Dobot Communication Protocol

Dobot Magician

TN01010101 V1.0.4 Date: 2016/12/20

Shenzhen Yuejiang Technology Co.,Ltd

Revised History

Version	Date	Reason
V1.0.0	2016/09/22	Create a document
V1.0.1	2016/11/18	Add description of communication parameters
V1.0.2	2016/11/21	Add BLE Reading & Writing Characteristic UUID
V1.0.3	2016/11/30	Add description of movement control (JOG PTP CP ARC)
V1.0.4	2016/12/20	Get the remaining space of instruction quene

Contents

1. Application Scope

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

2. Communication Protocol

2.1 Communication Parameters

1. USB to serial port

- Baud rate: 115200bps;
- Data bits:8-Bit;
- Stop bit: 1-Bit;
- Parity bit: Void.

2. Wi-Fi

- IP: Route and other distribution;
- COM port: 8899

3. 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.

4. TTL

- Baud Rate: 115200bps;
- Data Bits:8-Bit;
- Stop bit: 1-Bit;
- Parity bit: Void.

2.2 Protocol Introduction

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

The physical layer receives 8bite raw datas each time, which need make sure starting, ending and verifying the accuracy of data by setting up communication protocols. And the communication protocol includes packet header, packet load, checksum to make sure transferring the data accuarately.

2.2.1 Protocol Features

Dobot's communication protocol features are as follows:

1. Agreement instruction is not fixed length;

2. Protocol instruction consists of packet header, payload frame, payload frame, and check.

3. All communications are initiated by the host initiative, and for all communications instructions, the next crew to return (both read and write); for the queue instruction, which returns with 64-bit execution index value;

4. Instructions are divided into immediate instructions and queue instructions. The immediate instruction will be executed immediately, while the queue instruction will be placed in the lower machine queue for serial execution; all read operations are immediate instructions; for write (or set) operation, movement type of instruction should be the queue Commands (such as home, JOG, PTP,

etc.), set the parameters of the instruction can be not only immediate instructions also a queue instruction;

5. Before sending the queue command to the lower computer, the host should inquire the remaining space of the command queue of the lower computer (check once and send multiple commands);

6. The immediate instruction is always executed immediately; the completion of the execution of the queue instruction can be got from index by checking the quene command being executed and comparation with the index of the queue command (returned in the command mentioned in point 3) ;

7. The parameters in the command use little endian mode.

2.2.2 Checksum Calculation

In Dobot Magician's communication protocol, the send end checksum is calculated as follows:

1. Add all the contents of the Payload byte by byte (8 bits) to get a result R (8 bits);
2. The result R (8 bits) two complement, check byte.

2's complement. For an N-bit number, the second complement is equal to 2 ^ N minus the number. In this protocol, assuming that the result R is 0x0A, 2'complement, i.e., the result of the above check, 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 is:

1. The payload frame (Payload) in accordance with the contents of all bytes (8) by the byte-by-add to get a result A;

2. Result A is added to the check byte. If it is equal to 0, the checksum is correct.

2.2.3 The Protocol Classfication

It can be divided into the following parts according to 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, the communication protocol function ID is divided into following items shown in Figure 1:

Figure 1 Classification of functional items

Classification of functional items	Function ID area	Avaliabl 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

2.2.4 Other Explanations

1. An ID description is provided in each of the instruction descriptions below;
2. Ctrl in the following bytes, rw to Ctrl byte 0, isQueued Ctrl to the first byte.

2.3 Device Information

This command is used to set the device SN number, device name, and device version number. You can use the command to read current information of the device.

2.3.1 Set/Get DeviceSN

3. SetDeviceSN, the issued command package is shown as Figure 2 and the returned command package is in Figure 3;

Figure 2 The command package of Device SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	0	1	0	char* DeviceSN	Payload checksum

Figure 3 The retirned command package of SetDevice SN

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

4. GetDeviceSN, the issued command package is shown as Figure 4 and the returned command package is in Figure 5;

Figure 4 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

Figure 5 The returned command package of GetDevice SN

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	0	0	0	char* DeviceSN	Payload checksum

2.3.2 Set/Get DeviceName

1. SetDeviceName, the issued command package is shown as Figure 6 and the returned command package is in Figure 7;

Figure 6 The instruction packet of SetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	1	1	0	char* DeviceName	Payload checksum

Figure 7 The returned instruction packet of SetDeviceName

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

2. GetDeviceName, the issued instruction packet format is shown in Figure 8, and the returned instruction packet format is shown in Figure 9.

