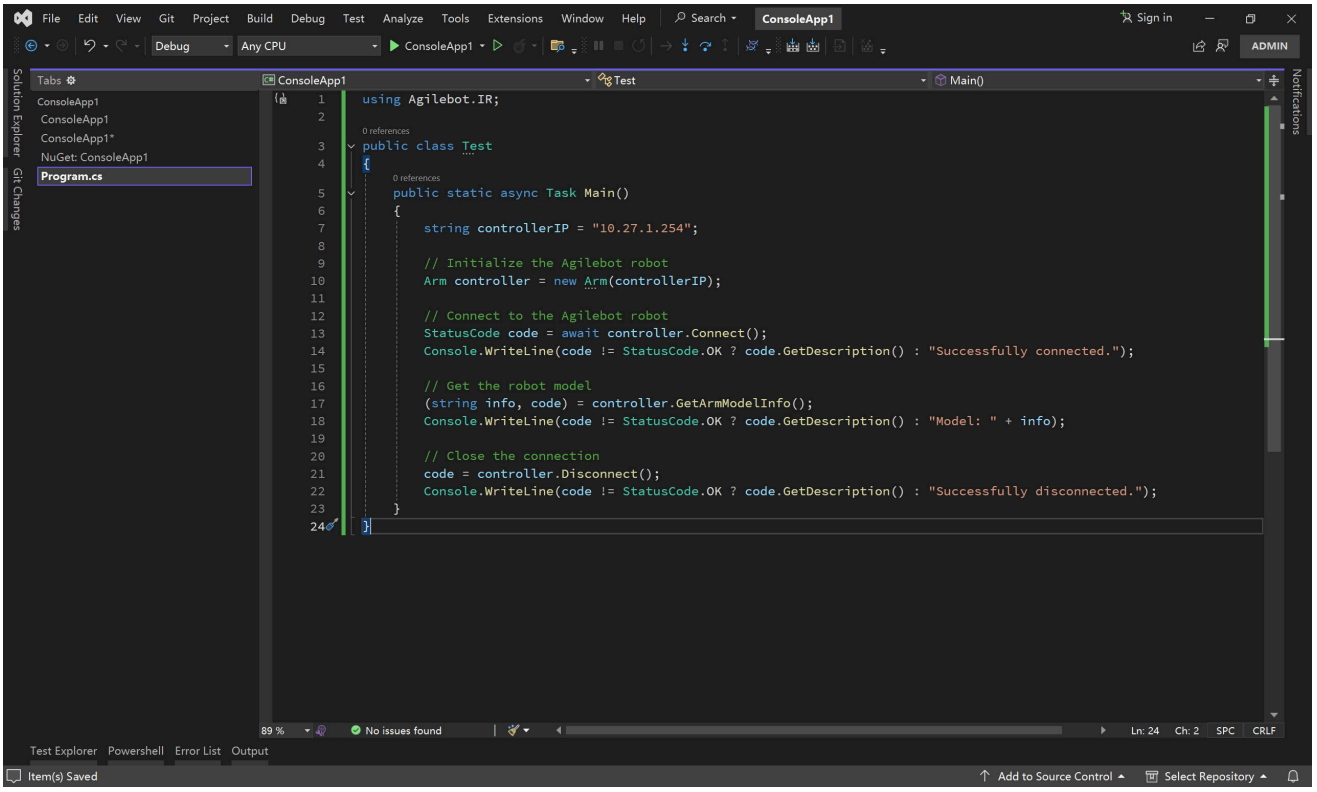
Proxy Files and Troubleshooting

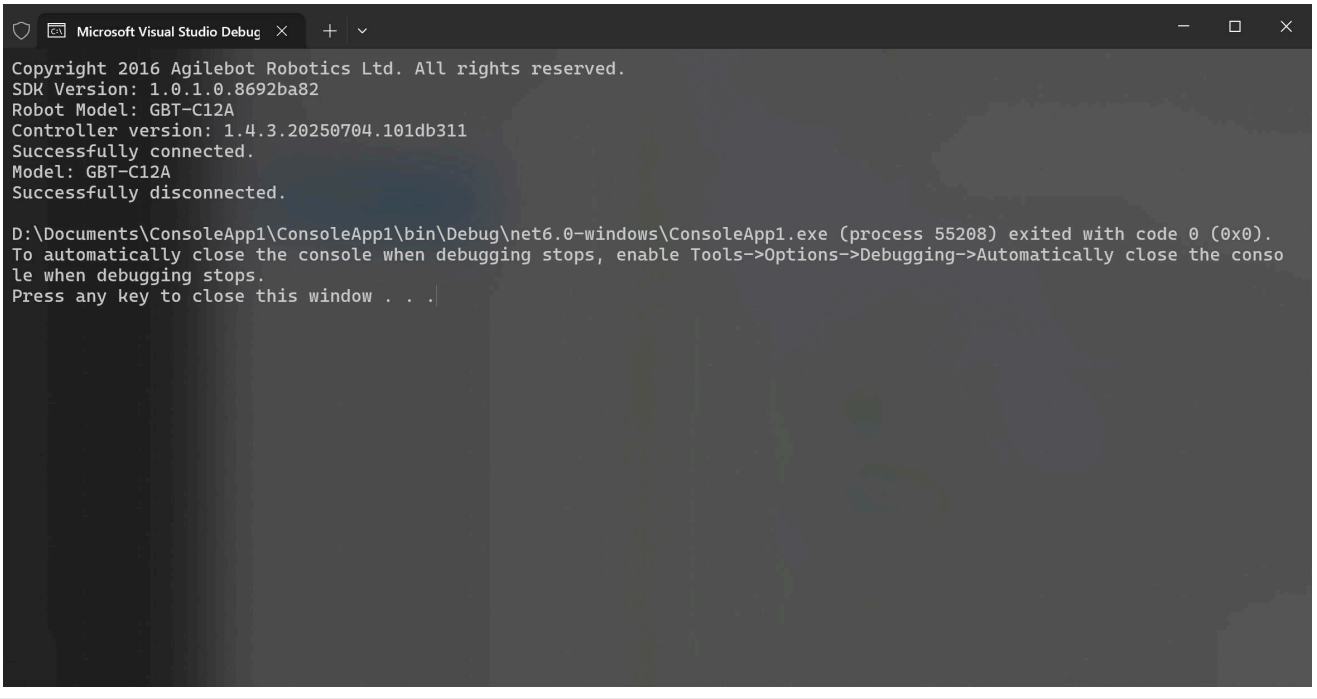
- After installing the SDK, the project automatically gains a `Tools` folder containing `controller_proxy_service_windows_amd64.exe`, which is required when using the local controller proxy. If the executable is missing, copy it manually into both the project folder and the build output directory.
- If the proxy service stays alive because the program exited unexpectedly, open Windows Task Manager, locate **controller_proxy_service_windows_amd64**, and end the process.
- While the proxy service is running, do **not** move the directory where the proxy service is located to another location.

Networking and Debugging Requirements

1. Before running your code, make sure the host PC is connected to the robot network or shares the same LAN as the robot.

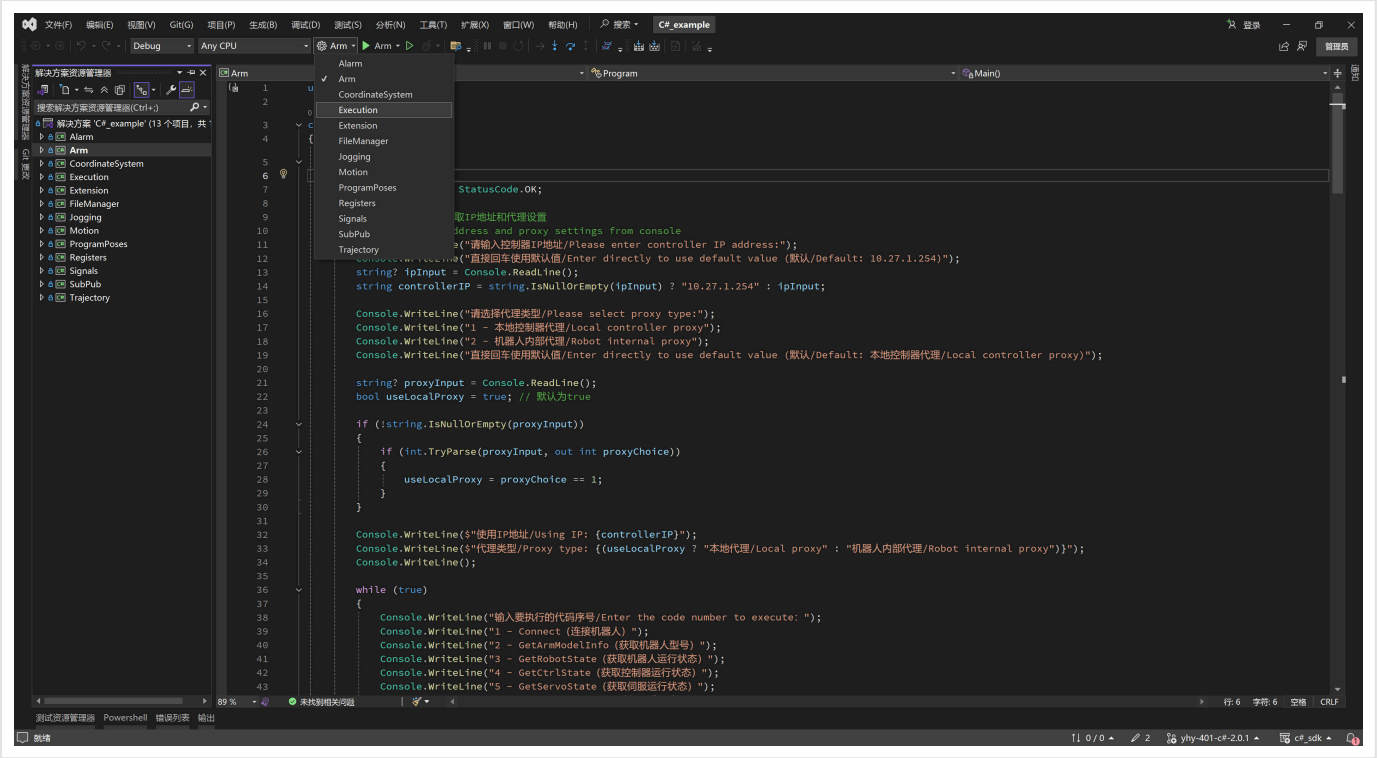
2. Keep the network stable during debugging to prevent the proxy service from dropping unexpectedly.





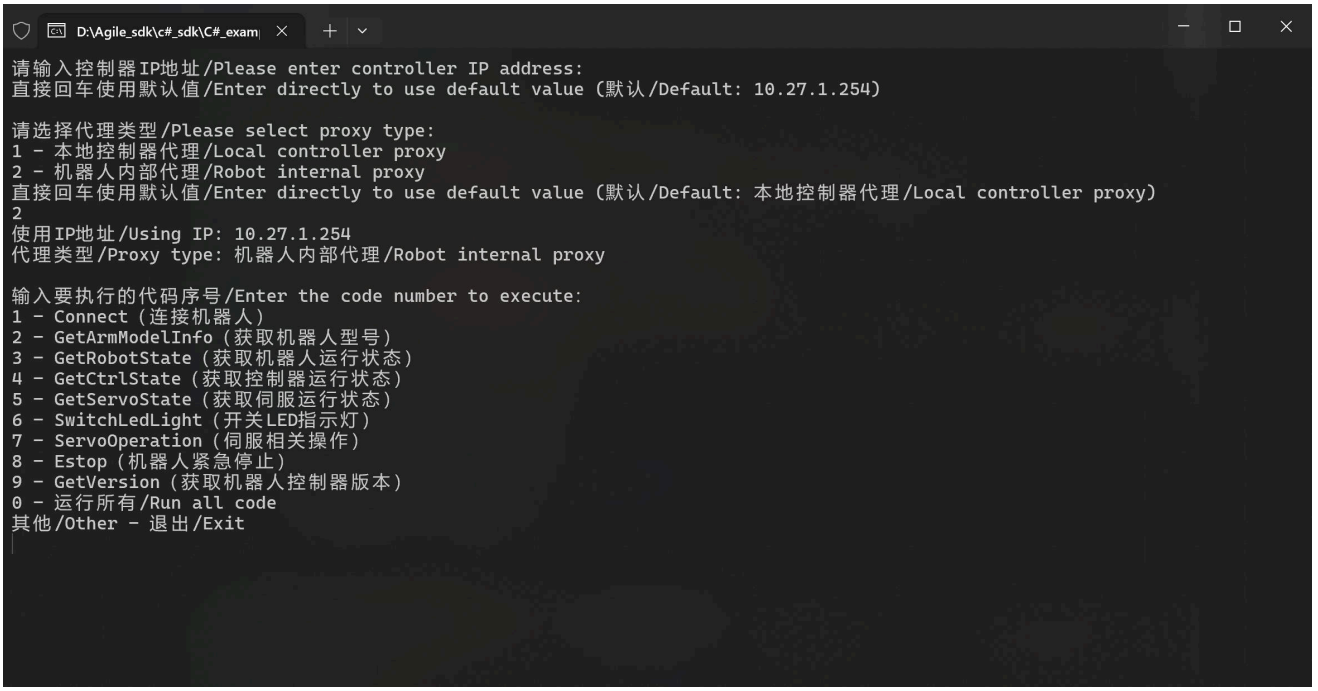
1.3 Example Program Usage

This chapter walks through the `C#_example` project that ships with the SDK. By switching the startup item you can quickly try out each major SDK class.



Run the Example

1. Open and run `C#_example` ; a console window appears automatically.
2. Enter the robot IP address when prompted.
3. Choose where to host the proxy service (robot controller or local PC).
4. Click **Start** to load the selected example.



Proxy Types

- **Robot Internal Proxy**: Uses the proxy built into the robot controller. Recommended for controller firmware **v7.7.0.0 or newer**.
- **Local Controller Proxy**: Uses the lightweight proxy shipped with the SDK and runs on the host PC. This is the only option when the controller firmware is **below v7.7.0.0**.
- For **Airbot** robots only the Robot Internal Proxy is supported; the local proxy cannot reach Airbot controllers.

2 Glossary

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

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

3 Data Structures

3.1 StatusCode

Description

Status codes returned by the interface.

Import

```c#
using Agilebot.IR;
```

Fields

Name	Enum Value	Description
OK	0	Execution successful
INCOMPATIBLE_VERSION	-1	Incompatible version
TIMEOUT	-3	Connection timeout
INTERFACE_NOT_IMPLEMENTED	-4	Interface not implemented
INDEX_OUT_OF_RANGE	-5	Index out of range
UNSUPPORTED_FILETYPE	-6	Unsupported file type
UNSUPPORTED_PARAMETER	-7	Unsupported robot parameter
UNSUPPORTED_SIGNALTYPE	-8	Unsupported IO signal type
PROGRAM_NOT_FOUND	-9	Program not found
PROGRAM_POSE_NOT_FOUND	-10	Program pose information not found

Name	Enum Value	Description
WRITE_PROGRAM_FAILED	-11	Failed to update program pose information
GET_ALARM_CODE_FAILED	-12	Failed to access alarm service to get alarm code
WRONG_POSITION_INFO	-13	Controller returns incorrect position information
UNSUPPORTED_TRA_TYPE	-14	Unsupported motion type
FILE_NOT_FOUND	-15	File or folder not found
FILE_ALREADY_EXIST	-16	File already exists
GET_ALARM_DESC_FAILED	-17	Failed to get alarm information based on alarm code
RESET_ALARM_ERRORS_FAILED	-18	Failed to reset alarm information
GET_ALL_ALARMS_FAILED	-19	Failed to get all alarm information
WRONG_DATA_FORMAT	-20	Incorrect data format received
CONNECT_FAILED	-21	Initialization connection failed, please check IP address or control cabinet service
POSE_INDEX_DUPLICATED	-23	Pose index duplicated
CONTROLLER_ERROR	-254	Controller error, please contact the developer
OTHER_REASON	-255	Other reasons

3.2 RobotState

Description

Robot operation status.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Name	Enum Value	Description
WRONG_DATA	-1	Unknown state
ROBOT_IDLE	0	Robot idle
ROBOT_RUNNING	1	Robot running
ROBOT_TEACHING	2	Robot teaching
ROBOT_IDLE_TO_RUNNING	101	Robot intermediate state, idle to running
ROBOT_IDLE_TO_TEACHING	102	Robot intermediate state, idle to teaching
ROBOT_RUNNING_TO_IDLE	103	Robot intermediate state, running to idle
ROBOT_TEACHING_TO_IDLE	104	Robot intermediate state, teaching to idle

---

3.3 CtrlState

Description

Controller operation status.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Name	Enum Value	Description
WRONG_DATA	-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	Controller intermediate state, any to emergency stop
CTRL_ESTOP_TO_ENGAGED	102	Controller intermediate state, emergency stop to enabled
CTRL_ESTOP_TO_TERMINATED	103	Controller intermediate state, emergency stop to terminated

3.4 ServoState

Description

Servo controller status.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Name	Enum Value	Description
WRONG_DATA	-1	Unknown servo controller state
SERVO_IDLE	1	Servo controller idle

Name	Enum Value	Description
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 TransformStatusEnum

Description

Enum for offline trajectory file conversion status.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Name	Enum Value	Description
TRANSFORM_START	0	Conversion task started
TRANSFORM_RUNNING	1	Conversion task in progress
TRANSFORM_SUCCESS	2	Conversion task completed successfully
TRANSFORM_FAILED	3	Conversion task failed
TRANSFORM_NOT_FOUND	4	Conversion task not found
TRANSFORM_UNKNOWN	-1	Unknown conversion task status

3.6 PayloadInfo

Description

The `PayloadInfo` class is used to store the robot's payload information, including payload ID, weight, center of mass, and moment of inertia. This information is crucial for kinematic and dynamic analysis of the robot under load conditions, especially for path planning and torque calculation.

Import

```c#
using Agilebot.IR.Motion;
```

Properties

Property	Type	Description
`Id`	`uint`	Payload ID, used to uniquely identify different payload configurations
`Comment`	`string`	Comment, used to describe additional information about the payload
`Weight`	`double`	Payload weight (unit: kilograms)
`MassCenter`	`MassCenter`	Payload center of mass (X, Y, Z coordinates)
`InertiaMoment`	`InertiaMoment`	Payload moment of inertia (LX, LY, LZ)

Example

```c#
PayloadInfo payload = new PayloadInfo
{
    Id = 1,
    Comment = "Sample Payload",
    Weight = 5.0,
    MassCenter = new MassCenter { X = 10.0, Y = 20.0, Z = 30.0 },
```

```
    InertiaMoment = new InertiaMoment { LX = 0.1, LY = 0.2, LZ = 0.3 }
};
```

3.6.1 MassCenter

Description

The `MassCenter` class is used to represent the center of mass of the payload, containing the X, Y, and Z coordinates. The center of mass is the geometric center of the payload in space and is important for robot motion control and torque calculation.

Import

```c#
using Agilebot.IR.Motion;
```

Properties

Property	Type	Description
`X`	`double`	X-coordinate of the center of mass (unit: millimeters)
`Y`	`double`	Y-coordinate of the center of mass (unit: millimeters)
`Z`	`double`	Z-coordinate of the center of mass (unit: millimeters)

3.6.2 InertiaMoment

Description

The `InertiaMoment` class is used to represent the moment of inertia of the payload, containing the LX, LY, and LZ components. The moment of inertia represents the payload's resistance to rotational changes and is important for robot dynamics analysis and control.

Import

```c#
using Agilebot.IR.Motion;
```

Properties

Property	Type	Description
LX	double	X-component of the moment of inertia (unit: kilograms·millimeters^2)
LY	double	Y-component of the moment of inertia (unit: kilograms·millimeters^2)
LZ	double	Z-component of the moment of inertia (unit: kilograms·millimeters^2)

3.7 TransformState

Description

Enum for offline trajectory file conversion status.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Enum Value	Value	Description
TRANSFORM_START	0	Conversion task started
TRANSFORM_RUNNING	1	Conversion task in progress
TRANSFORM_SUCCESS	2	Conversion task completed successfully
TRANSFORM_FAILED	3	Conversion task failed
TRANSFORM_NOT_FOUND	4	Conversion task not found
TRANSFORM_UNKNOWN	-1	Data error, unknown status

3.8 TCSType

Description

TCS coordinate system type.

Import

```c#
using Agilebot.IR.Types;
```

Fields

Name	Enum Value	Description
WRONG_TYPE	-1	Incorrect type
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

3.9 MotionPose

Description

Describes the robot's position structure. In the coordinate data, the distance in the XYZ direction is measured in millimeters (mm), and the angle data is measured in degrees (°). In some versions,

the angle information is in radians; see the function list return result description for details.

Import

```c#
using Agilebot.IR.Motion;
```

Properties

Property	Type	Description
CartData	BaseCartData	Cartesian data
Joint	Joint	Joint data
Pt	PoseType	Position type, defaults to Unknown

Example

```c#
MotionPose motionPose = new MotionPose();
motionPose.Pt = PoseType.Cart;
motionPose.CartData.Position = new Position{
    X  = 300,
    Y = 300,
    Z = 300,
    A = 0,
    B = 0,
    C = 0
};
motionPose.CartData.Posture = new Posture{
    WristFlip = 1,
    ArmUpDown = 1,
    ArmBackFront = 1,
    ArmLeftRight = 1,
    TurnCircle = new List<int>(9){0,0,0,0,0,0,0,0,0}
};

MotionPose motionPose2 = new MotionPose();
motionPose2.Pt = PoseType.Joint;
```

```
motionPose2.Joint = new Joint{
    J1 = 0,
    J2 = 0,
    J3 = 60,
    J4 = 60,
    J5 = 0,
    J6 = 0
};
```

3.10 BaseCartData

Description

Describes the robot's position and posture information in the Cartesian coordinate system. The spatial coordinates are measured in millimeters (mm), and the posture information includes wrist and arm postures as well as the rotation counts of each axis.

