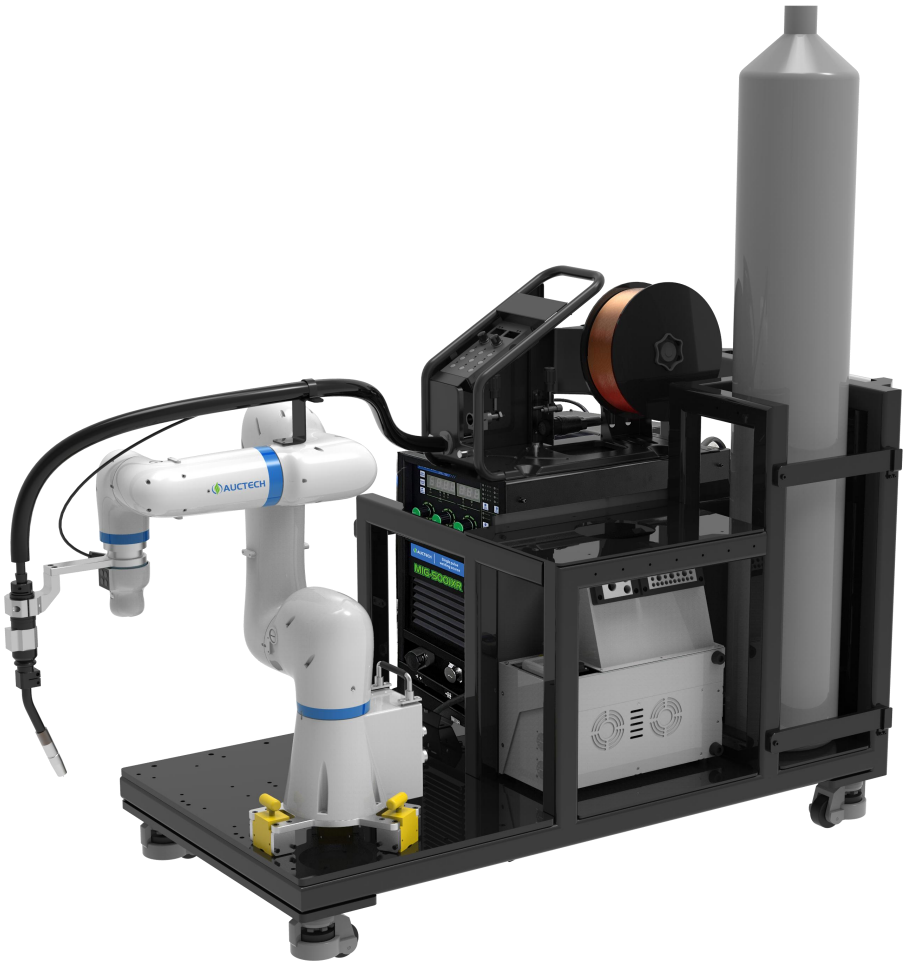
H Series Socket Cmd Communication Function User Manual



Guangzhou Auctech Automation Technology Ltd

Content

1 FEATURE DESCRIPTION

The Socket Cmd network terminal command is a standardized string-based control protocol agreed upon by robotic arm controllers. After establishing a Socket connection between the host computer and the robotic arm, sending the specified command string allows the robotic arm to perform designated actions or retrieve system status information.

When using the Socket Cmd network terminal command, the robotic arm acts as the server with the default communication port 23333. The IP address is the IP address of the robotic arm controller.

The IP address of the LAN0 network port on the IPC cabinet controller is as follows:

- 10.10.56.214

The integrated drive and control electrical cabinet has multiple network ports, each with the following IP address:

- MLAN: 10.10.57.213
- LAN2: 10.10.58.212
- LAN1: 10.10.59.211

To use Socket Cmd terminal commands, connect the host computer and robotic arm controller via a network cable (select the appropriate network port), then enable the Socket Cmd function in the teaching/working station software interface. The host computer acts as a client, connecting to the robotic arm controller's IP address and port number. By sending protocol strings, you can control the robotic arm.

2 PROTOCOL SPECIFICATION

2.1 Protocol Message Format

The data packet format sent by the host computer is as follows:

i: serial number	c: station comm- and	Break

The data message format returned by the robotic arm controller is as follows:

i: serial number	e: Error code	d: Command returns data content	Break

Data packet description:

(1) The data packet is encoded using UTF-8.

(2) The sequence number is of type uint, with a value range from 0 to 2147483648. Duplicate sequence numbers are generally not allowed in ongoing requests. However, if communication requests are processed sequentially within a single connection, duplicate values are permitted—for example, setting it to 0 or any other number. The sequence number of the sending message matches that of the returning message.

(3) For terminal commands, see Chapter 4: Terminal Command List.

(4) The error code is a 64-bit integer that indicates the result status of the request response.

- If the error code is 0, the request response is successful.
- If the error code is not 0, the request response fails.

(5) The separator is "@hs@". This is a fixed format.

Data packet example:

Terminal command description	Data packet requested by the host computer	Data message from the robotic arm
Enable on robotic arm	i:10000,c:mot.setGpEn(0,true)@hs@	i:10000,e:0,d:null@hs@
Get the robotic arm enabled status	i:10001,c:mot.getGpEn(0)@hs@	i:10001,e:0,d:true@hs@
Set the robotic arm load level to heavy	i:10002,c:mot.setRobLoad(0,”H”)@hs@	i:10002,e:0,d:null@hs@

2.2 Protocol Message Details

Take the enable function on the robotic arm in the above example:The host computer (client) sends: i: 10000, c: mot. setGpEn(0, true) @hs@

The service-side robotic arm returns: i: 10000, e: 0, d: null@hs@

(1) Packet sending details:

- i: 10000. The serial number can be customized based on actual needs, not necessarily starting from 10000. It can increment by 1 after each command, or remain unchanged.
- c: mot. setGpEn(0, true). Terminal command. In this command, 'mot' stands for motion module (other modules like IO are represented as 'io', register operations as 'var', etc.), while setGpEn() is the group enable configuration interface for the module. The parameters 0 and true control the interface: 0 sets the group number (default 0 for current robotic arm versions), and true enables the upper group, whereas false enables the lower group. For detailed interface specifications of other commands, refer to Chapter 4's Terminal Command List.
- @hs@. Delimiter. For fixed formatting, no changes needed.

(2) Return message details:

- i: 10000. Sequence number. Matches the sequence number of the sending message.
- e: 0. Error code. A value of 0 indicates a successful request response.
- d: null. The command returns data content. null indicates that the interface returns a void value or an empty string; both cases can be handled as void.
- @hs@. Delimiter. For fixed formatting, no changes needed.

2.3 Protocol Version

To switch protocols, send a specified command to set the protocol version. If no version is set, the latest protocol is used by default. The terminal command to set the protocol version is: sc. setProtoVer (version number).

Data packet example: i: 10000, c: sc. setProtoVer(1.0)@hs@

3 THE SERVER PUSHES MESSAGES TO THE CLIENT

The server pushes messages to the client using the data packet format described in Chapter 2.1 for the robotic arm response.

3.1 Alert Message Push

The client (host computer) continuously monitors the 23333 port on the server (robot arm) to receive alarm messages pushed by the robot arm. By default, the server sends alarm messages to all available connection channels. For optimal reception, it is recommended that the client maintain only one active connection channel with the server. The protocol specifies that messages containing a sequence number of -1 from the server are considered alarm messages, with the following format:

i:-1	e: Error code	d: alarm message content	Break

To simplify client parsing, alarm messages use standard JSON format.

The JSON data example for alarm messages is as follows:

{

"time":"2023-01-01 13:45:51.032" Time, format yyyy-MM-dd HH:mm:ss. SSS

,"errorCode":"302000c0230000" //Error code, a hexadecimal string

, "content": "[Warning] [Motion Layer] [0] [0] Failed to switch collaboration mode" //Alarm content

"errorLevel":3 //Alarm Level

}

The errorLevel alarm levels are defined as follows:

- 0: Unknown level
- 1: Representative Level INFO
- 2: Representative Level Note
- 3: Representative Level WARNING
- 4: Invalid rank

4 TERMINAL COMMAND LIST

In the robotic arm control system, the terminal commands for each module are described as follows:

Module name	Module Features
vm	Program Run Agent. Provides PRG program-related functions.
var	Register operation proxy. Provides register operations such as R, JR, LR, and area functions.
track	Conveyor tracking agent. Provides related operation functions for conveyor tracking.
sys	System function agent. Provides system alarm, registration authorization, and backup restoration operations.
servo	Servo operation proxy. Provides servo-related operation functions.
recover	Restore the operation agent. The provider name is used to obtain, set, and clear operation functions.
pdo	PDO operation handler. Provides PDO functions for retrieving and setting operations.
net	Network function proxy. Provides network connection, custom network, and other operations.
mot	Motion function agent. Provides operation functions such as axis group configuration and operation.
lib	Business function agent. Provides version information, shutdown, and restart operations.
io	Digital IO operation proxy. Provides functions for obtaining and setting digital IO ports.
home	Zero Point operation proxy. Provides zero point retrieval, setting, and calculation functions.
ext	External operation function agent. Provides external operation functions such as signals and programs.
dyn	Dynamical function agent. Provides operational functions such as load identification and collision detection.
collect	Collection action agent. Provides collection functionality to establish collection tasks for specified variables, enabling continuous data collection at designated intervals.
coder	Proxy for encoding and decoding functions. Provides encoder and decoder operation functions.
cali	Calibrate the operation proxy. Provides calibration functions for external tooling, collaborative groups, and external axes.

Algo	Algorithm function proxy. Provides joint coordinate to space coordinate operation function.
ain/aout	Simulate IO operation proxy. Provides simulated IO port functions such as access and configuration.

