Dobot Magician
Communication Protocol

Issue: V1.1.5

Date: 2019-08-05

Shenzhen Yuejiang Technology Co.,Ltd

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Yuejiang makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Yuejiang be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robotic arm is used on the premise of fully understanding the robotic arm and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related commands, Damages or losses will be happen in the using process, Dobot shall not be considered as a guarantee regarding to all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robotic arm.

Shenzhen Yuejiang Technology Co., Ltd

Address: 3F, Building NO.3, Tongfuyu Industrial Town, Nanshan District, Shenzhen, China

Website: www.dobot.cc

Preface

Purpose

The document is avaliale for communication protocol of commands or data interaction between Dobot Magician upper computer and Dobot Magician robot arm.

Intended Audience

This document is intended for:

- Customer Engineer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Change History

Date	Change Description
2019/08/05	Add the versions of sliding rail, color sensor, and infrared sensor
2019/05/05	Delete communication protocol GetQueuedCmdLeftspace
2018/07/12	Add description of Set/Get IRSwitch
2017/05/05	Modify the mistake of SetHOMECmd, Add description of interface command about rail, Add description of Get DeviceID, Add description of color sensor ports command
2016/12/20	Get the remaining space of command queue
2016/11/30	Add description of movement control (JOG PTP CP ARC)
2016/11/21	Add BLE Reading & Writing Characteristic UUID
2016/11/18	Add description of communication parameters
2016/09/22	Create a document

Symbol Conventions

The symbols that may be founded in this document are defined as follows.

Symbol	Description
⚠ DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury
⚠ WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robotic arm damage
⚠ NOTICE	Indicates a potentially hazardous situation which, if not avoided, can result in robotic arm damage, data

Symbol	Description
	loss, or unanticipated result
🕮NOTE	Provides additional information to emphasize or supplement important points in the main text

Contents

1. Communication Protocol

1.1 Communication Parameters

Table 1 Communication parameters description

Communication Parameters	Details	Description
USB to serial port	Baud rate	115200bps
	Data bits	8 Bit
	Stop bit	1 Bit
	Parity bit	Void
Wi-Fi	IP	Route and other distribution
	COM port	8899
BLE	Service UUID	0003CDD0-0000-1000-8000-00805F9B0131
	Read (Command) Port Characteristic UUID	0003CDD1-0000-1000-8000-00805F9B0131
	Write (Command) Port Characteristic UUID	0003CDD2-0000-1000-8000-00805F9B0131
TTL	Baud rate	115200bps
	Data bits	8 Bit
	Stop bit	1 Bit
	Parity bit	Void

1.2 Protocol Introduction

Dobot Magician can be controlled by PC/Android/iOS, achieving data transmission through certain communication protocols. The communication can be realized by USB-serial port, TLL level serial port, Wi-Fi (UDP).

Physical layer receives 8 bite raw data each time, which need to determine the starting, end of the data and verify the accuracy of the data by communication protocols. The communication protocol includes packet header, packet load and checksum to ensure the accurate transmission.

1.2.1 Protocol Features

Dobot Magician communication protocol features are as follows:

- Protocol command does not have fixed length.
- Protocol command consists of packet header, payload frame, and check.
- All communications are sent by the host initialtively, and for all communications commands, the client will return data (both read and write). For queue commands, the client returns data with 64-bit index.
- All commands are divided into immediate commands and queue commands. All read-operations are the immediate commands, which can be executed immediately.
- The queue commands will be placed in the queue of the client for serial execution. For write (or set) operation, motion commands should be queue commands (such as **homing** , **JOG**, **PTP**).

- Motion parameter commands are not only the immediate commands but also the queue commands.
- Before sending queue commands to client, the host should inquire the remaining space of command queue of client (check once and send multiple commands).
- The immediate command is always executed immediately. While the execution status of a queue command can be obtained by querying the index of the queue command being executed in the client and comparing with the index of this queue command (mentioned in point 3).
- The parameters in the commands use little endian mode.

1.2.2 Checksum Calculation

In Dobot Magician communication protocol, the checksum at the ending side is calculated as follows.

Step 1 Add all the contents of the Payload byte by byte (8 bits) to get a result R (8 bits);

Step 2 Get the 2's complement of the result **R** (8 bits), and put it into check byte.

📖NOTE

2's complement. For an N-bit number, the 2's complement is equal to 2 ^ N minus the number. In this protocol, assuming the result R is 0x0A, and the 2's complement, i.e., the result of the above checking is equal to (2 ^ 8 - 0x0A) = (256 - 10) = 246 = 0xF6.

At the receiving end, the method of verifying whether a frame of data is correct as follows:

Step 1 Add all the contents of the Payload byte by byte (8 bits) to get a result A;

Step 2 Add result **A** and the check byte. If the result is 0, the checksum is correct.

1.2.3 Protocol Classification

Protocols can be divided into the following parts according to their different implementation functions:

- Queue execution control command
- Related command of device information
- Common parameter command
- Home function command
- Handhold teaching command
- Jog mode command
- PTP mode command
- CP mode command
- TRACK mode command
- WAIT mode command
- TRIG trigger related command
- IO control command and so on

By classification, communication protocol function IDs are divided into the following items are shown in Table 2:

Table 2 Classification of functional items

Classification of functional items	Function ID area	Avaliable ID number
ProtocolFunctionDeviceInfoBase	[0, 10)	10
ProtocolFunctionPoseBase	[10, 20)	10
ProtocolFunctionALARMBase	[20, 30)	10
ProtocolFunctionHOMEBase	[30, 40)	10
ProtocolFunctionHHTBase	[40, 50)	10
ProtocolFunctionArmOrientationBase	[50, 60)	10
ProtocolFunctionEndEffectorBase	[60, 70)	10
ProtocolFunctionJOGBase	[70, 80)	10
ProtocolFunctionPTPBase	[80, 90)	10
ProtocolFunctionCPBase	[90, 100)	10
ProtocolFunctionARCBase	[100, 110)	10
ProtocolFunctionWAITBase	[110, 120)	10
ProtocolFunctionTRIGBase	[120, 130)	10
ProtocolFunctionEIOBase	[130, 140)	10
ProtocolFunctionCALBase	[140, 150)	10
ProtocolFunctionWIFIBase	[150, 160)	10
ProtocolFunctionQueuedCmdBase	[240, 250)	10
ProtocolMax	256	1

📖NOTE

- An ID description is provided in each of the command description in the following contents;
- In the following **Ctrl** byte, the bit 0 of **Ctrl** is **rw**, the bit 1 of **Ctrl** is **Queued**

📖NOTE

When **isQueue** = 1,that indicats the instruction is a queue command,which returns a 64-bit index. So the length is 2+8. When **isQueue** = 0, the instruction is an immediate command, which has no return. So the length is 2+0.

1.3 Device Information

These commands are used to set device SN number, device name, device version number, and read the current device information.

1.3.1 Set/Get Device SN

- This command is to set device serial number, the issued command package is shown in Table 3 and the returned command package is shown in Table 4.

Table 3 The command package of Device SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	0	1	0	char[n] DeviceSN	Payload checksum

Table 4 The returned command package of SetDevice SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	0	1	0	Empty	Payload checksum

- This command is to get device serial number, the issued command package is shown in Table 5 and the returned command package is shown in Table 6.

Table 5 The command package of GetDevice SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	0	0	0	Empty	Payload checksum

Table 6 The returned command package of GetDevice SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	0	0	0	char[n] DeviceSN	Payload checksum

1.3.2 Set/Get Device Name

- This command is to set device name, the issued command package is shown in Table 7 and the returned command package is shown in Table 8.

Table 7 The command packet of SetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	1	1	0	char[n] DeviceName	Payload checksum

Table 8 The returned command packet of SetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	1	1	0	Empty	Payload checksum

- This command is to get device name, the issued command packet format is shown in Table 9, and the returned command packet format is shown in Table 10.