Import

c#

```
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
Position	Position	Robot's spatial coordinates (X, Y, Z, A, B, C)
Posture	Posture	Robot's posture information (wrist, arm posture, and axis rotation counts)

Example

c#

```
BaseCartData cartData = new BaseCartData();
cartData.Position.X = 100.0;
cartData.Position.Y = 200.0;
```

```
cartData.Position.Z = 300.0;
cartData.Posture.ArmUpDown = 1;
cartData.Posture.ArmBackFront = -1;
Console.WriteLine(cartData.ToString());
```

3.10.1 Position

Description

Describes the robot's position and rotation angle coordinates in the Cartesian coordinate system. The distance in the X, Y, Z directions is measured in millimeters (mm), and the angles in the A, B, C directions are measured in degrees (°).

Import

c#

```
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
X	double	Distance in the X direction of the Cartesian coordinate system (unit: millimeters)
Y	double	Distance in the Y direction of the Cartesian coordinate system (unit: millimeters)
Z	double	Distance in the Z direction of the Cartesian coordinate system (unit: millimeters)
A	double	Angle in the A direction of the Cartesian coordinate system (unit: degrees)
B	double	Angle in the B direction of the Cartesian coordinate system (unit: degrees)
C	double	Angle in the C direction of the Cartesian coordinate system (unit: degrees)

Example

```c#
Position position = new Position();
position.X = 100.0;
position.Y = 200.0;
position.Z = 300.0;
position.A = 45.0;
position.B = 30.0;
position.C = 60.0;
Console.WriteLine(position.ToString());
```

3.10.2 Posture

Description

Describes the robot's posture information, including wrist and arm postures as well as the rotation counts of each axis. Posture information is used to define the robot's specific posture in space.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
WristFlip	int	Wrist flip posture. Range: -1, 0, 1. For a 6-axis robot J5 joint config: 1 = wrist flipped down, -1 = wrist flipped up.
ArmUpDown	int	Arm up/down posture. Range: -1, 0, 1. For a 6-axis robot J3 joint config: 1 = arm above (forward condition: joint-3 above the line from joint-4 to joint-2 and joint-3 angle < 0), -1 = arm below (joint-3 angle > 0).
ArmBackFront	int	Arm front/back posture. Range: -1, 0, 1. For a 6-axis robot J1 joint config: 1 = arm in front (collaborative robot facing forward, joint-2 on the left side of joint-1), -1 = arm behind (joint-2 on the right side of joint-1).

Property	Type	Description
ArmLeftRight	int	Arm left/right posture. Range: -1, 0, 1. For a 4-axis SCARA robot J2 joint config: 1 = SCARA arm on the right, -1 = SCARA arm on the left.
TurnCircle	List<int>	Multi-turn counts for each axis. Range: -1, 0, 1. When the axis is at 0°, turn count = 0. During linear or circular moves the controller auto-selects the turn count closest to the start pose, so the final value may differ from the taught posture. For axes 1, 4, 5, 6: ≥ 180° → value ≥ 1; -179.99° ~ 179.99° → 0; ≤ -180° → value ≤ -1.

Example

```c#
Posture posture = new Posture();
posture.TurnCircle = new List<int>(9){0,0,0,0,0,0,0,0,0};
posture.WristFlip = 1;
posture.ArmUpDown = 1;
posture.ArmBackFront = -1;
posture.ArmLeftRight = 1;
Console.WriteLine(posture.ToString());
```

3.11 Joint

Description

Describes the angle data of each robot joint. Each joint angle value is used to define the robot's specific position in joint space. The angle unit is typically degrees (°), but the specific unit should be confirmed based on the actual robot system.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
J1	double	Angle of the robot's first joint
J2	double	Angle of the robot's second joint
J3	double	Angle of the robot's third joint
J4	double	Angle of the robot's fourth joint
J5	double	Angle of the robot's fifth joint
J6	double	Angle of the robot's sixth joint
J7	double	Angle of the robot's seventh joint
J8	double	Angle of the robot's eighth joint
J9	double	Angle of the robot's ninth joint

Example

c#

```
Joint joint = new Joint();
joint.J1 = 45.0;
joint.J2 = 30.0;
joint.J3 = 60.0;
joint.J4 = 90.0;
joint.J5 = 120.0;
joint.J6 = 135.0;
joint.J7 = 150.0;
joint.J8 = 180.0;
joint.J9 = 225.0;
Console.WriteLine(joint.ToString());
```

Notes

- The unit of joint angles is typically degrees (°), but some robot systems may use radians (rad). Please confirm the unit based on the actual robot system documentation.
- The range of joint angles is usually limited by the robot hardware. Exceeding the range may cause errors or damage the equipment.

3.12 PoseType

Description

Defines the type of robot pose data, used to distinguish whether the data is joint angle data, Cartesian space coordinates, or unknown type. This enum is used to identify the format of robot pose data so that different types of data can be correctly processed in the program.

Import

```c#
using Agilebot.IR.Types;
```

Enum Values

Enum Value	Description
`Unknown`	Unknown type, indicating the pose data type is not defined
`Joint`	Joint angle data type, indicating the data is joint angles
`Cart`	Cartesian space coordinate data type, indicating the data is Cartesian coordinates

3.13 DHparam

Description

The `DHparam` class is used to describe the robot link parameters based on the Denavit-Hartenberg parameters (D-H parameters). These parameters are used to define the geometric relationships between robot joints and are the basis for robot kinematics and dynamics analysis.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`id`	`uint`	Unique identifier for the link, used to distinguish different links
`a`	`double`	Link length, representing the axial distance between adjacent joints (unit: millimeters)
`alpha`	`double`	Link twist angle, representing the angle between adjacent joint axes (unit: degrees or radians)
`d`	`double`	Joint distance, representing the distance along the current joint axis to the next joint (unit: millimeters)
`offset`	`double`	Joint angle offset, representing the initial angle offset of the joint (unit: degrees or radians)

Constructor

```c#
public DHparam(uint id, double d, double a, double alpha, double offset)
```

Notes

- **Unit consistency**: The units of `a` and `d` should be consistent (typically millimeters), and the units of `alpha` and `offset` should also be consistent (typically degrees or radians).
- **Angle unit**: In some robot systems, the angle unit may be radians instead of degrees. Please confirm and unify the units based on actual requirements.
- **D-H parameter definition**: The definition of D-H parameters depends on the specific robot model and coordinate system conventions. When using the `DHparam` class, ensure that the parameter definitions are consistent with the robot's actual geometric structure.

3.14 CartStatus

Description

The `CartStatus` class is used to represent the status of each axis in the Cartesian coordinate system. The status of each axis is represented by a boolean value, with `true` indicating the axis is available and `false` indicating it is not. This status class is commonly used in robot motion control to determine whether a particular axis can function properly.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`X`	`bool`	Status of the X direction, defaults to `true` (available)
`Y`	`bool`	Status of the Y direction, defaults to `true` (available)
`Z`	`bool`	Status of the Z direction, defaults to `true` (available)
`A`	`bool`	Status of the A direction, defaults to `true` (available)
`B`	`bool`	Status of the B direction, defaults to `true` (available)
`C`	`bool`	Status of the C direction, defaults to `true` (available)

3.15 JointStatus

Description

The `JointStatus` class is used to represent the status of each robot joint. The status of each joint is represented by a boolean value, with `true` indicating the joint is available and `false` indicating it is not. This status class is commonly used in robot motion control to determine whether a particular joint can function properly.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`J1`	`bool`	Status of Joint 1, defaults to `true` (available)
`J2`	`bool`	Status of Joint 2, defaults to `true` (available)
`J3`	`bool`	Status of Joint 3, defaults to `true` (available)
`J4`	`bool`	Status of Joint 4, defaults to `true` (available)
`J5`	`bool`	Status of Joint 5, defaults to `true` (available)
`J6`	`bool`	Status of Joint 6, defaults to `true` (available)
`J7`	`bool`	Status of Joint 7, defaults to `true` (available)
`J8`	`bool`	Status of Joint 8, defaults to `true` (available)
`J9`	`bool`	Status of Joint 9, defaults to `true` (available)

3.16 DragStatus

Description

The `DragStatus` class is used to represent the drag status of the robot arm, including the status of the Cartesian coordinate system and the joints. Additionally, it includes a flag

`IsContinuousDrag` to indicate whether the robot is in continuous drag mode. This status class is commonly used in robot drag control to determine the current drag mode and the status of each axis/joint.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`CartStatus`	`CartStatus`	Status of the Cartesian coordinate system
`JointStatus`	`JointStatus`	Status of the joints
`IsContinuousDrag`	`bool`	Whether the robot is in continuous drag mode, defaults to `false`

Constructor

```c#
public DragStatus()
```

- Initializes `CartStatus` and `JointStatus` , and sets `IsContinuousDrag` to `false` .

Example

```c#
DragStatus dragStatus = new DragStatus();
dragStatus.CartStatus.X = false; // X-axis unavailable
dragStatus.JointStatus.J3 = false; // Joint 3 unavailable
dragStatus.IsContinuousDrag = true; // Set to continuous drag mode
Console.WriteLine($"X-axis status: {dragStatus.CartStatus.X}, Joint 3 status: {drag
Status.JointStatus.J3}, Is continuous drag: {dragStatus.IsContinuousDrag}");
```

3.17 ProgramPose

Description

The `ProgramPose` class is used to represent a pose (position and orientation) in a program, which can be joint coordinates or Cartesian coordinates. This class includes a unique identifier for the pose, data (joint or Cartesian coordinate information), name, and comment. This class facilitates the management and manipulation of pose information in robot programs.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`Id`	`int`	Unique identifier for the pose
`PoseData`	`ProgramPoseData`	Pose data, including joint or Cartesian coordinate information
`Name`	`string`	Name of the pose
`Comment`	`string`	Comment for the pose

Constructor

```c#
public ProgramPose()
```

- Initializes `Id` , `PoseData` , `Name` , and `Comment` .

Example

```c#
ProgramPose programPose = new ProgramPose();
programPose.Id = 1; // Set the unique identifier for the pose
```

```
programPose.PoseData = new ProgramPoseData(); // Create pose data
programPose.Name = "Pose1"; // Set the name of the pose
programPose.Comment = "This is a sample pose"; // Set the comment for the pose
Console.WriteLine($"Pose ID: {programPose.Id}, Name: {programPose.Name}, Comment:
{programPose.Comment}");
```

3.17.1 ProgramPoseData

Description

The `ProgramPoseData` class is used to represent pose data in a program, including Cartesian space coordinates and posture information, joint angle information, and pose type. This class facilitates the storage and management of specific pose data.

Import

c#

```
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`CartData`	`ProgramCartData`	Cartesian data
`Joint`	`Joint`	Joint data
`Pt`	`PoseType`	Pose type, defaults to Unknown

3.17.2 ProgramCartData

Description

The `ProgramCartData` class is used to represent the Cartesian coordinate system data in a program. It references the `BaseCartData` class to include spatial coordinates and posture information, and determines the coordinate system type through the values of `Uf` and `Tf` . `Uf` represents the User Frame, and `Tf` represents the Tool Frame. If the values of `Uf` and `Tf` are

`-1` , it indicates the use of the system's default coordinate system. This class is used in robot programming to define and manage pose information in Cartesian space.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`BaseCart`	`BaseCartData`	Robot's Cartesian position and posture information
`Uf`	`int`	User Frame, `-1` indicates the use of the system's coordinate system
`Tf`	`int`	Tool Frame, `-1` indicates the use of the system's coordinate system

---

3.18 FileType

Description

The `FileType` enum is used to define the types of files allowed for upload. It distinguishes between different types of robot program files based on their source and format. This enum is used in the robot programming environment for file management, upload, and program parsing, helping the system correctly identify and process different types of program files.

Import

```c#
using Agilebot.IR.Types;
```

Enum Values

Enum Value	Description
`UserProgram`	Program files generated by the user through point selection, each program includes `.xml` and `.json` files.
`BlockProgram`	Program files generated by the user through block programming, each program includes `.block` , `.xml` , and `.json` files.
`TrajectoryProgram`	Offline trajectory program files, typically used for path planning in offline programming.

3.19 SignalType

Description

The `SignalType` enum is used to define the types of signals supported in the robot system. It distinguishes between various digital and analog signals based on their purpose and source. This enum is used in the robot control system for signal configuration, signal processing, and logic judgment, helping the system accurately identify and manage different types of signals.

Import

```c#
using Agilebot.IR.Types;
```

Enum Values

Enum Value	Description
`DI`	Digital Input, used to receive external digital signals.
`DO`	Digital Output, used to control external devices or actuators.
`RI`	Robot Input, used to receive digital signals from the robot's wrist.
`RO`	Robot Output, used to control actuators on the robot's wrist.
`UI`	User Input, used to receive user-defined digital signals.

Enum Value	Description
`UO`	User Output, used to output user-defined digital signals.
`TDI`	Tool Digital Input, used to receive digital signals from the tool end.
`TDO`	Tool Digital Output, used to control actuators on the tool end.
`GI`	Group Input, used to receive a combination of digital signals.
`GO`	Group Output, used to output a combination of digital signals.
`AI`	Analog Input, used to receive continuous analog signals.
`AO`	Analog Output, used to output continuous analog signals.
`TAI`	Tool Analog Input, used to receive analog signals from the tool end.

3.20 PoseRegister

Description

The `PoseRegister` class is used to represent a pose (position and orientation) in a PR register, which can be joint coordinates or Cartesian coordinates. This class includes a unique identifier for the pose, data (joint or Cartesian coordinate information), name, and comment. This class facilitates the management and manipulation of pose information in robot programs.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`Id`	`int`	Unique identifier for the pose

Property	Type	Description
PoseData	PoseRegisterData	Pose data, including joint or Cartesian coordinate information
Name	string	Name of the pose
Comment	string	Comment for the pose

Constructor

```c#
public PoseRegister()
```

- Initializes `Id` , `PoseData` , `Name` , and `Comment` .

Example

```c#
PoseRegister pose = new PoseRegister();
pose.Id = 1; // Set the unique identifier for the pose
pose.PoseData = new PoseRegisterData(); // Create pose data
pose.Name = "Pose1"; // Set the name of the pose
pose.Comment = "This is a sample pose"; // Set the comment for the pose
Console.WriteLine($"Pose ID: {pose.Id}, Name: {pose.Name}, Comment: {pose.Commen
t}");
```

3.20.1 PoseRegisterData

Description

The `PoseRegisterData` class is used to represent pose data in a PR register, including Cartesian space coordinates and posture information, joint angle information, and pose type. This class facilitates the storage and management of specific pose data.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`CartData`	`BaseCartData`	Cartesian data
`Joint`	`Joint`	Joint data
`Pt`	`PoseType`	Pose type, defaults to Unknown

3.22 Coordinate

Description

The `Coordinate` class is used to represent a coordinate system in the robot system. It includes basic information about the coordinate system, such as a unique identifier (ID), name, comment, motion group number, and specific pose data. This class is used in robot programming and control systems to define and manage the position and orientation of coordinate systems, facilitating motion planning and path control in programs.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`Id`	`int`	Unique identifier for the coordinate system
`Name`	`string`	Name of the coordinate system, used to identify and describe the coordinate system
`Comment`	`string`	Comment for the coordinate system, used to further explain the purpose or characteristics of the coordinate system

Property	Type	Description
GroupId	int	Motion group number to which the coordinate system belongs, used for classification and management of coordinate systems
Data	Position	Specific pose data of the coordinate system, including position and orientation information

Example

```c#
// Create a Coordinate instance
Coordinate coordinate = new Coordinate
{
    Id = 1, // Set the unique identifier
    Name = "UserCoordinate1", // Set the name
    Comment = "This is a user-defined coordinate system", // Set the comment
    GroupId = 1, // Set the motion group number
    Data = new Position { X = 100, Y = 200, Z = 300, A = 45, B = 30, C = 60 } // Se
t the pose data
};
```

3.22.1 CoordinateType

Description

The `CoordinateType` enum is used to define the type of coordinate system. It distinguishes between user coordinate systems and tool coordinate systems. This enum is used in robot programming and control systems to clearly specify the purpose of the coordinate system, helping the system correctly handle operations related to coordinate systems.

Import

```c#
using Agilebot.IR.Types;
```

Enum Values

Enum Value	Description
`UserCoordinate`	User coordinate system, used to define user-defined coordinate systems.
`ToolCoordinate`	Tool coordinate system, used to define the coordinate system of tools (e.g., end effectors).

3.22.2 CoordSummary

Description

The `CoordSummary` class is used to represent the summary information of a coordinate system. It includes the type, unique identifier, name, comment, and group ID of the coordinate system. This class is used in the robot programming environment to manage and store metadata of coordinate systems, facilitating quick access and manipulation of coordinate systems in programs.

Import

```c#
using Agilebot.IR.Types;
```

Properties

Property	Type	Description
`Type`	`CoordinateType`	Type of the coordinate system, which can be a user coordinate system or a tool coordinate system
`Id`	`int`	Unique identifier for the coordinate system
`Name`	`string`	Name of the coordinate system
`Comment`	`string`	Comment for the coordinate system, used to describe its purpose or characteristics
`GroupId`	`int`	Group ID to which the coordinate system belongs, used for classification and management of coordinate systems

Example

```c#
// Create a CoordSummary instance
CoordSummary coordSummary = new CoordSummary
{
    Type = CoordinateType.UserCoordinate, // Set to user coordinate system
    Id = 1, // Set the unique identifier
    Name = "UserCoord1", // Set the name
    Comment = "This is a user-defined coordinate system", // Set the comment
    GroupId = 0 // Set the group ID
};
```

4 Methods and Examples

4.1 Basic Operations of the Robot

Method Name	**Arm**(string `controllerIP` , string `teachPanelIP` = null, bool `localProxy` = true)
Description	Agilebot robot class constructor, which includes all available robot control interfaces. The robot must be initialized and connected before other functions can be used.
Request Parameters	`controllerIP` : string Robot controller IP address `teachPanelIP` : string Optional teach pendant IP; falls back to `controllerIP` when omitted `localProxy` : bool Whether to use local controller proxy service, default is true. When true, launches controller proxy service locally; when false, requires proxy service to be already installed in robot controller (requires robot software version 7.7 or later).
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.1 Connecting to the Robot

Method Name	**ConnectSync**()
Description	Establishes network connection with the Agilebot robot. The Arm constructor must be called first to initialize the robot instance.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.2 Checking the Connection with the Robot Arm

Method Name	**IsConnected**()
Description	Checks whether the network connection with the robot is valid.
Request Parameters	None
Return Value	bool: Connection status, true indicates connection is valid, false indicates connection is invalid or not connected
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.3 Disconnecting from the Robot

Method Name	**DisconnectSync**()
Description	Disconnects from the Agilebot robot and releases related resources.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/Connect.cs

```cs
using Agilebot.IR;

public class Connect
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
```

```
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接到捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 检查连接状态
            // [EN] Check the connection status
            var state = controller.IsConnected();
            Console.WriteLine("Connected: " + state);
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
```

```
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.4 Getting the Current Robot Model

Method Name	**GetArmModelInfo**()
Description	Gets the model information of the currently connected Agilebot robot.
Request Parameters	None
Return Value	string: Robot model string, e.g., "GBT-C5A" StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/GetArmModelInfo.cs

cs

```
using Agilebot.IR;

public class GetArmModelInfo
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
```

```
{
    // [ZH] 初始化捷勃特机器人
    // [EN] Initialize the Agilebot robot
    Arm controller = new Arm(
        controllerIP,
        useLocalProxy
    );

    // [ZH] 连接到捷勃特机器人
    // [EN] Connect to the Agilebot robot
    StatusCode code = controller.ConnectSync();
    if (code != StatusCode.OK)
    {
        Console.WriteLine(
            "Connect Robot Failed: "
                + code.GetDescription()
        );
        return code;
    }

    try
    {
        // [ZH] 获取机器人型号信息
        // [EN] Get the robot model information
        (string info, code) =
            controller.GetArmModelInfo();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Get Robot Model Failed: "
                    + code.GetDescription()
            );
        }
        else
        {
            Console.WriteLine("Model: " + info);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(
            $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
```

```
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.5 Getting the Robot's Operating State

Method Name	**GetRobotState**()
Description	Gets the current operating state of the Agilebot robot.
Request Parameters	None
Return Value	RobotState: Robot operating state enum value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/GetRobotState.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetRobotState
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 获取机器人运行状态
            // [EN] Get the robot running state
            (RobotState state, code) =
                controller.GetRobotState();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Get RobotState Failed: "
```

```
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("RobotState: " + state);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.6 Getting the Current Controller Operating State

Method Name	**GetCtrlState()**
Description	Gets the current operating state of the Agilebot robot controller.
Request Parameters	None
Return Value	CtrlState: Controller operating state enum value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/GetCtrlState.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetCtrlState
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
```

```
            );
            return code;
        }

        try
        {
            // [ZH] 获取控制器运行状态
            // [EN] Get the controller running state
            (CtrlState state, code) =
                controller.GetCtrlState();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Get CtrlState Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("CtrlState: " + state);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
```

```
                    code = disconnectCode;
                }
            }

            return code;
        }
    }
```

4.1.7 Getting the Current Servo State

Method Name	**GetServoState()**
Description	Gets the current state of the Agilebot robot servo system.
Request Parameters	None
Return Value	ServoState: Servo system state enum value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/GetServoState.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetServoState
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
```

```
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 获取伺服运行状态
            // [EN] Get the servo operating state
            (ServoState state, code) =
                controller.GetServoState();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Get ServoState Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("ServoState: " + state);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
```

```
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.8 Getting the Robot Controller Version

Method Name	**GetVersion**()
Description	Gets the software version information of the Agilebot robot controller.
Request Parameters	None
Return Value	string: Controller software version string StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/GetVersion.cs

cs

```cs
using Agilebot.IR;

public class GetVersion
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 获取机器人控制器版本
            // [EN] Get the robot controller version
            string version;
            (version, code) = controller.GetVersion();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Get version Failed: "
                        + code.GetDescription()
```

```
                );
            }
            else
            {
                Console.WriteLine("Version: " + version);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.9 Setting the Robot's LED Indicator Light

Method Name	**SwitchLedLight**(bool `mode`)
Description	Controls the on/off state of the Agilebot robot LED indicator light.
Request Parameters	`mode` : bool LED indicator light control mode, true indicates turn on, false indicates turn off
Return Value	StatusCode: Operation execution result
Compatibility	Only supports collaborative robots, requires controller version 1.3.6 and above, industrial robots not supported
Compatible robot software version	Collaborative (Copper): v7.5.1.3+ Industrial (Bronze): Not supported

Example Code

Arm/SwitchLedLight.cs

```cs
using Agilebot.IR;

public class SwitchLedLight
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
```

```
                + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 关闭灯光
            // [EN] Turn off the LED light
            code = controller.SwitchLedLight(false);
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Switch Led Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("Switch Led Light Off.");
            }

            Thread.Sleep(2000);

            // [ZH] 打开灯光
            // [EN] Turn on the LED light
            code = controller.SwitchLedLight(true);
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Switch Led Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("Switch Led Light On.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
```

```
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Disconnect from the robot
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.10 Robot Servo On

Method Name	**ServoOn**()
Description	Starts the servo system of the Agilebot robot, making the robot enter a controllable state.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.11 Robot Servo Off

Method Name	**ServoOff**()
Description	Turns off the servo system of the Agilebot robot, making the robot enter a safe stop state.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.1.12 Resetting the Robot Servo