Figure 8 The instruction packet of SetDeviceName

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

Figure 9 The returned instruction packet of GetDeviceName

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	1	0	0	char* DeviceName	Payload checksum

2.3.3 GetDeviceVersion

GetDeviceVersion, the issued instruction packet format is shown in Figure 10, and the returned instruction packet format is shown in Figure 11.

Figure 10 The instruction packet of GetDeviceVersion

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

Figure 11 The returned instruction 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

2.4 Pose

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

2.4.1 GetPose

GetPose, the issued instruction packet format is shown in Figure 12, and the returned instruction packet format is shown in Figure 13.

Figure 12 The instruction packet of GetPose

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

Figure 13 The returned instruction packet of GetPose

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	10	0	0	Pose（See ProgramProgram 1）	Payload checksum

Program 1 Pose Definition

```
typedef struct tagPose {
    float x;                //Robotic arm coordinate system x
    float y;                //Robotic arm coordinate system y
```

```
            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;
```

2.4.2 ResetPose

ResetPose, the issued instruction packet format is shown in Figure 14, and the returned instruction packet format is shown in Figure 15.

Figure 14 The instruction 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

Figure 15 The returned instruction 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 incoming rearArmAngle and frontArmAngle; when manual is 1, the rearArmAngle and frontArmAngle are incoming.

2.5 Alarm

2.5.1 GetAlarmsState

GetAlarmsState, the issued instruction packet format is shown in Figure 16, and the returned instruction packet format is shown in Figure 17.

Figure 16 The instruction packet of GetAlarmsState

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

Figure 17 The returned instruction 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

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.

2.5.2 ClearAllAlarmsState

ClearAllAlarmsState, the issued instruction packet format is shown in Figure 18, and the returned instruction packet format is shown in Figure 19.

Figure 18 The instruction packet of ClearAllAlarmsState

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

Figure 19 The returned instruction packet of ClearAllAlarmsState

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

2.6 Home

This part is Home function, including setting Hoem parameter, obtaining Hoem parameter, and setting Home position command.The default home position is corresponded into the coordinate values of 0 °, 45 °,45 °, 0 °, and users and call SetHOMEParams to reset Home coordinate according to personal requirement. After executing Home command, Dobot will move to home resetting.

Notice: In the process, Dobot indicator is in blue flashing, fine-tuning when moving in the near place of Home position, the buzzer will ring and the indicator is in green, then it hints that the execution of back to zero is successful.

2.6.1 Set/Get HOMEParams

1. SetHOMEParams, the issued instruction packet format is shown in Figure 20, and the returned instruction packet format is shown in Figure 21;

Figure 20 The instruction 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

Figure 21 The returned instruction packet of SetHOMEParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	30	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetHOMEParams, the issued instruction packet format is shown in Figure 22, and the returned instruction packet format is shown in Figure 23.