Table 9 The command packet of GetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	1	0	0	Empty	Payload checksum

Table 10 The returned command packet of GetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	1	0	0	char[n] DeviceName	Payload checksum

1.3.3 Get Device Version

This command is to get device version, the issued command packet format is shown in Table 11, and the returned command packet format is shown in Table 12.

Table 11 The command packet of GetDeviceVersion

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	2	0	0	Empty	Payload checksum

Table 12 The returned command packet of GetDeviceVersion

Header	Len	Payload						Checksum
		ID	Ctrl		Params			
			rw	isQueued				
0xAA 0xAA	2+3	2	0	0	uint8_t: majorVersion	uint8_t: minorVersion	uint8_t : revision	Payload checksum

1.3.4 Set/Get DeviceWithL

- This command is to set sliding rail enable status, the issued command packet format is shown in Table 13, and the returned command packet format is shown in Table 14.

Table 13 The command packet of Set DeviceWithL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	3	1	0	WithL withL (See Program 1)	Payload checksum

Table 14 The returned command packet of Set DeviceWithL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	3	1	0	Empty	Payload Checksum

Program 1 WithL definition

```
Typedef enum tagVersion{
    VER_V1,
    VER_V2
    }Version
Typedef struct tagWithL{
    Uint8_t isWithL;
    Version version;
    }WithL;
```

- This command is to get sliding rail enable status, the issued command packet format is shown in Table 15, and the returned command packet format is shown in Table 16.

Table 15 The command packet of Get DeviceWithL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	3	0	0	Empty	Payload checksum

Table 16 The returned command packet of Get DeviceWithL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	3	0	0	uint8_t:WithL	Payload checksum

1.3.5 Get DeviceTime

This comand is to get device time, the issued command packet format is shown in Table 17, and the returned command packet format is shown in Table 18.

Table 17 The command packet of GetDeviceTime

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	4	0	0	Empty	Payload checksum

Table 18 The returned command packet of GetDeviceTime

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	4	0	0	uint32_t：gSystick	Payload checksum

1.3.6 Get DeviceID

This command is to get device ID, the issued command packet format is shown in Table 19, and the returned command packet format is shown in Table 20.

Table 19 The command packet of Get DeviceID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	4	0	0	Empty	Payload checksum

Table 20 The returned command packet of Get DeviceID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	4	0	0	uint32_t[3]：DeviceID	Payload checksum

1.4 Real-time Pose

The function of setting the initial pose, obtaining the real-time pose, the kinematic parameters and so on.

1.4.1 GetPose

This command is to get the real-time pose of the Dobot, the issued command packet format is shown in Table 21, and the returned command packet format is shown in Table 22.

Table 21 The command packet of GetPose

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	10	0	0	Empty	Payload checksum

Table 22 The returned command packet of GetPose

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	10	0	0	Pose（See Program 2）	Payload checksum

Program 2 Pose definition

```
typedef struct tagPose {
float x;                    //Robotic arm coordinate system x
float y;                    //Robotic arm .
float z;                    //Robotic arm coordinate system z
float r;                    //Robotic arm coordinate system r
float jointAngle[4];        //Robotic arm 4 axis(The basement, rear arm, forearm,EndEffector) angles
} Pose;
```

1.4.2 Reset Pose

This command is to reset the real-time pose of the Dobot, the issued command packet format is shown in Table 23, and the returned command packet format is shown in Table 24.

Table 23 The command packet of ResetPose

Header	Len	Payload						Checksum
		ID	Ctrl		Params			
			rw	isQueued				
0xAA 0xAA	2+9	11	1	0	uint8_t: manual	float: rearArm Angle	float: frontArm Angle	Payload checksum

Table 24 The returned command packet of ResetPose

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	11	1	0	Empty	Payload checksum

NOTE

When **manual** is **0**, the attitude is automatically reset without **rearArmAngle** and **frontArmAngle**; when **manual** is **1**, the **rearArmAngle** and the **frontArmAngle** are required.

1.4.3 Get PoseL

This command is to get the real-time pose of sliding rail, the issued command packet format is shown in Table 25, and the returned command packet format is shown in Table 26.

Table 25 The command packet of GetPoseL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	13	0	0	Empty	Payload checksum

Table 26 The returned command packet of GetPoseL

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	13	0	0	float：PoseL	Payload checksum

1.5 Alarm

1.5.1 Get Alarms State

This command is to get alarm status, the issued command packet format is shown in Table 27, and the returned command packet format is shown in Table 28.

Table 27 The command packet of GetAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	20	0	0	Empty	Payloadchecksum

Table 28 The returned command packet of GetAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	11	0	0	uint8_t[16]:alarmsState	Payload checksum

NOTE

Each byte in the array alarmsState identifies the alarm status of 8 alarm items, with the MSB in the high order while the LSB in the low order. Refer to Dobot ALARM document of detailed definition for each alarm bit.

1.5.2 Clear All Alarms State

This command is to clear alaem status, the issued command packet format is shown in Table 29, and the returned command packet format is shown in Table 30.

Table 29 The command packet of ClearAllAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	21	1	0	Empty	Payload checksum

Table 30 The returned command packet of ClearAllAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	21	1	0	Empty	Payload checksum

1.6 Homing Function

This part is homing function, including setting homing parameter, obtaining homing parameter, and setting homing position command. The default home position is (0°, 45°,45°, 0°).

1.6.1 Set/Get HOMEParams

- This command is to set homing position, the issued command packet format is shown in Table 31, and the returned command packet format is shown in Table 32.

Table 31 The command packet of SetHOMEParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	30	1	0 or 1	HOMEParams（See Program 2）	Payload checksum

Table 32 The returned command packet of SetHOMEParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	30	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get homing position, the issued command packet format is shown in Table 33, and the returned command packet format is shown in Table 34.

Table 33 The command packet of GetHOMEParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	30	0	0	Empty	Payload checksum

Table 34 The returned command packet of GetHOMEParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	30	0	0	HOMEParams（See Program 3）	Payload checksum

Program 3 HOMEParams definition

```
typedef struct tagHOMEParams {
        float x; Dobot coordinates X;
        float y; Dobot coordinates y;
        float z; Dobot coordinates z;
        float r; Dobot coordinates r;
} HOMEParams;
```

1.6.2 SetHOMECmd

This command is to execute the homing function, the issued command packet format is shown in Table 35, and the returned command packet format is shown in Table 36.

Table 35 The command packet of SetHOMECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	31	1	1 or 0	HOMECmd（See Program 4）	Payload checksum

Table 36 The returned command packet of SetHOMECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	31	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 4 SetHOMECmd definition

```
typedef struct tagHOMECmd {
uint32_t reserved;    // Reserved for future use
} HOMECmd;
```

1.6.3 Set/Get AutoLeveling

- This command is to set automatic leveling, the issued command packet format is shown in Table 37, and the returned command packet format is shown in Table 38.

Table 37 The command packet of SetAutoLeveling

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	30	1	1 or 0	AutoLeveling（see Program 5）	Payload checksum

Table 38 The returned command packet of SetAutoLeveling

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	30	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get automatic leveling result, the issued command packet format is shown in Table 39, and the returned command packet format is shown in Table 40.

Table 39 The command packet of GetAutoLeveling

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	30	0	0	float：AutoLevelingResult	Payload checksum

Table 40 The returned command packet of GetAutoLeveling

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	30	0	0	float：AutoLevelingResult	Payload checksum

Program 5 AutoLevelingParams definition

```
typedef struct tagAutoLevelingParams {
        uint8_t IsAutoleveling;
        float Accuracy;
} AutoLevelingParams;
```

1.7 Handhold Teaching

1.7.1 Set/Get HHTTrigMode

- This command is to set eh hand-hold teaching mode, the issued command packet format is shown in Table 41, and the returned command packet format is shown in Table 42.

Table 41 The command packet of Set/Get HHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	40	1	0	HHTTrigMode（See Program 6）	Payload checksum

Table 42 The returned command packet of Set/Get HHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	40	1	0	Empty	Payload checksum

- This command is to get the hand-hold teaching mode, the issued command packet format is shown Table 43, and the returned command packet format is shown in Table 44.