Method Name	**ServoReset**()
Description	Resets the servo system of the Agilebot robot, clearing error states and preparing for restart.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/ServoOperation.cs

```cs
using Agilebot.IR;

public class ServoOperation
{
    public static StatusCode Run(
        string controllerIP,
```

```
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 机械臂伺服重置
            // [EN] Reset the robot arm servo
            code = controller.ServoReset();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Servo Reset Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("Servo Reset Success.");
            }

            Thread.Sleep(3000);

            // [ZH] 机械臂伺服关闭
```

```
            // [EN] Turn off the robot arm servo
            code = controller.ServoOff();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Servo Off Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("Servo Off Success.");
            }

            Thread.Sleep(3000);

            // [ZH] 机械臂伺服打开
            // [EN] Turn on the robot arm servo
            code = controller.ServoOn();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Servo On Failed: "
                        + code.GetDescription()
                );
            }
            else
            {
                Console.WriteLine("Servo On Success.");
            }

            Thread.Sleep(3000);
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
```

```
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.1.13 Emergency Stop

Method Name	**Estop()**
Description	Executes emergency stop of the Agilebot robot, immediately stopping all motion and entering a safe state.
Request Parameters	None
Return Value	StatusCode: Emergency stop operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Arm/Estop.cs

cs

```cs
using Agilebot.IR;

public class Estop
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接到捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        if (code != StatusCode.OK)
        {
            Console.WriteLine(
                "Connect Robot Failed: "
                    + code.GetDescription()
            );
            return code;
        }

        try
        {
            // [ZH] 触发机器人急停
            // [EN] Trigger the robot emergency stop
            code = controller.Estop();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    "Emergency Stop Failed: "
                        + code.GetDescription()
                );
            }
            else
```

```
            {
                Console.WriteLine("Emergency Stop Success");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.2 Robot Motion Control and Status

4.2.1 Getting Robot Parameters

4.2.1.1 Getting OVC Overall Velocity Coefficient

Method Name	**Motion.GetOVC()**
Description	Gets the current robot's OVC (Overall Velocity Control) global velocity ratio, with a range of 0~1.
Request Parameters	None
Return Value	double: Global velocity ratio value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.1.2 Getting OAC Overall Acceleration Coefficient

Method Name	**Motion.GetOAC()**
Description	Gets the current robot's OAC (Overall Acceleration Control) global acceleration ratio, with a range of 0~1.2.
Request Parameters	None
Return Value	double: Global acceleration ratio value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.1.3 Getting the Current TF

Method Name	**Motion.GetTF()**
Description	Gets the current TF (Tool Frame) tool coordinate system index used by the robot, with a range of 0~10.
Request Parameters	None
Return Value	int: TF tool coordinate system index StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.1.4 Getting the Current UF

Method Name	**Motion.GetUF()**
Description	Gets the current UF (User Frame) user coordinate system index used by the robot, with a range of 0~10.
Request Parameters	None
Return Value	int: UF user coordinate system index StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.1.5 Getting the Current TCS Teaching Coordinate System

Method Name	**Motion.GetTCS()**
Description	Gets the current TCS (Teach Coordinate System) teaching coordinate system type used by the robot, see TCSType for details.
Request Parameters	None
Return Value	TCSType: TCS teaching coordinate system type enum value StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/GetMotionParameters.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetMotionParameters
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取 OVC 全局速度比率
            // [EN] Get OVC global speed ratio
            double ovc;
            (ovc, code) = controller.Motion.GetOVC();
```

```
if (code == StatusCode.OK)
{
    Console.WriteLine($"OVC = {ovc}");
}
else
{
    Console.WriteLine(
        $"获取OVC失败: {code.GetDescription()}"
    );
}

// [ZH] 获取 OAC 全局加速度比率
// [EN] Get OAC global acceleration ratio
double oac;
(oac, code) = controller.Motion.GetOAC();
if (code == StatusCode.OK)
{
    Console.WriteLine($"OAC = {oac}");
}
else
{
    Console.WriteLine(
        $"获取OAC失败: {code.GetDescription()}"
    );
}

// [ZH] 获取当前使用的 TF
// [EN] Get current TF (Tool Frame)
int tf;
(tf, code) = controller.Motion.GetTF();
if (code == StatusCode.OK)
{
    Console.WriteLine($"TF = {tf}");
}
else
{
    Console.WriteLine(
        $"获取TF失败: {code.GetDescription()}"
    );
}

// [ZH] 获取当前使用的 UF
```

```
// [EN] Get current UF (User Frame)
int uf;
(uf, code) = controller.Motion.GetUF();
if (code == StatusCode.OK)
{
    Console.WriteLine($"UF = {uf}");
}
else
{
    Console.WriteLine(
        $"获取UF失败: {code.GetDescription()}"
    );
}

// [ZH] 获取当前使用的 TCS 示教坐标系
// [EN] Get current TCS teaching coordinate system
TCSType tcs;
(tcs, code) = controller.Motion.GetTCS();
if (code == StatusCode.OK)
{
    Console.WriteLine($"TCSType = {tcs}");
}
else
{
    Console.WriteLine(
        $"获取TCS失败: {code.GetDescription()}"
    );
}

// [ZH] 获取机器人软限位
// [EN] Get robot soft limits
List<List<double>> softLimit;
(softLimit, code) =
    controller.Motion.GetUserSoftLimit();
if (code == StatusCode.OK)
{
    Console.WriteLine("软限位信息:");
    for (int i = 0; i < softLimit.Count; i++)
    {
        Console.WriteLine(
            $"轴{i + 1}: 下限={softLimit[i][0]}, 上限={softLimit[i][1]}"
        );
```

```
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取软限位失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.2 Setting Robot Parameters

4.2.2.1 Setting OVC Overall Velocity Coefficient

Method Name	**Motion.SetOVC**(double `value`)
Description	Sets the current robot's OVC (Overall Velocity Control) global velocity ratio.
Request Parameters	`value` : double velocity ratio, range 0~1
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.2.2 Setting OAC Overall Acceleration Coefficient

Method Name	**Motion.SetOAC**(double `value`)
Description	Sets the current robot's OAC (Overall Acceleration Control) global acceleration ratio.
Request Parameters	`value` : double acceleration ratio, range 0~1.2
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.2.3 Setting the Current TF Tool Coordinate System Index

Method Name	**Motion.SetTF**(int `value`)
Description	Sets the current TF (Tool Frame) tool coordinate system index.
Request Parameters	`value` : int TF index, range 0~10
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.2.4 Setting the Current UF User Coordinate System Index

Method Name	**Motion.SetUF**(int `value`)
Description	Sets the current UF (User Frame) user coordinate system index.
Request Parameters	`value` : int UF index, range 0~10
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.2.5 Setting the Current TCS Teaching Coordinate System

Method Name	**Motion.SetTCS**(TCSType `value`)
Description	Sets the current TCS (Teach Coordinate System) teaching coordinate system.
Request Parameters	`value` : TCSType TCS teaching coordinate system type
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/SetMotionParameters.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class SetMotionParameters
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
```

```
Arm controller = new Arm(
    controllerIP,
    useLocalProxy
);

// [ZH] 连接捷勃特机器人
// [EN] Connect to the Agilebot robot
StatusCode code = controller.ConnectSync();
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 设置 OVC 全局速度比率
    // [EN] Set OVC global speed ratio
    code = controller.Motion.SetOVC(0.5);
    if (code == StatusCode.OK)
    {
        Console.WriteLine("设置OVC成功");
    }
    else
    {
        Console.WriteLine(
            $"设置OVC失败: {code.GetDescription()}"
        );
    }

    // [ZH] 设置 OAC 全局加速度比率
    // [EN] Set OAC global acceleration ratio
    code = controller.Motion.SetOAC(0.8);
    if (code == StatusCode.OK)
    {
        Console.WriteLine("设置OAC成功");
    }
```

```
else
{
    Console.WriteLine(
        $"设置OAC失败: {code.GetDescription()}"
    );
}

// [ZH] 设置当前使用的 TF 用户坐标系编号
// [EN] Set current TF (Tool Frame) user coordinate system number
code = controller.Motion.SetTF(2);
if (code == StatusCode.OK)
{
    Console.WriteLine("设置TF成功");
}
else
{
    Console.WriteLine(
        $"设置TF失败: {code.GetDescription()}"
    );
}

// [ZH] 设置当前使用的 UF 工具坐标系编号
// [EN] Set current UF (User Frame) tool coordinate system number
code = controller.Motion.SetUF(1);
if (code == StatusCode.OK)
{
    Console.WriteLine("设置UF成功");
}
else
{
    Console.WriteLine(
        $"设置UF失败: {code.GetDescription()}"
    );
}

// [ZH] 设置当前使用的 TCS 示教坐标系
// [EN] Set current TCS teaching coordinate system
code = controller.Motion.SetTCS(TCSType.TOOL);
if (code == StatusCode.OK)
{
    Console.WriteLine("设置TCS成功");
}
```

```
            else
            {
                Console.WriteLine(
                    $"设置TCS失败: {code.GetDescription()}"
                );
            }

            // [ZH] 设置UDP位置控制的相关参数
            // [EN] Set UDP position control related parameters
            code =
                controller.Motion.SetPositionTrajectoryParams(
                    10,
                    20,
                    10,
                    10
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("设置位置控制参数成功");
            }
            else
            {
                Console.WriteLine(
                    $"设置位置控制参数失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
```

```
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.3 Converting Cartesian Position to Joint Values

Method Name	**Motion.ConvertCartToJoint**(MotionPose `pose` , int `ufIndex` = 0, int `tfIndex` = 0)
Description	Converts pose data from Cartesian coordinates to joint coordinates.
Request Parameters	`pose` : MotionPose Robot pose data `ufIndex` : int User coordinate system index, default is 0 `tfIndex` : int Tool coordinate system index, default is 0
Return Value	MotionPose: Converted robot pose data StatusCode: Conversion operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/ConvertCartToJoint.cs

cs

```
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class ConvertCartToJoint
{
    public static StatusCode Run(
```

```
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 创建笛卡尔位姿
            // [EN] Create Cartesian pose
            MotionPose motionPose = new MotionPose();
            motionPose.Pt = PoseType.Cart;
            motionPose.CartData.Position = new Position
            {
                X = 300,
                Y = 300,
                Z = 300,
                A = 0,
                B = 0,
                C = 0,
            };
            motionPose.CartData.Posture = new Posture
            {
```

```
                WristFlip = 1,
                ArmUpDown = 1,
                ArmBackFront = 1,
                ArmLeftRight = 1,
                TurnCircle = new List<int>(9)
                {
                    0,
                    0,
                    0,
                    0,
                    0,
                    0,
                    0,
                    0,
                    0,
                },
            };

            // [ZH] 将笛卡尔点位转换成关节值点位
            // [EN] Convert Cartesian pose to joint pose
            MotionPose convertPose;
            (convertPose, code) =
                controller.Motion.ConvertCartToJoint(
                    motionPose
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("笛卡尔转关节成功:");
                Console.WriteLine(
                    $"关节值: J1={convertPose.Joint.J1}, J2={convertPose.Joint.J2},
J3={convertPose.Joint.J3}, J4={convertPose.Joint.J4}, J5={convertPose.Joint.J5}, J6
={convertPose.Joint.J6}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"笛卡尔转关节失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.4 Converting Joint Values to Cartesian Position

Method Name	**Motion.ConvertJointToCart**(MotionPose `pose` , int `ufIndex` = 0, int `tfIndex` = 0)
Description	Converts pose data from joint coordinates to Cartesian coordinates.
Request Parameters	`pose` : MotionPose Robot pose data `ufIndex` : int User coordinate system index, default is 0 `tfIndex` : int Tool coordinate system index, default is 0
Return Value	MotionPose: Converted robot pose data StatusCode: Conversion operation execution result

Method Name	**Motion.ConvertJointToCart**(MotionPose `pose` , int `ufIndex` = 0, int `tfIndex` = 0)
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/ConvertJointToCart.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class ConvertJointToCart
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }
```

```
        try
        {
            // [ZH] 创建关节位姿
            // [EN] Create joint pose
            MotionPose motionPose = new MotionPose();
            motionPose.Pt = PoseType.Joint;
            motionPose.Joint = new Joint
            {
                J1 = 0,
                J2 = 0,
                J3 = 60,
                J4 = 60,
                J5 = 0,
                J6 = 0,
            };

            // [ZH] 将关节值点位转换成笛卡尔点位
            // [EN] Convert joint pose to Cartesian pose
            MotionPose convertPose;
            (convertPose, code) =
                controller.Motion.ConvertJointToCart(
                    motionPose
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("关节转笛卡尔成功:");
                Console.WriteLine(
                    $"位置: X={convertPose.CartData.Position.X}, Y={convertPose.Car
tData.Position.Y}, Z={convertPose.CartData.Position.Z}"
                );
                Console.WriteLine(
                    $"姿态: A={convertPose.CartData.Position.A}, B={convertPose.Car
tData.Position.B}, C={convertPose.CartData.Position.C}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"关节转笛卡尔失败: {code.GetDescription()}"
                );
            }
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.5 Moving the Robot End Effector to a Specified Position

Method Name	**Motion.MoveJoint**(MotionPose `pose` , double `vel` = 1, double `acc` = 1)
Description	Moves the robot end effector to a specified position, using the fastest path (joint motion).
Request Parameters	`pose` : MotionPose Target position coordinates in Cartesian space or joint coordinate system `vel` : double Motion speed, range 0~1, representing multiple of maximum speed

Method Name	**Motion.MoveJoint**(MotionPose `pose` , double `vel` = 1, double `acc` = 1)
	`acc` : double Acceleration, range 0~1.2, representing multiple of maximum acceleration
Return Value	StatusCode: Motion command execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/MoveJoint.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class MoveJoint
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );
```

```
if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 创建关节位姿
    // [EN] Create joint pose
    MotionPose motionPose = new MotionPose();
    motionPose.Pt = PoseType.Joint;
    motionPose.Joint = new Joint
    {
        J1 = 10,
        J2 = 30,
        J3 = 30,
        J4 = 0,
        J5 = 0,
        J6 = 0,
    };

    // [ZH] 让机器人末端移动到指定的位置
    // [EN] Move robot end to specified position
    code = controller.Motion.MoveJoint(
        motionPose,
        0.5,
        0.8
    );
    if (code == StatusCode.OK)
    {
        Console.WriteLine("关节运动请求成功");
    }
    else
    {
        Console.WriteLine(
            $"关节运动失败: {code.GetDescription()}"
        );
    }
}
catch (Exception ex)
{
    Console.WriteLine(
```

```
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.6 Moving the Robot End Effector Along a Straight Line to a Specified Position

Method Name	**Motion.MoveLine**(MotionPose `pose` , double `vel` = 100, double `acc` = 1)
Description	Moves the robot end effector along a straight line to a specified position, using a linear path between two points.
Request Parameters	`pose` : MotionPose Target position coordinates in Cartesian space or joint coordinate system `vel` : double Motion speed, range 0~5000mm/s, representing robot end effector movement speed `acc` : double Acceleration, range 0~1.2, representing multiple of maximum acceleration

Method Name	**Motion.MoveLine**(MotionPose `pose` , double `vel` = 100, double `acc` = 1)
Return Value	StatusCode: Motion command execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/MoveLine.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class MoveLine
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
```

```
            return code;
        }

        try
        {
            // [ZH] 创建关节位姿
            // [EN] Create joint pose
            MotionPose motionPose = new MotionPose();
            motionPose.Pt = PoseType.Joint;
            motionPose.Joint = new Joint
            {
                J1 = 20,
                J2 = 40,
                J3 = 40,
                J4 = 5,
                J5 = 5,
                J6 = 5,
            };

            // [ZH] 让机器人末端沿直线移动到指定的位置
            // [EN] Move robot end in straight line to specified position
            code = controller.Motion.MoveLine(
                motionPose,
                100,
                1.0
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("直线运动请求成功");
            }
            else
            {
                Console.WriteLine(
                    $"直线运动失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
```

```
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.7 Moving the Robot End Effector Along an Arc to a Specified Position

Method Name	**Motion.MoveCircle**(MotionPose `pose1` , MotionPose `pose2` , double `vel` = 100, double `acc` = 1)
Description	Moves the robot end effector along an arc to a specified position.
Request Parameters	`pose1` : MotionPose Robot motion intermediate pose `pose2` : MotionPose Robot motion final pose `vel` : double Motion speed, range 0~5000mm/s, representing robot end effector movement speed `acc` : double Acceleration, range 0~1.2, representing multiple of maximum acceleration
Return Value	StatusCode: Motion command execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/MoveCircle.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class MoveCircle
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 创建第一个位姿（途径点）
            // [EN] Create first pose (waypoint)
            MotionPose motionPose1 = new MotionPose();
```

```
motionPose1.Pt = PoseType.Joint;
motionPose1.Joint = new Joint
{
    J1 = 0,
    J2 = 0,
    J3 = 60,
    J4 = 60,
    J5 = 0,
    J6 = 0,
};

// [ZH] 创建第二个位姿（终点）
// [EN] Create second pose (endpoint)
MotionPose motionPose2 = new MotionPose();
motionPose2.Pt = PoseType.Joint;
motionPose2.Joint = new Joint
{
    J1 = 0,
    J2 = 30,
    J3 = 70,
    J4 = 40,
    J5 = 0,
    J6 = 0,
};

// [ZH] 让机器人末端沿弧线移动到指定的位置
// [EN] Move robot end in arc to specified position
code = controller.Motion.MoveCircle(
    motionPose1,
    motionPose2,
    100,
    1.0
);
if (code == StatusCode.OK)
{
    Console.WriteLine("弧线运动请求成功");
}
else
{
    Console.WriteLine(
        $"弧线运动失败: {code.GetDescription()}"
    );
```

```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.8 Getting the Current Pose of the Robot

Method Name	**Motion.GetCurrentPose**(PoseType `pt` , int `ufIndex` = 0, int `tfIndex` = 0)
Description	Gets the current pose of the robot, which can be pose information in Cartesian space or joint coordinate system.
Request Parameters	`pt` : PoseType Pose type `ufIndex` : int When using PoseType.CART, user coordinate system index must be provided, default is 0

Method Name	**Motion.GetCurrentPose**(PoseType `pt` , int `ufIndex` = 0, int `tfIndex` = 0)
	`tfIndex` : int When using PoseType.CART, tool coordinate system index must be provided, default is 0
Return Value	MotionPose: Robot pose data StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/GetCurrentPose.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class GetCurrentPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );
```

```
        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人的当前位姿（笛卡尔坐标）
            // [EN] Get robot current pose (Cartesian coordinates)
            MotionPose cartPose;
            (cartPose, code) =
                controller.Motion.GetCurrentPose(
                    PoseType.Cart,
                    0,
                    0
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("当前笛卡尔位姿:");
                Console.WriteLine(
                    $"位置: X={cartPose.CartData.Position.X}, Y={cartPose.CartData.
Position.Y}, Z={cartPose.CartData.Position.Z}"
                );
                Console.WriteLine(
                    $"姿态: A={cartPose.CartData.Position.A}, B={cartPose.CartData.
Position.B}, C={cartPose.CartData.Position.C}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"获取笛卡尔位姿失败: {code.GetDescription()}"
                );
            }

            // [ZH] 获取机器人的当前位姿（关节坐标）
            // [EN] Get robot current pose (joint coordinates)
            MotionPose jointPose;
            (jointPose, code) =
                controller.Motion.GetCurrentPose(
                    PoseType.Joint,
```

```
                    0,
                    0
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("当前关节位姿:");
                Console.WriteLine(
                    $"关节值: J1={jointPose.Joint.J1}, J2={jointPose.Joint.J2}, J3=
{jointPose.Joint.J3}, J4={jointPose.Joint.J4}, J5={jointPose.Joint.J5}, J6={jointPo
se.Joint.J6}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"获取关节位姿失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
```

```
    }
}
```

4.2.9 Getting the Robot's DH Parameters

Method Name	**Motion.GetDHParam()**
Description	Gets the robot's DH (Denavit-Hartenberg) parameters.
Request Parameters	None
Return Value	List<DHparam>: DH parameter list StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

Motion/GetDHParam.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetDHParam
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );
```

```
        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人的DH参数
            // [EN] Get robot DH parameters
            List<DHparam> dhParamsList;
            (dhParamsList, code) =
                controller.Motion.GetDHParam(1);
            if (code == StatusCode.OK)
            {
                Console.WriteLine("获取DH参数成功:");
                for (int i = 0; i < dhParamsList.Count; i++)
                {
                    var dh = dhParamsList[i];
                    Console.WriteLine(
                        $"轴{i + 1}: Alpha={dh.alpha}, A={dh.a}, D={dh.d}, Offset=
{dh.offset}"
                    );
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取DH参数失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.10 Setting the Robot's DH Parameters

Method Name	**Motion.SetDHParam**(List<DHparam> dHparams)
Description	Sets the robot's DH (Denavit-Hartenberg) parameters.
Request Parameters	dHparams : List<DHparam> DH parameter list
Return Value	StatusCode: Set operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

Motion/SetDHParam.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class SetDHParam
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 先获取当前的DH参数
            // [EN] First get current DH parameters
            List<DHparam> dhParamsList;
            (dhParamsList, code) =
                controller.Motion.GetDHParam(1);
            if (code != StatusCode.OK)
```

```
            {
                Console.WriteLine(
                    $"获取DH参数失败: {code.GetDescription()}"
                );
                return code;
            }

            Console.WriteLine(
                "获取DH参数成功，准备设置相同的参数..."
            );

            // [ZH] 设置DH参数（这里设置为相同的参数作为示例）
            // [EN] Set DH parameters (set same parameters as example)
            code = controller.Motion.SetDHParam(
                dhParamsList
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("设置DH参数成功");
            }
            else
            {
                Console.WriteLine(
                    $"设置DH参数失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
```

```
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.2.11 Getting the Robot Axis Lock Status

Method Name	**Motion.GetDragSet**()
Description	Gets the current robot axis lock status, which only applies to teaching movements.
Request Parameters	None
Return Value	DragStatus: Axis lock status, True indicates the axis is movable, False indicates it is locked StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

4.2.12 Setting the Robot Axis Lock Status

Method Name	**Motion.SetDragSet**(DragStatus `dragStatus`)
Description	Sets the current robot axis lock status, which only applies to teaching movements.
Request Parameters	`dragStatus` : DragStatus Axis lock status, default is all True: unlocked state
Return Value	StatusCode: Result of function execution

Method Name	**Motion.SetDragSet**(DragStatus `dragStatus`)
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

4.2.13 Enabling Drag Teaching for the Robot

Method Name	**Motion.EnableDrag**(bool `dragState`)
Description	Enables or disables drag teaching for the robot.
Request Parameters	`dragState` : bool The drag state of the robot, true indicates entering drag mode, false indicates exiting drag mode
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): Not supported

Example Code

Motion/DragControl.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class DragControl
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
```