Figure 22 The instruction packet of GetHOMEParams

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

Figure 23 The returned instruction packet of GetHOMEParams

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

Program 2 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;
```

2.6.2 SetHOMECmd

SetHOMECmd, the issued instruction packet format is shown in Figure 24, and the returned instruction packet format is shown in Figure 25.

Figure 24 The instruction packet of SetHOMECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	31	1	1	HOMECmd（See Program 3）	Payload checksum

Figure 25 The returned instruction packet of SetHOMECmd

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

Program 3 HOMECmd Definition

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

2.7 Handhold Teaching

Handhold teaching instruction, for configuration of related commands and obtained information, including enabling / disabling of hand-held teaching mode, access to handheld teaching of enabled information and access to obtain a new increased point.

2.7.1 Set/Get HHTTrigMode

1. SetHHTTrigMode, the issued instruction packet format is shown in Figure 26, and the returned instruction packet format is shown in Figure 27;

Figure 26 The instruction packet of Set/Get HHTTrigMode

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

Figure 27 The returned instruction packet of Set/Get HHTTrigMode

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

2. GetHHTTrigMode, the issued instruction packet format is shown in Figure 28, and the returned instruction packet format is shown in Figure 29.

Figure 28 The instruction packet of GetHHTTrigMode

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

Figure 29 The returned instruction packet of GetHHTTrigMode

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

Program 4 HHTTrigMode 的 Definition

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

2.7.2 Set/Get HHTTrigOutputEnabled

1. SetHHTTrigOutputEnabled, the issued instruction packet format is shown in Figure 30, and the returned instruction packet format is shown in Figure 31;

Figure 30 The instruction packet of SetHHTTrigOutputEnabled

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

Figure 31 The returned instruction packet of SetHHTTrigOutputEnabled

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

2. GetHHTTrigOutputEnabled, the issued instruction packet format is shown in Figure 32, and the returned instruction packet format is shown in Figure 33.

Figure 32 The instruction packet of GetHHTTrigOutputEnabled

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

Figure 33 The returned instruction packet of GetHHTTrigOutputEnabled

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+1	41	0	0	uint8_t: isEnabled	Payload checksum

2.7.3 GetHHTTrigOutput

GetHHTTrigOutput, the issued instruction packet format is shown in Figure 34, and the returned instruction packet format is shown in Figure 35.

Figure 34 The instruction packet of GetHHTTrigOutput

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

Figure 35 The returned instruction packet of GetHHTTrigOutput

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

2.8 ArmOrientation

2.8.1 Set/Get ArmOrientation

Note: This command is currently only applicable to SCARA models.

1. SetArmOrientation, the issued instruction packet format is shown in Figure 36, and the returned instruction packet format is shown in Figure 37;

Figure 36 The instruction packet of SetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	50	1	0 or 1	ArmOrientation（See Program 5）	Payload checksum

Figure 37 The returned instruction packet of SetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	50	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetArmOrientation, the issued instruction packet format is shown in Figure 38, and the returned instruction packet format is shown in Figure 39.

Figure 38 The instruction packet of GetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	50	0	0	Empty	Payload checksum

Figure 39 The returned instruction packet of GetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	50	0	0	ArmOrientation（See Program 5）	Payload checksum

Program 5 ArmOrientation Definition

```
typedef enum tagArmOrientation {
    LeftyArmOrientation,
    RightyArmOrientation
} ArmOrientation;
```

2.9 EndEffector

2.9.1 Set/Get EndEffectorParams

1. SetEndEffectorParams, the issued instruction packet format is shown in Figure 40, and the returned instruction packet format is shown in Figure 41.

Figure 40 The instruction packet of SetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	60	1	0 or 1	EndEffectorParams（See Program 6）	Payload checksum

Figure 41 The returned instruction packet of SetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	60	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetEndEffectorParams, the issued instruction packet format is shown in Figure 42, and the returned instruction packet format is shown in Figure 43.

Figure 42 The instruction packet of GetEndEffectorParams

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

Figure 43 The returned instruction packet of GetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	60	0	0	EndEffectorParams（See Program 6）	Payload checksum

Program 6 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;
```

2.9.2 Set/Get EndEffectorLaser

1. SetEndEffectorLaser, the issued instruction packet format is shown in Figure 44, and the returned instruction packet format is shown in Figure 45;

Figure 44 The instruction packet of SetEndEffectorLaser

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

Figure 45 The returned instruction packet of SetEndEffectorLaser

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	61	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Notice: If the controlling is enabled, the laser is On.

2. GetEndEffectorLaser, the issued instruction packet format is shown in Figure 46, and the returned instruction packet format is shown in Figure 47.