4.1 Run the Program on the VM

4.1.1 Set Execution Mode

Command format	vm.setMode(int mode)
Command	Set execution mode
Interface parameters	mode: mode. 0 indicates NORMAL (normal mode); 1 indicates OPT (optimized mode)
Returned value	null

4.1.2 Loading Program

Command format	vm.load(string path, string entry)
Command	Loading program
Interface parameters	path: Path. This is a reserved parameter. It is recommended to leave it empty. entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.3 Run the Executable

Command format	vm.start(string entry)
Command	Run the executable
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.4 Run the Program Step by Step

Command format	vm.step(string entry)
Command	Run the program step by step

Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.5 Programmatic Jump

Command format	vm.jump(string entry, int lineNum)
Command	Programmatic jump
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" lineNum: Line number
Returned value	null

4.1.6 Roll Back the Execution Procedure

Command format	vm.back(string entry)
Command	Roll back the execution proce- dure
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.7 Pause the Program

Command format	vm.pause(string entry)
Command	Pause the program
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.8 Stop Executing the Program

Command format	vm.stop(string entry)
Command	Stop executing the program
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"

Returned value	null

4.1.9 Uninstall

Command format	vm.unload(string entry)
Command	Uninstall
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.10 Edit the Program

Command format	vm.enterEdit(string entry)
Command	Enter the editing program. This command is valid only when in ST_READY or ST_PAUSED status.
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.11 Exit Editing Program

Command format	vm.exitEdit(string entry)
Command	Exit the editor. This command is valid only in the ST_WAITING state.
Interface Parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	null

4.1.12 Insert or Modify a Line of Program Content

Command format	vm.modifyProgLine(string entry, string progName, int lineNum, bool isInsert, string content)
Command	Insert or modify a line of program content. This command is valid only in the ST_WAITING state.
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" progName: program file name lineNum: Line number

	isInsert: Insert or modify the program. True inserts content below the line number; false replaces the content at the line number. content: a line of content
Returned value	❖ 0: Failed ❖ 1: indicates success

4.1.13 Delete Program Content

Command format	vm.deleteProgLine(string entry, string progName, int lineNum, int count)
Command	Delete program content. This command is valid only in the ST_WAITING state.
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" progName: program file name lineNum: Line number count: number of lines in the program
Returned value	null

4.1.14 Add or Modify Points

Command format	vm.modifyP(string entry, string progName, int indexP, string value)
Command	Add or modify a point. This command is valid only when in ST_WAITING status.
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" progName: program file name indexP: P-point index. If the corresponding index for P-point does not exist, a new P- point is added; if it already exists, the P-point is modified. value: point position. The key points are " {{1.0,2.0,3.0,4.0,5.0,6.0}}"; the spatial point position is " {1,2,0, {1.0,2.0,3.0,4.0,5.0,6.0}}", where the first three digits represent tool number 1, workpiece number 2, and the form bit 0
Returned value	❖ 0: Failed ❖ 1: indicates success

4.1.15 Delete Point

Command format	vm.deleteP(string entry, string progName, int indexP)
Command	Delete point. This command is valid only when in ST_WAITING state.

Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" progName: program file name indexP: P-point index. If the corresponding index for P-point does not exist, a new P- point is added; if it already exists, the P-point is modified.
Returned value	null

4.1.16 Get App Status

Command format	vm.progState(string entry)
Command	Get app status
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	❖ 1: Indicates ST_UNLOAD (not loaded) ❖ 2: Indicates ST_READY (Ready) ❖ 3: Indicates ST_RUNNING (Running) ❖ 4: Indicates ST_PAUSED (Pause) ❖ 7: Indicates ST_FAULT (Error) ❖ 8: Indicates ST_LOADING (Loading) ❖ 9: Indicates ST_WAITING (Waiting)

4.1.17 Is It Loaded?

Command format	vm.isLoaded(string entry)
Command	Is it loaded?
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	❖ 0: Not loaded ❖ 1: Loaded

4.1.18 Get the Current Row

Command format	vm.getCurLineNo(string entry)
Command	Get the current row
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"

Returned value	❖ 1 to n: indicates row numbers

4.1.19 Get the Name of the Current Running Program

Command format	vm.getCurProgName(string entry)
Command	Get the name of the current running program
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Return value	Program name

4.1.20 Get the List of Loaded Main Programs

Command format	vm.mainProgNames()
Command	Get the list of loaded main programs
Interface parameters	null
Returned value	Program list. Format: string queue in parentheses, separated by half-width commas.For example, "[ABC.PRG,DEF.PRG]"

4.1.21 Get App Information

Command format	vm.getProgInfo(string entry)
Command	Get app information
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG"
Returned value	Content For example, "ret=0, cur_prog=" *** ", line_no=0, back_prog=" *** ", line_of_back=0, state=1, uf_num=-1, ut_num=-1" Where: "ret" is the return value, "cur_prog" is the current program name, "line_no" is the line number, " back_prog" is the pointer to the previous program name, "line_of_back" is the pointer to the previous line number, "state" is the runtime state, "uf_num" is the runtime workpiece number, and "ut_num" is the runtime tool number

4.1.22 Get an Int Variable

Command format	vm.getVariableOfInt(string entry, string npName, string varName)

Command	Get the int type variable
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" npName: Variable space name. This is a reserved parameter and can be empty. varName: variable name. For example, "UFRAME_NUM", "CNT", "J_VEL", etc.
Returned value	Variate-value

4.1.23 Get a Double Variable

Command format	vm.getVariableOfDouble(string entry, string npName, string varName)
Command	Get a double variable
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" npName: Variable space name. This is a reserved parameter and can be empty. varName: variable name. For example, "UFRAME_NUM", "CNT", "J_VEL", etc.
Returned value	Variate-value
Command format	vm.setVariableOfDouble(string entry, string npName, string varName, double value)
Command	Set double type variable

4.1.24 Set Double Type Variable

Command format	vm.setVariableOfDouble(string entry, string npName, string varName, double value)
Command	Set double type variable
Interface parameters	entry: entry point. For example, "TEST.PRG" or "/X/TEST.PRG" npName: Variable space name. This is a reserved parameter and canbe empty varName: variable name. For example, "UFRAME_NUM", "CNT", "J_VEL", etc. value: variable value
Returned value	null

4.2 Register Operation Agent Var

4.2.1 Get Variable Value

Command format	var.getVarValue(string strVarFullName)
Command	Get variable value
Interface parameters	strVarFullName: The full name of the variable. For example, "group[0].tfnum"
Returned value	Price

4.2.2 Set Variable Values

Command format	var.setVarValue(string strVarFullName, string strValue)
Command	Set variable values
Interface parameters	strVarFullName: The full name of the variable. For example, "group[0].tfnum" strValue: Value
Return value	null

4.2.3 Set the Variable Value to Another Variable

Command format	var.setVarValueByVar(string strVarFullName, string strSrcVarFullName)
Command	Set the variable value to another variable
Interface parameters	strVarFullName: The full name of the variable. For example, "group[0].tfnum" strSrcVarFullName: The full name of another variable, such as "group[1].tfnum".
Returned value	null

4.2.4 Save Axis Data

Command format	var.saveAx(int8_t axId, string file)
Command	Save axis data
Interface parameters	axId: Axis number. The range is 0 to n-1. file:file name. para: parameter; home: origin-related parameter; limit: soft limit-related parameter

Returned value	null

4.2.5 Save All Axis Data

Command format	var.saveAx(string file)
Command	Save all axis data
Interface parameters	file:file name. para: parameter; home: origin-related parameter; limit: soft limit-related parameter
Returned value	null

4.2.6 Save Group Data

Command format	var.saveGp(int8_t gpId, string file)
Command	Save group data
Interface parameters	gpId: Group ID. The range is 0 to n-1. file:file name. para: parameter; JR: JR register value; LR: LR register value; UT: UT register value; UF: UF register value; area: area-related parameter; track: track-related parameter
Returned value	null

4.2.7 Save All Group Data

Command format	var.saveGp(string file)
Command	Save all group data
Interface parameters	file:file name. para: parameter; JR: JR register value; LR: LR register value; UT: UT register value; UF: UF register value; area: area-related parameter; track: track-related parameter
Returned value	null

4.2.8 Save Public Data

Command format	var.saveCommon(string file)
Command	Save public data

Interface parameters	file: file name. R represents the R register value; system represents the system parameter; mb represents the Modbus parameter.
Returned value	null

4.2.9 Set R Register

Command format	var.setR(int32_t index, double value)
Command	Set R register
Interface parameters	index: index. The range is from 0 to n-1. value: floating-point number
Returned value	null

4.2.10 Get R Register

Command format	var.getR(int32_t index)
Command	Get R register
Interface parameters	index: index. The range is from 0 to n-1.
Returned value	Floating-point value

4.2.11 Get Multiple R Registers

Command format	var.getR(int32_t index, int32_t len)
Command	Get multiple R registers
Interface parameters	index: starting index. The range is from 0 to n-1 len: quantity
Returned value	A list of floating-point values. Format: a string queue separated by half-width commas and enclosed in parentheses.For example: "[1.0,2.0]"

4.2.12 Set JR Register

Command format	var.setJR(int8_t gpId, int32_t index, string value)
Command	Set JR register

Interface parameters	gpId: Group ID. The range is 0 to n-1. index: index. The range is from 0 to n-1.value: data points. Format: a sequence of numbers separated by semicolons in curly braces. For example: " {{7,8}}"
Returned value	null

4.2.13 Get JR Register

Command format	var.getLR(int8_t gpId, int32_t index)
Command	Get JR register
Interface parameters	gpId: Group ID. The range is 0 to n-1.index: index. The range is from 0 to n-1.
Returned value	Data points. Format: a sequence of numbers separated by semicolons and enclosed in curly braces.For example: " {{7,8}}"

4.2.14 Get Multiple JR Registers

Command format	var.getJR(int8_t gpId, int32_t index, int32_t len)
Command	Get multiple JR registers
Interface parameters	gpId: Group ID. The range is 0 to n-1. index: starting index. The range is from 0 to n-1 len: quantity
Returned value	Data point list. Format: a sequence of joint coordinate data separated by half-width commas in parentheses, for example: " {{7,8}}"

4.2.15 Set the LR Register

Command format	var.setLR(int8_t gpId, int32_t index, string value)
Command	Set the LR register

Interface parameters	gpId: Group ID. The range is 0 to n-1. index: index. The range is from 0 to n-1.value: point data. For example, " {1,2,3, {7,8}}", where " 1,2,3" are tool number, workpiece number, and shape bit, and " {7,8}" is point data, a sequence of numbers separated by curly braces and semicolons.
Returned value	null

4.2.16 Get LR Register

Command format	var.getLR(int8_t gpId, int32_t index)
Command	Get LR register
Interface parameters	gpId: Group ID. The range is 0 to n-1.index: index. The range is from 0 to n-1.
Returned value	Data pointsFor example, the format {1,2,3, {7,8}} uses {1,2,3} for tool number, workpiece number, and shape bit, respectively, and {7,8} for point data, which is a sequence enclosed in curly braces and separated by semicolons.