```
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取当前机器人的轴锁定状态
            // [EN] Get current robot axis lock status
            DragStatus dragStatus;
            (dragStatus, code) =
                controller.Motion.GetDragSet();
            if (code == StatusCode.OK)
            {
                Console.WriteLine("获取轴锁定状态成功:");
                Console.WriteLine(
                    $"X轴: {dragStatus.CartStatus.X}, Y轴: {dragStatus.CartStatus.
Y}, Z轴: {dragStatus.CartStatus.Z}"
                );
                Console.WriteLine(
                    $"连续拖动: {dragStatus.IsContinuousDrag}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"获取轴锁定状态失败: {code.GetDescription()}"
                );
            }

            // [ZH] 修改当前机器人的轴锁定状态
```

```
// [EN] Modify current robot axis lock status
if (code == StatusCode.OK)
{
    dragStatus.CartStatus.X = false;
    dragStatus.IsContinuousDrag = true;
    code = controller.Motion.SetDragSet(
        dragStatus
    );
    if (code == StatusCode.OK)
    {
        Console.WriteLine("设置轴锁定状态成功");
    }
    else
    {
        Console.WriteLine(
            $"设置轴锁定状态失败: {code.GetDescription()}"
        );
    }
}

// [ZH] 启动拖动（注意：实际使用中需要谨慎）
// [EN] Enable drag (Note: use with caution in practice)
if (code == StatusCode.OK)
{
    Console.WriteLine(
        "注意：启动拖动功能，请确保安全！"
    );
    code = controller.Motion.EnableDrag(true);
    if (code == StatusCode.OK)
    {
        Console.WriteLine("启动拖动成功");

        // [ZH] 等待一段时间后停止拖动
        // [EN] Wait for a while then stop drag
        Console.WriteLine(
            "等待3秒后停止拖动..."
        );
        Thread.Sleep(3000);

        code = controller.Motion.EnableDrag(
            false
        );
```

```
                    if (code == StatusCode.OK)
                    {
                        Console.WriteLine("停止拖动成功");
                    }
                    else
                    {
                        Console.WriteLine(
                            $"停止拖动失败: {code.GetDescription()}"
                        );
                    }
                }
                else
                {
                    Console.WriteLine(
                        $"启动拖动失败: {code.GetDescription()}"
                    );
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
```

```
    }
}
```

4.2.14 Entering Real-Time Position Control Mode

Method Name	**Motion.EnterPositionControl()**
Description	Enters real-time position control mode, allowing precise position control of the robot.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Note	After entering real-time control mode, control commands must be sent via UDP.
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.15 Exiting Real-Time Position Control Mode

Method Name	**Motion.ExitPositionControl()**
Description	Exits real-time position control mode, returning to the default robot control state.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Note	After exiting, the robot will no longer accept real-time control commands.
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.16 Setting Subscription Parameters

Method Name	**Motion.SetUDPFeedbackParams**(bool `flag` , string `ip` , int `interval` , int `feedbackType` , List<int> `DOList` = null)
Description	Configures the subscription parameters for the robot to push data to a specified IP address.
Request Parameters	`flag` : bool Whether to enable UDP data pushing; `ip` : string IP address of the recipient; `interval` : int Interval for sending data (unit: milliseconds); `feedbackType` : int Feedback data format (0: XML format); `DOList` : List<int> List of DO signals to be obtained (up to ten, optional)
Return Value	StatusCode: Result of function execution
Note	The parameter settings are only effective when the UDP data pushing function is enabled.
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Example Code

Motion/PositionControl.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class PositionControl
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
```

```
        controllerIP,
        useLocalProxy
    );

    // [ZH] 连接捷勃特机器人
    // [EN] Connect to the Agilebot robot
    StatusCode code = controller.ConnectSync();
    Console.WriteLine(
        code != StatusCode.OK
            ? code.GetDescription()
            : "连接成功/Successfully connected."
    );

    if (code != StatusCode.OK)
    {
        return code;
    }

    try
    {
        // [ZH] 设置UDP反馈参数
        // [EN] Set UDP feedback parameters
        code = controller.Motion.SetUDPFeedbackParams(
            true,
            "192.168.1.1",
            10,
            0
        );
        if (code == StatusCode.OK)
        {
            Console.WriteLine("设置UDP反馈参数成功");
        }
        else
        {
            Console.WriteLine(
                $"设置UDP反馈参数失败: {code.GetDescription()}"
            );
        }

        // [ZH] 进入实时位置控制模式
        // [EN] Enter real-time position control mode
        code = controller.Motion.EnterPositionControl();
```

```
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "进入实时位置控制模式成功"
            );

            // [ZH] 在此可以插入发送UDP数据控制机器人的代码
            // [EN] Insert UDP data control code here
            Console.WriteLine(
                "注意：在实时位置控制模式下，需要通过UDP发送控制指令"
            );
            Console.WriteLine("等待2秒...");
            Thread.Sleep(2000);

            // [ZH] 退出实时位置控制模式
            // [EN] Exit real-time position control mode
            code =
                controller.Motion.ExitPositionControl();
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "退出实时位置控制模式成功"
                );
            }
            else
            {
                Console.WriteLine(
                    $"退出实时位置控制模式失败: {code.GetDescription()}"
                );
            }
        }
        else
        {
            Console.WriteLine(
                $"进入实时位置控制模式失败: {code.GetDescription()}"
            );
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(
            $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
```

```
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

Data Push Description

Name	Field	Description
RIst: Cartesian Position	X	Value in the X direction in the tool coordinate system, unit is millimeters
	Y	Value in the Y direction in the tool coordinate system, unit is millimeters
	Z	Value in the Z direction in the tool coordinate system, unit is millimeters
	A	Rotation around the X axis in the tool coordinate system, unit is degrees
	B	Rotation around the Y axis in the tool coordinate system, unit is degrees
	C	Rotation around the Z axis in the tool coordinate

Name	Field	Description
		system, unit is degrees
AIPos: Joint Position	A1-A6	Values of the six joints, unit is degrees
EIPos: Additional Axis Data	EIPos	Additional axis data
WristBtnState: Wrist Button State	Button State	1 = Button pressed, 0 = Button released
	DragModel	Drag button state
	RecordJoint	Teach 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 segment return value, format is `program name: line number`
IPOC: Timestamp	IPOC	Timestamp

4.2.17 Getting the Robot's Soft Limits

Method Name	**Motion.GetUserSoftLimit()**
Description	Gets the current soft limits of the robot.
Request Parameters	None
Return Value	List<List<double>>: Robot's soft limits, the first layer of the list represents each axis, and the second layer represents the lower and upper limit values of each

Method Name	**Motion.GetUserSoftLimit**()
	axis StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.18 Specifying UDP Position Control Parameters

Method Name	**Motion.SetPositionTrajectoryParams**(int `maxTimeoutCount` , int `timeout` , double `wristElbowThreshold` , double `shoulderThreshold`)
Description	Specifies the parameters related to UDP position control.
Request Parameters	`maxTimeoutCount` : Maximum number of timeouts; `timeout` : Timeout period (i.e., send interval, default is 20ms); `wristElbowThreshold` : Threshold for wrist/elbow approaching singularity; `shoulderThreshold` : Threshold for approaching shoulder singularity;
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.19 Payload-Related Interfaces

4.2.19.1 Getting the Current Active Payload

Method Name	**Motion.Payload.GetCurrentPayload**()
Description	Gets the currently active payload information.
Request Parameters	None
Return Value	int: Index of the currently active payload StatusCode: Result of function execution

Method Name	**Motion.Payload.GetCurrentPayload**()
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.19.2 Getting the Corresponding Payload

Method Name	**Motion.Payload.GetPayloadById**(int `index`)
Description	Gets the corresponding payload.
Request Parameters	`index` : Payload index
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.19.3 Activating the Corresponding Payload

Method Name	**Motion.Payload.SetCurrentPayload**(int `index`)
Description	Activates the corresponding payload.
Request Parameters	`index` : Payload index
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	The payload ID must exist in the current device.

4.2.19.4 Getting All Payload Information

Method Name	**Motion.Payload.GetAllPayloadInfo**()
Description	Gets detailed information of all payloads.
Request Parameters	None

Method Name	**Motion.Payload.GetAllPayloadInfo**()
Return Value	Dictionary<uint, string>: Returns a dictionary of payload information StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.19.5 Adding a Payload

Method Name	**Motion.Payload.AddPayload**(PayloadInfo `payload`)
Description	Adds a new payload.
Request Parameters	`payload` : PayloadInfo Payload object
Return Value	StatusCode: Result of function execution
Note	The new payload ID must not exist in the current device and must be between 1 and 10.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.2.19.6 Deleting a Specified Payload

Method Name	**Motion.Payload.DeletePayload**(int `index`)
Description	Deletes the payload with the specified index.
Request Parameters	`index` : int Payload index
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	Note: The currently active payload cannot be deleted. If you want to delete the active payload, please activate another payload first and then delete the current one.

4.2.19.7 Updating a Specified Payload

Method Name	**Motion.Payload.UpdatePayload**(PayloadInfo `payload`)
Description	Updates the information of the specified payload.
Request Parameters	`payload` : PayloadInfo Payload object
Return Value	StatusCode: Result of function execution
Note	The payload ID must exist in the current device.
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Motion/PayloadControl.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;

public class PayloadControl
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
```

```
            : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取负载列表
            // [EN] Get payload list
            Dictionary<int, string> payloadList;
            (payloadList, code) =
                controller.Motion.Payload.GetAllPayloadInfo();
            if (code == StatusCode.OK)
            {
                Console.WriteLine("获取负载列表成功:");
                foreach (var p in payloadList)
                {
                    Console.WriteLine(
                        $"负载ID: {p.Key}, 描述: {p.Value}"
                    );
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取负载列表失败: {code.GetDescription()}"
                );
            }

            // [ZH] 获取当前激活的负载
            // [EN] Get current active payload
            int currentPayload;
            (currentPayload, code) =
                controller.Motion.Payload.GetCurrentPayload();
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前激活的负载ID: {currentPayload}"
                );
```

```
}
else
{
    Console.WriteLine(
        $"获取当前负载失败: {code.GetDescription()}"
    );
}

// [ZH] 添加新负载
// [EN] Add new payload
PayloadInfo payload = new()
{
    Id = 3,
    Comment = "测试负载",
    Weight = 1.0,
    MassCenter = new()
    {
        X = 1,
        Y = 2,
        Z = 3,
    },
    InertiaMoment = new()
    {
        LX = 10,
        LY = 20,
        LZ = 30,
    },
};

code = controller.Motion.Payload.AddPayload(
    payload
);
if (code == StatusCode.OK)
{
    Console.WriteLine("添加负载成功");
}
else
{
    Console.WriteLine(
        $"添加负载失败: {code.GetDescription()}"
    );
}
```

```
            // [ZH] 设置当前激活的负载
            // [EN] Set current active payload
            if (code == StatusCode.OK)
            {
                code =
                    controller.Motion.Payload.SetCurrentPayload(
                        3
                    );
                if (code == StatusCode.OK)
                {
                    Console.WriteLine("设置当前负载成功");
                }
                else
                {
                    Console.WriteLine(
                        $"设置当前负载失败: {code.GetDescription()}"
                    );
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }
```

```
        return code;
    }
}
```

4.2.19.8 Checking if Axis 3 is Horizontal

Method Name	**Motion.Payload.CheckAxisThreeHorizontal**()
Description	Checks if Axis 3 is horizontal.
Request Parameters	None
Return Value	double: The horizontal angle of Axis 3 StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	The horizontal angle must be between -1 and 1 to perform payload identification.

4.2.19.9 Getting the Payload Identification State

Method Name	**Motion.Payload.GetPayloadIdentifyState**()
Description	Gets the state of payload identification.
Request Parameters	None
Return Value	PayloadIdentifyState: Payload identification state StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.19.10 Starting Payload Identification

Method Name	**Motion.Payload.StartPayloadIdentify**(double `weight` , double `angle`)
Description	Starts payload identification.
Request Parameters	`weight` : double Payload weight (use -1 for unknown weight) `angle` : double Allowed rotation angle of Axis 6 (30-90 degrees)
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	You must enter the payload identification state before starting payload identification.

4.2.19.11 Getting the Payload Identification Result

Method Name	**Motion.Payload.PayloadIdentifyResult**()
Description	Gets the result of payload identification.
Request Parameters	None
Return Value	PayloadInfo: Payload identification result StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.19.12 Starting Interference Check for Payload Identification

Method Name	**Motion.Payload.InterferenceCheckForPayloadIdentify**(double `weight` , double `angle`)
Description	Starts interference check for payload identification to check for potential collisions.
Request Parameters	`weight` : double Payload weight (use -1 for unknown weight) `angle` : double Allowed rotation angle of Axis 6 (30-90 degrees)
Return Value	StatusCode: Result of function execution

Method Name	**Motion.Payload.InterferenceCheckForPayloadIdentify**(double `weight` , double `angle`)
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.19.13 Entering Payload Identification State

Method Name	**Motion.Payload.PayloadIdentifyStart**()
Description	Enters the payload identification state.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.19.14 Exiting Payload Identification State

Method Name	**Motion.Payload.PayloadIdentifyDone**()
Description	Exits the payload identification state.
Request Parameters	None
Return Value	StatusCode: Result of function execution
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

4.2.19.15 Full Payload Identification Process

Method Name	**Motion.Payload.PayloadIdentify**(double `weight` = -1, double `angle` = 90)
Description	Complete payload identification process, including all the interfaces mentioned above. For general payload identification, this interface is sufficient.
Request Parameters	`weight` : double Payload weight (use -1 for unknown weight) `angle` : double Allowed rotation angle of Axis 6 (30-90 degrees)

Method Name	**Motion.Payload.PayloadIdentify**(double `weight` = -1, double `angle` = 90)
Return Value	PayloadInfo: Payload identification result StatusCode: Result of function execution
Note	The returned payload can be added to the robot or saved to an existing payload in the robot. The full process steps are: 1. Enter payload identification state 2. Start payload identification 3. Get payload identification result 4. Exit payload identification state
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported

Example Code

Motion/PayloadIdentify.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Motion;
using Agilebot.IR.Types;

public class PayloadIdentify
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
```

```
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人模式
            // [EN] Get robot mode
            (UserOpMode opMode, StatusCode opCode) =
                controller.GetOpMode();
            if (opCode == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前机器人模式/Current robot mode: {opMode}"
                );
                if (opMode != UserOpMode.AUTO)
                {
                    Console.WriteLine(
                        $"负载测定执行必须在机器人自动模式下/Payload identification ex
ecution must be in automatic mode"
                    );
                    return StatusCode.OtherReason;
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取机器人模式失败/Failed to get robot mode: {opCode.GetDescri
ption()}"
                );
            }

            // [ZH] 检测3轴是否水平
            // [EN] Check if axis 3 is horizontal
            double horizontalAngle;
```

```
(horizontalAngle, code) =
    controller.Motion.Payload.CheckAxisThreeHorizontal();
if (code == StatusCode.OK)
{
    Console.WriteLine(
        $"3轴水平角度: {horizontalAngle}"
    );
    if (Math.Abs(horizontalAngle) > 1)
    {
        Console.WriteLine(
            "警告：3轴水平角度超出范围(-1~1)，无法进行负载测定"
        );
        return StatusCode.OtherReason;
    }
}
else
{
    Console.WriteLine(
        $"检测3轴水平失败: {code.GetDescription()}"
    );
}

// [ZH] 获取负载测定状态
// [EN] Get payload identification state
PayloadIdentifyState identifyState;
(identifyState, code) =
    controller.Motion.Payload.GetPayloadIdentifyState();
if (code == StatusCode.OK)
{
    Console.WriteLine(
        $"负载测定状态: {identifyState}"
    );
}
else
{
    Console.WriteLine(
        $"获取负载测定状态失败: {code.GetDescription()}"
    );
}

// [ZH] 执行完整的负载测定流程
// [EN] Execute complete payload identification process
```

```
            PayloadInfo payload;
            (payload, code) =
                controller.Motion.Payload.PayloadIdentify(
                    -1,
                    90
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine("负载测定成功:");
                Console.WriteLine(
                    $"负载重量: {payload.Weight}"
                );
                Console.WriteLine(
                    $"质心位置: X={payload.MassCenter.X}, Y={payload.MassCenter.Y},
Z={payload.MassCenter.Z}"
                );
                Console.WriteLine(
                    $"惯性矩: LX={payload.InertiaMoment.LX}, LY={payload.InertiaMom
ent.LY}, LZ={payload.InertiaMoment.LZ}"
                );

                // [ZH] 保存负载到机器人中
                // [EN] Save payload to robot
                payload.Id = 6;
                code = controller.Motion.Payload.AddPayload(
                    payload
                );
                if (code == StatusCode.OK)
                {
                    Console.WriteLine(
                        "保存负载到机器人成功"
                    );
                }
                else
                {
                    Console.WriteLine(
                        $"保存负载失败: {code.GetDescription()}"
                    );
                }
            }
            else
            {
```

```
                Console.WriteLine(
                    $"负载测定失败: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            Console.WriteLine(
                disconnectCode != StatusCode.OK
                    ? disconnectCode.GetDescription()
                    : "Successfully disconnected."
            );
        }

        return code;
    }
}
```

4.3 Robot Program Execution Class

4.3.1 Executing a Specified Program

Method Name	**Execution.Start**(string `programName`)
Description	Starts execution of the specified program in the robot controller.
Request Parameters	`programName` : string Name of the program to be executed
Return Value	StatusCode: Program start operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.3.2 Stopping the Currently Executing Program

Method Name	**Execution.Stop**(string `programName` = null)
Description	Stops the currently executing program or stops the robot's current motion command.
Request Parameters	`programName` : string Name of the program to be stopped, default is null, meaning stop the currently running program or motion command
Return Value	StatusCode: Stop operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.3.3 Returning Details of All Running Programs

Method Name	**Execution.AllRunningPrograms**()
Description	Gets detailed information of all running programs, including program IDs and program names.
Request Parameters	None
Return Value	Dictionary<string, int>: List of running program IDs and program names StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.3.4 Pausing Program Execution

Method Name	**Execution.Pause**(string `programName` = null)
Description	Pauses the currently executing program or pauses the robot's current motion.
Request Parameters	`programName` : string Name of the program to be paused, when not passed, defaults to controlling the currently running program or executing action
Return Value	StatusCode: Pause operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.3.5 Resuming Program Execution

Method Name	**Execution.Resume**(string `programName`)
Description	Continues running a program in paused state or resumes the robot's paused motion.
Request Parameters	`programName` : string Name of the program to be resumed, when not passed, defaults to controlling the currently running program or executing action
Return Value	StatusCode: Resume operation execution result

Method Name	**Execution.Resume**(string programName)
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Execution/ProgramExecution.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class ProgramExecution
{
    /// <summary>
    /// 测试程序执行完整流程功能
    /// 验证程序的启动、暂停、恢复和停止等完整操作流程
    /// </summary>
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
```

```
            return code;
        }

        try
        {
            Console.WriteLine(
                "开始程序执行完整流程/Starting Program Execution Complete Flow"
            );

            // [ZH] 获取测试文件路径
            // [EN] Get test file path
            string file_user_program = GetTestFilePath(
                "test_prog.xml"
            );

            // [ZH] 设置程序名称
            // [EN] Set program name
            string progName = "test_prog";

            // [ZH] 上传用户程序文件
            // [EN] Upload user program file
            code = controller.FileManager.Upload(
                file_user_program,
                FileType.UserProgram,
                true
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"用户程序文件上传成功/User Program File Upload Success: {progNa
me}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"用户程序文件上传失败/User Program File Upload Failed: {code.Ge
tDescription()}"
                );
                return code;
            }
```

```
// [ZH] 等待
// [EN] Wait
Thread.Sleep(3000);

// [ZH] 启动程序
// [EN] Start program
code = controller.Execution.Start(progName);
if (code == StatusCode.OK)
{
    Console.WriteLine(
        $"程序启动成功/Program Started Successfully: {progName}"
    );
}
else
{
    Console.WriteLine(
        $"程序启动失败/Program Start Failed: {code.GetDescription()}"
    );
    return code;
}
Thread.Sleep(2000);

// [ZH] 获取所有正在运行的程序列表
// [EN] Get all running programs list
Dictionary<string, int> progList;
(progList, code) =
    controller.Execution.AllRunningPrograms();
if (code == StatusCode.OK)
{
    Console.WriteLine(
        "获取运行程序列表成功/Get Running Programs List Success"
    );
    Console.WriteLine(
        $"运行程序数量/Running Programs Count: {progList.Count}"
    );
    foreach (var prog in progList)
    {
        Console.WriteLine(
            $"  程序/Program: {prog.Key}, 状态/Status: {prog.Value}"
        );
    }
}
```

```
            else
            {
                Console.WriteLine(
                    $"获取运行程序列表失败/Get Running Programs List Failed: {code.G
etDescription()}"
                );
                return code;
            }
            Thread.Sleep(2000);

            // [ZH] 暂停程序
            // [EN] Pause program
            code = controller.Execution.Pause(progName);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"程序暂停成功/Program Paused Successfully: {progName}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"程序暂停失败/Program Pause Failed: {code.GetDescription()}"
                );
                return code;
            }
            Thread.Sleep(2000);

            // [ZH] 恢复程序
            // [EN] Resume program
            code = controller.Execution.Resume(progName);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"程序恢复成功/Program Resumed Successfully: {progName}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"程序恢复失败/Program Resume Failed: {code.GetDescription()}"
                );
```

```
            return code;
        }
        Thread.Sleep(2000);

        // [ZH] 停止程序
        // [EN] Stop program
        code = controller.Execution.Stop(progName);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                $"程序停止成功/Program Stopped Successfully: {progName}"
            );
        }
        else
        {
            Console.WriteLine(
                $"程序停止失败/Program Stop Failed: {code.GetDescription()}"
            );
            return code;
        }

        // [ZH] 删除用户程序文件
        // [EN] Delete user program file
        code = controller.FileManager.Delete(
            progName,
            FileType.UserProgram
        );
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                $"用户程序文件删除成功/User Program File Delete Success: {progNa
me}"
            );
        }
        else
        {
            Console.WriteLine(
                $"用户程序文件删除失败/User Program File Delete Failed: {code.Ge
tDescription()}"
            );
            return code;
        }
```

```
            Console.WriteLine(
                "程序执行完整流程结束/Program Execution Complete Flow Test Complete
d"
            );
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }

    /// <summary>
    /// 获取test_files文件夹中文件的路径示例方法
    /// 展示如何获取当前程序目录下的test_files文件夹中的文件路径
    /// </summary>
    private static string GetTestFilePath(string fileName)
    {
        // [ZH] 获取当前程序集的目录
        // [EN] Get current assembly directory
        string? codeFilePath =
```

```
            new System.Diagnostics.StackTrace(true)
                .GetFrame(0)
                ?.GetFileName();
        if (string.IsNullOrEmpty(codeFilePath))
        {
            throw new InvalidOperationException(
                "无法获取当前文件路径/Cannot get current file path"
            );
        }

        string? codeDirectory = Path.GetDirectoryName(
            codeFilePath
        );
        if (string.IsNullOrEmpty(codeDirectory))
        {
            throw new InvalidOperationException(
                "无法获取当前目录路径/Cannot get current directory path"
            );
        }

        // [ZH] 构建test_files文件夹路径
        // [EN] Build test_files folder path
        string testFilesDirectory = Path.Combine(
            codeDirectory,
            "test_files"
        );
        // [ZH] 构建文件完整路径
        // [EN] Build complete file path
        string filePath = Path.Combine(
            testFilesDirectory,
            fileName
        );
        return filePath;
    }
}
```