Figure 46 The instruction packet of GetEndEffectorLaser

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

Figure 47 The instruction 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

Notice: If the controlling is enabled, the laser is On. (isCtrlEnabled） , laser（isOn）。

2.9.3 Set/Get EndEffectorSuctionCup

1. SetEndEffectorSuctionCup, the issued instruction packet format is shown in Figure 48, and the returned instruction packet format is shown in Figure 49;

Figure 48 he instruction packet of SetEndEffectorSuctionCup

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

Figure 49 The returned instruction packet of SetEndEffectorSuctionCup

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	62	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Notice: The controlling (isCtrlEnabled） Suction cup（isSucked）

2. GetEndEffectorSuctionCup, the issued instruction packet format is shown in Figure 50, and the returned instruction packet format is shown in Figure 51.

Figure 50 The instruction packet of GetEndEffectorSuctionCup

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+0	62	0	0	Empty	Payload checksum

Figure 51 The returned instruction packet of GetEndEffectorSuctionCup

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

Notice: The controlling (isCtrlEnabled） Suction cup（isSucked）

2.9.4 Set/Get EndEffectorGripper

1. SetEndEffectorGripper is gripped or released, the issued instruction packet format is shown in Figure 52, and the returned instruction packet format is shown in Figure 53;

Figure 52 The instruction packet of EndEffector gripped or released

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

Figure 53 The returned instruction packet of EndEffector gripped or released

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	63	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Note: isCtrlEnabled or isGripped.

2. SetEndEffectorGripper, the issued instruction packet format is shown in Figure 54, and the returned instruction packet format is shown in Figure 55.

Figure 54 The instruction packet of SetEndEffectorGripper

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

Figure 55 The returned instruction packet of SetEndEffectorGripper

Header	Len	Payload	Checksum

		ID	Ctrl		Params		
			rw	isQueued			
0xAA 0xAA	2+2	63	0	0	uint8_t: isCtrlEnable	uint8_t: isGriped	Payload checksum

Note: isCtrlEnabled or isGripped

2.10 JOG

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

2.10.1 Set/Get JOGJointParams

1. SetJOGJointParams, the issued instruction packet format is shown in Figure 56, and the returned instruction packet format is shown in Figure 57;

Figure 56 The instruction packet of SetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	70	1	0 or 1	JOGJointParams（See Program 7）	Payload checksum

Figure 57 The returned instruction packet of SetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	70	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Note: In the teaching of the joint movement, we need to set the joint speed and acceleration parameters, this group of instructions related to the command need to be set in advance when in the joint movement. The command will set the speed and acceleration of four joints.

2. GetJOGJointParams, the issued instruction packet format is shown in Figure 58, and the returned instruction packet format is shown in Figure 59.

Figure 58 The instruction packet of GetJOGJointParams

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

Figure 59 The returned instruction packet of GetJOGJointParams

Header	Len	Payload	Checksum

		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	70	0	0	JOGJointParams（See Program 7）	Payload checksum

Program 7 JOGJointParams Definition

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

2.10.2 Set/Get JOGCoordinateParams

1. SetJOGCoordinateParams, the issued instruction packet format is shown in Figure 60, and the returned instruction packet format is shown in Figure 61;

Figure 60 The instruction packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	1	0 or 1	JOGCoordinateParams（See Program 8）	Payload checksum

Figure 61 The returned instruction packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	71	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Note: The difference between this command and parameter command of single joint movement is that this command sets the parameters of the coordinate system, which are the speed and acceleration of the X, Y, Z and R axes, respectively.

2. GetJOGCoordinateParams, the issued instruction packet format is shown in Figure 62, and the returned instruction packet format is shown in Figure 63.

Figure 62 The instruction packet of GetJOGCoordinateParams

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

Figure 63 The returned instruction packet of GetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	0	0	JOGCoordinateParams （See Program 8）	Payload checksum

Program 8 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;
```

2.10.3 Set/Get JOGCommonParams

1. SetJOGCommonParams, the issued instruction packet format is shown in Figure 64, and the returned instruction packet format is shown in Figure 65;

Figure 64 The instruction packet of SetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	1	0 or 1	JOGCommonParams （See Program 9）	Payload checksum

Figure 65 The returned instruction packet of SetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	72	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetJOGCommonParams, the issued instruction packet format is shown in Figure 66, and the returned instruction packet format is shown in Figure 67.

Figure 66 The instruction packet of GetJOGCommonParams

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

Figure 67 The returned instruction packet of GetJOGCommonParams

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+8	72	0	0	JOGCommonParams （See Program 9）	Payload checksum

Program 9 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;
```

2.10.4 SetJOGCmd

SetJOGCmd, the issued instruction packet format is shown in Figure 68, and the returned instruction packet format is shown in Figure 69.