4.2.17 Get Multiple LR Registers

Command format	var.getLR(int8_t gpId, int32_t index, int32_t len)
Command	Get multiple LR registers
Interface parameters	gpId: Group ID. The range is 0 to n-1.index: index. The range is from 0 to n-1. len: quantity
Returned value	Data point list. Format: a sequence of Cartesian coordinate data separated by half-width commas and enclosed in square brackets, for example: "[{1,2,3, {7,8}}], {1,2,3, {7,8}}]"

4.2.18 Configure the Region Origin

Command format	var.setAreaOrigin(int8_t gpId, int8_t iIndex, int8_t ufNum, string strOrigin)
Command	Configure the region origin

Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. ufNum: The origin relative to the part number. The range is-1 to 15. l strOrigin: XYZ data from the origin. Format: a sequence of numbers separated by semicolons in parentheses. For example: " {1.0,2.0,3.0}"
Returned value	null

4.2.19 Configure Area Shape

Command format	var.setAreaShape(int8_t gpId, int8_t iIndex, int8_t type, string strSize)
Command	Configure area shape
Interface Parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. l Type: Shape Type l strSize: Shape data. Format: a sequence of numbers separated by half- width commas and enclosed in curly braces. For example: " {1.0,2.0,3.0,4.0}"
Returned value	null

4.2.20 Configure the Scaling Value

Command format	var.setAreaOffset(int8_t gpId, int8_t iIndex, float offset)
Command	Configure the scaling value
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. offset: Scale value. Unit: mm
Returned value	null

4.2.21 Configure Region Type

Command format	var.setAreaType(int8_t gpId, int8_t iIndex, int8_t type)
Command	Configure region type

Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. type: region type
Returned value	null

4.2.22 Get Area Type

Command format	var.getAreaType(int8_t gpId, int8_t iIndex)
Command	Get area type
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1.
Returned value	Area type

4.2.23 Get Area Type

Command format	var.getAreaType(int8_t gpId)
Command	Get area type
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Return value	Data list. Format: a semi-colon-separated data queue.For example: " {1,2,...}"

4.2.24 Configure the Output Signal Inversion for the Region

Command format	var.setAreaOutRev(int8_t gpId, int8_t iIndex, bool en)
Command	Configure the output signal inversion for the region
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. en: Reverse
Returned value	null

4.2.25 Configure Shared Area Processing Mode

Command format	var.setShareAreaMode(int8_t gpId, int8_t iIndex, int8_t mode)
Command	Configure shared area processing mode
Interface Parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. mode: processing mode
Returned value	null

4.2.26 Enable Shared Area

Command format	var.setShareAreaEn(int8_t gpId, int8_t iIndex, bool en)
Command	Enable shared area
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1. en: Enable
Returned value	null

4.2.27 Enable Shared Area

Command format	var.getShareAreaEn(int8_t gpId, int8_t iIndex)
Command	Enable shared area
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1.
Returned value	❖ 0: Not enabled ❖ 1: Enabled

4.2.28 Enable Shared Area

Command format	var.getShareAreaEn(int8_t gpId)
Command	Enable shared area
Interface parameters	gpId: Group ID. The range is 0 to n-1.

Returned value	Data list. Format: a semi-colon-separated data queue.For example: " {true, false,...}"

4.2.29 Get Regional Output

Command format	var.getAreaOut(int8_t gpId, int8_t iIndex)
Command	Get regional output
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1.
Returned value	❖ 0: No output ❖ 1: Output completed

4.2.30 Get Regional Output

Command format	var.getAreaOut(int8_t gpId)
Command	Get regional output
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Data list. Format: a semi-colon-separated data queue.For example: " {true, false,...}"

4.2.31 Get Regional Configuration Information

Command format	var.getAreaConfig(int8_t gpId, int8_t iIndex)
Command	Get regional configuration information
Interface parameters	gpId: Group ID. The range is 0 to n-1. iIndex:Range index. The range is from 0 to n-1.
Returned value	Information For example: "ufnum=" 0 ", origin=" {1.0,2.0,3.0,} ",shape_type=" 0 ",size=" {1.0,2.0,3.0,4.0,} ",offset=" 1.0 ",type=" 0 ",out_rev=" false ",share_mode=" 0" "ufnum" is the part number, "origin" is the origin data, "shape_type" is the shape type, " size" is the shape size, "offset" is the scaling value, "type" is the area type, "out_rev" is the output signal inversion for the area, and "share_mode" is the shared area processing mode.

4.3 Track Conveyor Belt

4.3.1 Calibrate Conveyor

Command format	track.calibrateConveyer(int8_t gpId, int8_t trackId, int8_t type, string strData)
Command	Calibrate the conveyor belt. After successful calibration, the data will be set in memory but not saved to the file.
Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. Range from 0 to n-1 type: calibration method code strData: Calibration data. Format: A dot data queue separated by semicolons. For example: " {1,2,};{3,4,};{5,6,}"
Returned value	Calibration results For example, "ret=0x0, data={7,8,}" where "ret" is the calibration return value and "data" is the calibration result.

4.3.2 Get the Current Hardware Trigger Number

Command format	track.getHardTrig(int8_t gpId, int8_t trackId)
Command	Get the current hardware trigger number. This ensures that the visual and controller processes are the same hardware trigger.
Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. Range from 0 to n-1
Returned value	Hardware trigger number. The normal range is 1 to 255.

4.3.3 Configure Software Triggers

Command format	track.createSoftTrig(int8_t gpId, int8_t trackId)
Command	Configure software triggers. This ensures that the visual and controller processes are triggered by the same software.
Interface parameters	gpId: Group ID. The range is 0 to n-1.trackId: tracking number. The range is 0 to n-1.
Returned value	Software-generated number. The normal range is 1 to 255.

4.3.4 Update Software Triggers

Command format	track.updateSoftTrig(int8_t gpId, int8_t trackId, int32_t trigId)
Command	Update the software trigger. Call this command after createSoftTrig to reduce the impact of network communication latency on encoder values.
Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. The range is 0 to n-1. trigId: software trigger ID
Returned value	null

4.3.5 Add Target

Command format	track.addTarget(int8_t gpId, int8_t trackId, string data, int32_t type, int32_t trigId)
Command	Add target. Ensure the visual and controller handle the same hardware/software trigger with the correct trigger number.
Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. The range is 0 to n-1. data: Target location. Format: a sequence of numbers separated by half-width commas in curly braces. For example: " {1.0,2.0,3.0,4.0,5.0,6.0,}" corresponds to xyzabc.l Type: Target type trigId: Hardware trigger/software trigger number
Returned value	null

4.3.6 Add a Target With TP Point Offset Parameter

Command format	track.addTargetWithOffset(int8_t gpId, int8_t trackId, string data, string offset, int32_t type, int32_t trigId)
Command	Add a target with TP point offset parameters. Ensure the visual and controller processes the same hardware/software trigger with the correct trigger number.

Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. The range is 0 to n-1.data: Target location. Format: a sequence of numbers separated by half-width commas in curly braces. For example, " {1.0,2.0,3.0,4.0,5.0,6.0,}" corresponds to xyzabc.offset: The offset value for tracking target position data. Format: a sequence of numbers separated by semicolons in curly braces. For example: " {1.0,2.0,3.0,4.0,5.0,6.0,}" corresponds to the offset values of xyzabc.type: target type trigId: Hardware trigger/software trigger number
Returned value	null

4.3.7 Get Location

Command format	track.getPosition(int8_t gpId, int8_t trackId, int32_t trigId)
Command	Get the position. Given the correct software trigger number, the position is relative to the conveyor belt position of a software trigger record.
Interface Parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. The range is 0 to n-1. trigId: software trigger ID
Returned value	Position data in string format. Format: a sequence of numbers separated by half-width commasand enclosed in curly braces, for example: " {1.0,2.0,3.0,4.0,5.0,6.0,}" corresponding to xyzabc.

4.3.8 Get the Current Encoder Value

Command format	track.getCurrentEncoder(int8_t gpId, int8_t trackId)
Command	Get the current encoder value
Interface parameters	gpId: Group ID. The range is 0 to n-1. trackId: tracking number. The range is 0 to n-1.
Returned value	Encoder value

4.4 System Function Agent Sys

4.4.1 System Reset

Command format	sys.reset()
Command	System reset

Interface parameters	null
Returned value	null

4.4.2 Is There an Alarm in the System?

Command format	sys.hasError()
Command	Is there an alarm in the system?
Interface parameters	null
Returned value	Alert count

4.4.3 Query Alarm Details

Command format	sys.queryError(uint64_t errCode)
Command	Query alarm details
Interface parameters	errCode: error code
Returned value	Alert details For example: "reason=xxx,elim=yyy", where "reason" is the reason and "elim" is the exclusion measure

4.4.4 Is the System Ready?

Command format	sys.isReady()
Command	Is the system ready?
Interface parameters	null
Returned value	❖ 0: Not ready ❖ 1: Ready

4.4.5 Upgrade

Command format	sys.prepareUpgrade()
Command	Upgrade is required. Record device model, load, and other information for parameter adaptation after upgrade.

Interface parameters	null
Returned value	null

4.4.6 Preparing to Update User PLC

Command format	sys.prepareUserPlc()
Command	Preparing to update user PLC
Interface parameters	null
Returned value	null

4.4.7 Register

Command format	sys.register(string strKey, string strValue)
Command	Register
Interface parameters	strKey: key strValue: Value
Returned value	null

4.4.8 Use SN to Obtain Authorization

Command format	sys.genSn()
Command	Use SN to obtain authorization
Interface parameters	null
Returned value	sn a sign or object indicating number

4.4.9 empower

Command format	sys.authorize(string strCode)
Command	Empower

Interface parameters	strCode: Code
Returned value	null

4.4.10 Remaining Authorization Time

Command format	sys.getRemainTime()
Command	Remaining authorization time
Interface parameters	null
Returned value	Time. Unit: seconds.

4.4.11 Allow

Command format	sys.isAuthorized()
Command	Allow
Interface parameters	null
Returned value	❖ 0: Unauthorized ❖ 1: Authorized

4.4.12 Set Language

Command format	sys.setLang(string strLang)
Command	Set language
Interface parameters	strLang: Language name. "ch" means Chinese; "en" means English
Returned value	❖ 0: Failed to set ❖ 1: Configuration succeeded

4.4.13 Back-Up

Command format	sys.backupFiles(string strIntent)
Command	Back-up

Interface parameters	strIntent: Intent. "MANU_CALIB_DATA" indicates the factory calibration parameters.
Returned value	Name of the backup package file

4.4.14 Restore Backup

Command format	sys.recoverFiles(string strIntent, string strFileName)
Command	Restore backup
Interface parameters	strIntent: Intent. "MANU_CALIB_DATA" indicates the factory calibration parameters. strFileName: The name of the backup file
Returned value	null

4.5 Servo Proxy

4.5.1 Get the Servo Parameter Names for the Specified Axis

Command format	servo.getServoParaNameList(int8_t gpId, int8_t axIndex)
Command	Get the servo parameter names for the specified axis
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1.
Return value	Name list. Format: Ascending semi-ASCII comma-separated data queue.For example: ["P005", "P006", "P007"]

4.5.2 Get the Formatting Information for the Specified Axis and Servo Parameters

Command format	servo.getServoParaInfoFormat(int8_t gpId, int8_t axIndex, string paraName)
Command	Get the formatting information for the specified axis and servo parameters
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1. paraName:parameter name

Returned value	Information For example: "name=" P005 ", chName=" parameter ", unitName=" mm ", minValue=1.0, maxValue=1000.0" Where "name" is the parameter's English name, "chName" is the parameter's Chinese name, "unitName " is the unit name, "minValue" is the minimum value, and "maxValue" is the maximum value

4.5.3 Print Servo Parameter Information

Command format	servo.printServoParaInfo(int8_t gpId, int8_t axIndex)
Command	Print servo parameter information
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1.
Returned value	List the formatting information for each servo parameter corresponding to the specified axis.