4.3.6 Executing a BAS Script Program

Method Name	**Execution.ExecuteBasScript**(BasScript `script`)
Description	Executes a user-defined BAS script program.
Request Parameters	`script` : BasScript User-defined BAS script program
Return Value	StatusCode: Script execution operation execution result
Note	BAS script program pause, resume, and stop methods are the same as regular programs.
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported Industrial Robot: v7.6.0.0+

Example Code

Execution/ExecuteBasScript.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.BasScript;
using Agilebot.IR.Execution;
using Agilebot.IR.Types;

public class ExecuteBasScript
{
    /// <summary>
    /// 测试执行Bas脚本功能
    /// 验证能否成功执行包含条件判断、运动控制和赋值操作的Bas脚本
    /// </summary>
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );
```

```
        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            Console.WriteLine(
                "开始执行Bas脚本程序/Starting Execute BasScript"
            );

            // [ZH] 创建BAS脚本程序
            // [EN] Create BAS script program
            BasScript script = new BasScript("test_bas");

            // [ZH] 添加条件判断到脚本
            // [EN] Add conditional statement to script
            code = script.Logical.IF(
                RegisterType.R,
                1,
                OtherType.VALUE,
                0
            );
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"添加条件判断失败/Add Conditional Statement Failed: {code.GetDe
scription()}"
                );
                return code;
            }

            // [ZH] 添加运动控制到脚本
```

```
            // [EN] Add motion control to script
            BasScript.ExtraParam param =
                new BasScript.ExtraParam();
            param.Acceleration(80);
            code = script.Motion.MoveJoint(
                MovePoseType.PR,
                1,
                SpeedType.VALUE,
                30,
                SmoothType.SD,
                10,
                extraParam: param
            );
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"添加运动控制失败/Add Motion Control Failed: {code.GetDescripti
on()}"
                );
                return code;
            }

            // [ZH] 添加赋值操作到脚本
            // [EN] Add assignment operation to script
            code = script.AssignValue(AssignType.R, 1, 99);
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"添加赋值操作失败/Add Assignment Operation Failed: {code.GetDes
cription()}"
                );
                return code;
            }

            // [ZH] 结束条件判断
            // [EN] End conditional statement
            code = script.Logical.END_IF();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"结束条件判断失败/End Conditional Statement Failed: {code.GetDe
scription()}"
```

```
                );
                return code;
            }

            // [ZH] 等待上一个测试结束
            // [EN] Wait for previous test to end
            Thread.Sleep(1000);

            // [ZH] 执行BAS脚本程序
            // [EN] Execute BAS script program
            code = controller.Execution.ExecuteBasScript(
                script
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "BAS脚本执行成功/Execute BasScript Success"
                );
                Console.WriteLine(
                    "脚本包含条件判断、运动控制和赋值操作/Script includes conditional
statements, motion control and assignment operations"
                );
            }
            else
            {
                Console.WriteLine(
                    $"BAS脚本执行失败/Execute BasScript Failed: {code.GetDescription
()}"
                );
            }

            Console.WriteLine(
                "执行Bas脚本测试完成/Execute BasScript Test Completed"
            );
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
```

```
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.4 Program Information Read/Write Operations

4.4.1 Reading the Value of a Specified Pose in a Program

Method Name	**ProgramPoses.Read**(string `programName` , int `index` , FileType `ft` = FileType.UserProgram)
Description	Reads the pose data at the specified index in the specified program.
Request Parameters	`programName` : string Specified program name `index` : int Specified pose index `ft` : FileType File type
Return Value	ProgramPose Robot pose data in the program StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/ReadProgramPose.cs

```cs
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class ReadProgramPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
```

```
        controllerIP,
        useLocalProxy
    );

    // [ZH] 连接捷勃特机器人
    // [EN] Connect to the Agilebot robot
    StatusCode code = controller.ConnectSync();
    Console.WriteLine(
        code != StatusCode.OK
            ? code.GetDescription()
            : "连接成功/Successfully connected."
    );

    if (code != StatusCode.OK)
    {
        return code;
    }

    try
    {
        // [ZH] 设置程序名称和位姿点索引
        // [EN] Set program name and pose index
        string progName = "test_prog";
        int index = 1;

        // [ZH] 读取指定程序中指定位姿点值
        // [EN] Read specified pose value in specified program
        ProgramPose pose;
        (pose, code) = controller.ProgramPoses.Read(
            progName,
            index
        );
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "读取程序位姿点成功/Read Program Pose Success"
            );
            Console.WriteLine(
                $"位姿信息/Pose Info: {pose}"
            );
        }
        else
```

```
            {
                Console.WriteLine(
                    $"读取程序位姿点失败/Read Program Pose Failed: {code.GetDescript
ion()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.4.2 Wirting the Value of a Specified Pose in a Program

Method Name	**ProgramPoses.Write**(string `programName` , int `index` , ProgramPose `value` , FileType `ft` = FileType.UserProgram)
Description	Modifies the pose data at the specified index in the specified program.
Request Parameters	`programName` : string Specified program name `index` : int Specified pose index `value` : ProgramPose Robot pose data in the program `ft` : FileType File type
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/WriteProgramPose.cs

```cs
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class WriteProgramPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
```

```
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置程序名称和位姿点索引
            // [EN] Set program name and pose index
            string progName = "test_prog";
            int index = 2;

            // [ZH] 生成随机位姿点
            // [EN] Generate random pose
            ProgramPose rndPose =
                ProgramPose.GenerateRandomPose(index);

            // [ZH] 修改指定程序中指定位姿点值
            // [EN] Write specified pose value in specified program
            code = controller.ProgramPoses.Write(
                progName,
                index,
                rndPose
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入程序位姿点成功/Write Program Pose Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入程序位姿点失败/Write Program Pose Failed: {code.GetDescrip
tion()}"
                );
            }
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.4.3 Adding a Pose to a Specified Program

Method Name	**ProgramPoses.Add**(string `programName` , int `index` , ProgramPose `value` , FileType `ft` = FileType.UserProgram)
Description	Adds pose data at the specified index position in the specified program.
Request Parameters	`programName` : string Specified program name `index` : int Specified pose index

Method Name	**ProgramPoses.Add**(string `programName` , int `index` , ProgramPose `value` , FileType `ft` = FileType.UserProgram)
	`value` : ProgramPose Robot pose data in the program `ft` : FileType File type
Return Value	StatusCode: Add operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/AddProgramPose.cs

```cs
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class AddProgramPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );
```

```
        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置程序名称和位姿点索引
            // [EN] Set program name and pose index
            string progName = "test_prog";
            int index = 3;

            // [ZH] 生成随机位姿点
            // [EN] Generate random pose
            ProgramPose rndPose =
                ProgramPose.GenerateRandomPose(index);

            // [ZH] 添加指定程序中指定位姿点
            // [EN] Add specified pose in specified program
            code = controller.ProgramPoses.Add(
                progName,
                index,
                rndPose
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "添加程序位姿点成功/Add Program Pose Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"添加程序位姿点失败/Add Program Pose Failed: {code.GetDescripti
on()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
```

```
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.4.4 Deleting a Specified Pose from a Program

Method Name	**ProgramPoses.Delete**(string `programName` , int `index` , FileType `ft` = FileType.UserProgram)
Description	Deletes the pose at the specified index in the specified program.
Request Parameters	`programName` : string Specified program name `index` : int Specified pose index `ft` : FileType File type
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/DeleteProgramPose.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class DeleteProgramPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置程序名称和位姿点索引
            // [EN] Set program name and pose index
            string progName = "test_prog";
```

```
            int index = 3;

            // [ZH] 删除指定程序中指定位姿点
            // [EN] Delete specified pose in specified program
            code = controller.ProgramPoses.Delete(
                progName,
                index
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "删除程序位姿点成功/Delete Program Pose Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"删除程序位姿点失败/Delete Program Pose Failed: {code.GetDescri
ption()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
```

```
                    code = disconnectCode;
                }
            }

            return code;
        }
    }
```

4.4.5 Retrieving All Poses from a Specified Program

Method Name	**ProgramPoses.ReadAllPoses**(string `programName` , FileType `ft` = FileType.UserProgram)
Description	Gets all pose information from the specified program.
Request Parameters	`programName` : string Specified program name `ft` : FileType File type
Return Value	List<ProgramPose>: Pose data list StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/ReadAllProgramPoses.cs

cs

```
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class ReadAllProgramPoses
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
```

```
{
    // [ZH] 初始化捷勃特机器人
    // [EN] Initialize the Agilebot robot
    Arm controller = new Arm(
        controllerIP,
        useLocalProxy
    );

    // [ZH] 连接捷勃特机器人
    // [EN] Connect to the Agilebot robot
    StatusCode code = controller.ConnectSync();
    Console.WriteLine(
        code != StatusCode.OK
            ? code.GetDescription()
            : "连接成功/Successfully connected."
    );

    if (code != StatusCode.OK)
    {
        return code;
    }

    try
    {
        // [ZH] 设置程序名称
        // [EN] Set program name
        string progName = "test_prog";

        // [ZH] 读取指定程序中所有位姿点
        // [EN] Read all poses in specified program
        List<ProgramPose> poses;
        (poses, code) =
            controller.ProgramPoses.ReadAllPoses(
                progName
            );
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "读取所有程序位姿点成功/Read All Program Poses Success"
            );
            Console.WriteLine(
                $"位姿点数量/Number of poses: {poses.Count}"
```

```
                );

                for (int i = 0; i < poses.Count; i++)
                {
                    Console.WriteLine(
                        $"位姿点 {i + 1}/Pose {i + 1}: {poses[i]}"
                    );
                }
            }
            else
            {
                Console.WriteLine(
                    $"读取所有程序位姿点失败/Read All Program Poses Failed: {code.Get
Description()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
```

```
    }
}
```

4.4.6 Converting Pose Types in Robot Programs

Method Name	**ProgramPoses.ConvertPose**(ProgramPose pose, PoseType toType)
Description	Converts robot poses in the program between joint coordinates and Cartesian space coordinates.
Request Parameters	`pose` : ProgramPose Robot pose data in the program `toType` : PoseType Desired coordinate type after conversion
Return Value	ProgramPose: Converted pose data StatusCode: Conversion operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

ProgramPoses/ConvertProgramPose.cs

```cs
using Agilebot.IR;
using Agilebot.IR.ProgramPoses;
using Agilebot.IR.Types;

public class ConvertProgramPose
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
```

```
    useLocalProxy
);

// [ZH] 连接捷勃特机器人
// [EN] Connect to the Agilebot robot
StatusCode code = controller.ConnectSync();
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 设置程序名称和位姿点索引
    // [EN] Set program name and pose index
    string progName = "test_prog";
    int cartIndex = 1;

    // [ZH] 先读取一个位姿点
    // [EN] First read a pose
    ProgramPose cartPose;
    (cartPose, code) = controller.ProgramPoses.Read(
        progName,
        cartIndex
    );
    if (code != StatusCode.OK)
    {
        Console.WriteLine(
            $"读取位姿点失败/Read Pose Failed: {code.GetDescription()}"
        );
        return code;
    }

    // [ZH] 转换位姿点类型（从笛卡尔坐标转换为关节坐标）
    // [EN] Convert pose type (from Cartesian to Joint coordinates)
    ProgramPose pose;
```

```
            (pose, code) =
                controller.ProgramPoses.ConvertPose(
                    cartPose,
                    PoseType.Joint
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "转换程序位姿点成功/Convert Program Pose Success"
                );
                Console.WriteLine(
                    $"原始位姿/Original Pose: {cartPose}"
                );
                Console.WriteLine(
                    $"转换后位姿/Converted Pose: {pose}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"转换程序位姿点失败/Convert Program Pose Failed: {code.GetDescr
iption()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
```

```
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.5 IO Signals

4.5.1 Reading the Value of a Specified Type and Port IO

Method Name	**Signals.Read**(SignalType `type` , int `index`)
Description	Reads the IO signal value of the specified type and port.
Request Parameters	`type` : SignalType IO signal type to read `index` : int IO port index to read, starting from 1
Return Value	int: IO signal value, 1 represents high level, 0 represents low level StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Signals/ReadSignal.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class ReadSignal
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );
```

```
        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置IO信号类型和索引
            // [EN] Set IO signal type and index
            SignalType type = SignalType.DI;
            int index = 1;

            // [ZH] 读取指定类型指定端口IO的值
            // [EN] Read specified type and port IO value
            int res;
            (res, code) = controller.Signals.Read(
                type,
                index
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "读取IO信号成功/Read Signal Success"
                );
                Console.WriteLine(
                    $"{type}: {index} 的值为/has value {res}"
                );
                Console.WriteLine(
                    $"信号状态/Signal Status: {(res == 1 ? "高电平/High Level" : "低
电平/Low Level")}"
                );
            }
```

```
            else
            {
                Console.WriteLine(
                    $"读取IO信号失败/Read Signal Failed: {code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.5.2 Writing the Value of a Specified Type and Port IO

Method Name	**Signals.Write**(SignalType `type` , int `index` , double `value`)
Description	Writes the IO signal value of the specified type and port.
Request Parameters	`type` : SignalType IO signal type to write `index` : int IO port index to write, starting from 1 `value` : double Signal value to write, 1 represents high level, 0 represents low level
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	UI/UO signals can only be read, not written

Example Code

Signals/WriteSignal.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class WriteSignal
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
```

```
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置IO信号类型、索引和值
            // [EN] Set IO signal type, index and value
            SignalType type = SignalType.DO;
            int index = 1;
            int value = 1;

            // [ZH] 写指定类型指定端口IO的值
            // [EN] Write specified type and port IO value
            code = controller.Signals.Write(
                type,
                index,
                value
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入IO信号成功/Write Signal Success"
                );
                Console.WriteLine(
                    $"{type}: {index} 设置为/set to value {value}"
                );
                Console.WriteLine(
                    $"信号状态/Signal Status: {(value == 1 ? "高电平/High Level" :
"低电平/Low Level")}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入IO信号失败/Write Signal Failed: {code.GetDescription()}"
                );
```

```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.5.3 Batch Write DO Signals

Method Name	**Signals.MultiWrite**(SignalType `type` , List<int> `ioData`)
Description	Batch write multiple DO ports with a flattened list of port/value pairs.
Request Parameters	`type` : SignalType DO only `ioData` : List<int> like `[port1, state1, port2, state2, ...]` , length must be

Method Name	**Signals.MultiWrite**(SignalType `type` , List<int> `ioData`)
	even
Return Value	StatusCode: write result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	Only DO supports batch write; UI/UO are read-only, DI/RI support single-point read only

Example Code

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class Test
{
    public static async Task Main()
    {
        string controllerIP = "10.27.1.254";
        Arm controller = new Arm(controllerIP);
        StatusCode code = await controller.Connect();
        Console.WriteLine(code != StatusCode.OK ? code.GetDescription() : "Successf
ully connected.");

        // Batch write DO1=1, DO2=0
        code = controller.Signals.MultiWrite(SignalType.DO, new List<int> { 1, 1,
2, 0 });
        Console.WriteLine(code != StatusCode.OK ? code.GetDescription() : "MultiWri
te Success");

        code = controller.Disconnect();
        Console.WriteLine(code != StatusCode.OK ? code.GetDescription() : "Successf
ully disconnected.");
    }
}
```

4.5.4 Batch Read DO Signals

Method Name	**Signals.MultiRead**(SignalType `type` , List<int> `indexes`)
Description	Batch read multiple DO ports; returned values align with the input order.
Request Parameters	`type` : SignalType DO only `indexes` : List<int> ports to read, at least one index required
Return Value	List<int> values (ordered as input) plus StatusCode
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+
Note	Only DO supports batch read; UI/UO are read-only, DI/RI support single-point read only

Example Code

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class Test
{
    public static async Task Main()
    {
        string controllerIP = "10.27.1.254";
        Arm controller = new Arm(controllerIP);
        StatusCode code = await controller.Connect();
        Console.WriteLine(code != StatusCode.OK ? code.GetDescription() : "Successf
ully connected.");

        // Batch read DO1, DO2
        (List<int> values, StatusCode readCode) = controller.Signals.MultiRead(Sign
alType.DO, new List<int> { 1, 2 });
        if (readCode == StatusCode.OK)
        {
            Console.WriteLine($"MultiRead Success: DO1={values[0]}, DO2={values
[1]}");
        }
```