Table 43 The command packet of GetHHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	40	0	0	Empty	Payload checksum

Table 44 The returned command packet of GetHHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	40	1	0	HHTTrigMode（See Program 6）	Payload checksum

Program 6 HHTTrigMode definition

```
typedef enum tagHHTTrigMode {
TriggeredOnKeyReleased,          //Update when release the key
TriggeredOnPeriodicInterval    //Timed update
} HHTTrigMode;
```

1.7.2 Set/Get HHTTrigOutputEnabled

- This command is to set status of the hand-holding teaching function, the issued command packet format is shown in Table 45, and the returned command packet format is shown in Table 46.

Table 45 The command packet of SetHHTTrigOutputEnabled

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	41	1	0	uint8_t: isEnabled	Payload checksum

Table 46 The returned command packet of SetHHTTrigOutputEnabled

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	41	1	0	Empty	Payload checksum

- This command is to get the status of the hand-hold teaching, the issued command packet format is shown in Table 47, and the returned command packet format is shown in Table 48.

Table 47 The command packet of GetHHTTrigOutputEnabled

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	41	0	0	Empty	Payload checksum

Table 48 The returned command packet of GetHHTTrigOutputEnabled

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	41	0	0	uint8_t: isEnabled	Payload checksum

1.7.3 Get HHTTrigOutput

This command is to get the hand-hold teaching trigger single, the issued command packet format is shown in Table 49, and the returned command packet format is shown in Table 50.

Table 49 The command packet of GetHHTTrigOutput

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	42	0	0	Empty	Payload checksum

Table 50 The returned command packet of GetHHTTrigOutput

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	42	0	0	uint8_t: isTriggered	Payload checksum

1.8 EndEffector

1.8.1 Set/Get EndEffectorParams

- This command is to set the offset of the end-effector, the issued command packet format is shown in Table 51, and the returned command packet format is shown in Table 52.

Table 51 The command packet of SetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	60	1	1 or 0	EndEffectorParams（See Program 7）	Payload checksum

Table 52 The returned command packet of SetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	60	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the offset of the end-effector, the issued command packet format is shown in Table 53, and the returned command packet format is shown in Table 54.

Table 53 The command packet of GetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	60	0	0	Empty	Payload checksum

Table 54 The returned command packet of GetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	60	0	0	EndEffectorParams（See Program 7）	Payload checksum

Program 7 EndEffectorParams definition

```
typedef struct tagEndEffectorParams {
float xBias;EndEffector of X axis direction length;
float yBias; EndEffector of Y axis direction length;
float zBias; EndEffector of z axis direction length;
} EndEffectorParams;
```

1.8.2 Set/Get EndEffectorLaser

- This command is to set the status of the laser, the issued command packet format is shown in Table 55, and the returned command packet format is shown in Table 56.

Table 55 The command packet of SetEndEffectorLaser

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	61	1	1 or 0	uint8_t: enableCtrl	uint8_t: on	Payload checksum

Table 56 The returned command packet of SetEndEffectorLaser

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	61	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

⚠NOTICE

enableCtrl indicates whether the end-effector is enabled and **on** indicates whether the laser is enabled..

- This command is to get the status of the laser, the issued command packet format is shown in Table 57, and the returned command packet format is shown in Table 58.

Table 57 The command packet of GetEndEffectorLaser

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	61	0	0	Empty	Payload checksum

Table 58 The command packet of GetEndEffectorLaser

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	61	0	0	uint8_t: isCtrlEnabled	uint8_t: isOn	Payload checksum

1.8.3 Set/Get EndEffectorSuctionCup

- This command is to set the status of the air pump, the issued command packet format is shown in Table 59, and the returned command packet format is shown in Table 60.

Table 59 The command packet of SetEndEffectorSuctionCup

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	62	1	1 or 0	uint8_t: isCtrlEnabled	uint8_t: issucked	Payload checksum

Table 60 The returned command packet of SetEndEffectorSuctionCup

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	62	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

NOTICE

isCtrlEnabled indicates whether the end-effector is enabled and **issucked** indicates whether the suction cup is enabled..

- This command is to get the status of the air pump, the issued command packet format is shown in Table 61, and the returned command packet format is shown in Table 162.

Table 61 The command packet of GetEndEffectorSuctionCup

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	62	0	0	Empty	Payload checksum

Table 62 The returned command packet of GetEndEffectorSuctionCup

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	62	0	0	uint8_t: isCtrlEnable	uint8_t: issucked	Payload checksum

1.8.4 Set/Get EndEffectorGripper

- This command is to set the status of the gripper, the issued command packet format is shown in Table 63, and the returned command packet format is shown in Table 64.

Table 63 The command packet of EndEffector gripped or released

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	63	1	1 or 0	uint8_t: isCtrlEnable	uint8_t: isGripped	Payload checksum

Table 64 The returned command packet of EndEffector gripped or released

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	63	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

📖NOTE

isCtrlEnabled indicates whether the end-effector is enabled and **is Gripped** indicates whether the gripper is enabled..

- This command is to get the status of the gripper, the issued command packet format is shown in Table 65, and the returned command packet format is shown in Table 66.

Table 65 The command packet of GetEndEffectorGripper

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	63	0	0	Empty	Payload checksum

Table 66 The returned command packet of GetEndEffectorGripper

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	63	0	0	uint8_t: isCtrlEnable	uint8_t: isGripped	Payload checksum

1.9 JOG

Set / get parameters including joints, coordinate system parameters, jog public parameters and the execution of jog function.

1.9.1 Set/Get JOGJointParams

- This command is to set the velocity and acceleration of the joints coordinate axes in jogging mode, the issued command packet format is shown in Table 67, and the returned command packet format is shown in Table 68.

Table 67 The command packet of SetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	70	1	1 or 0	JOGJointParams（See Program 8）	Payload checksum

Table 68 The returned command packet of SetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	70	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

NOTE

In the teaching of the joint movement, we need to set the joint velocity and acceleration parameters. This command will set the velocity and acceleration of four joints.

- This command is to get the velocity and acceleration of the joints coordinate axes in jogging mode, the issued command packet format is shown in Table 69, and the returned command packet format is shown in Table 70.

Table 69 The command packet of GetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	70	0	0	Empty	Payload checksum

Table 70 The returned command packet of GetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	70	0	0	JOGJointParams（See Program 8）	Payload checksum

Program 8 JOGJointParams definition

```
typedef struct tagJOGJointParams{
float velocity[4];//Joint velocity of 4 axis
float acceleration[4]; //Joint acceleration of 4 axis
}JOGJointParams;
```

1.9.2 Set/Get JOGCoordinateParams

- This command is to set the velocity and acceleration of the Cartesian coordinate axes in jogging mode, the issued command packet format is shown in Table 71, and the returned command packet format is shown in Table 72.

Table 71 The command packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	1	1 or 0	JOGCoordinateParams（See Program 9）	Payload checksum

Table 72 The returned command packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	71	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

📖NOTE

This command sets the parameters of the coordinate system, which are the velocity and acceleration of the X, Y, Z and R axes, respectively.

- This command is to get the velocity and acceleration of the Cartesian coordinate axes in jogging mode, the issued command packet format is shown in Table 73, and the returned command packet format is shown in Table 74.

Table 73 The command packet of GetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	71	0	0	Empty	Payload checksum

Table 74 The returned command packet of GetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	0	0	JOGCoordinateParams（See Program 9）	Payload checksum

Program 9 JOGCoordinateParams definition

```
typedef struct tagJOGCoordinateParams {
float velocity[4];//Coornite velocity of 4 axis(x,y,z,r)
float acceleration[4];//Coordinate acceleration of 4 zxis(x,y,z,r)
} JOGCoordinateParams;
```

1.9.3 Set/Get JOGCommonParams

- This command is to set the velocity ratio and acceleration ratio of the sliding rail in jogging mode, the issued command packet format is shown in Table 75, and the returned command packet format is shown in Table 76.