4.5.4 Read the Parameters of the Servo Corresponding to the Specified Axis

Command format	servo.readServoPara(int8_t gpId, int8_t axIndex)
Command	Read the parameters of the servo corresponding to the specified axis
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1.
Returned value	Count values in a column. Format: data queue separated by half-width commas.For example, "[1.0, 1000.0,NAN]" corresponds to parameters sorted by name in ascending order.

4.5.5 Read the Specified Servo Parameters for the Designated Axis

Command format	servo.readServoPara(int8_t gpId, int8_t axIndex, string paraName)
Command	Read the specified servo parameters for the designated axis
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1. paraName:parameter name
Returned value	Price

4.5.6 Write the Specified Servo Parameters for the Designated Axis

Command format	servo.writeServoPara(int8_t gpId, int8_t axIndex, string paraName, double value)
Command	Write the specified servo parameters for the designated axis
Interface parameters	gpId: Group ID. The range is 0 to n-1.l axIndex: Axis index within the group. The range is 0 to n-1. paraName:parameter name value: price
Returned value	null

4.6 Resume Recovery

4.6.1 Set the Recovery Program Name and Save

Command format	recover.setAndSaveRecoverProg(string entry)
Command	Set the recovery program name and save
Interface parameters	entry: enter the mouth
Returned value	null

4.6.2 Clear and Restore the Program Name and Save

Command format	recover.clearAndSaveRecoverProg()
Command	Clear and restore the program name and save
Interface parameters	null
Returned value	null

4.6.3 Get Recovery Program Name

Command format	recover.getRecoverProg()
Command	Get recovery program name
Interface parameters	null
Returned value	Resume program name

4.7 PDO Proxy PDO

4.7.1 Set up PDO

Command format	pdo.setPDO(int id, int padr, int sadr, double data)
Command	Set up PDO
Interface parameters	ID: Axis number padr: primary addressl SADR: secondary address data: PDO value
Returned value	null

4.7.2 Get PDO

Command format	pdo.getPDO(int id, int padr, int sadr)
Command	Gain PDO
Interface Parameters	ID: Axis number padr: primary addressl SADR: secondary address
Returned value	PDO price

4.8 Network Proxy

4.8.1 Get the Network Connection ID List

Command format	net.getSessionIdList()
Command	Get the network connection ID list
Interface parameters	null
Returned value	Network connection ID list. Format: a queue of integers separated by half-width commas in parentheses.For example: "[1,2]"

4.8.2 End Network Connection

Command format	net.killSession(int64_t iSessionId)
Command	End network connection
Interface parameters	iSessionId: Network connection ID
Returned value	null

4.8.3 Get Connection Information

Command format	net.getSessionInfo(int64_t iSessionId)
Command	Get connection informat- ion
Interface parameters	iSessionId: Network connection ID
Returned value	Information For example: "type=" TCP ", ip=" /123.234.34.45 ", port=" 12345"Where "type" is the type, "ip" is the IP address, and "port" is the port number

4.8.4 Enable Custom Network

Command format	net.enableCustomedNetwork(int8_t index, string strIp, string strMask)
Command	Enable custom network. Restart to apply.
Interface parameters	index: index number. The range is 0 to 1. strip: IP address strMask: subnet mask
Returned value	null

4.8.5 Disable Custom Network

Command format	net.disableCustomedNetwork(int8_t index)
Command	Disable custom network. Restart to take eff- ect.
Interface parameters	index: index number. The range is 0 to 1.
Returned value	null

4.8.6 Get Custom Network Configuration Information

Command format	net.getCustomedNetworkConfigInfo()
Command	Get custom network configuration information
Interface parameters	null
Returned value	JSON format configuration informationFor example: {"configList": [{"en": true, "ip": " 192.168.0.2", "mask": "255.255.255.0"}...]} where "configList" is the configuration list, "en" indicates whether the custom network is enabled (false means disabled, true means enabled), "ip" is the IP address, and "mask" is the subnet mask.

4.8.7 Save Custom Network Configuration

Command format	net.saveCustomedNetworkConfig()
Command	Save custom network configuration
Interface parameters	null
Returned value	null

4.9 Motion Function Agent

4.9.1 Get the Motion Layer Version Information

Command format	mot.getMotionVer()
Command	Get the motion layer version information
Interface parameters	null
Returned value	Version informa- tion

4.9.2 Print Control

Command format	mot.setDebugLog(int32_t mode, bool flag)
Command	Print control
Interface parameters	mode: pattern flag: Set
Returned value	null

4.9.3 Get Print Control Settings

Command format	mot.getDebugLog(int32_t mode)
Command	Get print control settings
Interface parameters	mode: pattern
Returned value	❖ 0 indicates no setting ❖ 1 indicates set

4.9.4 Set Operation Mode

Command format	mot.setOpMode(int8_t mode)
Command	Set operation mode

Interface parameters	mode: mode. 1 indicates manual T1 mode; 2 indicates manual T2 mode; 3 indicates automatic operation mode; 4 indicates external operation mode
Returned value	null

4.9.5 Get Operation Mode

Command format	mot.getOpMode()
Command	Get operation mode
Interface parameters	null
Returned value	Mode. 1 indicates manual T1 mode; 2 indicates manual T2 mode; 3 indicates automatic operation mode; 4 indicates external operation mode

4.9.6 Set Manual Motion Type

Command format	mot.setManualMode(int8_t mode)
Command	Set manual motion type
Interface parameters	mode: type. 0 indicates continuous motion; 1 indicates intermittent motion.
Returned value	null

4.9.7 Get Manual Motion Type

Command format	mot.getManualMode()
Command	Get manual motion type
Interface parameters	null
Returned value	Type. 0 indicates continuous motion; 1 indicates intermittent motion.

4.9.8 Set Manual Increment Distance

Command format	mot.setInchLen(double mode)

Command	Set manual increment distance
Interface parameters	len: distance. The unit depends on the coordinate system: degrees for joints and millimeters for Cartesian.
Returned value	null

4.9.9 Get Manual Incremental Distance

Command format	mot.getInchLen()
Command	Get manual incremental dista- nce
Interface parameters	null
Returned value	Distance size. Units are determined by the coordinate system: degrees for joints and millimeters for Cartesian.

4.9.10 Get Manual Run Status

Command format	mot.getManualStat()
Command	Get manual run status
Interface parameters	null
Returned value	Status. 0 means stop; 1 means accelerate; 2 means constant speed; 3 means decelerate

4.9.11 Set Auto Zoom

Command format	mot.setVord(int32_t vord)
Command	Set auto zoom
Interface parameters	term: multiplier. Range from 1 to 100, in percentage.
Returned value	null

4.9.12 Get Auto Zoom

Command format	mot.getVord()

Command	Get auto zoom
Interface parameters	null
Returned value	Factor. Range from 1 to 100, in percentage.

4.9.13 Set Manual Zoom

Command format	mot.setJogVord(int32_t vord)
Command	Set manual zoom
Interface parameters	term: multiplier. Range from 1 to 100, in percentage.
Returned value	null

4.9.14 Get Manual Speed

Command format	mot.getJogVord()
Command	Get manual speed
Interface parameters	null
Returned value	Factor. Range from 1 to 100, in percentage.

4.9.15 Jerk

Command format	mot.setEstop(bool en)
Command	Jerk
Interface parameters	en: Enable. true means on; false means off
Returned value	null

4.9.16 Get Emergency Stop Status

Command format	mot.getEstop()

Command	Get emergency stop status
Interface parameters	null
Returned value	❖ 0: indicates closed ❖ 1: indicates open

4.9.17 Enable Group

Command format	mot.setGpEn(int8_t gpId, bool en)
Command	Enable group
Interface parameters	gpId: Group ID. The range is 0 to n-1. en: Enable. true means on; false means off
Returned value	null

4.9.18 Get Group Enable Status

Command format	mot.getGpEn(int8_t gpId)
Command	Get group enable status
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	❖ 0: indicates the closed state ❖ 1: indicates the open state

4.9.19 Group Drag-and-Teach

Command format	mot.setGpDrag(int8_t gpId, bool en)
Command	Group Drag-and-Teach
Interface parameters	gpId: Group ID. The range is 0 to n-1. en: Enable. true means on; false means off
Returned value	null

4.9.20 Get Group Drag-and-Drop Teaching Status

Command format	mot.getGpDrag(int8_t gpId)
Command	Get group drag-and-drop teaching status
Interface Parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	❖ 0: indicates the closed state ❖ 1: indicates the open state

4.9.21 Reset Group

Command format	mot.gpReset(int8_t gpId)
Command	Reset group
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	null

4.9.22 Set the Zero Line

Command format	mot.setUseSimulate(int8_t gpId, bool en)
Command	Set the zero line
Interface parameters	gpId: Group ID. The range is 0 to n-1. en: switch
Returned value	null

4.9.23 Zero-Point Calibration

Command format	mot.setHomePosition(int8_t gpId, string strPos, int32_t mask)
Command	Zero-point calibration
Interface parameters	gpId: Group ID. The range is 0 to n-1.

	strPos: Zero point setting value. For example: {0, -90,0, 0, 90,0} mask: Enable mask. The lower 9 bits are valid, with the least significant bit representing the first axis in the group and the 9th bit representing the 9th axis in the group.
Returned value	null

4.9.24 Get Zero

Command format	mot.getHomePosition(int8_t gpId)
Command	Get Zero
Interface Parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.25 Set Soft Limit

Command format	mot.setJointLimit(int8_t gpId, string strLimitPos, string strLimitNeg, int32_t mask)
Command	Set soft limit
Interface parameters	gpId: Group ID. The range is 0 to n-1. strLimitPos: Forward setting value. For example, {90,90,90,90,90,90} strLimitNeg: A negative threshold value. For example, {-90, -90, -90, -90, -90, -90} mask: Enable mask. The lower 9 bits are valid, with the least significant bit representing the first axis in the group and the 9th bit representing the 9th axis in the group.
Returned value	null

4.9.26 Get Soft Limit

Command format	mot.getJointLimit(int8_t gpId)
Command	Get soft limit
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Location information For example: "pos={1, 1,}, neg={-1,-1,}, mask=0x1ff"Here, "pos" is the positive setpoint, "neg" is the negative setpoint, and "mask" is the enable mask.