```
        code = controller.Disconnect();
        Console.WriteLine(code != StatusCode.OK ? code.GetDescription() : "Successf
ully disconnected.");
    }
}
```

4.6 Register Information

4.6.1 R Numeric Register Operations

4.6.1.1 Reading the Value of an R Register

Method Name	**Registers.Read_R**(int `index`)
Description	Reads the value of an R numeric register.
Request Parameters	`index` : int R register number to read
Return Value	double: R register numeric value StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.1.2 Writing the Value of an R Register

Method Name	**Registers.Write_R**(int `index` , double `value`)
Description	Writes the value of an R numeric register.
Request Parameters	`index` : int R register number to write `value` : double R register numeric value to write
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.1.3 Deleting an R Register

Method Name	**Registers.Delete_R**(int `index`)
Description	Deletes the specified R numeric register.
Request Parameters	`index` : int R register number to delete
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Registers/RRegisterOperations.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class RRegisterOperations
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );
```

```
        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置寄存器索引和值
            // [EN] Set register index and value
            int index = 1;
            double value = 9.9;

            // [ZH] 写入R寄存器
            // [EN] Write R register
            code = controller.Registers.Write_R(
                index,
                value
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入R寄存器成功/Write R Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入R寄存器失败/Write R Register Failed: {code.GetDescription
()}"
                );
            }

            // [ZH] 读取R寄存器
            // [EN] Read R register
            double readValue;
            (readValue, code) = controller.Registers.Read_R(
                index
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"读取R寄存器成功/Read R Register Success: 值/Value = {readValu
```

```
e}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"读取R寄存器失败/Read R Register Failed: {code.GetDescription
()}"
                );
            }

            // [ZH] 删除R寄存器
            // [EN] Delete R register
            code = controller.Registers.Delete_R(index);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "删除R寄存器成功/Delete R Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"删除R寄存器失败/Delete R Register Failed: {code.GetDescription
()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
```

```
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.6.2 MR Motion Register Operations

4.6.2.1 Reading the Value of an MR Register

Method Name	**Registers.Read_MR**(int `index`)
Description	Reads the value of an MR motion register.
Request Parameters	`index` : int MR register number to read
Return Value	int: MR register numeric value StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.2.2 Writing the Value of an MR Register

Method Name	**Registers.Write_MR**(int `index` , int `value`)
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

Method Name	**Registers.Write_MR**(int `index` , int `value`)
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.2.3 Deleting an MR Register

Method Name	**Registers.Delete_MR**(int `index`)
Description	Deletes the specified MR motion register.
Request Parameters	`index` : int MR register number to delete
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Registers/MRRegisterOperations.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class MRRegisterOperations
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );
```

```
        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置寄存器索引和值
            // [EN] Set register index and value
            int index = 1;
            int value = 9;

            // [ZH] 写入MR寄存器
            // [EN] Write MR register
            code = controller.Registers.Write_MR(
                index,
                value
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入MR寄存器成功/Write MR Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入MR寄存器失败/Write MR Register Failed: {code.GetDescriptio
n()}"
                );
            }

            // [ZH] 读取MR寄存器
```

```
            // [EN] Read MR register
            int readValue;
            (readValue, code) =
                controller.Registers.Read_MR(index);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"读取MR寄存器成功/Read MR Register Success: 值/Value = {readVal
ue}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"读取MR寄存器失败/Read MR Register Failed: {code.GetDescription
()}"
                );
            }

            // [ZH] 删除MR寄存器
            // [EN] Delete MR register
            code = controller.Registers.Delete_MR(index);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "删除MR寄存器成功/Delete MR Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"删除MR寄存器失败/Delete MR Register Failed: {code.GetDescripti
on()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
```

```
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.6.3 SR String Register Operations

4.6.3.1 Reading the Value of an SR Register

Method Name	**Registers.Read_SR**(int `index`)
Description	Reads the value of an SR string register.
Request Parameters	`index` : int SR register number to read
Return Value	string: SR register string value StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.3.2 Writing the Value of an SR Register

Method Name	**Registers.Write_SR**(int `index` , string `value`)
Description	Writes the value of an SR string register.
Request Parameters	`index` : int SR register number to write `value` : string SR register string value to write
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.3.3 Deleting an SR Register

Method Name	**Registers.Delete_SR**(int `index`)
Description	Deletes the specified SR string register.
Request Parameters	`index` : int SR register number to delete
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Registers/SRRegisterOperations.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class SRRegisterOperations
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
```

```
// [ZH] 初始化捷勃特机器人
// [EN] Initialize the Agilebot robot
Arm controller = new Arm(
    controllerIP,
    useLocalProxy
);

// [ZH] 连接捷勃特机器人
// [EN] Connect to the Agilebot robot
StatusCode code = controller.ConnectSync();
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 设置寄存器索引和值
    // [EN] Set register index and value
    int index = 1;
    string value = "test";

    // [ZH] 写入SR寄存器
    // [EN] Write SR register
    code = controller.Registers.Write_SR(
        index,
        value
    );
    if (code == StatusCode.OK)
    {
        Console.WriteLine(
            "写入SR寄存器成功/Write SR Register Success"
        );
    }
    else
    {
```

```
            Console.WriteLine(
                $"写入SR寄存器失败/Write SR Register Failed: {code.GetDescriptio
n()}"
            );
        }

        // [ZH] 读取SR寄存器
        // [EN] Read SR register
        string readValue;
        (readValue, code) =
            controller.Registers.Read_SR(index);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                $"读取SR寄存器成功/Read SR Register Success: 值/Value = {readVal
ue}"
            );
        }
        else
        {
            Console.WriteLine(
                $"读取SR寄存器失败/Read SR Register Failed: {code.GetDescription
()}"
            );
        }

        // [ZH] 删除SR寄存器
        // [EN] Delete SR register
        code = controller.Registers.Delete_SR(index);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "删除SR寄存器成功/Delete SR Register Success"
            );
        }
        else
        {
            Console.WriteLine(
                $"删除SR寄存器失败/Delete SR Register Failed: {code.GetDescripti
on()}"
            );
        }
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.6.4 PR Pose Register Operations

4.6.4.1 Reading the Value of a PR Register

Method Name	**Registers.Read_PR**(int `index`)
Description	Reads the value of a PR pose register.
Request Parameters	`index` : int PR register number to read

Method Name	**Registers.Read_PR**(int `index`)
Return Value	PoseRegister: PR register pose data StatusCode: Read operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.4.2 Writing the Value of a PR Register

Method Name	**Registers.Write_PR**(int `index` , PoseRegister `value`)
Description	Writes the value of a PR pose register.
Request Parameters	`index` : int PR register number to write `value` : PoseRegister PR register pose data to write
Return Value	StatusCode: Write operation execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.1+ Industrial (Bronze): v7.6.0.0+

4.6.4.3 Deleting a PR Register

Method Name	**Registers.Delete_PR**(int `index`)
Description	Deletes the specified PR pose register.
Request Parameters	`index` : int PR register number to delete
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Registers/PRRegisterOperations.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Registers;
```

```
using Agilebot.IR.Types;

public class PRRegisterOperations
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置寄存器索引
            // [EN] Set register index
            int index = 1;

            // [ZH] 生成位姿寄存器
            // [EN] Generate pose register
            var pose = new PoseRegister
            {
                Id = 1,
                Name = "Test",
```

```
            Comment = "Test",
            PoseRegisterData = new PoseRegisterData
            {
                Pt = PoseType.Joint,
                Joint = new Joint
                {
                    J1 = 6.6,
                    J2 = 6.6,
                    J3 = 6.6,
                    J4 = 6.6,
                    J5 = 6.6,
                    J6 = 6.6,
                },
                CartData = null,
            },
        };

        // [ZH] 写入PR寄存器
        // [EN] Write PR register
        code = controller.Registers.Write_PR(pose);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "写入PR寄存器成功/Write PR Register Success"
            );
        }
        else
        {
            Console.WriteLine(
                $"写入PR寄存器失败/Write PR Register Failed: {code.GetDescriptio
n()}"
            );
        }

        // [ZH] 读取PR寄存器
        // [EN] Read PR register
        PoseRegister readValue;
        (readValue, code) =
            controller.Registers.Read_PR(index);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
```

```
                    $"读取PR寄存器成功/Read PR Register Success: ID = {readValue.I
d}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"读取PR寄存器失败/Read PR Register Failed: {code.GetDescription
()}"
                );
            }

            // [ZH] 删除PR寄存器
            // [EN] Delete PR register
            code = controller.Registers.Delete_PR(index);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "删除PR寄存器成功/Delete PR Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"删除PR寄存器失败/Delete PR Register Failed: {code.GetDescripti
on()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
```

```
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

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

4.6.5.1 Reading the Value of an MH Register

Method Name	**Registers.Read_MH**(int `index`)
Description	Gets the value of an MH register.
Request Parameters	`index` : int Register number to get
Return Value	Register information StatusCode: Function execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

4.6.5.2 Reading the Value of an MI Register

Method Name	**Registers.Read_MI**(int `index`)
Description	Gets the value of an MI register.

Method Name	**Registers.Read_MI**(int `index`)
Request Parameters	`index` : int Register number to get
Return Value	Register information StatusCode: Function execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

4.6.5.3 Writing the Value of an MH Register

Method Name	**Registers.Write_MH**(int `index` , int `value`)
Description	Updates the value of an MH register.
Request Parameters	`index` : int Register number `value` : int Register information to update
Return Value	StatusCode: Function execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

4.6.5.4 Writing the Value of an MI Register

Method Name	**Registers.Write_MI**(int `index` , int `value`)
Description	Updates the value of an MI register.
Request Parameters	`index` : int Register number `value` : int Register information to update
Return Value	StatusCode: Function execution result
Compatible robot software version	Collaborative (Copper): v7.6.0.0+ Industrial (Bronze): v7.6.0.0+

Example Code

Registers/ModbusRegisterOperations.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class ModbusRegisterOperations
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 设置寄存器索引和值
            // [EN] Set register index and value
            int index = 1;
            int writeValue = 8;

            // [ZH] 写入MH保持寄存器
            // [EN] Write MH holding register
            code = controller.Registers.Write_MH(
```

```
                index,
                writeValue
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入MH保持寄存器成功/Write MH Holding Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入MH保持寄存器失败/Write MH Holding Register Failed: {code.G
etDescription()}"
                );
            }

            // [ZH] 写入MI输入寄存器
            // [EN] Write MI input register
            code = controller.Registers.Write_MI(
                index,
                writeValue + 1
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "写入MI输入寄存器成功/Write MI Input Register Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"写入MI输入寄存器失败/Write MI Input Register Failed: {code.Get
Description()}"
                );
            }

            // [ZH] 读取MH保持寄存器
            // [EN] Read MH holding register
            int mhValue;
            (mhValue, code) = controller.Registers.Read_MH(
                index
```

```
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"读取MH保持寄存器成功/Read MH Holding Register Success: 值/Valu
e = {mhValue}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"读取MH保持寄存器失败/Read MH Holding Register Failed: {code.Ge
tDescription()}"
                );
            }

            // [ZH] 读取MI输入寄存器
            // [EN] Read MI input register
            int miValue;
            (miValue, code) = controller.Registers.Read_MI(
                index
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"读取MI输入寄存器成功/Read MI Input Register Success: 值/Value
= {miValue}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"读取MI输入寄存器失败/Read MI Input Register Failed: {code.GetD
escription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
```

```
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.7 Trajectory Control

4.7.1 Setting the Offline Trajectory File

Method Name	**Trajectory.SetOffLineTrajectoryFile**(string `path`)
Description	Sets the offline trajectory file to be executed.
Request Parameters	`path` : string Offline trajectory file path, such as example file A.trajectory A.trajectory trajectory file format is a **text file**, described as follows: - Line 1: 6 represents 6 axes, 0.001 represents 1ms interval between two points, 8093 represents a total of 8093 trajectory points - Line 2: Represents the initial positions of the 6 axes - Lines 3-8095: Represent trajectory points, including positions, velocities, accelerations, torque feedforward, do ports, and values of do ports for the 6 axes - do_port represents the used do port (range 1-24) - do_port is -1, indicating no IO signal will be triggered at this position - do_port is 1, do_state is 1, indicating do1 port will trigger ON signal at this position - do_port is 1, do_state is 0, indicating do1 port will trigger OFF signal at this position Users upload the offline file to the robot controller root directory using FileManager.upload, then execute the trajectory using instructions 4.7.2 and 4.7.3
Return Value	StatusCode: Set operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.7.2 Moving the Robot to the Start Point of the Offline Trajectory

Method Name	**Trajectory.PrepareOfflineTrajectory**()
Description	Moves the robot to the start point of the offline trajectory at a safe speed.

Method Name	**Trajectory.PrepareOfflineTrajectory**()
Request Parameters	None
Return Value	StatusCode: Prepare operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.7.3 Starting Execution of the Offline Trajectory File

Method Name	**Trajectory.ExecuteOfflineTrajectory**()
Description	Starts the execution of the offline trajectory file.
Request Parameters	None
Return Value	StatusCode: Execute operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.7.4 Convert CSV Trajectory File to Trajectory Format

Method	**Trajectory.TransformCsvToTrajectory**(string `fileName`)
Description	Converts a trajectory CSV file into the trajectory format and saves it to the controller's trajectory file directory.
Request Parameter	`fileName` : string – name of the CSV trajectory file.
Return Value	string: path of the converted trajectory file. StatusCode: result of the conversion operation.
Compatible Robot Software Versions	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.7.5 Query Trajectory Conversion Status

Method	**Trajectory.CheckTransformStatus**(string `fileName`)
Description	Queries the working status of the TransformCsvToTrajectory process.
Request Parameter	`fileName` : string – result returned by the TransformCsvToTrajectory interface.
Return Value	TransformState: conversion state. StatusCode: result of the query operation.
Compatible Robot Software Versions	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Trajectory/OfflineTrajectory.cs

```cs
using System.IO;
using Agilebot.IR;
using Agilebot.IR.Trajectory;
using Agilebot.IR.Types;

public class OfflineTrajectory
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
```

```
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人模式
            // [EN] Get robot mode
            (UserOpMode opMode, StatusCode opCode) =
                controller.GetOpMode();
            if (opCode == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前机器人模式/Current robot mode: {opMode}"
                );
                if (opMode != UserOpMode.AUTO)
                {
                    Console.WriteLine(
                        $"离线轨迹执行必须在机器人自动模式下/Offline trajectory execut
ion must be in automatic mode"
                    );
                    return StatusCode.OtherReason;
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取机器人模式失败/Failed to get robot mode: {opCode.GetDescri
ption()}"
                );
            }

            // [ZH] 添加程序文件到机器人中
            // [EN] Add program file to robot
```

```
            string file_user_program = GetTestFilePath(
                "test.csv"
            );
            StatusCode ret_code =
                controller.FileManager.Upload(
                    file_user_program,
                    FileType.TmpFile,
                    true
                );
            if (ret_code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"上传文件失败/Upload file failed: {ret_code.GetDescription()}"
                );
                return ret_code;
            }
            Console.WriteLine(
                "文件上传成功/File upload success"
            );

            // [ZH] 测试CSV转换为轨迹文件功能
            // [EN] Test CSV to trajectory file conversion functionality
            string csvFilename = "test.csv";
            (
                string trajFileName,
                StatusCode transformCode
            ) =
                controller.Trajectory.TransformCsvToTrajectory(
                    csvFilename
                );

            if (transformCode != StatusCode.OK)
            {
                Console.WriteLine(
                    $"CSV转换失败/CSV conversion failed: {transformCode.GetDescript
ion()}"
                );
                return transformCode;
            }
            Console.WriteLine(
                $"CSV转换成功/CSV conversion success, trajectory file: {trajFileNam
e}"
```

```
            );

            // [ZH] 检查转换状态
            // [EN] Check conversion status
            var startTime = System.DateTime.Now;
            TransformState state;
            StatusCode statusCode;
            do
            {
                (state, statusCode) =
                    controller.Trajectory.CheckTransformStatus(
                        System.IO.Path.GetFileName(
                            trajFileName
                        )
                    );
                if (statusCode != StatusCode.OK)
                {
                    Console.WriteLine(
                        $"检查转换状态失败/Check transform status failed: {statusCod
e.GetDescription()}"
                    );
                    return statusCode;
                }

                Console.WriteLine(
                    $"转换状态/Transform state: {state}"
                );
                Thread.Sleep(2000); // 等待2秒

                if (
                    System.DateTime.Now - startTime
                    > System.TimeSpan.FromSeconds(60)
                )
                {
                    Console.WriteLine(
                        "转换状态检查超时/Transform status check timeout"
                    );
                    break;
                }
            } while (
                state != TransformState.TRANSFORM_SUCCESS
                && state != TransformState.TRANSFORM_FAILED
```

```
            );

            if (state == TransformState.TRANSFORM_FAILED)
            {
                Console.WriteLine(
                    "CSV转换失败/CSV conversion failed"
                );
                return StatusCode.OtherReason;
            }

            // [ZH] 转换任务成功并进行了结果查询后 服务端不会继续保存转换任务的状态
            // [EN] After the conversion task is successful and the result is queri
ed, the server will not continue to save the conversion task status
            (
                TransformState finalState,
                StatusCode finalCode
            ) = controller.Trajectory.CheckTransformStatus(
                System.IO.Path.GetFileName(trajFileName)
            );
            if (finalCode != StatusCode.OK)
            {
                Console.WriteLine(
                    $"最终状态检查失败/Final status check failed: {finalCode.GetDesc
ription()}"
                );
                return finalCode;
            }
            Console.WriteLine(
                $"最终转换状态/Final transform state: {finalState}"
            );

            // [ZH] 设置轨迹文件
            // [EN] Set trajectory file
            code =
                controller.Trajectory.SetOffLineTrajectoryFile(
                    "test_torque.trajectory"
                );
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"设置轨迹文件失败/Set trajectory file failed: {code.GetDescript
ion()}"
```

```
                );
                return code;
            }
            Console.WriteLine(
                "设置轨迹文件成功/Set trajectory file success"
            );

            // [ZH] 准备离线轨迹
            // [EN] Prepare offline trajectory
            code =
                controller.Trajectory.PrepareOfflineTrajectory();
            if (code != StatusCode.OK)
            {
                Console.WriteLine(
                    $"准备离线轨迹失败/Prepare offline trajectory failed: {code.GetD
escription()}"
                );
                return code;
            }
            Console.WriteLine(
                "准备离线轨迹成功/Prepare offline trajectory success"
            );

            // [ZH] 等待机器人和伺服器空闲
            // [EN] Wait for robot and servo to be idle
            startTime = System.DateTime.Now;
            RobotState robotStatus;
            ServoState servoStatus;
            StatusCode robotStatusCode;
            StatusCode servoStatusCode;

            do
            {
                (robotStatus, robotStatusCode) =
                    controller.GetRobotState();
                if (robotStatusCode != StatusCode.OK)
                {
                    Console.WriteLine(
                        $"获取机器人状态失败/Get robot state failed: {robotStatusCod
e.GetDescription()}"
                    );
                    return robotStatusCode;
```

```
                }

                (servoStatus, servoStatusCode) =
                    controller.GetServoState();
                if (servoStatusCode != StatusCode.OK)
                {
                    Console.WriteLine(
                        $"获取伺服状态失败/Get servo state failed: {servoStatusCode.
GetDescription()}"
                    );
                    return servoStatusCode;
                }

                Console.WriteLine(
                    $"机器人状态/Robot state: {robotStatus}, 伺服状态/Servo state:
{servoStatus}"
                );

                if (
                    robotStatus == RobotState.ROBOT_IDLE
                    && servoStatus == ServoState.SERVO_IDLE
                )
                {
                    Console.WriteLine(
                        "机器人和伺服器已空闲/Robot and servo are idle"
                    );
                    break;
                }

                Thread.Sleep(2000); // 等待2秒

                if (
                    System.DateTime.Now - startTime
                    > System.TimeSpan.FromSeconds(60)
                )
                {
                    Console.WriteLine(
                        "等待机器人和伺服器空闲超时/Waiting for robot and servo idle
timeout"
                    );
                    break;
                }
```

```
            } while (true);

            // [ZH] 执行离线轨迹
            // [EN] Execute offline trajectory
            code =
                controller.Trajectory.ExecuteOfflineTrajectory();
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "执行离线轨迹成功/Execute offline trajectory success"
                );
                Console.WriteLine(
                    "机器人开始执行轨迹程序/Robot started executing trajectory progra
m"
                );
            }
            else
            {
                Console.WriteLine(
                    $"执行离线轨迹失败/Execute offline trajectory failed: {code.GetD
escription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution/Exception o
ccurred during execution: {ex.Message}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
```

```
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }

    /// <summary>
    /// 获取test_files文件夹中文件的路径示例方法
    /// 展示如何获取当前程序目录下的test_files文件夹中的文件路径
    /// </summary>
    private static string GetTestFilePath(string fileName)
    {
        // 获取当前程序集的目录
        string? codeFilePath =
            new System.Diagnostics.StackTrace(true)
                .GetFrame(0)
                ?.GetFileName();
        if (string.IsNullOrEmpty(codeFilePath))
        {
            throw new InvalidOperationException(
                "无法获取当前文件路径/Cannot get current file path"
            );
        }

        string? codeDirectory = Path.GetDirectoryName(
            codeFilePath
        );
        if (string.IsNullOrEmpty(codeDirectory))
        {
            throw new InvalidOperationException(
                "无法获取当前目录路径/Cannot get current directory path"
            );
        }

        // 构建test_files文件夹路径
        string testFilesDirectory = Path.Combine(
            codeDirectory,
            "test_files"
        );
```

```
        // 构建文件完整路径
        string filePath = Path.Combine(
            testFilesDirectory,
            fileName
        );
        return filePath;
    }
}
```

4.8 Alarm Information

4.8.1 Getting the Most Severe Alarm

Method Name	**Alarm.GetTopAlarm()**
Description	Gets the most severe alarm information.
Request Parameters	None
Return Value	string: Alarm information string StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Alarm/GetTopAlarm.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetTopAlarm
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );
```

```
// [ZH] 连接捷勃特机器人
// [EN] Connect to the Agilebot robot
StatusCode code = controller.ConnectSync();
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 获取最严重的一条报警
    // [EN] Get the most severe alarm
    string topError;
    (topError, code) =
        controller.Alarm.GetTopAlarm();
    if (code == StatusCode.OK)
    {
        Console.WriteLine(
            "获取最严重报警成功/Get Top Alarm Success"
        );
        if (string.IsNullOrEmpty(topError))
        {
            Console.WriteLine(
                "当前无报警/No current alarms"
            );
        }
        else
        {
            Console.WriteLine(
                $"最严重报警/Most Severe Alarm: {topError}"
            );
        }
    }
    else
    {
        Console.WriteLine(
```

```
                    $"获取最严重报警失败/Get Top Alarm Failed: {code.GetDescription
()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.8.2 Getting All Active Alarms

Method Name	**Alarm.GetAllActiveAlarms()**
Description	Gets all currently active alarm information.

Method Name	**Alarm.GetAllActiveAlarms()**
Request Parameters	None
Return Value	List<string>: Alarm information list StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Alarm/GetAllActiveAlarms.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class GetAllActiveAlarms
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
```

```
{
    return code;
}

try
{
    // [ZH] 获取所有的活动的报警
    // [EN] Get all active alarms
    List<string> errors;
    (errors, code) =
        controller.Alarm.GetAllActiveAlarms();
    if (code == StatusCode.OK)
    {
        Console.WriteLine(
            "获取所有活动报警成功/Get All Active Alarm Success"
        );
        Console.WriteLine(
            $"活动报警数量/Active Alarm Count: {errors.Count}"
        );

        if (errors.Count == 0)
        {
            Console.WriteLine(
                "当前无活动报警/No active alarms"
            );
        }
        else
        {
            Console.WriteLine(
                "活动报警列表/Active Alarm List:"
            );
            for (int i = 0; i < errors.Count; i++)
            {
                Console.WriteLine(
                    $"  {i + 1}. {errors[i]}"
                );
            }
        }
    }
    else
    {
        Console.WriteLine(
```

```
                    $"获取所有活动报警失败/Get All Active Alarm Failed: {code.GetDes
cription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.8.3 Resetting Alarms

Method Name	**Alarm.ResetAlarms**()
Description	Resets errors.

Method Name	**Alarm.ResetAlarms**()
Request Parameters	None
Return Value	StatusCode: Function execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.9 File Service Class

4.9.1 Uploading a Local File to the Robot

Method Name	**FileManager.Upload**(string `filePath` , FileType `ft` , bool `overWriting` = false)
Description	Uploads a local file to the robot controller.
Request Parameters	`filePath` : string Absolute path of the local file to be uploaded `ft` : FileType Type of the file to be uploaded `overWriting` : bool Whether to overwrite existing file in robot controller, default is false (no overwrite)
Note	For USER_PROGRAM and BLOCK_PROGRAM, provide the full path to the `.xml` / `.block` file; the system also uploads the same-name `.json` (and `.xml` for BlockProgram).
Return Value	StatusCode: Upload operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.9.2 Downloading a Robot File to a Local Machine

Method Name	**FileManager.Download**(string `fileName` , FileType `ft` , string `savePath`)
Description	Downloads a file from the robot controller to local.
Request Parameters	`fileName` : string Name of the file to be downloaded `ft` : FileType Type of the file to be downloaded `savePath` : string Local save path for the downloaded file
Note	For `UserProgram` / `BlockProgram` / `TrajectoryProgram` , specify only the program name (filename without extension). For `TmpFile` , provide the full filename with extension.

Method Name	**FileManager.Download**(string `fileName` , FileType `ft` , string `savePath`)
Return Value	StatusCode: Download operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.9.3 Deleting a File from the Robot

Method Name	**FileManager.Delete**(string `fileName` , FileType `ft`)
Description	Deletes a file from the robot controller.
Request Parameters	`fileName` : string Name of the file to be deleted `ft` : FileType Type of the file to be deleted
Note	For `UserProgram` / `BlockProgram` / `TrajectoryProgram` , specify only the program name (filename without extension). For `TmpFile` , provide the full filename with extension.
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

FileManager/UserProgramOperations.cs

```cs
using System.Collections.Generic;
using System.IO;
using Agilebot.IR;
using Agilebot.IR.FileManager;
using Agilebot.IR.Types;

public class UserProgramOperations
{
    /// <summary>
    /// 测试用户程序文件的完整操作流程：上传、下载、搜索和删除
```

```
    /// </summary>
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            Console.WriteLine(
                "开始用户程序文件操作测试/Starting User Program File Operations Test"
            );

            // [ZH] 获取测试文件路径
            // [EN] Get test file path
            string file_user_program = GetTestFilePath(
                "test_prog.xml"
            );
            string fileName = "test_prog";
            string save_path = GetTestFilePath("download");

            // [ZH] 上传用户程序文件
```

```
            // [EN] Upload user program file
            code = controller.FileManager.Upload(
                file_user_program,
                FileType.UserProgram,
                true
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"用户程序文件上传成功/User Program File Upload Success: {fileNa
me}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"用户程序文件上传失败/User Program File Upload Failed: {code.Ge
tDescription()}"
                );
                return code;
            }

            // [ZH] 等待下载
            // [EN] Wait before download
            Thread.Sleep(1000);

            // [ZH] 下载用户程序文件
            // [EN] Download user program file
            code = controller.FileManager.Download(
                fileName,
                FileType.UserProgram,
                save_path
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"用户程序文件下载成功/User Program File Download Success: {file
Name}"
                );
            }
            else
            {
```

```
                Console.WriteLine(
                    $"用户程序文件下载失败/User Program File Download Failed: {code.
GetDescription()}"
                );
                return code;
            }

            // [ZH] 搜索用户程序文件
            // [EN] Search user program file
            List<string> results = new List<string>();
            (results, code) = controller.FileManager.Search(
                fileName
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"用户程序文件搜索成功/User Program File Search Success"
                );
                Console.WriteLine(
                    $"搜索结果数量/Search Results Count: {results.Count}"
                );
                foreach (var result in results)
                {
                    Console.WriteLine(
                        $"  找到文件/Found File: {result}"
                    );
                }
            }
            else
            {
                Console.WriteLine(
                    $"用户程序文件搜索失败/User Program File Search Failed: {code.Ge
tDescription()}"
                );
                return code;
            }

            // [ZH] 等待删除
            // [EN] Wait before delete
            Thread.Sleep(1000);