Table 75 The command packet of SetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	1	1 or 0	JOGCommonParams（See Program 10）	Payload checksum

Table 76 The returned command packet of SetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	72	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration of the slidingrail in jogging mode, the issued command packet format is shown in Table 77, and the returned command packet format is shown in Table 78.

Table 77 The command packet of GetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	72	0	0	Empty	Payload checksum

Table 78 The returned command packet of GetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	0	0	JOGCommonParams （See Program 10）	Payload checksum

Program 10 JOGCommonParams definition

```
typedef struct tagJOGCommonParams {
float velocityRatio;//Velocity ratio,share joint jog and coordinated jog
float accelerationRatio; //Acceleration ratio,share joint jog and coordinated jog
} JOGCommonParams;
```

1.9.4 SetJOGCmd

This command is to execute the jogging command, the issued command packet format is shown in Table 79, and the returned command packet format is shown in Table 80.

Table 79 The command packet of SetJOGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	73	1	1 or 0	JOGCmd（See Program 11）	Payload checksum

Table 80 The returned command packet of SetJOGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	73	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 11 JOGCmd definition

```
typedef struct tagJOGCmd {
uint8_t isJoint;//Jog mode 0-coordinate jog   1-Joint jog
uint8_t cmd;//Jog command(Value range0~8)
}JOGCmd;
//The detailed description of JOGCmd
enum {
    IDEL,           //Void
    AP_DOWN,      // X+/Joint1+
    AN_DOWN,      // X-/Joint1-
    BP_DOWN,      // Y+/Joint2+
    BN_DOWN,      // Y-/Joint2-
    CP_DOWN,      // Z+/Joint3+
    CN_DOWN,      // Z-/Joint3-
    DP_DOWN,      // R+/Joint4+
    DN_DOWN       // R-/Joint4-
};
```

1.9.5 Set/Get JOGLParams

- This command is to set the velocity and acceleration of the sliding rail in jogging mode, the issued command packet format is shown in Table 81, and the returned command packet format is shown in Table 82.

Table 81 The command packet of Set/Get JOGLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	74	1	1 or 0	JOGLParams（See Program 12）	Payload checksum

Table 82 The returned command packet of Set/Get JOGLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	74	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration of the sliding rail in jogging mode, the issued command packet format is shown in Table 83, and the returned command packet format is shown in Table 84.

Table 83 The command packet of GetJOGLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	74	0	0	Empty	Payload checksum

Table 84 The returned command packet of GetJOGLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	74	0	0	JOGLParams（See Program 12）	Payload checksum

Program 12 JOGLParams definition

```
typedef struct tagJOGLParams{
float velocity;          // Joint velocity of JOGL
float acceleration;   // Joint acceleration of JOGL
}JOGLParams;
```

1.10 PTP

Playback commands are used for the relevant motion parameters setting and configuration, including joint parameters, coordinate system parameters, ratio parameters, and other related parameters.

1.10.1 Set/Get PTPJointParams

These commands are used to set and get the playback speed parameters, including the velocity and acceleration of joint coordinate axes. The speed set by this command is only applied to playback motion and does not work for the jogging movement.

- This command is to set the velocity and acceleration of the joint coordinate axes in PTP mode. The issued command packet format is shown in Table 85, and the returned command packet format is shown in Table 86.

Table 85 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	1	1 or 0	PTPJointParams（See Program 13）	Payload checksum

Table 86 The returned command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	80	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration of the joint coordinate axes in PTP mode, the issued command packet format is shown inTable 87， and the returned command packet format is shown in Table 88.

Table 87 The command packet of GetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	80	0	0	Empty	Payload checksum

Table 88 The returned command packet of GetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	0	0	PTPJointParams（See Program 13）	Payload checksum

Program 13 PTPJointParams definition

```
typedef struct tagPTPJointParams {
float velocity[4];          //In PTP mode, joint velocity of 4 axis
float acceleration[4];      // In PTP mode, joint acceleration of 4 axis
} PTPJointParams;
```

1.10.2 Set/Get PTPCoordinateParams

- This command is to set the velocity and acceleration of the Cartesian coordinate axes in PTP mode, the issued command packet format is shown in Table 89, and the returned command packet format is shown in Table 90.

Table 89 The command packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	1	1 or 0	PTPCoordinateParams （See Program 14）	Payload checksum

Table 90 The returned command packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	81	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration of the Cartesian coordinate axes in PTP mode, the issued command packet format is shown in Table 91, and the returned command packet format is shown in Table 92.

Table 91 The command packet of GetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	81	0	0	Empty	Payload checksum

Table 92 The returned command packet of GetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	0	0	PTPCoordinateParams （See Program 14）	Payload checksum

Program 14 PTPCoordinateParams definition

```
typedef struct tagPTPCoordinateParams {
float xyzVelocity;        //In PTP mode, coordinate velocity of xyz 3 axis
float rVelocity;          //In PTP mode, end-effector velocity
float xyzAcceleration;//In PTP mode, coordinate acceleration of xyz 3 axis
float rAccleration; // In PTP mode, end-effector acceleration
} PTPCoordinateParams;
```

1.10.3 Set/Get PTPJumpParams

- This command is to set the lifting height and maximum lifting height in JUMP mode, the issued command packet format is shown in Table 93, and the returned command packet format is shown in Table 94.

Table 93 The command packet of SetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	82	1	1 or 0	PTPJumpParams（See Program 15）	Payload checksum

Table 94 The returned command packet of SetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	82	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get This command is to get the lifting height and the maximum lifting height in JUMP mode, the issued command packet format is shown in Table 95, and the returned command packet format is shown in Table 96.

Table 95 The command packet of GetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	82	0	0	Empty	Payload checksum

Table 96 The returned command packet of GetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	82	0	0	PTPJumpParams（See Program 15）	Payload checksum

Program 15 PTPJumpParams definition

```
typedef struct tagPTPJumpParams {
float jumpHeight;          //Movement rising distance in Jump mode
float zLimit;     //Movement of the maximum rising height limitation in Jump mode
} PTPJumpParams;
```

1.10.4 Set/Get PTPCommonParams

- This command is to set the velocity ratio and the acceleration ratio in PTP mode, the issued command packet format is shown in Table 97, and the returned command packet format is shown in Table 98.

Table 97 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	1	1 or 0	PTPCommonParams（See Program 16）	Payload checksum

Table 98 The returned command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	83	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity ratio and the acceleration ratio in PTP mode, the issued command packet format is shown in Table 99, and the returned command packet format is shown in Table 100.

Table 99 The command packet of GetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	83	0	0	Empty	Payload checksum

Table 100 The returned command packet of GetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	0	0	PTPCommonParams（See Program 16）	Payload checksum

Program 16 PTPCommonParams definition

```
typedef struct tagPTPCommonParams {
float velocityRatio;          //Velocity ratio in PTP mode, share joint and coordinate mode
float accelerationRatio;      //Acceleration ratio in PTP mode, share joint and coordinate mode
} PTPCommonParams;
```

1.10.5 SetPTPCmd

This command is to execute PTP command, the issued command packet format is shown in Table 101, and the returned command packet format is shown in Table 102.

Table 101 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	84	1	1 or 0	PTPCmd（See Program 17）	Payload checksum

Table 102 The returned command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	84	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 17 PTPCmd definition

```
typedef struct tagPTPCmd {
uint8_t ptpMode;              //PTP mode (Value range 0~8)
float x;                      //x,y,z,r is the parameter of ptpMode, which can be set as the coordinates
                                //Joint angles ,coordinates orangle increments
float y;
float z;
float r;
} PTPCmd;
Among these,the value of ptpMode as follows:
enum {
    JUMP_XYZ,           // JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system
    MOVJ_XYZ,          // MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system
    MOVL_XYZ,          //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system
    JUMP_ANGLE,        // JUMP mode, (x,y,z,r) is the target point in Joint coordinate system
    MOVJ_ANGLE,        // MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system
    MOVL_ANGLE,        // MOVL mode, (x,y,z,r) is the target point in Joint coordinate system
    MOVJ_INC,          // MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system
    MOVL_INC,          // MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system
    MOVJ_XYZ_INC,   // MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
                        system
```

```
    JUMP_MOVL_XYZ, // JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
                   system
};
```

1.10.6 Set/Get PTPLParams

These commands are used to set and receive the playback speed parameters of rail, including the speed acceleration as well as the linear velocity and acceleration. The speed set by this command is only applied to playback motion and does not work for the teaching movement.