4.9.27 Set Current Limit

Command format	mot.setELimit(int8_t gpId, string strEPos, string strENeg, int32_t mask)
Command	Set current limit
Interface parameters	gpId: Group ID. The range is 0 to n-1. strLimitPos: Forward setting value. For example, {90,90,90,90,90,90} strLimitNeg: A negative threshold value. For example, {-90, -90, -90, -90, -90, -90} mask: Enable mask. The lower 9 bits are valid, with the least significant bit representing the first axis in the group and the 9th bit representing the 9th axis in the group.
Returned value	null

4.9.28 Get Current Limit

Command format	mot.getELimit(int8_t gpId)
Command	Get current limit
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Location information For example: "pos={1, 1,}, neg={-1,-1,}, mask=0x1ff"Here, "pos" is the positive setpoint, "neg" is the negative setpoint, and "mask" is the enable mask.

4.9.29 Single-Axis Movement Started

Command format	mot.startJog(int8_t gpId, int8_t axId, int8_t direc)
Command	Single-axis movement star- ted
Interface parameters	gpId: Group ID. The range is 0 to n-1.axId: Axis number. 0 to 5 are internal axes; 6 to 8 are additional axes. dirc: direction. 0 indicates forward; 1 indicates backward.
Returned value	null

4.9.30 Single-Axis Movement Stopped

Command format	mot.stopJog(int8_t gpId)
Command	Single-axis movement stopped
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	null

4.9.31 Set Work Coordinate System

Command format	mot.setWorkFrame(int8_t gpId, int8_t frame)
Command	Set work coordinate system
Interface parameters	gpId: Group ID. The range is 0 to n-1. frame: Work coordinate system. 0 represents the axis coordinate system; 1 represents the world coordinate system; 2 represents the user coordinate system; 3 represents the tool coordinate system.
Returned value	null

4.9.32 Get Work Coordinate System

Command format	mot.getWorkFrame(int8_t gpId)
Command	Get work coordinate system
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	❖ 0: represents the axis coordinate system ❖ 1: World coordinate system ❖ 2: User coordinate system❖ 3: Tool coordinate system

4.9.33 Set Device

Command format	mot.setRobType(int8_t gpId, string strTypeName, string strLoadType)
Command	Set the device model. Restart the system to apply the change.

Interface parameters	gpId: Group ID. The range is 0 to n-1. strTypeName: Device namel strLoadType: Load name. "H" indicates heavy load; "M" indicates medium load; "L" indicates light load.
Returned value	null

4.9.34 Set Load

Command format	mot.setRobLoad(int8_t gpId, string strLoadType)
Command	Set the load. Restart the system to apply
Interface parameters	gpId: Group ID. The range is 0 to n-1.l strLoadType: Load name. "H" indicates heavy load; "M" indicates medium load; "L" indicates light load.
Returned value	null

4.9.35 Get Robot Type

Command format	mot.getRobType(int8_t gpId)
Command	Get robot type
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Robot type

4.9.36 Get the Robot Type Name

Command format	mot.getRobTypeName(int8_t gpId)
Command	Get the robot type name
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Robot type name

4.9.37 Get Robot Load

Command format	mot.getRobLoad(int8_t gpId)
Command	Get robot load

Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	❖ "H": heavy load ❖ "M": Chinese❖ "L": underloading

4.9.38 Get Available Model Names

Command format	mot.getRobTypeNameList()
Command	Get available model names
Interface parameters	null
Returned value	List of model names. Format: data queue separated by half-width commas.For example: ["PUMA", "SCARA"]

4.9.39 Get Internal Axis Numbers

Command format	mot.getRobAxisCount(int8_t gpId)
Command	Get internal axis numb- ers
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Number of axles

4.9.40 Add Extra Axis

Command format	mot.setAuxAxisCount(int8_t gpId, int32_t cnt)
Command	Add extra axis
Interface parameters	gpId: Group ID. The range is 0 to n-1. cnt: axis number. Range: 0 to 3
Returned value	null

4.9.41 Get Additional Axis Numbers

Command format	mot.getAuxAxisCount(int8_t gpId)
Command	Get additional axis numb- ers

Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Number of axles

4.9.42 Get the Starting Axis Number of the Additional Axis

Command format	mot.getAuxAxisBegin(int8_t gpId)
Command	Get the starting axis number of the additional axis
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Return value	Starting axis number

4.9.43 Configure Collaboration Group

Command format	mot.setRobCoordConfig(int8_t gpId, int8_t coordId, string axisId, string type)
Command	Configure collaboration group
Interface parameters	gpId: Group ID. The range is 0 to n-1. coordId: Collaboration group number. The range is 0 to n-1. axisId: Axis number. Format: a sequence of numbers separated by half-width commas in curly braces, for example: " {6,7,8,}". Note: The number must be 3.l Type: Axis usage. Format: a sequence of numbers separated by half-width commas in curly braces, for example, " {1, 1, 1,}". Note: The sequence must contain 3 data points.
Returned value	❖ 0: Configuration failed ❖ 1: Configuration succeeded

4.9.44 Get Collaborative Group Configuration

Command format	mot.getRobCoordConfig(int8_t gpId, int8_t coordId)
Command	Get collaborative group config-uration
Interface parameters	gpId: Group ID. The range is 0 to n-1.coordId: Collaboration group number. The range is 0 to n-1.

Returned value	Collaboration Group Configuration For example: "axisId={6,7,8,}, type={1, 1, 1,}" AxisId is the axis number, and type is the axis purpose

4.9.45 Get the Mask of the Current Collaborative Motion Axes for the Axis Group

Command format	mot.getCoordAxisMask(int8_t gpId)
Command	Get the mask of the current collaborative motion axes for the axis group
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	A mask for co-ordinated motion axes. The lower 9 bits are valid, with the least significant bit representing the first axis in the group and the 9th bit representing the 9th axis in the group. Axes with a value of 1 are co-ordinated.

4.9.46 Get Joint Coordinate Data

Command format	mot.getJntData(int8_t gpId)
Command	Get joint coordinate data
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Return value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.47 Get Additional Axis Coordinate Data

Command format	mot.getAuxData(int8_t gpId)
Command	Get additional axis coordinate data
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.48 Get Joint Feedback Current Data

Command format	mot.getJntEData(int8_t gpId)

Command	Get joint feedback current data
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.49 Get Additional Shaft Feedback Current Data

Command format	mot.getAuxEData(int8_t gpId)
Command	Get additional shaft feedback current data
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Return value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.50 Get Cartesian Coordinate Data

Command format	mot.getLocData(int8_t gpId)
Command	Get Cartesian coordinate data
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.51 Get Morphology

Command format	mot.getConfig(int8_t gpId)
Command	Get Morphology
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Morphology

4.9.52 Check if the Checkpoint Is Reachable

Command format	mot.isReachable(int8_t gpId, bool isJoint, int8_t ufNum, int8_t utNum, int32_t config, string strPos, int8_t type)

Command	Check if the checkpoint is reachable
Interface parameters	gpId: Group ID. The range is 0 to n-1. isJoint: Is it a joint? ufNum: Workpiece number. Range: -1 to 15. utNum: Tool number. Range: -1 to 15 config: morphic position strPos: Point data. Format: a sequence of numbers separated by half- width commas in curly braces, for example: " {1.0,2.0,3.0,4.0,5.0,6.0,}" type: type
Returned value	null

4.9.53 Motion Position

Command format	mot.moveTo(int8_t gpId, bool isJoint, int8_t ufNum, int8_t utNum, int32_t config, string strPos, bool isLinear)
Command	Motion Position
Interface parameters	gpId: Group ID. The range is 0 to n-1. isJoint: Is it a joint? ufNum: Workpiece number. Range: -1 to 15. utNum: Tool number. Range: -1 to 15 config: morphic position strPos: Point data. Format: a sequence of numbers separated by half- width commas in curly braces, for example: " {1.0,2.0,3.0,4.0,5.0,6.0,}" isLinear: Is linear motion
Returned value	null

4.9.54 Circular Motion Positioning

Command format	mot.moveRad(int8_t gpId, string strMidPos, string strEndPos, bool isToMid)
Command	Circular motion positioning
Interface parameters	gpId: Group ID. The range is 0 to n-1. strMidPos: Midpoint. Format: Joint point as " {{1.0,2.0,3.0,4.0,5.0,6.0,}}"; spatial point as " {1,2,0, {1.0,2.0,3.0,4.0,5.0,6.0,}}". strEndPos: End point isToMid: Whether to reach the midpoint. True means moving to the midpoint; false means moving to the endpoint.

Returned value	null

4.9.55 Set the Workpiece Coordinate System

Command format	mot.setWorkpiece(int8_t gpId, int8_t index, string data)
Command	Set the workpiece coordinate system
Interface parameters	gpId: Group ID. The range is 0 to n-1. index: part number. Range from 0 to 15 data: point data. Format: a sequence of numbers separated by half- width commas in curly braces, for example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"
Returned value	null

4.9.56 Get Part Coordinates

Command format	mot.getWorkpiece(int8_t gpId, int8_t index)
Command	Get part coordinates
Interface parameters	gpId: Group ID. The range is 0 to n-1. index: part number. Range from 0 to 15
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.57 Set Tool Coordinate System

Command format	mot.setTool(int8_t gpId, int8_t index, string data)
Command	Set tool coordinate system
Interface parameters	gpId: Group ID. The range is 0 to n-1. index: part number. Range from 0 to 15 data: point data. Format: a sequence of numbers separated by half- width commas in curly braces, for example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"
Returned value	null

4.9.58 Get Tool Coordinate System

Command format	mot.getTool(int8_t gpId, int8_t index)
Command	Get tool coordinate system
Interface parameters	gpId: Group ID. The range is 0 to n-1. index: part number. Range from 0 to 15
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.9.59 Set up the Installation Tool

Command format	mot.setInstallToolType(int8_t gpId, int8_t type)
Command	Set up the installation tool
Interface parameters	gpId: Group ID. The range is 0 to n-1. type: method type. 0 indicates a flange tool; 1 indicates an external tool.
Returned value	null

4.9.60 Get the Installation Tool

Command format	mot.getInstallToolType(int8_t gpId)
Command	Get the installation tool
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	❖ 0: indicates flange too l❖ 1: indicates external tools

4.9.61 Set Collaboration Group Number

Command format	mot.setCoordNum(int8_t gpId, int8_t num)
Command	Set collaboration group number
Interface parameters	gpId: Group ID. The range is 0 to n-1.num: Collaboration group number. Range: -1 to 2
Returned value	null