            // [ZH] 删除用户程序文件
```

```
            // [EN] Delete user program file
            code = controller.FileManager.Delete(
                fileName,
                FileType.UserProgram
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    $"用户程序文件删除成功/User Program File Delete Success: {fileNa
me}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"用户程序文件删除失败/User Program File Delete Failed: {code.Ge
tDescription()}"
                );
                return code;
            }

            Console.WriteLine(
                "用户程序文件操作测试完成/User Program File Operations Test Complete
d"
            );
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
```

```
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }

    /// <summary>
    /// 获取test_files文件夹中文件的路径示例方法
    /// 展示如何获取当前程序目录下的test_files文件夹中的文件路径
    /// </summary>
    private static string GetTestFilePath(string fileName)
    {
        // [ZH] 获取当前程序集的目录
        // [EN] Get current assembly directory
        string? codeFilePath =
            new System.Diagnostics.StackTrace(true)
                .GetFrame(0)
                ?.GetFileName();
        if (string.IsNullOrEmpty(codeFilePath))
        {
            throw new InvalidOperationException(
                "无法获取当前文件路径/Cannot get current file path"
            );
        }

        string? codeDirectory = Path.GetDirectoryName(
            codeFilePath
        );
        if (string.IsNullOrEmpty(codeDirectory))
        {
            throw new InvalidOperationException(
                "无法获取当前目录路径/Cannot get current directory path"
            );
        }

        // [ZH] 构建test_files文件夹路径
        // [EN] Build test_files folder path
```

```
        string testFilesDirectory = Path.Combine(
            codeDirectory,
            "test_files"
        );
        // [ZH] 构建文件完整路径
        // [EN] Build complete file path
        string filePath = Path.Combine(
            testFilesDirectory,
            fileName
        );
        return filePath;
    }
}
```

4.9.4 Search Files by Filename Pattern

Method Name	**FileManager.Search**(string `pattern` , ref List<string> `fl`)
Description	Searches for files on the robot controller that match the filename pattern.
Request Parameters	`pattern` : string Filename match pattern `fl` : ref List<string> Returned file list
Return Value	StatusCode: Search operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.10 BasScript Script Program Class

Method Name	**BasScript(** `name`)
Description	BasScript script program class constructor, corresponds to program instructions in teaching pendant program writing.
Request Parameters	`name` : string Script program name
Compatible robot software version	Collaborative (Copper): v7.5.2.0+ Industrial (Bronze): Not supported
Note	All methods in the BasScript script program class have the same compatible robot software version requirements as this class.

4.10.1 Motion to Point Instruction

Method Name	**BasScript.BasMotion.MoveJoint**(poseType, poseIndex, speedType, speedValue, smoothType, smoothDistance, extraParam)
Description	Executes joint motion instruction, corresponds to MoveJoint instruction in teaching pendant program writing.
Request Parameters	`poseType` : Pose type `poseIndex` : Pose index `speedType` : Speed type `speedValue` : Speed value `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.2 Linear Motion to Point Instruction

Method Name	**BasScript.BasMotion.MoveLine**(poseType, poseIndex, speedType, speedValue, smoothType, smoothDistance, extraParam)
Description	Executes linear motion instruction, corresponds to MoveLine instruction in teaching pendant program writing.
Request Parameters	`poseType` : Pose type `poseIndex` : Pose index `speedType` : Speed type `speedValue` : Speed value `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.3 Arc Motion to Point Instruction

Method Name	**BasScript.BasMotion.MoveCircle**(poseType1, poseIndex1, poseType2, poseIndex2, speedType, speedValue, smoothType, smoothDistance, extraParam)
Description	Executes arc motion instruction, corresponds to MoveCircle instruction in teaching pendant program writing.
Request Parameters	`poseType1` : Intermediate point pose type `poseIndex1` : Intermediate point pose index `poseType2` : End point pose type `poseIndex2` : End point pose index `speedType` : Speed type `speedValue` : Speed value `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.4 Jump Point-to-Point Motion Instruction

Method Name	**BasScript.BasMotion.Jump**(poseType, poseIndex, speedValue, speedRatio, limZType, limZValue, smoothType, smoothDistance, extraParam)
Description	JUMP instruction, robot point-to-point motion to specified position
Request Parameters	`poseType` : Target pose storage type `poseIndex` : Target position index `speedValue` : Motion speed value `speedRatio` : Motion speed ratio `limZType` : Z-axis limit type `limZValue` : Z-axis limit value `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.5 Jump3 Three-Point Jump Instruction

Method Name	**BasScript.BasMotion.Jump3**(poseType, poseIndex, speedValue, speedRatio, smoothType, smoothDistance, extraParam)
Description	JUMP3 instruction, robot point-to-point motion to specified position
Request Parameters	`poseType` : Target pose storage type `poseIndex` : 3 target position indices `speedValue` : Motion speed value `speedRatio` : Motion speed ratio `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.6 Jump3CP Three-Point Jump CP Instruction

Method Name	**BasScript.BasMotion.Jump3CP**(poseType, poseIndex, speedValue, smoothType, smoothDistance, extraParam)
Description	JUMP3CP instruction, robot point-to-point motion to specified position
Request Parameters	`poseType` : Target pose storage type `poseIndex` : 3 target position indices `speedValue` : Motion speed value `smoothType` : Smooth type `smoothDistance` : Smooth distance `extraParam` : Extra parameter
Return Value	StatusCode: Motion instruction execution result

4.10.7 Extra Parameter Class

Method Name	**ExtraParam.Acceleration**(value)
Description	Sets additional acceleration parameter
Request Parameters	`value` : double Acceleration value, range 1~120
Return Value	StatusCode: Parameter setting execution result

Method Name	**ExtraParam.RTCP**()
Description	Sets RTCP (Real-Time Control Protocol) parameter
Request Parameters	None
Return Value	StatusCode: Parameter setting execution result

Method Name	**ExtraParam.Offset**(index)
Description	Sets coordinate offset parameter
Request Parameters	`index` : int PR index for offset
Return Value	StatusCode: Parameter setting execution result

Method Name	**ExtraParam.TB**(second, type, name)
Description	Sets delay parameter to execute program instruction after current instruction runs
Request Parameters	`second` : double Delay in seconds `type` : string Instruction type `name` : string Program name
Return Value	StatusCode: Parameter setting execution result

Method Name	**ExtraParam.TB**(second, type, index, status)
Description	Sets delay parameter to assign value to specified IO after current instruction runs
Request Parameters	`second` : double Delay in seconds `type` : string IO type `index` : int IO index `status` : int Status to assign
Return Value	StatusCode: Parameter setting execution result

Method Name	**ExtraParam.SKIP**(index)
Description	Sets jump instruction parameter
Request Parameters	`index` : int Jump to the specified LABEL index
Return Value	StatusCode: Parameter setting execution result

4.10.8 AssignValue Assignment Instruction

Method Name	**BasScript.AssignValue**(param1, index, param2, value, optIndex, optValue)
Description	Assignment instruction
Request Parameters	param1: Type of parameter 1 index: Index of parameter 1 param2: Type of parameter 2 value: Value of parameter 2

Method Name	**BasScript.AssignValue**(param1, index, param2, value, optIndex, optValue)
	optIndex: Additional index for parameter 1 optValue: Additional value for parameter 2
Return Value	StatusCode: Result of function execution

4.10.9 AssignValue Assignment Instruction

Method Name	**BasScript.AssignValue**(param, index, value)
Description	Assign a value to a variable
Request Parameters	param: Parameter type (AssignType) index: Index (integer) value: Value (IOStatus, double, or string)
Return Value	StatusCode: Result of function execution

4.10.10 IF Conditional Instruction

Method Name	**BasScript.BasLogical.IF**(param1, index, param2, value, operatorType)
Description	Adds a logical IF statement to the script
Request Parameters	param1: First parameter, type RegisterType or IOType index: Index (integer) param2: Second parameter, type RegisterType, IOType, or OtherType value: Value, type index, number, string, or IOStatus operatorType: Boolean operator, default is equal
Return Value	StatusCode: Result of function execution

4.10.11 ELSE_IF Conditional Branch Instruction

Method Name	**BasScript.BasLogical.ELSE_IF**(param1, index, param2, value, operatorType)
Description	Adds a logical ELSE IF statement to the script
Request Parameters	param1: First parameter, type RegisterType or IOType index: Index (integer) param2: Second parameter, type RegisterType, IOType, or OtherType value: Value, type index, number, string, or IOStatus operatorType: Boolean operator, default is equal
Return Value	StatusCode: Result of function execution

4.10.12 ELSE Instruction

Method Name	**BasScript.BasLogical.ELSE**()
Description	Adds a logical ELSE statement to the script
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.13 END_IF End Conditional Instruction

Method Name	**BasScript.BasLogical.END_IF**()
Description	Ends the logical IF statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.14 WHILE Loop Instruction

Method Name	**BasScript.BasLogical.WHILE**(param1, index, param2, value, operatorType)
Description	Adds a logical WHILE statement to the script
Request Parameters	param1: First parameter, type RegisterType or IOType index: Index (integer) param2: Second parameter, type RegisterType, IOType, or OtherType value: Value, type index, number, string, or IOStatus operatorType: Boolean operator, default is equal
Return Value	StatusCode: Result of function execution

4.10.15 END_WHILE End Loop Instruction

Method Name	**BasScript.BasLogical.END_WHILE**()
Description	Ends the logical While statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.16 SWITCH Multi-Branch Selection Instruction

Method Name	**BasScript.BasLogical.SWITCH**(param, index)
Description	Adds a logical SWITCH statement to the script
Request Parameters	param: Parameter, type RegisterType or IOType index: Index of the parameter
Return Value	StatusCode: Result of function execution

4.10.17 CASE Branch Instruction

Method Name	**BasScript.BasLogical.CASE**(param, value)
Description	Adds a logical CASE statement to the script
Request Parameters	param: Parameter, type RegisterType, IOType, or OtherType value: Value, type index, number, string
Return Value	StatusCode: Result of function execution

4.10.18 DEFAULT Branch Instruction

Method Name	**BasScript.BasLogical.DEFAULT**()
Description	Adds a logical DEFAULT statement to the script
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.19 END_SWITCH End Multi-Branch Selection Instruction

Method Name	**BasScript.BasLogical.END_SWITCH**()
Description	Ends the logical SWITCH statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.20 SKIP_CONDITION Skip Condition Instruction

Method Name	**BasScript.BasLogical.SKIP_CONDITION**(param1, index, param2, value, operatorType)
Description	Adds a logical SKIP CONDITION statement to the script

Method Name	**BasScript.BasLogical.SKIP_CONDITION**(param1, index, param2, value, operatorType)
Request Parameters	param1: First parameter, type RegisterType or IOType index: Index of parameter 1 param2: Second parameter, type RegisterType, IOType, or OtherType value: Value, type index, number, string, or IOStatus operatorType: Boolean operator, default is equal
Return Value	StatusCode: Result of function execution

4.10.21 WAIT Wait Condition Instruction

Method Name	**BasScript.BasStructure.WAIT**(param1, index, param2, value, operatorType)
Description	Adds a logical WAIT COND statement to the script
Request Parameters	param1: First parameter, type RegisterType or IOType index: Index of parameter 1 param2: Second parameter, type ValuesType, IOType, or OtherType value: Value, type index, number, string, or IOStatus operatorType: Boolean operator, default is equal
Return Value	StatusCode: Result of function execution

4.10.22 WAIT_TIME Wait Time Instruction

Method Name	**BasScript.BasStructure.WAIT_TIME**(param, value)
Description	WAIT TIME waits for a certain amount of time
Request Parameters	param: Parameter type value: Time value to wait
Return Value	StatusCode: Result of function execution

4.10.23 GOTO Jump Instruction

Method Name	**BasScript.BasLogical.GOTO**(index)
Description	GOTO jump statement
Request Parameters	index: Index of the target label
Return Value	StatusCode: Result of function execution

4.10.24 LABEL Instruction

Method Name	**BasScript.BasLogical.LABEL**(index)
Description	LABEL statement
Request Parameters	index: Index of the label
Return Value	StatusCode: Result of function execution

4.10.25 BREAK Break Out of Loop Instruction

Method Name	**BasScript.BasLogical.BREAK**()
Description	BREAK statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.26 CONTINUE Skip Loop Instruction

Method Name	**BasScript.BasLogical.CONTINUE**()
Description	CONTINUE statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.27 PAUSE Instruction

Method Name	**BasScript.BasStructure.PAUSE**()
Description	PAUSE statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.28 ABORT Instruction

Method Name	**BasScript.BasStructure.ABORT**()
Description	ABORT statement
Request Parameters	None
Return Value	StatusCode: Result of function execution

4.10.29 CALL Synchronous Program Call Instruction

Method Name	**BasScript.BasStructure.CALL**(name)
Description	CALL synchronous program call
Request Parameters	name: Program name

Method Name	**BasScript.BasStructure.CALL**(name)
Return Value	StatusCode: Result of function execution

4.10.30 RUN Asynchronous Program Call Instruction

Method Name	**BasScript.BasStructure.RUN**(name)
Description	RUN asynchronous program call
Request Parameters	name: Program name
Return Value	StatusCode: Result of function execution

4.10.31 LOAD Load Program Instruction

Method Name	**BasScript.BasStructure.LOAD**(param, value)
Description	LOAD load program
Request Parameters	param: Parameter, R register, SR register, number, or string value: Value of the parameter, number or string
Return Value	StatusCode: Result of function execution

4.10.32 UNLOAD Unload Program Instruction

Method Name	**BasScript.BasStructure.UNLOAD**(param, value)
Description	UNLOAD unload program
Request Parameters	param: Parameter, R register, SR register, number, or string value: Value of the parameter, number or string
Return Value	StatusCode: Result of function execution

4.10.33 EXEC Execute Program Instruction

Method Name	**BasScript.BasStructure.EXEC**(param, value)
Description	EXEC execute program
Request Parameters	param: Parameter, R register, SR register, number, or string value: Value of the parameter, number or string
Return Value	StatusCode: Result of function execution

4.10.34 OPEN Open Socket Connection Instruction

Method Name	**BasScript.BasSocket.OPEN**(index)
Description	SOCKET OPEN open socket connection
Request Parameters	index: SK register index
Return Value	StatusCode: Result of function execution

4.10.35 CLOSE Close Socket Connection Instruction

Method Name	**BasScript.BasSocket.CLOSE**(index)
Description	SOCKET CLOSE close socket connection
Request Parameters	index: SK register index
Return Value	StatusCode: Result of function execution

4.10.36 CONNECT Socket Connection Instruction

Method Name	**BasScript.BasSocket.CONNECT**(index)
Description	SOCKET CONNECT connect socket
Request Parameters	index: SK register index
Return Value	StatusCode: Result of function execution

4.10.37 SEND Send Socket Data Instruction

Method Name	**BasScript.BasSocket.SEND**(index, msgType, value)
Description	SOCKET SEND send data via socket
Request Parameters	index: SK register index msgType: Message type value: Message content or index
Return Value	StatusCode: Result of function execution

4.10.38 RECV Receive Socket Data Instruction

Method Name	**BasScript.BasSocket.RECV**(index, msgLength, msgType, value)
Description	SOCKET RECV receive socket data
Request Parameters	index: SK register index msgLength: Message length msgType: Message type value: Message content or index
Return Value	StatusCode: Result of function execution

4.10.39 READ_MH Read Modbus Holding Register Instruction

Method Name	**BasScript.BasModbus.READ_MH**(index, id, address, length, rIndex)
Description	ReadMH read Modbus holding register
Request Parameters	index: Channel index id: Modbus ID address: Register address length: Register length rIndex: R register index to write to
Return Value	StatusCode: Result of function execution

4.10.40 READ_MI Read Modbus Input Register Instruction

Method Name	**BasScript.BasModbus.READ_MI**(index, id, address, length, rIndex)
Description	ReadMI read Modbus input register
Request Parameters	index: Channel index id: Modbus ID address: Register address length: Register length rIndex: R register index to write to
Return Value	StatusCode: Result of function execution

4.10.41 WRITE_MH Write Modbus Holding Register Instruction

Method Name	**BasScript.BasModbus.WRITE_MH**(index, id, address, length, valueType, value)
Description	ModbusWriteMH write to Modbus holding register
Request Parameters	index: Channel index id: Modbus ID address: Register address length: Register length

Method Name	**BasScript.BasModbus.WRITE_MH**(index, id, address, length, valueType, value)
	valueType: Value type value: Value or index
Return Value	StatusCode: Result of function execution

4.10.42 FIND Find Vision Program Instruction

Method Name	**BasScript.BasVision.FIND**(name)
Description	VISION FIND find vision program
Request Parameters	name: Vision program name
Return Value	StatusCode: Result of function execution

4.10.43 GET_OFFSET Get Vision Program Offset Instruction

Method Name	**BasScript.BasVision.GET_OFFSET**(name, index, labelIndex)
Description	VISION GET OFFSET get vision program offset
Request Parameters	name: Vision program name index: Vision register index labelIndex: Label index
Return Value	StatusCode: Result of function execution

4.10.44 GET_QUANTITY Get Vision Program Result Instruction

Method Name	**BasScript.BasVision.GET_QUANTITY**(name, index)
Description	VISION GET QUANTITY get vision program result

Method Name	**BasScript.BasVision.GET_QUANTITY**(name, index)
Request Parameters	name: Vision program name index: R register index
Return Value	StatusCode: Result of function execution

4.10.45 SetParam Set Parameter Instruction

Method Name	**BasScript.SetParam**(type, valueType, value)
Description	SET PARAM set parameter
Request Parameters	type: Parameter type valueType: Value type value: Value
Return Value	StatusCode: Result of function execution

4.11 Coordinate System Class

4.11.1 Getting Information of a Specified Coordinate System

Method Name	**CoordinateSystem.Get**(CoordinateType `type` , int `index`)
Description	Gets the corresponding coordinate system information based on the specified coordinate system type and index.
Request Parameters	`type` : CoordinateType Coordinate system type `index` : int Coordinate system index
Return Value	Coordinate: Coordinate system information data StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.11.2 Updating Coordinate System Information

Method Name	**CoordinateSystem.Update**(CoordinateType `type` , Coordinate `coordinate`)
Description	Updates the corresponding coordinate system based on the specified coordinate system type and coordinate system information.
Request Parameters	`type` : CoordinateType Coordinate system type `coordinate` : Coordinate Coordinate system information to be updated
Return Value	StatusCode: Update operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.11.3 Adding Coordinate System Information

Method Name	**CoordinateSystem.Add**(CoordinateType `type` , Coordinate `coordinate`)
Description	Adds a new coordinate system based on the specified coordinate system type and coordinate system information.
Request Parameters	`type` : CoordinateType Coordinate system type `coordinate` : Coordinate Coordinate system information to be added
Return Value	StatusCode: Add operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.11.4 Deleting Information of a Specified Coordinate System

Method Name	**CoordinateSystem.Delete**(CoordinateType `type` , int `index`)
Description	Deletes the corresponding coordinate system information based on the specified coordinate system type and index.
Request Parameters	`type` : CoordinateType Coordinate system type `index` : int Coordinate system index
Return Value	StatusCode: Delete operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

4.11.5 Getting a List of Coordinate System Information

Method Name	**CoordinateSystem.GetCoordinateList**(CoordinateType `type`)
Description	Gets a list of all coordinate system information based on the specified coordinate system type.
Request Parameters	`type` : CoordinateType Coordinate system type
Return Value	List<[CoordSummary][#3.22.2]>: Coordinate system information list StatusCode: Get operation execution result
Compatible robot software version	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

CoordinateSystem/TFCoordinateTest.cs

```cs
using Agilebot.IR;
using Agilebot.IR.CoordinateSystem;
using Agilebot.IR.Types;

public class TFCoordinateTest
{
    /// <summary>
    /// 测试 TF 坐标系的计算、添加、获取列表、获取单个坐标系、更新和删除操作
    /// </summary>
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
```

```
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

if (code != StatusCode.OK)
{
    return code;
}

try
{
    // [ZH] 准备测试数据
    // [EN] Prepare test data
    var poseData = new List<Position>
    {
        new Position(
            847.0999429718556,
            166.7999999999656,
            276.8195498896624,
            90,
            0,
            -70
        ),
        new Position(
            809.0227439212846,
            166.79999999994843,
            459.80354972094295,
            90,
            0,
            -45
        ),
        new Position(
            717.1223240422377,
            166.79999999993265,
            654.0891675073312,
            90,
            0,
            -30
        ),
        new Position(
```

```
                572.917828754028,
                166.79999999992168,
                825.1862002007621,
                90,
                0,
                -40
            ),
        };

        Console.WriteLine(
            "开始TF坐标系测试/Starting TF Coordinate Test"
        );

        // [ZH] 计算坐标系
        // [EN] Calculate coordinate system
        Coordinate calculatedCoord = new Coordinate();
        (Position coord, StatusCode calculateCode) =
            controller.CoordinateSystem.Calculate(
                CoordinateType.ToolCoordinate,
                poseData
            );

        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "计算TF坐标系成功/Calculate TF Coordinate Success"
            );
        }
        else
        {
            Console.WriteLine(
                $"计算TF坐标系失败/Calculate TF Coordinate Failed: {code.GetDesc
ription()}"
            );
            return code;
        }
        calculatedCoord.Id = 5;
        calculatedCoord.Data = coord;

        // [ZH] 删除可能存在的坐标系
        // [EN] Delete existing coordinate if exists
        StatusCode deleteCode =
```

```
            controller.CoordinateSystem.Delete(
                CoordinateType.ToolCoordinate,
                calculatedCoord.Id
            );
        Console.WriteLine(
            $"删除现有坐标系/Delete Existing Coordinate: {deleteCode.GetDescript
ion()}"
        );

        // [ZH] 添加坐标系
        // [EN] Add coordinate system
        StatusCode addCode =
            controller.CoordinateSystem.Add(
                CoordinateType.ToolCoordinate,
                calculatedCoord
            );
        if (addCode == StatusCode.OK)
        {
            Console.WriteLine(
                "添加TF坐标系成功/Add TF Coordinate Success"
            );
        }
        else
        {
            Console.WriteLine(
                $"添加TF坐标系失败/Add TF Coordinate Failed: {addCode.GetDescrip
tion()}"
            );
            return addCode;
        }

        // [ZH] 获取坐标系列表
        // [EN] Get coordinate list
        List<CoordSummary> listRes;
        (listRes, code) =
            controller.CoordinateSystem.GetCoordinateList(
                CoordinateType.ToolCoordinate
            );
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "获取TF坐标系列表成功/Get TF Coordinate List Success"
```

```
                );
                Console.WriteLine(
                    $"坐标系列表数量/Coordinate List Count: {listRes.Count}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"获取TF坐标系列表失败/Get TF Coordinate List Failed: {code.GetD
escription()}"
                );
                return code;
            }