- This command is to set the velocity and acceleration of the sliding rail in PTP mode, the issued command packet format is shown in Table 103, and the returned command packet format is shown in Table 104.

Table 103 The command packet of SetPTPLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	85	1	1 or 0	PTPLParams（See Program 18）	Payload checksum

Table 104 The returned command packet of SetPTPLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	85	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration of the sliding rail in PTP mode, the issued command packet format is shown in Table 105, and the returned command packet format is shown in Table 106.

Table 105 The command packet of GetPTPLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	85	0	0	Empty	Payload checksum

Table 106 The returned command packet of GetPTPLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	85	0	0	PTPLParams（See Program 18）	Payload checksum

Program 18 PTPParams definition

```
typedef struct tagPTPLParams {
float velocity;          // In PTP mode, joint velocity of 4 axis
float acceleration;    // In PTP mode, joint acceleration of 4 axis
} PTPLParams;
```

1.10.7 SetPTPWithLCmd

This command is to execute PTP command with the sliding rail, the issued command packet format is shown in Table 107, and the returned command packet format is shown in Table 108.

Table 107 The command packet of SetPTPWithLCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+21	86	1	1 or 0	PTPWithLCmd（See Program 19）	Payload checksum

Table 108 The returned command packet of SetPTPWithLCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	86	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

Program 19 PTPWithLCmd definition

```
typedef struct tag PTPWithLCmd {
uint8_t ptpMode;          //PTP mode (Value range 0~9)
float x;                  //x,y,z,r is the parameter of ptpMode, as the coordinates
//Joint angle or coordinates/angle increments
float y;
float z;
float r;
float l;                  //The distance that sliding rail moves
```

```
} PTPWithLCmd;
Details for ptpMode:
enum {
JUMP_XYZ,        //JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVJ_XYZ,        //MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVL_XYZ,        //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system
JUMP_ANGLE,      //JUMP mode, (x,y,z,r) is the target point in Joint coordinate system
MOVJ_ANGLE,      //MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system
MOVL_ANGLE,      //MOVL mode, (x,y,z,r) is the target point in  Joint coordinate system
MOVJ_INC,        //MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system
MOVL_INC,        //MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system
MOVJ_XYZ_INC,    //MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
JUMP_MOVL_XYZ,   //JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
};
```

1.10.8 Set/Get PTPJump2Params

These two commands Set/Get PTPJump2Params divide the lifting height into the starting height and end height.

- This command is to set the extended parameters in JUMP mode, the issued command packet format is shown in Table 109, and the returned command packet format is shown in Table 110.

Table 109 The command packet of SetPTPJumpLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	87	1	1 or 0	PTPJump2Params（See Program 20）	Payload checksum

Table 110 The returned command packet of SetPTPJumpLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	87	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to set the extended parameters in JUMP mode, the issued command packet format is shown in Table 111, and the returned command packet format is shown in Table 112.

Table 111 The command packet of GetPTPJumpLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	87	0	0	Empty	Payload checksum

Table 112 The returned command packet of GetPTPJumpLParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	87	0	0	PTPJump2Params（See Program 20）	Payload checksum

Program 20 PTPJump2Params definition

```
typedef struct tagPTPJump2Params {
float startjumpHeight;//Movement of the starting lifting height in Jump mode
float endJumpHeight;  //Movement of the end lifting height in Jump mode
float zLimit; //Movement of the maximum lifting height in Jump mode
} PTPJump2Params;
```

1.10.9 SetPTPPOCmd

The command **SetPTPPOCmd** can manipulate I/O at a certain moment when running PTP command, rather than using EIO command to control the output state of I/O after the movement.

The issued command packet format is shown in Table 113, and the returned command packet format is shown in Table 114.

Table 113 The command packet of SetPTPPOCmd

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+17+4*n	88	1	1 or 0	PTPCmd（See Program 21）	POCmd（n）	Payload checksum

Table 114 The returned command packet of SetPTPPOCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	88	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 21 PTPCmd and POCmd definition

```
typedef struct tagPTPCmd {
uint8_t ptpMode;     //PTP mode (Value range 0~9)
float x;                //x,y,z,r is the parameter of ptpMode, as the coordinates
//Joint angle or coordinates/angle increments
float y;
float z;
float r;
} PTPCmd;
Details for ptpMode:
enum {
JUMP_XYZ,         //JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVJ_XYZ,          //MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVL_XYZ,          //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system
JUMP_ANGLE,        //JUMP mode, (x,y,z,r) is the target point in Joint coordinate system
MOVJ_ANGLE,        //MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system
MOVL_ANGLE,        //MOVL mode, (x,y,z,r) is the target point in   Joint coordinate system
MOVJ_INC,              //MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system
MOVL_INC,          //MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system
MOVJ_XYZ_INC,       //MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
JUMP_MOVL_XYZ,      //JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
};
Struct POCmd definition as follows：
Typedef struct tagPOCmd{
        Uint8_t ratio;      //Percentage of movement completion
        Uint16_t address //EIO number
        Uint8_t      level  //Output state
}POCmd；
```

1.10.10 SetPTPPOWithLCmd

When the Dobot run PTP movement with a sliding rail, this command SetPTPPOWithLCmd can manipulate I/O at a certain moment, rather than using EIO command to control the output state of IO after the movement.

The issued command packet format is shown in Table 115, and the returned command packet format is shown in Table 116.

Table 115 The command packet of SetPTPPOWithLCmd

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+21+4*n	89	1	1 or 0	PTPCmd（See Program 22）	POCmd（n）	Payload checksum

Table 116 The returned command packet of SetPTPPOWithLCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	89	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 22 PTPPOWithLCmd definition

```
typedef struct tagPTPCmd {
uint8_t ptpMode;     //PTP mode (Value range 0~9)
float x;                //x,y,z,r is the parameter of ptpMode, as the coordinates
//Joint angle or coordinates/angle increments
float y;
float z;
float r;
float l;                 //The distance that Sliding rail moves
} PTPCmd;
Details for ptpMode:
enum {
JUMP_XYZ,         //JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVJ_XYZ,          //MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system
MOVL_XYZ,          //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system
JUMP_ANGLE,        //JUMP mode, (x,y,z,r) is the target point in Joint coordinate system
MOVJ_ANGLE,        //MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system
MOVL_ANGLE,        //MOVL mode, (x,y,z,r) is the target point in   Joint coordinate system
MOVJ_INC,          //MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system
MOVL_INC,          //MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system
MOVJ_XYZ_INC,      //MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
JUMP_MOVL_XYZ,     //JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate
system
};

Struct POCmd definition as follows:
Typedef struct tagPOCmd{
    Uint8_t ratio；    //Percentage of movement completion
    Uint16_t address //EIO number
    Uint8_t    level  //Output state
}POCmd；
```

1.11 CP

The commands of continuous trajectory is used for motion setting and configuration related to continuous trajectory, which includes joint parameter, coordinate parameter, functional parameter and so on. The function is corresponded to Dobot CP, realizing the function of writing, drawing, laser engraving and other functions related to continuous trajectory.

1.11.1 Set/Get CPParams

This two commands are to set and get parameters of continuous trajectory, including planned acceleration, joint velocity and acceleration. This command is only available for continuous trajectory motion.

- This command is to get the motion parameters in CP mode. The issued command packet format is shown in Table 117, and the returned command packet format is shown in Table 118.

Table 117 The command packet of SetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	1	1 or 0	CPParams（See Program 23）	Payload checksum

Table 118 The returned command packet of SetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	90	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the velocity and acceleration in CP mode, the issued command packet format is shown in Table 119, and the returned command packet format is shown in Table 120.