4.9.62 Get Collaborative Group Number

Command format	mot.getCoordNum(int8_t gpId)
Command	Get collaborative group number
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Collaboration Group Number

4.9.63 Set Part Number

Command format	mot.setWorkpieceNum(int8_t gpId, int8_t num)
Command	Set part number
Interface parameters	gpId: Group ID. The range is 0 to n-1.num: Workpiece number. Range: -1 to 15
Returned value	null

4.9.64 Get Part Number

Command format	mot.getWorkpieceNum(int8_t gpId)
Command	Get part number
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Workpiece ID

4.9.65 Set Tool Number

Command format	mot.setToolNum(int8_t gpId, int8_t num)
Command	Set tool number
Interface parameters	gpId: Group ID. The range is 0 to n-1. num: Tool number. Range: -1 to 15
Returned value	null

4.9.66 Get Tool ID

Command format	mot.getToolNum(int8_t gpId)
Command	Get tool ID
Interface parameters	gpId: Group ID. The range is 0 to n-1.
Returned value	Tool ID

4.9.67 Demarcate

Command format	mot.gpCalibrate(string what, int8_t gpId, int8_t num, int8_t type, string strData)
Command	Calibration. Data becomes effective after successful calibration, but it is not saved.
Interface parameters	what: The object to be calibrated. "BASE" represents the workpiece; "TOOL" represents the tool; "EXT_TOOL" represents the external tool; "EXT_BASE" represents the external workpiece; "COORD" represents the collaborative group.gpId: Group ID. The range is 0 to n-1. num: Workpiece number or tool number. Range: -1 to 15. type: calibration method code. 6 indicates the 6-point tool method; 5 indicates the 5-point collaborative group method; 4 indicates the 4-point tool method; 3 indicates the 3-point workpiece method, 3-point collaborative group method, 3-point external tool method, or 3-point external workpiece method; 2 indicates the 2-point tool method. strData: Calibration data. Format: A dot data queue separated by semicolons, for example: " {1,2,};{3,4,};{5,6,}."
Returned value	Calibration results For example: "ret=0x0, data={7,8,}""ret" is the calibration return value, and "data" is the calibration result

4.9.68 Calibrate the Additional Shaft Reduction Ratio

Command format	mot.calibrateAuxAxisGearRatio(string strJsonInput)
Command	Calibrate the additional shaft reduction ratio. The data becomes effective after calibration, but it will not be saved.

Interface parameters	strJSONInput: JSON-formatted calibration input. Format: {"gpId": 0, "auxAxId": 6, " auxAxType": "L", "sampleList": [{"xyz": [123.0,...], "enc": 123.0,...]}} where "gpId" is the group number (0 to n-1); "auxAxId" is the auxiliary axis number (0 to 2); "auxAxType " is the auxiliary axis type ("L" for linear, "R" for rotational); "sampleList" is the calibration data sample list (3 samples required for rotational axes); "xyz" is the XYZ coordinates of the sample; "enc" is the auxiliary axis encoder value.
Returned value	Reduction gear ratio

4.9.69 Get the Axis Reduction Ratio

Command format	mot.getGearRatio(int8_t gpId, int8_t axId)
Command	Get the axis reduction ratio
Interface parameters	gpId: Group ID. The range is 0 to n-1.axId: Axis number within the group. The range is 0 to 8.
Returned value	Reduction gear ratio

4.9.70 20-Point Calibration

Command format	mot.calibrateWith20Points(string strJsonInput)
Command	20 Dot calibration. Data becomes effective after successful calibration, but it will not be saved.
Interface parameters	strJSONInput: JSON-formatted calibration input data. Format: {"gpId": 0, "utNum": 0, "sampleList": [{"jnt": [123.0,...]},...]}, where "gpId" is the group number (0 to n-1), " utNum" is the tool number (0 to 15), and "sampleList" is the calibration data sample list. 10 20 samples; "jnt" represents joint coordinates, with values from 1 to 9
Returned value	JSON format calibration results Format: {"ec": 0, "UT": "***", "zero": "***"}"ec" stands for 64-bit error code; "UT" stands for tool value string; "zero" stands for joint zero point value string

4.10 Business Function Agent Lib

4.10.1 Get the Business Layer Version Information

Command format	lib.versions()

Command	Get the business layer version information
Interface parameters	null
Returned value	Version information string

4.10.2 Set Controller Name

Command format	lib.setName(string name)
Command	Set controller name
Interface parameters	name: name
Returned value	null

4.10.3 Get Controller Version Information

Command format	lib.getControllerVer()
Command	Get controller version information
Interface parameters	null
Returned value	Overall version information for the controller

4.10.4 Shut Down

Command format	lib.shutdown()
Command	Turn off control system
Interface parameters	null
Returned value	null

4.10.5 Restart

Command format	lib.reboot()
Command	Restart control system
Interface	null

parameters	
Returned value	null

4.10.6 Get Controller Information

Command format	lib.getIpcInfo()
Command	Get controller information
Interface parameters	null
Returned value	Controller information

4.10.7 Sync Memory Data to the Hard Drive

Command format	lib.syncDisk()
Command	Sync memory data to the hard drive
Interface parameters	null
Returned value	null

4.10.8 Clear Hard Drive Data

Command format	lib.clearDisk()
Command	Clear hard drive data
Interface parameters	null
Returned value	null

4.11 Digital IO Operation Agent IO

4.11.1 Get the Number of Digital Input Ports

Command format	io.getMaxDinNum()
Command	Get the number of digital input ports
Interface	null

parameters	
Returned value	Quantity

4.11.2 Get the Number of Digital Output Ports

Command format	io.getMaxDoutNum()
Command	Get the number of digital output ports
Interface parameters	null
Returned value	Quantity

4.11.3 Get the Number of Numeric Input Groups

Command format	io.getMaxDinGrp()
Command	Get the number of digit input groups. 32 inputs form a group. Group 0 includes digits 0 to 31, group 1 includes digits 32 to 63, and so on.
Interface parameters	null
Returned value	Quantity

4.11.4 Get the Number of Digital Output Groups

Command format	io.getMaxDoutGrp()
Command	Get the number of digital output groups. 32 outputs are grouped, with group 0 including digital outputs 0 to 31, group 1 including digital outputs 32 to 63, and so on.
Interface parameters	null
Returned value	Quantity

4.11.5 Get the Numeric Input Group Value

Command format	io.getDinGrp(int32_t grpIndex)
Command	Get the numeric input group value

Interface parameters	grpIndex: Group number. Range from 0 to n-1.
Returned value	Group value. A 32-bit unsigned integer, where the least significant bit represents the value of the group's digit input 0 and the most significant bit represents the value of the group's digit input 31.

4.11.6 Get the Digital Output Group Value

Command format	io.getDoutGrp(int32_t grpIndex)
Command	Get the digital output group value
Interface parameters	grpIndex: Group number. Range from 0 to n-1.
Returned value	Group value. A 32-bit unsigned integer, where the least significant bit represents the value of the group' s digital output 0, and the most significant bit represents the value of the group's digital output 31.

4.11.7 Get the Digital Input Group Status

Command format	io.getDinMaskGrp(int32_t grpIndex)
Command	Get the digital input group status
Interface parameters	grpIndex: Group number. Range from 0 to n-1.
Returned value	Group status. A 32-bit unsigned integer where the least significant bit (LSB) indicates the status of digital input 0 in the group, and the most significant bit (MSB) indicates the status of digital input 31.

4.11.8 Get the Digital Output Group Status

Command format	io.getDoutMaskGrp(int32_t grpIndex)
Command	Get the digital output group status
Interface Parameters	grpIndex: Group number. Range from 0 to n-1.
Returned value	Group status. A 32-bit unsigned integer where the least significant bit indicates the status of digital output 0 in the group, and the most significant bit indicates the status of digital output 31.

4.11.9 Set the Digital Input Port Value

Command format	io.setDin(int32_t portIndex, bool value)

Command	Set the numeric input port value
Interface parameters	portIndex: port number. The range is 0 to n-1. value: price
Returned value	null

4.11.10 Set the Digital Output Port Value

Command format	io.setDout(int32_t portIndex, bool value)
Command	Set the digital output port value
Interface parameters	portIndex: port number. The range is 0 to n-1. value: price
Returned value	null

4.11.11 Get the Digital Input Port Value

Command format	io.getDin(int32_t portIndex)
Command	Get the digital input port value
Interface parameters	portIndex: port number. The range is 0 to n-1.
Returned value	Price

4.11.12 Get the Digital Output Port Value

Command format	io.getDout(int32_t portIndex)
Command	Get the digital output port value
Interface parameters	portIndex: port number. The range is 0 to n-1.
Returned value	Price

4.11.13 Set the Numeric Input Status Value

Command format	io.setDinMaskBit(int32_t portIndex, bool stat)

Command	Set the digital input status value. When the digital input status is "VIRTUAL", you can set the value using setDin. When the digital input status is "REAL", the value is read from the IO module, and setting the value through setDin is invalid.
Interface parameters	portIndex: port number. The range is 0 to n-1. stat: status. true means virtual; false means real
Returned value	null

4.11.14 Set the Digital Output Status Value

Command format	io.setDoutMaskBit(int32_t portIndex, bool stat)
Command	Set the digital output status. When the digital output status is "VIRTUAL", you can set the value using setDout, but the IO module output cannot be triggered. When the digital output status is "REAL", you can set the value using setDout and trigger the IO module output.
Interface parameters	portIndex: port number. The range is 0 to n-1. stat: status. true means virtual; false means real
Returned value	null

4.12 Midnight Operation Agent Home

4.12.1Set Zero Point

Command format	home.setHome(int gpID, string home, int mask)
Command	Set zero point
Interface parameters	gpID: Group number. The range is 0 to n-1. home: zero point. For example: " {0, -90, 180,0, 90,0, 0, 0, 0}" mask: mask
Returned value	null

4.12.2Get Zero

Command format	home.getHome(int gpID)

Command	Get Zero
Interface parameters	gpID: Group number. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,0.0,0.0,0. 0}"

4.12.3Calculate Zero Point from Zero Deviation

Command format	home.setHomeByOffset(int gpID, string home, string offset, int mask)
Command	Calculate the zero point using zero-point deviation. When encoder values are lost, you can use
Interface parameters	gpID: Group number. The range is 0 to n-1. home: zero point. For example: " {0, -90, 180,0, 90,0, 0, 0, 0}" offset: zero offset. For example: " {0,0,0,0,0,0,0,0,0}" mask: mask
Return value	null