Figure 68 The instruction packet of SetJOGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	73	1	1	JOGCmd（See Program 10）	Payload checksum

Figure 69 The returned instruction packet of SetJOGCmd

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

Program 10 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-
};
```

2.11 PTP

Playback function, for playback the relevant motion setting and configuration. These include joint parameters, coordinate system parameters, scale parameters, and other related parameters.。

2.11.1 Set/Get PTPJointParams

These commands are used to set and receive the playback speed parameters, including the speed acceleration of a single joint 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.

1. SetPTPJointParams, used for controlling the speed of playback, which can achieve the fast or slow movement. The issued instruction packet format is shown in Figure 70, and the returned instruction packet format is shown in Figure 71;

Figure 70 The instruction packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	1	0 or 1	PTPJointParams（See Program 11）	Payload checksum

Figure 71 The returned instruction packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenghr	80	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetPTPJointParams, the issued instruction packet format is shown in Figure 72, and the returned instruction packet format is shown in Figure 73.

Figure 72 The instruction packet of GetPTPJointParams

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

Figure 73 The returned instruction packet of GetPTPJointParams

Header	Len	Payload	Checksum

		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	0	0	PTPJointParams（See Program 11）	Payload checksum

Program 11 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;
```

2.11.2 Set/Get PTPCoordinateParams

1. SetPTPCoordinateParams, the issued instruction packet format is shown in Figure 74, and the returned instruction packet format is shown in Figure 75;

Figure 74 The instruction packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	1	0 or 1	PTPCoordinateParams（See Program 12）	Payload checksum

Figure 75 The returned instruction packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	81	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetPTPCoordinateParams, the issued instruction packet format is shown in Figure 76, and the returned instruction packet format is shown in Figure 77.

Figure 76 The instruction packet of GetPTPCoordinateParams

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

Figure 77 The returned instruction packet of GetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+16	81	0	0	PTPCoordinateParams（See Program 12）	Payload checksum

Program 12 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;
```

2.11.3 Set/Get PTPJumpParams

1. SetPTPJumpParams, the issued instruction packet format is shown in Figure 78, and the returned instruction packet format is shown in Figure 79;

Figure 78 The instruction packet of SetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	82	1	0 or 1	PTPJumpParams（See Program 13）	Payload checksum

Figure 79 The returned instruction packet of SetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	82	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetPTPJumpParams, the issued instruction packet format is shown in Figure 80, and the returned instruction packet format is shown in Figure 81.

Figure 80 The instruction packet of GetPTPJumpParams

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

Figure 81 The returned instruction packet of GetPTPJumpParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+8	82	0	0	PTPJumpParams（See Program 13）	Payload checksum

Program 13 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;
```

2.11.4 Set/Get PTPCommonParams

1. SetPTPJointParams, the issued instruction packet format is shown in Figure 82, and the returned instruction packet format is shown in Figure 83;

Figure 82 The instruction packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	1	0 or 1	PTPCommonParams （See Program 14）	Payload checksum

Figure 83 The returned instruction packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	83	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetPTPJointParams, the issued instruction packet format is shown in Figure 84, and the returned instruction packet format is shown in Figure 85.