            // [ZH] 获取单个坐标系
            // [EN] Get single coordinate
            Coordinate getCoord;
            (getCoord, code) =
                controller.CoordinateSystem.Get(
                    CoordinateType.ToolCoordinate,
                    calculatedCoord.Id
                );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "获取TF坐标系成功/Get TF Coordinate Success"
                );
                Console.WriteLine(
                    $"坐标系名称/Coordinate Name: {getCoord.Name}"
                );
            }
            else
            {
                Console.WriteLine(
                    $"获取TF坐标系失败/Get TF Coordinate Failed: {code.GetDescriptio
n()}"
                );
                return code;
            }

            // [ZH] 更新坐标系
            // [EN] Update coordinate system
```

```
            getCoord.Name = "test";
            StatusCode updateCode =
                controller.CoordinateSystem.Update(
                    CoordinateType.ToolCoordinate,
                    getCoord
                );
            if (updateCode == StatusCode.OK)
            {
                Console.WriteLine(
                    "更新TF坐标系成功/Update TF Coordinate Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"更新TF坐标系失败/Update TF Coordinate Failed: {updateCode.GetD
escription()}"
                );
                return updateCode;
            }

            // [ZH] 删除坐标系
            // [EN] Delete coordinate system
            deleteCode = controller.CoordinateSystem.Delete(
                CoordinateType.ToolCoordinate,
                calculatedCoord.Id
            );
            if (deleteCode == StatusCode.OK)
            {
                Console.WriteLine(
                    "删除TF坐标系成功/Delete TF Coordinate Success"
                );
            }
            else
            {
                Console.WriteLine(
                    $"删除TF坐标系失败/Delete TF Coordinate Failed: {deleteCode.GetD
escription()}"
                );
                return deleteCode;
            }
```

```
            Console.WriteLine(
                "TF坐标系测试完成/TF Coordinate Test Completed"
            );
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.12 Robot Jogging Motion

4.12.1 Robot Jogging Motion

Method Name	**Jogging.Move**(int `ajNum` , MoveMode `moveMode` , double `stepLength` = 0, double `stepAngle` = 0)
Description	Controls the robot to move continuously or by a specified increment.
Request Parameters	`ajNum` : int, value 1–6 corresponds to joint numbers [1–6], or x, y, z, rx, ry, rz in Cartesian space, depending on the currently selected coordinate system. Positive values indicate movement in the positive direction, negative values indicate movement in the negative direction. `moveMode` : MoveMode, motion mode of the manipulator; supports incremental or continuous motion. `stepLength` : double, step length in mm or degrees (only effective in incremental motion mode). `stepAngle` : double, step angle in degrees (only effective in incremental motion mode).
Return Value	StatusCode: Status code indicating whether the jogging operation succeeded.
Compatible Robot Software Versions	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Jogging/StepJogging.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class StepJogging
{
    public static StatusCode Run(
        string controllerIP,
```

```
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人模式
            // [EN] Get robot mode
            (UserOpMode opMode, StatusCode opCode) =
                controller.GetOpMode();
            if (opCode == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前机器人模式/Current robot mode: {opMode}"
                );
                if (
                    opMode != UserOpMode.UNLIMITED_MANUAL
                    && opMode != UserOpMode.LIMIT_MANUAL
                )
                {
                    Console.WriteLine(
                        $"示教运动必须在机器人手动模式下/Jogging must be in manual mo
```

```
de"
                    );
                    return StatusCode.OtherReason;
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取机器人模式失败/Failed to get robot mode: {opCode.GetDescri
ption()}"
                );
            }

            // [ZH] 设置单步示教运动参数
            // [EN] Set step jogging parameters
            int ajNum = 1; // 轴序号，正数表示正方向运动
            MoveMode moveMode = MoveMode.Stepping; // 单步运动模式
            double stepLength = 5.0; // 步长，单位为mm或角度
            double stepAngle = 5.0; // 轴旋转角度，单位为角度

            Console.WriteLine(
                "开始单步示教运动/Starting Step Jogging"
            );
            Console.WriteLine(
                $"轴序号/Axis Number: {ajNum}"
            );
            Console.WriteLine(
                $"运动模式/Move Mode: {moveMode}"
            );
            Console.WriteLine(
                $"步长/Step Length: {stepLength}"
            );

            // [ZH] 执行单步示教运动
            // [EN] Execute step jogging movement
            code = controller.Jogging.Move(
                ajNum,
                moveMode,
                stepLength,
                stepAngle
            );
            if (code == StatusCode.OK)
```

```
            {
                Console.WriteLine(
                    "单步示教运动执行成功/Step Jogging Executed Successfully"
                );
                Console.WriteLine(
                    $"轴{ajNum}向正方向移动{stepLength}单位/Axis {ajNum} moved {step
Length} units in positive direction"
                );
            }
            else
            {
                Console.WriteLine(
                    $"单步示教运动执行失败/Step Jogging Execution Failed: {code.GetD
escription()}"
                );
            }

            // [ZH] 等待一秒后执行反向运动
            // [EN] Wait one second then execute reverse movement
            Thread.Sleep(1000);

            // [ZH] 执行反向单步运动
            // [EN] Execute reverse step movement
            int reverseAjNum = -ajNum; // 负数表示负方向运动
            code = controller.Jogging.Move(
                reverseAjNum,
                moveMode,
                stepLength,
                stepAngle
            );
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "反向单步示教运动执行成功/Reverse Step Jogging Executed Successfu
lly"
                );
                Console.WriteLine(
                    $"轴{Math.Abs(reverseAjNum)}向负方向移动{stepLength}单位/Axis {M
ath.Abs(reverseAjNum)} moved {stepLength} units in negative direction"
                );
            }
            else
```

```
            {
                Console.WriteLine(
                    $"反向单步示教运动执行失败/Reverse Step Jogging Execution Failed:
{code.GetDescription()}"
                );
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.12.2 Multi-Axis Simultaneous Continuous Motion

Method Name	**Jogging.MultiMove**(int[] `ajNums`)
Description	Controls the robot to perform continuous motion on multiple axes simultaneously.
Request Parameters	`ajNums` : int[], values 1–6 correspond to joint numbers [1–6], or x, y, z, rx, ry, rz in Cartesian space, depending on the currently selected coordinate system. Positive values indicate movement in the positive direction, negative values indicate movement in the negative direction.
Return Value	StatusCode: Status code indicating whether the jogging operation succeeded.
Compatible Robot Software Versions	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Jogging/MultiJogging.cs

```cs
using Agilebot.IR;
using Agilebot.IR.Jogging;
using Agilebot.IR.Types;

public class MultiJogging
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
```

```
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人模式
            // [EN] Get robot mode
            (UserOpMode opMode, StatusCode opCode) =
                controller.GetOpMode();
            if (opCode == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前机器人模式/Current robot mode: {opMode}"
                );
                if (
                    opMode != UserOpMode.UNLIMITED_MANUAL
                    && opMode != UserOpMode.LIMIT_MANUAL
                )
                {
                    Console.WriteLine(
                        $"示教运动必须在机器人手动模式下/Jogging must be in manual mo
de"
                    );
                    return StatusCode.OtherReason;
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取机器人模式失败/Failed to get robot mode: {opCode.GetDescri
ption()}"
                );
            }

            Console.WriteLine(
```

```
            "开始多轴示教运动/Starting Multi-axis Jogging"
        );
        Console.WriteLine(
            "演示多轴运动/Demo multi-axis step movements"
        );

        // [ZH] 多轴运动
        // [EN] Multi-axis step movement
        Console.WriteLine(
            "\n=== 多轴运动/Multi-axis Step Movement ==="
        );
        int[] axes = { 1, 2, 3 }; // 正方向运动

        code = controller.Jogging.MultiMove(axes);
        if (code == StatusCode.OK)
        {
            Console.WriteLine(
                "连续示教运动启动成功/Continuous Jogging Started Successfully"
            );
            Console.WriteLine(
                "运动3秒后自动停止/Moving for 3 seconds then auto stop"
            );

            // [ZH] 运动3秒
            // [EN] Move for 3 seconds
            Thread.Sleep(3000);

            // [ZH] 停止示教运动
            // [EN] Stop jogging movement
            controller.Jogging.Stop();
            Console.WriteLine(
                "示教运动已停止/Jogging Movement Stopped"
            );
        }
        else
        {
            Console.WriteLine(
                $"连续示教运动启动失败/Continuous Jogging Start Failed: {code.Ge
tDescription()}"
            );
        }
```

```
            Console.WriteLine(
                "\n多轴示教运动完成/Multi-axis Jogging Completed"
            );
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.12.3 Stop Robot Jogging Motion

Method Name	**Jogging.Stop()**
Description	Stops the robot jogging motion.

Method Name	**Jogging.Stop**()
Request Parameters	None
Return Value	void
Notes	This method is only required to stop motion when in continuous mode.
Compatible Robot Software Versions	Collaborative (Copper): v7.5.0.0+ Industrial (Bronze): v7.5.0.0+

Example Code

Jogging/ContinuousJogging.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.Types;

public class ContinuousJogging
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );
```

```
        if (code != StatusCode.OK)
        {
            return code;
        }

        try
        {
            // [ZH] 获取机器人模式
            // [EN] Get robot mode
            (UserOpMode opMode, StatusCode opCode) =
                controller.GetOpMode();
            if (opCode == StatusCode.OK)
            {
                Console.WriteLine(
                    $"当前机器人模式/Current robot mode: {opMode}"
                );
                if (
                    opMode != UserOpMode.UNLIMITED_MANUAL
                    && opMode != UserOpMode.LIMIT_MANUAL
                )
                {
                    Console.WriteLine(
                        $"示教运动必须在机器人手动模式下/Jogging must be in manual mo
de"
                    );
                    return StatusCode.OtherReason;
                }
            }
            else
            {
                Console.WriteLine(
                    $"获取机器人模式失败/Failed to get robot mode: {opCode.GetDescri
ption()}"
                );
            }

            // [ZH] 设置示教运动参数
            // [EN] Set jogging parameters
            int ajNum = 3; // 轴序号，正数表示正方向运动
            MoveMode moveMode = MoveMode.Continuous; // 连续运动模式
```

```
            Console.WriteLine(
                "开始连续示教运动/Starting Continuous Jogging"
            );
            Console.WriteLine(
                $"轴序号/Axis Number: {ajNum}"
            );
            Console.WriteLine(
                $"运动模式/Move Mode: {moveMode}"
            );

            // [ZH] 启动连续示教运动
            // [EN] Start continuous jogging movement
            code = controller.Jogging.Move(ajNum, moveMode);
            if (code == StatusCode.OK)
            {
                Console.WriteLine(
                    "连续示教运动启动成功/Continuous Jogging Started Successfully"
                );
                Console.WriteLine(
                    "运动3秒后自动停止/Moving for 3 seconds then auto stop"
                );

                // [ZH] 运动3秒
                // [EN] Move for 3 seconds
                Thread.Sleep(3000);

                // [ZH] 停止示教运动
                // [EN] Stop jogging movement
                controller.Jogging.Stop();
                Console.WriteLine(
                    "示教运动已停止/Jogging Movement Stopped"
                );
            }
            else
            {
                Console.WriteLine(
                    $"连续示教运动启动失败/Continuous Jogging Start Failed: {code.Ge
tDescription()}"
                );
            }
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            code = StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }

        return code;
    }
}
```

4.13 Robot Subscription & Publish Interface

4.13.1 Connect to WebSocket Server

Method Name	**SubPub.Connect()**
Description	Connects to the robot controller WebSocket server
Request Parameters	None
Return Value	Task: Asynchronous connection operation result
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.13.2 Disconnect from WebSocket Server

Method Name	**SubPub.Disconnect()**
Description	Disconnects from the robot controller WebSocket server
Request Parameters	None
Return Value	Task: Asynchronous disconnect operation result
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.13.3 Subscribe to Robot Status

Method Name	**SubPub.SubscribeStatus**(RobotTopicType[] `topicTypes` , int `frequency` = 200)
Description	Adds robot status data subscription
Request Parameters	`topicTypes` : RobotTopicType[] List of robot topic types to subscribe `frequency` : int Subscription frequency in Hz, default is 200
Return Value	Task: Asynchronous subscription operation result
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.13.4 Subscribe to Registers

Method Name	**SubPub.SubscribeRegister**(RegTopicType `regType` , int[] `regIds` , int `frequency` = 200)
Description	Adds register data subscription
Request Parameters	`regType` : RegTopicType Register type `regIds` : int[] List of register IDs to subscribe `frequency` : int Subscription frequency in Hz, default is 200
Return Value	Task: Asynchronous subscription operation result
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.13.5 Subscribe to I/O Signals

Method Name	**SubPub.SubscribeIO**((IOTopicType, int)[] `ioList` , int `frequency` = 200)
Description	Subscribes to IO signal data, including digital inputs and outputs
Request Parameters	`ioList` : (IOTopicType, int)[] IO list, each element is `(IO type, IO ID)` `frequency` : int Subscription frequency in Hz, default is 200

Method Name	**SubPub.SubscribeIO**((IOTopicType, int)[] `ioList` , int `frequency` = 200)
Return Value	Task: Asynchronous subscription operation result
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

4.13.6 Start Receiving Messages

Method Name	**SubPub.StartReceiving**(Func<Dictionary<string, object>, Task> `onMessageReceived`)
Description	Starts receiving subscription messages and processes received data through callback function
Request Parameters	`onMessageReceived` : Func<Dictionary<string, object>, Task> Message receiving callback function
Return Value	Task: Asynchronous receiving task
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

Example Code

SubPub/CallbackReceiving.cs

```cs
using Agilebot.IR;
using Agilebot.IR.SubPub;
using Agilebot.IR.Types;

public class CallbackReceiving
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
```

```
// [ZH] 初始化捷勃特机器人
// [EN] Initialize the Agilebot robot
Arm controller = new Arm(
    controllerIP,
    useLocalProxy
);

// [ZH] 连接捷勃特机器人
// [EN] Connect to the Agilebot robot
StatusCode code = controller.ConnectSync();
Console.WriteLine(
    code != StatusCode.OK
        ? code.GetDescription()
        : "连接成功/Successfully connected."
);

// [ZH] 初始化捷勃特机器人SubPub
// [EN] Initialize the Agilebot robot SubPub
var subPub = controller.SubPub;

try
{
    Console.WriteLine(
        "开始回调方式接收消息测试/Starting Callback Receiving Test"
    );

    // [ZH] 连接到WebSocket服务器
    // [EN] Connect to WebSocket server
    subPub.Connect().Wait();
    Console.WriteLine(
        "WebSocket连接成功/WebSocket Connected Successfully"
    );

    // [ZH] 订阅机器人状态
    // [EN] Subscribe to robot status
    var topicTypes = new RobotTopicType[]
    {
        RobotTopicType.TopicCurrentJoint,
        RobotTopicType.TopicRobotStatus,
    };
    subPub
        .SubscribeStatus(topicTypes, frequency: 100)
```

```
    .Wait();
Console.WriteLine(
    "机器人状态订阅成功/Robot Status Subscription Successful"
);

// [ZH] 订阅寄存器
// [EN] Subscribe to registers
var regIds = new int[] { 1, 2, 3 };
subPub
    .SubscribeRegister(
        RegTopicType.R,
        regIds,
        frequency: 100
    )
    .Wait();
Console.WriteLine(
    "寄存器订阅成功/Register Subscription Successful"
);

// [ZH] 订阅IO
// [EN] Subscribe to IO
var ioList = new (IOTopicType, int)[]
{
    (IOTopicType.DI, 0),
    (IOTopicType.DO, 1),
};
subPub
    .SubscribeIO(ioList, frequency: 100)
    .Wait();
Console.WriteLine(
    "IO订阅成功/IO Subscription Successful"
);

int messageCount = 0;
int maxMessages = 10; // 接收10条消息后停止

Console.WriteLine(
    "开始接收消息/Starting to receive messages..."
);

// [ZH] 开始接收消息（回调方式）
// [EN] Start receiving messages (callback method)
```

```
            subPub
                .StartReceiving(async message =>
                {
                    messageCount++;
                    Console.WriteLine(
                        $"\n=== 收到第{messageCount}条消息/Received Message #{messag
eCount} ==="
                    );
                    foreach (var kv in message)
                    {
                        Console.WriteLine(
                            $"{kv.Key}: {kv.Value}"
                        );
                    }

                    // [ZH] 接收指定数量消息后主动断开
                    // [EN] Disconnect after receiving specified number of messages
                    if (messageCount >= maxMessages)
                    {
                        Console.WriteLine(
                            $"已接收{maxMessages}条消息，准备断开连接/Received {maxMe
ssages} messages, preparing to disconnect"
                        );
                        subPub.Disconnect().Wait();
                        Console.WriteLine(
                            "WebSocket断开成功/WebSocket Disconnected Successfully"
                        );
                    }

                    await Task.CompletedTask;
                })
                .Wait();

            Console.WriteLine(
                "回调方式接收消息测试完成/Callback Receiving Test Completed"
            );
            return StatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
```

```
e}"
            );
            return StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }
    }
}
```

4.13.7 Receive Next Text Message

Method Name	**SubPub.Receive**()
Description	Receives the next text message and returns it
Request Parameters	None
Return Value	Task<Dictionary<string, object>>: Received message dictionary
Compatible robot software version	Collaborative (Copper): v7.7.0.0+ Industrial (Bronze): v7.7.0.0+

Example Code

SubPub/PollingReceiving.cs

cs

```cs
using Agilebot.IR;
using Agilebot.IR.SubPub;
using Agilebot.IR.Types;

public class PollingReceiving
{
    public static StatusCode Run(
        string controllerIP,
        bool useLocalProxy = true
    )
    {
        // [ZH] 初始化捷勃特机器人
        // [EN] Initialize the Agilebot robot
        Arm controller = new Arm(
            controllerIP,
            useLocalProxy
        );

        // [ZH] 连接捷勃特机器人
        // [EN] Connect to the Agilebot robot
        StatusCode code = controller.ConnectSync();
        Console.WriteLine(
            code != StatusCode.OK
                ? code.GetDescription()
                : "连接成功/Successfully connected."
        );

        // [ZH] 初始化捷勃特机器人SubPub
        // [EN] Initialize the Agilebot robot SubPub
        var subPub = controller.SubPub;

        try
        {
            Console.WriteLine(
                "开始轮询方式接收消息测试/Starting Polling Receiving Test"
            );

            // [ZH] 连接到WebSocket服务器
            // [EN] Connect to WebSocket server
            subPub.Connect().Wait();
            Console.WriteLine(
```

```
    "WebSocket连接成功/WebSocket Connected Successfully"
);

// [ZH] 订阅机器人状态
// [EN] Subscribe to robot status
var topicTypes = new RobotTopicType[]
{
    RobotTopicType.TopicCurrentJoint,
    RobotTopicType.TopicRobotStatus,
};
subPub
    .SubscribeStatus(topicTypes, frequency: 100)
    .Wait();
Console.WriteLine(
    "机器人状态订阅成功/Robot Status Subscription Successful"
);

// [ZH] 订阅寄存器
// [EN] Subscribe to registers
var regIds = new int[] { 1, 2, 3 };
subPub
    .SubscribeRegister(
        RegTopicType.R,
        regIds,
        frequency: 100
    )
    .Wait();
Console.WriteLine(
    "寄存器订阅成功/Register Subscription Successful"
);

// [ZH] 订阅IO
// [EN] Subscribe to IO
var ioList = new (IOTopicType, int)[]
{
    (IOTopicType.DI, 0),
    (IOTopicType.DO, 1),
};
subPub
    .SubscribeIO(ioList, frequency: 100)
    .Wait();
Console.WriteLine(
```

```
            "IO订阅成功/IO Subscription Successful"
        );

        int messageCount = 0;
        int maxMessages = 10; // 接收10条消息后停止

        Console.WriteLine(
            "开始轮询接收消息/Starting to poll messages..."
        );

        // [ZH] 循环接收消息直到达到期望数量
        // [EN] Loop to receive messages until reaching desired count
        do
        {
            messageCount++;
            try
            {
                // [ZH] 接收单条消息
                // [EN] Receive single message
                var message = subPub.Receive().Result;
                Console.WriteLine(
                    $"\n=== 收到第{messageCount}条消息/Received Message #{messag
eCount} ==="
                );
                foreach (var kv in message)
                {
                    Console.WriteLine(
                        $"{kv.Key}: {kv.Value}"
                    );
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"接收消息时发生异常/Exception while receiving message: {ex.
Message}"
                );
                break;
            }
        } while (messageCount < maxMessages);

        // [ZH] 断开连接
```

```
            // [EN] Disconnect
            subPub.Disconnect().Wait();
            Console.WriteLine(
                "WebSocket断开成功/WebSocket Disconnected Successfully"
            );

            Console.WriteLine(
                "轮询方式接收消息测试完成/Polling Receiving Test Completed"
            );
            return StatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine(
                $"执行过程中发生异常/Exception occurred during execution: {ex.Messag
e}"
            );
            return StatusCode.OtherReason;
        }
        finally
        {
            // [ZH] 关闭连接
            // [EN] Close the connection
            StatusCode disconnectCode =
                controller.Disconnect();
            if (disconnectCode != StatusCode.OK)
            {
                Console.WriteLine(
                    disconnectCode.GetDescription()
                );
                if (code == StatusCode.OK)
                    code = disconnectCode;
            }
        }
    }
}
```

Agilebot C# SDK Update Notes

2.0.3.* Update (2025/12/17)

1. Arm constructor now includes teachPanelIP parameter.
2. Removed System.Text.Json dependency.
3. Added synchronous connection interface ConnectSync.

2.0.2.* Update (2025/12/12)

1. Fixed the incorrect read/write order of data in the PR register struct.

2.0.1.* Update (2025/10/21)

1. Fixed the stuttering issue during continuous jogging.
2. Fixed a build-time error where the proxy executable might fail to copy.

Update 2.0.0.* (2025-09-10)

1. Refactored the underlying request pattern and added a local controller proxy service.
2. Introduced subscription functionality.
3. Reorganized the BasScript class structure.
4. Full support for both .NET Framework and .NET (Core/5+/6+).

Version 1.0.1.0 Update (July 7, 2025)

1. Added the old register interface class `RegistersOld` to be compatible with robot versions prior to 7.6.0.0
2. Added the Estop emergency braking interface
3. Fixed the example programs in the documentation

Version 1.0.0.0 Update (May 30, 2025)

1. Implemented using RPC method.
2. Synchronized all interface definitions with the Python version.

Help

AI Coding Support

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

Preparation

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

- **SDK Documentation**: https://dev.sh-agilebot.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 robot's current position, coordinate system number, servo status, and other information.

Reference materials:
SDK Documentation: https://dev.sh-agilebot.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 robot's current position, coordinate system number, servo status, and other information.

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