4.12.4Get Zero Offset

Command format	home.getHomeOffset(int gpID)
Command	Get zero offset
Interface parameters	gpID: Group number. The range is 0 to n-1.
Returned value	Data points. Format: curly braces expanded, semicolons separated numbersFor example: " {1.0,2.0,3.0,4.0,5.0,6.0,0.0,0.0,0. 0}"

4.12.5Calculate Zero Point from Encoder Values

Command format	home.setHomeByEncoder(int gpID, string home, int mask)
Command	Calculate zero point from encoder values
Interface parameters	gpID: Group number. The range is 0 to n-1. home: zero point. For example: " {0, -90, 180,0, 90,0, 0, 0, 0}" mask: mask

Returned value	null

4.13 External Operation Function Agent Ext

4.13.1 Add External Input Signal

Command format	ext.addExtIn(int iDin, string sig)
Command	Add external input signal
Interface parameters	iDin: External input signal index. Range from 0 to n-1. sig: External input signal name
Returned value	null

4.13.2 Remove External Input Signal

Command format	ext.delExtIn(int iDin, string sig)
Command	Remove external input signal
Interface parameters	iDin: External input signal index. Range from 0 to n-1. sig: External input signal name
Returned value	null

4.13.3 Get External Input Signal Configuration List

Command format	ext.getExtInList()
Command	Get external input signal configuration list
Interface parameters	null
Returned value	External input signal configuration list. Format: data queue in square brackets, separated by half-width commas, for example: "[iDin=1, sig=iENABLE,...]" Where "iDin" is the index of the external input signal, and "sig" is the name of the external input signal

4.13.4 Add External Output Signal

Command format	ext.addExtOut(int iDout, string sig)

Command	Add external output signal
Interface parameters	iDout: External output signal index. Range from 0 to n-1. sig: External output signal name
Returned value	null

4.13.5 Delete External Output Signal

Command format	ext.delExtOut(int iDout, string sig)
Command	Delete external output signal
Interface parameters	iDout: External output signal index. Range from 0 to n-1. sig: External output signal name
Returned value	null

4.13.6 Get External Output Signal Configuration List

Command format	ext.getExtOutList()
Command	Get external output signal configuration list
Interface parameters	null
Returned value	External output signal configuration list. Format: configuration data queue separated by half- width commas in parentheses, for example: "[iDout= 1, sig= \" oENABLE_STATE\ ",...]" Where "iDout" is the index of the external output signal, and "sig" is the name of the external output signal

4.13.7 Modify the Reference Register Configurat- ion

Command format	ext.modRefRegData(int iRef, int iGroup, string typeReg, int iReg, double precision)
Command	Modify the reference register configurat- ion
Interface parameters	iRef: Reference Signal Index iGroup: Bind group number typeReg:The bound register type. "JR" indicates the JR register. iReg: Bind register index precision: accuracy

Returned value	null

4.13.8 Delete Reference Register Configuration

Command format	ext.delRefRegData(int iRef)
Command	Delete reference register configuration
Interface parameters	iRef: Reference Signal Index
Returned value	null

4.13.9 Get the Address Register Configuration

Command format	ext.getRefRegData(int32_t iRef)
Command	Get the address register configuration
Interface parameters	iRef: Reference Signal Index
Returned value	Reference register configuration For example: "iRef= 1, iGroup=0, typeReg=" JR ", iReg=99, precision=1.000000"Where "iRef" is the reference signal index, "sig" is the external output signal name, "iGroup" is the binding group number, "typeReg" is the binding register type, "iReg" is the binding register index, and "precision" is the precision

4.13.10 Get the Configuration List of the Reference Register

Command format	ext.getRefRegDataInfo()
Command	Get the configuration list of the reference register
Interface parameters	null
Returned value	Reference register configuration list. Format: data queue in parentheses, separated by half-width commas. For example: "[" iRef= 1,iGroup=0,typeReg= "JR",iReg=99,precision=1.000000 ",...]", where " iRef" is the reference signal index, "sig" is the external output signal name, "iGroup" is the binding group number, "typeReg" is the binding register type, "iReg" is the binding register index, and " precision" is the precision

4.13.11 Set External Program

Command format	ext.setExtProgName(int index, string entry)

Command	Set external program
Interface parameters	index: configuration item number entry: entry point path
Returned value	null

4.13.12 Get the External Program Mapping Table

Command format	ext.getExtProgNameMap()
Command	Get the external program mapping table
Interface parameters	null
Returned value	Mapping table. Format: [] with half-width commas to separate external program configuration mapping tables, for example: "[1=X.PRG,2=/a/Y.PRG]".

4.13.13 Set the External Program Number Binding Register

Command format	ext.setR4ExtProg(int iR)
Command	Set the external program number binding register
Interface parameters	iR: R register index. Range from 0 to n-1
Returned value	null

4.13.14 Access the External Program Counter Register

Command format	ext.getR4ExtProg()
Command	Access the external program counter register
Interface parameters	null
Returned value	R register index

4.13.15 Save Hook Content

Command format	ext.saveHook()

Command	Save content with encoding/decoding and external input/output configurations
Interface parameters	null
Returned value	null

4.13.16 Preserve

Command format	ext.save()
Command	Save the contents ofthe referenced register configuration and external running programs
Interface parameters	null
Returned value	null

4.14 Dynamic Function Proxy Dyn

4.14.1 Set Collaboration Mode

Command format	dyn.setDynamicMode(bool en)
Command	Set collaboration mode
Interface parameters	null
Returned value	null

4.14.2 Get Collaboration Mode

Command format	dyn.getDynamicMode()
Command	Get collaboration mode
Interface parameters	null
Returned value	null

4.14.3 Initialize the Dynamic Identification Module

Command format	dyn.initIdentification(int8_t sampleNum, int8_t isLoadIden)

Command	Initialize the dynamic identification module
Interface parameters	sampleNum: Number of times the incentive trajectory runsl isLoadIden: Is it a load identifier?
Returned value	null

4.14.4 Set Load to Identify Joint Range of Motion

Command format	dyn.genLoadIdenTraj(string strLimit)
Command	Set load to identify joint range of motion
Interface parameters	strLimit: Axis movement range. Format: A dot data queue separated by semicolons, for example: " {2, 10,-10};{3, 10,-10};{4, 10,-10};{5, 10,-10}". {2, 10,-10} represents the positive/negative limits of the first three axes, followed by the positive/negative limits of the fourth, fifth, and sixth axes.
Returned value	null

4.14.5 Get Load Incentive Trajectory Coefficients

Command format	dyn.getLoadIdenTrajCoff()
Command	Get load incentive trajectory coefficients
Interface parameters	null
Returned value	Trajectory coefficient. Format: a sequence of numbers separated by half-width commas in parentheses.For example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.14.6 Run the Trajectory

Command format	dyn.runIdenTraj(int8_t trajCnt, int8_t sampleCnt)
Command	Run the trajectory
Interface parameters	trajCnt: Incentive trajectory sequence numberl sampleCnt: The sequence number of the incentive trajectory
Returned value	null

4.14.7 Low-Speed Operation Excitation Trajectory

Command format	dyn.runIdenTrajSafely(int8_t trajCnt)
Command	Low-speed operation excitation trajectory
Interface parameters	trajCnt: Incentive trajectory sequence number
Returned value	null

4.14.8 Stop the Trajectory

Command format	dyn.stopCurrentTraj()
Command	Stop the stimulus trajectory
Interface parameters	null
Returned value	null

4.14.9 Get the Starting Point of the Identification Trajectory

Command format	dyn.getStartPos(int8_t trajCnt)
Command	Get the starting point of the identification trajectory
Interface parameters	trajCnt: Incentive trajectory sequence number
Returned value	Joint position. Format: braces with a sequence separated by half-width commas.For example: " {1.0,2.0,3.0,4.0,5.0,6.0,}"

4.14.10 Drive Dynamics Identification

Command format	dyn.identify()
Command	Drive dynamics identification
Interface parameters	null
Returned value	null

4.14.11 Run Load Identification

Command format	dyn.identifyLoad(int8_t num)
Command	Run load identification
Interface parameters	num: Load number
Returned value	null

4.14.12 Get the Status of the Incentive Trajectory

Command format	dyn.getIdenTrajState()
Command	Get the status of the incentive trajectory
Interface parameters	null
Returned value	❖ 0: indicates that the incentive trajectory has completed ❖ 1: indicates that the incentive trajectory is running ❖ 2: indicates that the incentive trajectory failed to run

4.14.13 Load Balancing

Command format	dyn.switchLoad(int8_t loadNum)
Command	Load balancing
Interface parameters	loadNum: Load sequence number
Returned value	null

4.14.14 Clear Load

Command format	dyn.clearLoad()
Command	Clear load. Switch to no-load.
Interface parameters	null
Returned value	null

4.14.15 Switch Dynamic Module

Command format	dyn.switchMode(int8_t mode)
Command	Switch dynamic module
Interface parameters	mode: dynamic module. 0 indicates drag mode; 1 indicates position control mode (torque feedforward mode), which is the default mode; 2 indicates collision response mode
Returned value	null

4.14.16 Set Collision Sensitivity

Command format	dyn.setCollisionSen(int8_t axis, int8_t sens)
Command	Set collision sensitivity
Interface parameters	axis: axis number sens: collision sensitivity
Returned value	null

4.14.17 Get Collision Sensitivity

Command format	dyn.getCollisionSen(int8_t axis)
Command	Get collision sensitivity
Interface parameters	axis: axis number
Returned value	Collision sensitivity

4.15 Collect Operation Proxy

4.15.1 Configure Collection Tasks

Command format	collect.config(int32_t iSessionId, string strConfig)
Command	Configure collection tasks
Interface parameters	iSessionId: Network connection ID strConfig: Configuration information
Returned value	null

4.15.2 Add Collection Task

Command format	collect.addTask(int32_t iSessionId, int32_t tid, int32_t iSampleKT, string strVarList)
Command	Add collection task
Interface parameters	iSessionId: Network connection ID tid: Task ID iSampleKT: Sampling interval multiplier strVarList: Variable string list
Returned value	null

4.15.3 Delete Collection Task

Command format	collect.delTask(int32_t iSessionId, int32_t tid)
Command	Delete collection task
Interface parameters	iSessionId: Network connection ID tid: Task ID
Returned value	null

4.15.4 Start All Collection Tasks

Command format	collect.start(int32_t iSessionId)
Command	Start all collection tasks
Interface parameters	iSessionId: Network connection ID
Returned value	null

4.15.5 Start Collection

Command format	collect.start(int32_t iSessionId, int32_t tid)
Command	Start collection
Interface parameters	iSessionId: Network connection ID tid: Task ID
Returned	null

value	

4.15.6 Stop All Collection Tasks

Command format	collect.stop(int32_t iSessionId)
Command	Stop all collection tasks
Interface parameters	iSessionId: Network connection ID
Returned value	null