Figure 84 The instruction packet of GetPTPJointParams

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

Figure 85 The returned instruction packet of GetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	0	0	PTPCommonParams （See Program 14）	Payload checksum

Program 14 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;
```

2.11.5 SetPTPCmd

SetPTPJointParams, the issued instruction packet format is shown in Figure 86, and the returned instruction packet format is shown in Figure 87.

Figure 86 The instruction packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	84	1	1	PTPCmd（See Program 15）	Payload checksum

Figure 87 The returned instruction packet of SetPTPJointParams

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

Program 15 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, as the coordinates
//Joint angle or coordinates/angle increments
        float y;
        float z;
        float r;
    } PTPCmd;
    Among these,the value of ptpMode as follows:
    enum {
        JUMP_XYZ, //Jump mode, the parameters for the target point coordinates
        MOVJ_XYZ,           //Joint movement, the parameters for the target point coordinates
        MOVL_XYZ,           //Liner movement, the parameters for the target point coordinates
        JUMP_ANGLE,        // Jump mode, the parameters for the target point coordinates
```

```
            MOVJ_ANGLE,        // Joint movement, the parameters for the target point coordinates
            MOVL_ANGLE,        // Liner movement, the parameters for the target point coordinates
            MOVJ_INC,          // Joint movement increment mode, the parameter is for the target
point joint angle increment
            MOVL_INC,          // Liner movement increment mode, the parameter is the target
point joint angle increment
            MOVJ_XYZ_INC,    // Joint movement increment mode, the parameter is the target point
joint angle increment
            JUMP_MOVL_XYZ, //Jump movement,the movement is MOVL
      };
```

2.12 CP

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

2.12.1 Set/Get CPParams

The commands are applied to set and get parameters of continuous trajectory, including acceleration preset, joint velocity and acceleration. One thing to note that the velocity of this command is only available for continuous trajectory motion.

1. The aim of setting parameters of continuous trajectory（SetCPParams）is to control its motion speed. The issued instruction packet format is shown in Figure 88, and the returned instruction packet format is shown in Figure 89;

Figure 88 The instruction packet of SetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	1	0 or 1	CPParams（See Program 16）	Payload checksum

Figure 89 The returned instruction packet of SetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	90	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetCPParams, the issued instruction packet format is shown in Figure 90, and the returned instruction packet format is shown in Figure 91.

Figure 90 The instruction packet of GetCPParams

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

Figure 91 The returned instruction packet of GetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	0	0	CPParams（See Program 16）	Payload checksum

Program 16 CPParams Definition

```
typedef struct tagCPParams {
    float planAcc;              // Maximum value of planned acceleration
    float junctionVel;    // Maximum value of junction acceleration
    union {
        float acc; //Maximum value of actual acceleration,using in non-real-time mode
        float period;         //Interpolation cycle, real-time mode
    };
    uint8_t realTimeTrack; //0—non real time mode; 1—non real time mode
    } CPParams;
```

2.12.2 SetCPCmd

SetCPCmd, the issued instruction packet format is shown in Figure 92, and the returned instruction packet format is shown in Figure 93.

Figure 92 The instruction packet of SetCPCmd

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

Figure 93 The returned instruction packet of SetCPCmd

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

Program 17 CPCmd Definition

```
typedef struct tagCPCmd {
        uint8_t cpMode;             //CP mode    0-relative mode    1-absolute mode
        float x;                    //x-coordinate increment / x coordinate
        float y;                    //y-coordinate increment / y coordinate
        float z;                    //z-coordinate increment/ z coordinate
        union {
        float velocity;    // Reserved
        float power;            //Laser power
                }
} CPCmd;
```

2.12.3 SetCPLECmd

Execute the function of continuous path laser engraving commands, the issued instruction packet is shown as Figure 94, and the returned instruction packet is shown as Figure 95.

Figure 94 The instruction packet of SetCPLECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	92	1	1	CPCmd（见 Program 18）	Payload checksum

Figure 95 The returned instruction packet of SetCPLECmd

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

Program 18 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 0~100
        }
} CPCmd;
```

2.13 ARC

2.13.1 Set/Get ARCParams

1. Set the circular arc interpolation parameters（SetARCParams）, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 96 The instruction packet of SetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	1	0 or 1	ARCParams（See Program 19）	Payload checksum

Figure 97 The returned instruction packet of SetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	100	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetARCParams, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 98 The instruction packet of GetARCParams

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

Figure 99 The returned instruction packet of GetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	0	0	ARCParams（See Program 19）	Payload checksum

Program 19 ARCParams Definition