Table 119 The command packet of GetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	90	0	0	Empty	Payload checksum

Table 120 The returned command packet of GetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	0	0	CPParams（See Program 23）	Payload checksum

Program 23 CPParams definition

```
typedef struct tagCPParams {
float planAcc;          // Maximum planned accelerations
float junctionVel;   // Maximum junction acceleration
union {
        float acc;       //Maximum actual acceleration,used in non-real-time mode
        float period;      //Interpolation cycle, used in real-time mode
};
uint8_t realTimeTrack;   //0: Non-real time mode; 1: Real time mode
} CPParams;
```

1.11.2 SetCPCmd

This command is to execute the CP command, the issued command packet format is shown in Table 121, and the returned command packet format is shown in Table 122.

Table 121 The command packet of SetCPCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	91	1	1 or 0	CPCmd（See Program 24）	Payload checksum

Table 122 The returned command packet of SetCPCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	91	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 24 CPCmd definition

```
typedef struct tagCPCmd {
  uint8_t cpMode;        //CP mode,   0: Relative mode   1：Absolute mode
```

```
  float x;                  //x-coordinate increment(Relative mode) / x-coordinate(Absolute mode)
  float y;                  //y-coordinate increment(Relative mode)/ y-coordinate(Absolute mode)
  float z;                 // z-coordinate increment(Relative mode) / z-coordinate(Absolute mode)
union {
float velocity;    // Reserved
float power;        //Laser power
    }
} CPCmd;
```

1.11.3 SetCPLECmd

This command is to execute the function of continuous path laser engraving commands, the issued command packet is shown in Table 123, and the returned command packet is shown in Table 124.

Table 123 The command packet of SetCPLECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	92	1	1 or 0	CPCmd（See Program 25）	Payload checksum

Table 124 The returned command packet of SetCPLECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	92	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 25 SetCPLECmd definition

```
typedef struct tagCPCmd {
    uint8_t cpMode;        //CP mode,  0: Relative mode  1：Absolute mode
    float x;                  //x-coordinate increment(Relative mode) / x-coordinate(Absolute mode)
    float y;                  //y-coordinate increment(Relative mode)/ y-coordinate(Absolute mode)
    float z;                 // z-coordinate increment(Relative mode) / z-coordinate(Absolute mode)
    union {ssss
        float velocity; //Reserved
        float power;    // Laser power 0~100
    }
} CPCmd;
} CPCmd;
```

1.12 ARC

1.12.1 Set/Get ARCParams

- This command is to set the velocity and acceleration in ARC mode, the issued command packet format is shown in Table 125, and the returned command packet format is shown in Table 126.

Table 125 The command packet of SetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	1	1 or 0	ARCParams（See Program 26）	Payload checksum

Table 126 The returned command packet of SetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	100	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to set the velocity and acceleration in ARC mode, the issued command packet format is shown in Table 127, and the returned command packet format is shown in Table 128.

Table 127 The command packet of GetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	100	0	0	Empty	Payload checksum

Table 128 The returned command packet of GetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	0	0	ARCParams（See Program 26）	Payload checksum

Program 26 ARCParams definition

```
typedef struct tagARCParams {
float xyzVelocity;          // Cartesian coordinate axis (X,Y,Z) velocity
```

```
float rVelocity;              // Cartesian coordinate axis (R) velocity
float xyzAcceleration;   // Cartesian coordinate axis (X,Y,Z) acceleration
flaot rAcceleration;      // Cartesian coordinate axis (R) acceleration
} ARCParams;
```

1.12.2 SetARCCmd

This command is to execute the ARC command, the issued command packet format is shown in Table 129, and the returned command packet format is shown in Table 130.

Table 129 The command packet of SetARCCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	101	1	1 or 0	ARCCmd（See Program 27）	Payload checksum

Table 130 The returned command packet of SetARCCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	101	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 27 ARCCmd definition

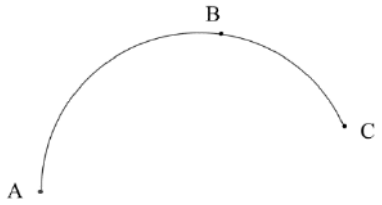
```
typedef struct tagARCCmd {
struct{
float x;
float y;
float z;
float r;
} cirPoint;        //Any circular point

struct {
float x;
float y;
float z;
float r;
} toPoint;             //Circular ending point
} ARCCmd;
```

📖NOTE

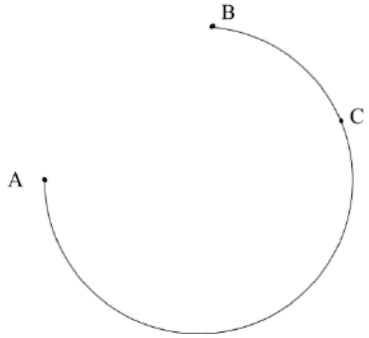
- arc track is the space of the arc, from the current point, any point on the arc and the end point of the arc together to determine the three points;
- The arc always passes from one point on the arc to the end point.

Circular trajectory shown as follows:

- A is the current point, B is any point on the arc, C is the end point.



- A is the current point, C is any point on the arc, B is the end point.



1.13 WAIT

1.13.1 SetWAITCmd

This comnd is to execute the waiting command, the issued command packet format is shown in Table 131, and the returned command packet format is shown in Table 132.

Table 131 The command packet of SetWAITCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	110	1	1 or 0	WAITCmd（See Program 28）	Payload checksum

Table 132 The returned command packet of SetWAITCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	110	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 28 SetWAITCmd definition

```
typedef struct tagWAITCmd {
    uint32_t timeout;          //Unit ms
} WAITCmd;
```

1.14 TRIG

1.14.1 SetTRIGCmd

This command is to execute the triggering command, the issued command packet format is shown in Table 133, and the returned command packet format is shown in Table 134.

Table 133 The command packet of SetTRIGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	120	1	1 or 0	TRIGCmd（See Program 29）	Payload checksum

Table 134 The returned command packet of SetTRIGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	120	1	1 or 0	uint64_t: queuedCmdIndex or Empty	Payload checksum

Program 29 SetTRIGCmd definition

```
typedef struct tagTRIGCmd {
uint8_t address;              // EIO address:1-20
uint8_t mode;                 //Triggering mode. 0: Level trigger.1:A/D trigger
uint8_t condition;            //Triggering condition Level: 0, equal. 1, unequal
                                A/D: 0, less than. 1,less than or equal
                                2, greater than or equal. 3, greater than
uint16_t threshold;           //Triggering threshold. Level : 0,1 .A/D: 0-4095
}TRIGCmd;
```

1.15 EIO

1.15.1 Set/Get IOMultiplexing

- This command is to set the I/O multiplexing, the issued command packet format is shown in Table 135, and the returned command packet format is shown in Table 136.

Table 135 The command packet of Set I/O Multiplexing

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	130	1	1 or 0	IOMultiplexing（See Program 30）	Payload checksum

Table 136 The returned command packet of Set I/O Multiplexing

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	130	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the I/O multiplexing, the issued command packet format is shown in Table 137, and the returned command packet format is shown in Table 138.

Table 137 The command packet of Get I/O Multiplexing

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	130	0	0	Empty	Payload checksum

Table 138 The returned command packet of Get I/O Multiplexing

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	130	0	0	IOMultiplexing（See Program 30）	Payload checksum

Program 30 IOMultiplexing definition

```
typedef struct tagIOMultiplexing {
uint8_t address;            //EIO addressing（Value range 1~20）
```

```
uint8_t multiplex;          //EIO function
} IOMultiplexing;
```

In which the values mutiplexsupported shown as in Program 31.

Program 31 IOFunction definition

```
typedef enum tagIOFunction {
IOFunctionDummy,      //Do not config function
IOFunctionPWM,        //PWM Output
IOFunctionDO,         //IO Output
IOFunctionDI,         //IO Output
IOFunctionADC,        //AD Input
} IOFunction;
```

1.15.2 Set/Get IODO

- This command is to set the I/O output, the issued command packet format is shown in Table 139 and the returned command packet format is shown in Table 140.