4.15.7 Stop Collection

Command format	collect.stop(int32_t iSessionId, int32_t tid)
Command	Stop collection
Interface parameters	iSessionId: Network connection ID tid: Task ID
Returned value	null

4.15.8 View Existing Collection Tasks

Command format	collect.getExistTasks(int32_t iSessionId)
Command	View existing collection tasks
Interface parameters	iSessionId: Network connection ID
Return value	Task ID (tid) list. Format: queue separated by half-width commas and enclosed in parentheses.For example: "[1,2,3]"

4.15.9 Get the Running Collection Task List

Command format	collect.getRunningTasks(int32_t iSessionId)
Command	Get the running collection task list
Interface parameters	iSessionId: Network connection ID

Returned value	Task ID (tid) list. Format: queue separated by half-width commas and enclosed in parentheses.For example: "[1,2,3]"

4.15.10 Get the Current Timestamp

Command format	collect.getCurTimestamp()
Command	Get the current timestamp
Interface parameters	null
Returned value	Time stamp (tick count)

4.16 Proxy for Encoding and Decoding Functions

4.16.1 Add Encoder

Command format	coder.addEncoder(int32_t iDout, int32_t lenDout, int32_t iR)
Command	Add encoder
Interface parameters	iDout: The starting index for the digital output. The range is from 0 to n-1.l lenDout: Number of digits for the digital output. Range: 1 to n. iR: R register index. Range from 0 to n-1
Returned value	null

4.16.2 Delete Encoder

Command format	coder.delEncoder(int32_t iDout, int32_t lenDout, int32_t iR)
Command	Delete encoder
Interface parameters	iDout: The starting index for the digital output. The range is from 0 to n-1.l lenDout: Number of digits for the digital output. Range: 1 to n. iR: R register index. Range from 0 to n-1
Returned value	null

4.16.3 Get Encoder List

Command	coder.getEncoderList()

format	
Command	Get encoder list
Interface parameters	null
Returned value	Decoder list. Format: [] with half-width commas separating decoder data queues, e.g., "[" iDout=31, lenDout=3, iR=5 ",...]" Where "iDout" is the starting index of DOUT, "lenDout" is the number of DOUT bits, and "iR" is the R register index

4.16.4 Add Decoder

Command format	coder.addDecoder(int iR, int iDin, int lenDin)
Command	Add decoder
Interface parameters	iR: R register index. Range from 0 to n-1iDin: The starting index for digital input. The range is 0 to n-1.l lenDin: Number of digits for numeric input. Range from 1 to n.
Returned value	null

4.16.5 Delete Decoder

Command format	coder.delDecoder(int iR, int iDin, int lenDin)
Command	Delete decoder
Interface parameters	iR: R register index. Range from 0 to n-1iDin: The starting index for digital input. The range is 0 to n-1.l lenDin: Number of digits for numeric input. Range from 1 to n.
Returned value	null

4.16.6 Get Decoder List

Command format	coder.getDecoderList()
Command	Get decoder list
Interface Parameters	null

Return value	Decoder list. Format: [] with half-width commas separating decoder data queues, e.g., "[iR= 1, iDin=2, lenDin=3,...]". Where "iR" is the R register index, "iDin" is the DIN start index, and "lenDin" is the DIN bit count

4.16.7 Preserve

Command format	coder.save()
Command	Save content with encoding/decoding and external input/output configurations
Interface parameters	null
Returned value	null

4.17 Calibrate the Operation Agent

4.17.1 Collaborative Group Calibration

Command format	cali.calibrate(int8_t gpId, int8_t num, int8_t type, string strData)
Command	Collaborative Group Ca- libration
Interface parameters	gpId: Group ID. The range is 0 to n-1. num: Collaboration group number. Range: -1 to 2type: calibration method code. 5 indicates the 5-point method; 3 indicates the 3-point method.strData: Calibration data. Format: A dot data queue separated by semicolons, for example: "{1,2,};{3,4,};{5,6,}."
Returned value	Calibration results For example: "ret=0x0, data={{7,8,}, {9, 10,}}" Here, "ret" is the calibration return value, and "data" is the calibration result, which is a two-dimensional matrix.

4.17.2 External Shaft Reduction Ratio Calibration

Command format	cali.caliExtAxisRatio(bool isLine, string strCali, string strEnc)
Command	External shaft reduction ratio calibration

Interface parameters	isLine: Whether the axis is a straight line. True indicates a straight line axis; false indicates a rotating axis.strCali:Calibration data. Format: a dot data queue separated by semicolons, for example: " {1,2,};{3,4,};{5,6,}." strEnc: Encoded data. Format: a dot data queue separated by semicolons, for example: " {1,2,3}"
Returned value	Calibration res- ults

4.17.3 External Tool Calibration

Command format	cali.caliExtTool(string strCali)
Command	External Tool Calibration
Interface parameters	strCali:Calibration data. Format: a dot data queue separated by semicolons, for example: " {1,2,};{3,4,};{5,6,}."
Returned value	Calibration results For example: "ret=0x0, data={7,8,}""ret" is the calibration return value, and "data" is the calibration result

4.17.4 External Workpiece Calibra- tion

Command format	cali.caliExtWorkPiece(string strCali)
Command	External workpiece calibra- tion
Interface parameters	strCali:Calibration data. Format: a dot data queue separated by semicolons, for example: " {1, 2,};{3,4,};{5,6,};{7,8,}." Here, {1,2,} represents spatial point 1; {3,4,} represents spatial point 2; {5,6,} represents spatial point 3; {7,8,} represents the external tool calibration values (XYZABC).
Returned value	Calibration res- ults For example: "ret=0x0, data={9, 10,}"Where "ret" is the calibration return value and "data" is the calibration result

4.18 Algorithmic Agent Algo

4.18.1 Convert Joint Coordinates to Space Coordinates 1

Command format	Algo.calcJPosToCPos(int ToolNum, int WorkNum, string strJPos)
Command	Convert joint coordinates to spatial coordin- ates

Interface parameters	ToolNum: Tool number WorkNum: Workpiece number strJPos: Joint angle. Format: A dot data queue separated by semicolons, for example: " {1,2,}".
Returned value	Calibration results For example: "ret=0x0, data={{3,4,}, {5,6,}}"Here, "ret" is the calibration return value, "data" is the calibration result, {3,4,} is the flange position in the base coordinate, and {5,6,} is the tool point position in the workpiece.

4.18.2 Transform Joint Coordinates to Space Coordinates 2

Command format	Algo.calcCPosToJPosByApproach(int ToolNum, int WorkNum, int Cfg, string strCPos, string strJPos)
Command	Convert joint coordinates to spatial coordinates. This command calculates the angles of adjacent joints.
Interface parameters	ToolNum: Tool number WorkNum: Workpiece number Cfg: Morph strCPos: Spatial angle. Format: A dot data queue separated by semicolons, for example: " {1,2,}".strJPos: Joint angle. Format: A dot data queue separated by semicolons, for example: " {3,4,}"
Returned value	Calibration res- ults For example: "ret=0x0, data={5,6,}"Where "ret" is the calibration return value and "data" is the calibration result

4.18.3 Joint Coordinate to Space Coordinate 3

Command format	Algo.calcCPosToJPosByAnalysis(int ToolNum, int WorkNum, int State, int Cfg, string strCPos)
Command	Convert joint coordinates to spatial coordinates. This command solves the analytical solution.

Interface parameters	ToolNum: Tool number WorkNum: Workpiece numberl State: The type of spatial position. 0 indicates the flange's spatial position in the base coordinate system; 1 indicates the tool point's spatial position in the base coordinate system; 2 indicates the tool point's spatial position in the work coordinate system.Cfg: Morph strCPos: Spatial angle. Format: a dot data queue separated by semicolons, for example: " {1,2,}".
Returned value	Calibration results For example: "ret=0x0, data={3,4,}""ret" is the calibration return value, and "data" is the calibration result

4.19 Simulate IO Operations for the Ain/aout Agent

4.19.1 Get the Number of IO Variables

Command format	ain/aout.getSize()
Command	Get the maximum number of IO variables supported by the variable frame
Interface parameters	null
Returned value	Quantity

4.19.2 Get the Actual Port Count

Command format	ain/aout.getRealSize()
Command	Get the actual port count
Interface parameters	null
Returned value	Quantity

4.19.3 Get Port Value

Command format	ain/aout.getValue(size_t index)
Command	Get port value
Interface parameters	index: port number. The range is 0 to n-1.

Returned value	Price

4.19.4 Set Port Value

Command format	ain/aout.setValue(size_t index, float value)
Command	Set port value
Interface parameters	index: port number. The range is 0 to n-1. value: price
Returned value	null

4.19.5 Get Port Status

Command format	ain/aout.getMask(size_t index)
Command	Get the port status. When the status is "VIRTUAL", the value set by setValue is stored in the process memory. When the status is "REAL", the value is stored in the variable frame (shared memory).
Interface parameters	index: port number. The range is 0 to n-1.
Returned value	❖ true: indicates a virtual state ❖ false: indicates the actual status

4.19.6 Set Port Status

Command format	ain/aout.setMask(size_t index, bool bVirtual)
Command	Set the port status. When the status is "VIRTUAL", the value set by setValue is stored in the process memory. When the status is "REAL", the value is stored in the variable frame (shared memory).
Interface parameters	index: port number. The range is 0 to n-1. bVirtual: Status. True indicates virtual status; false indicates real status.
Returned value	null

4.19.7 Get IO Group Status

Command format	ain/aout.getMaskGrp(size_t grpIndex)
Command	Get the IO group status. 32 ports form one group.

Interface parameters	grpIndex: Group number. Range from 0 to n-1.
Returned value	Group status. A 32-bit unsigned integer where the least significant bit represents the status of port 0 in the group, and the most significant bit represents the status of port 31. ❖ true: indicates a virtual state❖ false: indicates the actual status

4.19.8 Set IO Group Status

Command format	ain/aout.setMaskGrp(size_t grpIndex, uint32_t mask)
Command	Set the IO group status. 32 ports form one group.
Interface parameters	grpIndex: Group number. The range is 0 to n-1. mask: Group status. A 32-bit unsigned integer where the least significant bit (LSB) represents the status of port 0 in the group, and the most significant bit (MSB) represents the status of port 31. True indicates virtual status; false indicates real status.
Returned value	Error code❖ ERR_VAR_MGR_OK: indicates normal ❖ Other: indicates an exception

Guangzhou Auctech Automation Technology Ltd

Add: Room903, BuildingE1, Design City, No.7, Hexian North Street, Helong Street, Baiyun District, Guangzhou City, CHINA

Tel:+862084898493

E-mail:info@auctech.com.cn

Web:www.auctech.com.cn