```
typedef struct tagARCParams {
        float xyzVelocity;            // Circular motion of xyz axis speed
        float rVelocity;                   // EndEffector rotation speed of circular motion
        float xyzAcceleration; // Circular motion xyz axis acceleration
        flaot rAcceleration;      // EndEffector rotation acceleration of circular motion
    } ARCParams;
```

2.13.2 SetARCCmd

SetARCCmd, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 100 The instruction packet of SetARCCmd

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

Figure 101 The returned instruction packet of SetARCCmd

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

Program 20 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;
```

WAIT Circular path Description:

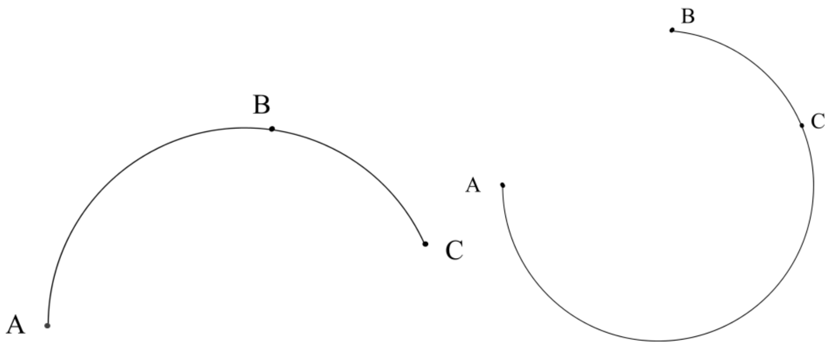
1、arc track is the space of the arc, from the current point, any point on the arc and the end point of the arc together to determine the three points;

The arc always passes from one point on the arc to the end point.

Circular trajectory shown as follows:

(a) A is the current point, B is any point on the arc, C is the end point;

(b) A is the current point, C is any point on the arc, B is the end point.



(a) Starting point A, ending point C (b) Starting point A, ending point B

2.14 WAIT

2.14.1 SetWAITCmd

SetWAITCmd, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 102 The instruction packet of SetWAITCmd

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

Figure 103 The returned instruction packet of SetWAITCmd

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

Program 21 WAITCmd Definition

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

2.15 TRIG

2.15.1 SetTRIGCmd

SetTRIGCmd, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 104 The instruction packet of SetTRIGCmd

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

Figure 105 The returned instruction packet of SetTRIGCmd

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

Program 22 WAITCmd Definition

```
typedef struct tagWAITCmd {
    uint8_t address;            //EIO addressing（Value range 1~20）
    uint8_t mode;               //Trigger mode 0—IO trigger   1—AD trigger
    uint16_t threshold;         //Trigger condition IO value—0/1   ADCvalue—0~4095
} TRIGCmd;
```

2.16 EIO

2.16.1 Set/Get IOMultiplexing

1. SetIOMultiplexing, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 106 The instruction packet of Set I/O Multiplexing

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

Figure 107 The returned instruction packet of Set I/O Multiplexing

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	Payload length	130	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetIOMultiplexing, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 108 The instruction 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

Figure 109 The returned instruction 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 ）	Payload checksum

Program 23 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 24:

Program 24 IOFunction Definition

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

2.16.2 Set/Get IODO

1. SetIODO, the issued instruction packet format is shown in Figure 110, and the returned instruction packet format is shown in Figure 111;

Figure 110 The instruction packet of SetIODO

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

Figure 111 The returned instruction packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload length	131	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. GetIODO, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 112 The instruction packet of GetIODO

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

Figure 113 The returned instruction packet of GetIODO

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

Program 25 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;
```

2.16.3 Set/Get IOPWM

1. Set I/O PWM output（SetIOPWM）, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 114 The instruction packet of SetIOPWM

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+9	132	1	0 or 1	IOPWM（See Program ）	Payload checksum

Figure 115 The returned instruction packet of SetIOPWM

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	132	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

2. Get I/O PWM（GetIOPWM）, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 116 PWMThe instruction packet of GetIOPWM

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

Figure 117 PWMThe returned instruction packet of GetIOPWM

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

Program 26 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;
```

2.16.4 GetIODI

GetIODI, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 118 The instruction packet of GetIODI

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+0	133	0	0	Empty	Payload checksum