Table 139 The command packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	131	1	1 or 0	IODO（See Program 32）	Payload checksum

Table 140 The returned command packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	131	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the I/O output, the issued command packet format is shown in Table 141, and the returned command packet format is shown in Table 142.

Table 141 The command packet of GetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	131	0	0	Empty	Payload checksum

Table 142 The returned command packet of GetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	131	0	0	IODO（See Program 32）	Payload checksum

Program 32 IODO definition

```
typedef struct tagIODO {
uint8_t address;          //EIO addressing(Value range 1~20)
uint8_t level;            //Level output 0-Low level 1-High level
} IODO;
```

1.15.3 Set/Get IOPWM

- This command is to set the I/O PWM output（SetIOPWM），the issued command packet format is shown in Table 143, and the returned command packet format is shown in Table 144.

Table 143 The command packet of SetIOPWM

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+9	132	1	1 or 0	IOPWM（See Program 33）	Payload checksum

Table 144 The returned command packet of SetIOPWM

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	132	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the I/O PWM, the issued command packet format is shown in Table 145, and the returned command packet format is shown in Table 146.

Table 145 The command packet of GetIOPWM

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	132	0	0	Empty	Payload checksum

Table 146 The returned command packet of GetIOPWM

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+9	132	0	0	IOPWM（See Program 33）	Payload checksum

Program 33 IOPWM definition

```
typedef struct tagIOPWM {
uint8_t address;          //EIO addressing（Value range 1~20）
float frequency;          //PWM frequency 10HZ~1MHz
float dutyCycle;          //PWM duty ratio 0~100
} IOPWM;
```

1.15.4 Get IODI

This command is to get the I/O input, the issued command packet format is shown in Table 147, and the returned command packet format is shown in Table 147.

Table 147 The command packet of GetIODI

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	133	0	0	Empty	Payload checksum

Table 148 The returned command packet of GetIODI

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	133	0	0	IODI（See Program 34）	Payload checksum

Program 34 IODI definition

```
typedef struct tagIODI {
uint8_t address;          //EIO addressing(Value range 1~20)
uint8_t level;          //Input IO level 0-low level 1-high level
}IODI;
```

1.15.5 GetIOADC

This command is to get the A/D input, the issued command packet format is shown in Table 149, and the returned command packet format is shown in Table 150.

Table 149 The command packet of GetIOADC

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	134	0	0	Empty	Payload checksum

Table 150 The returned command packet of GetIOADC

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+3	134	0	0	IOADC（See Program 35）	Payload checksum

Program 35 IOADC definition

```
typedef struct tagIOADC{
uint8_t address;        //EIO addressing（Value range 1~20）
uint16_t value;         //Input value of ADC, range of 0~4095
}IOADC;
```

1.15.6 Set EMotor

This command is to set the velocity of the extended motor, the issued command packet format is shown in Table 151, and the returned command packet format is shown in Table 152.

Table 151 The command packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	135	1	1 or 0	EMotor（See Program 36）	Payload checksum

Table 152 The returned command packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	135	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

Program 36 EMotor definition

```
typedef struct tagEMotor{
    uint8_t index;          //Value range 0/1   0-Stepper1    1-Stepper2
    uint8_t insEnabled; //Motor control is enabled
    float speed;             //Motor control velocity(Number of pulse of per second)
}EMotor;
```

1.15.7 Set/Get ColorSensor

- This command is to enable the color sensor, the issued command packet format is shown in Table 153, and the returned command packet format is shown in Table 154.

Table 153 The command packet of SetColorSensor

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+3	137	1	1 or 0	Device ColorSense （See Program 37）	Payload checksum

Table 154 The returned command packet of SetColorSensor

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	137	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get the color sensor value, the issued command packet format is shown in Table 155, the returned command packet format is shown in Table 156.

Table 155 The command packet of GetColorSensor

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	137	0	0	Empty	Payload checksum

Table 156 The returned command packet of GetColorSensor

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+3	137	0	0	Color（See Program 38）	Payload checksum

Program 37 Device definition

```
Typedef enum tagVersion{
    VER_V1,
    VER_V2
}Version;
Typedef struct tagDevice{
    uint8_t isEnable;
    Port port;
    Version version;
}Device _attribute_((_packed_));
```

Program 38 Color definition

```
typedef struct tagColor {
    uint8_t r;
    uint8_t g;
    uint8_t b;
} Clolor;
```

1.15.8 Set/Get IRSwitch

- This command is to set infrared sensor, the issued command packet format is shown in Table 157, and the returned command packet format is shown in Table 158.

Table 157 The command packet of SetIRSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	138	1	1 or 0	Device IRSense (See Program 37)	Payload checksum

Table 158 The returned command packet of SetIRSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	138	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

- This command is to get infrared sensor, the issued command packet format is shown in Table 159, and the returned command packet format is shown in Table 160.

Table 159 The command packet of GetIRSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	138	0	0	Empty	Payload checksum

Table 160 The returned command packet of GetIRSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	138	0	0	Uint8_t state	Payload checksum

1.16 Calibration (CAL)

Angle sensors of forearm and rear arm may have a static offset due to angle sensor welding, machine status, and so on. We can get this static error by means of various means (such as leveling, compared with the standard source) and write it to the device through this API.

1.16.1 Set/Get AngleSensorStaticError

- This command is to set the angle sensor static error, the issued command packet format is shown in Table 161, and the returned command packet format is shown Table 162.

Table 161 The command packet of SetAngleSensorStaticError

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+8	140	1	0	float: rearArmAngleError	float: frontArmAngleError	Payload checksum

Table 162 The returned command packet of SetAngleSensorStaticError

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	140	1	0	Empty	Payload checksum

- This command is to get the angle sensor static error, the issued command packet format is shown Table 163, and the returned cpmmand packet format is shown Table 164.

Table 163 The command packet of GetAngleSensorStaticError

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	140	0	0	Empty	Payload checksum

Table 164 The returned command packet of GetAngleSensorStaticError

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+8	140	0	0	float: rearArmAngleError	float: frontArmAngleError	Payload checksum

1.17 WIFI

1.17.1 Set/Get WIFIConfigMode

- This command is to enable WIFI, the issued command packet format is shown in Table 165, and the returned command packet format is shown in Table 166.

Table 165 The command packet of SetWIFIConfigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	150	1	0	uint8_t: enable	Payload checksum

Table 166 The returned command packet of SetWIFIConfigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	150	1	0	Empty	Payload checksum

- This command is to get WIFI status, the issued command packet format is shown in Table 167, and the returned command packet format is shown in Table 168.

Table 167 The command packet of GetWIFIConfigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	150	0	0	Empty	Payload checksum

Table 168 The returned command packet of GetWIFIConfigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	150	0	0	uint8_t: enable	Payload checksum

1.17.2 Set/Get WIFISSID

- This command is to set SSID, the issued command packet format is shown in Table 169, and the returned command packet format is shown in Table 170.

Table 169 The command packet of SetWIFISSID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	151	1	0	char[n] ssid	Payload checksum

Table 170 The returned command packet of SetWIFISSID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	151	1	0	Empty	Payload checksum

- This command is to get SSID, the issued command packet format is shown in Table 171, and the returned command packet format is shown in Table 172.

Table 171 The command packet of GetWIFISSID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	151	0	0	Empty	Payload checksum

Table 172 The returned command packet of GetWIFISSID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	151	0	0	char[n] ssid	Payload checksum

1.17.3 Set/Get WIFIPassword

- This command is to set network password, the issued command packet format is shown in Table 173, and the returned command packet format is shown in Table 174.

Table 173 The command packet of SetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	152	1	0	char[n] password	Payload checksum

Table 174 The returned command packet of SetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	152	1	0	Empty	Payload checksum

- This command is to get network password, the issued command packet format is shown in Table 175, and the returned command packet format is shown in Table 176.

Table 175 The command packet of GetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	152	0	0	Empty	Payload checksum

Table 176 The returned command packet of GetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+n	152	0	0	char[n] password	Payload checksum

1.17.4 Set/Get WIFIIPAddress

- This command is to set IP address, the issued command packet format is shown in Table 177, and the returned command packet format is shown in Table 178.

Table 177 The command packet of setting IP

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	153	1	0	WIFIIPAddress（See Program 39）	Payload checksum

Table 178 The command packet of setting returned IP

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	153	1	0	Empty	Payload checksum

- This command is to get IP address, the issued command packet format is shown in Table 179, and the returned command packet format is shown in Table 180.

Table 179 The command packet of GetWIFIIPAddress

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	153	0	0	Empty	Payload checksum

Table 180 The command packet of GetWIFIIPAddress

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	153	0	0	WIFIIPAddress（See Program 39）	Payload checksum

Program 39 WIFIIPAddress definition

```
typedef struct tagWIFIIPAddress {
uint8_t dhcp;
uint8_t addr[4];
} WIFIIPAddress;
```

1.17.5 Set/Get WIFINetmask

- This command is to set netmask, the issued command packet format is shown in Table 181, and the returned command packet format is shown in Table 182.

Table 181 The command packet of SetWIFINetmask

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	154	1	0	WIFINetmask（See Program 40）	Payload checksum

Table 182 The returned command packet of SetWIFINetmask

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	154	1	0	Empty	Payload checksum

- This command is to get netmask, the the issued command packet format is shown in Table 183, and the returned command packet format is shown in Table 184.

Table 183 The command packet of GetWIFINetmask

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	154	0	0	Empty	Payload checksum

Table 184 The returned command packet of GetWIFINetmask

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	154	0	0	WIFINetmask（See Program 40）	Payload checksum

Program 40 WIFINetmask definition

```
typedef struct tagWIFINetmask {
```

```
uint8_t addr[4];
} WIFINetmask;
```

1.17.6 Set/Get WIFIGateway

- This command is to set gateway, the issued command packet format is shown in Table 185, and the returned command packet format is shown in Table 186.

Table 185 The command packet of SetWIFIGateway

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	155	1	0	WIFIGateway（See Program 41）	Payload checksum

Table 186 The returned command packet of SetWIFIGateway

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	155	1	0	Empty	Payload checksum

- This command is to get gateway, the issued command packet format is shown in Table 187, and the returned command packet format is shown in Table 188.

Table 187 The command packet of GetWIFIGateway

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	155	0	0	Empty	Payload checksum

Table 188 The returned command packet of GetWIFIGateway

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	155	0	0	WIFIGateway（See Program 41）	Payload checksum

Program 41 WIFIGateway definition

```
typedef struct tagWIFIGateway {
```

```
uint8_t addr[4];
} WIFIGateway;
```

1.17.7 Set/Get WIFIDNS

- This command is to set DNS, the issued command packet format is shown in Table 189, and the returned command packet format is shown in Table 190.

Table 189 The command packet of SetWIFIDNS

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	156	1	0	WIFIDNS（See Program 42）	Payload checksum

Table 190 The returned command packet of SetWIFIDNS

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	156	1	0	Empty	Payload checksum

- This command is to get DNS, the issued command packet format is shown in Table 191, and the returned command packet format is shown in Table 192.

Table 191 The command packet of GetWIFIDNS

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	156	0	0	Empty	Payload checksum

Table 192 The returned command packet of GetWIFIDNS

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	156	0	0	WIFIDNS（See Program 42）	Payload checksum

Program 42 WIFIDNS definition

```
typedef struct tagWIFIDNS {
```

```
uint8_t addr[4];
} WIFIDNS;
```

1.17.8 GetWIFIConnectStatus

This command is to get WIFI connection status, the issued command packet format is shown in Table 193, and the returned command packet format is shown in Table 194.

Table 193 The command packet of GetWIFIConnectStatus

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	157	0	0	Empty	Payload checksum

Table 194 The returned command packet of GetWIFIConnectStatus

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	157	0	0	uint8_t: isConnected	Payload checksum

1.18 Losing-Step Detection

1.18.1 SetLostStepParams

This command is to set losing-step threshold, the issued command packet format is shown in Table 195, and the returned command packet format is shown in Table 196.

Table 195 The command packet of SetLostStepParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	170	1	0	float: value	Payload checksum

Table 196 The returned command packet of SetLostStepParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	170	1	0	Empty	Payload checksum

1.18.2 SetLostStepCmd

This command is to execute losing-step, the issued command packet format is shown in Table 197, and the returned command packet format is shown in Table 198.

Table 197 The command packet of of SetLostStepCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	171	1	1 or 0	Empty	Payload checksum

Table 198 The returned command packet of SetLostStepCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	171	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum

1.19 Queued execution control commands

Queued execution control commands are used to set related parameters of the queue command execution, including the command execution mode (online / offline), the current state of the queue command buffer, the execution status of the queue command (TRUE / FALSE), the queue command execution control (START / PAUSE / STOP).

1.19.1 Set QueuedCmdStartExec

This command is to start in command queue, the issued command packet format is shown in Table 199, and the returned command packet format is shown in Table 200.

Table 199 The command packet of SetQueuedCmdStartExec

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	240	1	0	Empty	Payload checksum

Table 200 The returned command packet of SetQueuedCmdStartExec

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	240	1	0	Empty	Payload checksum

1.19.2 Set QueuedCmdStopExec

This command is to stop in command queue, the issued command packet format is shown in Table 201, and the returned command packet format is shown in Table 202.

Table 201 The command packet of SetQueuedCmdStopExec

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	241	1	0	Empty	Payload checksum

Table 202 The returned command packet of SetQueuedCmdStopExec

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	241	1	0	Empty	Payload checksum

1.19.3 Set QueuedCmdForceStopExec

This command is to stop in command queue forcedly, the issued command packet format is shown in Table 203, and the returned command packet format is shown in Table 204.

Table 203 The command packet of SetQueuedCmdForceStopExec,

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	242	1	0	Empty	Payload checksum

Table 204 The returned command packet of SetQueuedCmdForceStopExec,

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	242	1	0	Empty	Payload checksum

1.19.4 Set QueuedCmdStartDownload

This command is to download something, the issued command packet format is shown in Table 205, and the returned command packet format is shown in Table 206.

Table 205 The command packet of SetQueuedCmdStartDownload

Header	Len	Payload					Checksum
		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+8	243	1	0	uint32_t: totalLoop	uint32: linePerLoop	Payload checksum

Table 206 The returned command packet of SetQueuedCmdStartDownload

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	243	1	0	Empty	Payload checksum

NOTE

Dobot controller supports storing commands in the external Flash of the controller, which can then be executed by pressing the keys on the controller, that is, offline function.

1.19.5 Set QueuedCmdStopDownload

This command is to stop to download something, the issued command packet format is shown in Table 207, and the returned command packet format is shown in Table 208.

Table 207 The command packet of SetQueuedCmdStopDownload

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	244	1	0	Empty	Payload checksum

Table 208 The returned command packet of SetQueuedCmdStopDownload

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	244	1	0	Empty	Payload checksum

1.19.6 Set QueuedCmdClear

This command is to clear command queue, the issued command packet format is shown in Table 209, and the returned command packet format is shown in Table 210.

Table 209 The command packet of SetQueuedCmdClear

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	245	1	0	Empty	Payload checksum

Table 210 The returned command packet of SetQueuedCmdClear

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	245	1	0	Empty	Payload checksum

1.19.7 Get QueuedCmdCurrentIndex

This command is to get command index, the issued command packet format is shown in Table 211, and the returned command packet format is shown in Table 212.

Table 211 The command packet of GetQueuedCmdCurrentIndex

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	246	0	0	Empty	Payload checksum

Table 212 The returned command packet of GetQueuedCmdCurrentIndex

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	246	0	0	uint64_t: queuedCmdCurrentIndex	Payload checksum

NOTE

There is a 64-bit internal count index in Dobot controller command queue mechanism. The counter is automatically incremented after the controller executes a command. By this internal index, you can check how many queue commands the controller have been executed, and which command is executing currently.