Figure 119 The returned instruction packet of GetIODI

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

Program 27 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;
```

2.16.5 GetIOADC

GetIOADC, the issued instruction packet format is shown in Figure 120, and the returned instruction packet format is shown in Figure 121.

Figure 120 The instruction packet of GetIOADC

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

Figure 121 The returned instruction packet of GetIOADC

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

Program 28 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;
```

2.16.6 SetEMotor

SetIODO, the issued instruction packet format is shown in Figure 122, and the returned instruction packet format is shown in 123;

Figure 122 The instruction packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	135	1	0 or 1	EMotor（见 Program 20）	Payload checksum

Figure 123 The returned instruction packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload length	135	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Program 20 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;
```

2.17 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.

2.17.1 Set/Get AngleSensorStaticError

1. SetAngleSensorStaticError, the issued instruction packet format is shown in Figure 124 , and the returned instruction packet format is shown in Figure ;

Figure 124 The instruction 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

Figure 125 The returned instruction packet of SetAngleSensorStaticError

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

2. GetAngleSensorStaticError, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 126 The instruction packet of GetAngleSensorStaticError

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

Figure 127 The returned instruction 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

2.18 WIFI

2.18.1 Set/Get WIFIConfigMode

1. SetWIFIConfigMode, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 128 The instruction packet of SetWIFIConfigMode

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

Figure 129 The returned instruction packet of SetWIFIConfigMode

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

2. GetWIFIConfigMode, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 130 The instruction packet of GetWIFIConfigMode

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

Figure 131 The returned instruction packet of GetWIFIConfigMode

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

2.18.2 Set/Get WIFISSID

1. SetWIFISSID, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 132 The instruction packet of SetWIFISSID

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	151	1	0	char* ssid	Payload checksum

Figure 133 The returned instruction packet of SetWIFISSID

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

2. GetWIFISSID, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 134 The instruction packet of GetWIFISSID

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

Figure 135 The returned instruction packet of GetWIFISSID

Header	Len	Payload	Checksum

		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	151	0	0	char* ssid	Payload checksum

2.18.3 Set/Get WIFIPassword

1. SetWIFIPassword, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 136 The instruction packet of SetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	152	1	0	char* password	Payload checksum

Figure 137 The returned instruction packet of SetWIFIPassword

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

2. GetWIFIPassword, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 138 The instruction packet of GetWIFIPassword

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

Figure 139 The returned instruction packet of GetWIFIPassword

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	Payload lenght	152	0	0	char* password	Payload checksum

2.18.4 Set/Get WIFIIPAddress

1. Set IP（SetWIFIIPAddress）, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 140 The instruction packet of setting IP

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

Figure 141 The instruction packet of setting returned IP

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

2. GetWIFIIPAddress, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 142 The instruction packet of GetWIFIIPAddress

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

Figure 143 The instruction packet of GetWIFIIPAddress

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

Program 30 WIFIIPAdress Definition

```
typedef struct tagWIFIIPAdress {
    uint8_t dhcp;
    uint8_t addr[4];
} WIFIIPAdress;
```

2.18.5 Set/Get WIFINetmask

1. SetWIFINetmask, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 144 The instruction packet of SetWIFINetmask

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+4	154	1	0	WIFINetmask（See Program ）	Payload checksum

Figure 145 The returned instruction packet of SetWIFINetmask

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

2. GetWIFINetmask, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 146 The instruction packet of GetWIFINetmask

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

Figure 147 The returned instruction packet of GetWIFINetmask

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

Program 31 WIFINetmask Definition

```
typedef struct tagWIFINetmask {
    uint8_t addr[4];
} WIFINetmask;
```

2.18.6 Set/Get WIFIGateway

1. SetWIFIGateway, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 148 The instruction packet of SetWIFIGateway

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

Figure 149 The returned instruction packet of SetWIFIGateway

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

2. GetWIFIGateway, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 150 The instruction packet of GetWIFIGateway

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

Figure 151 The returned instruction packet of GetWIFIGateway

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

Program 32 WIFIGateway Definition

```
typedef struct tagWIFIGateway {
    uint8_t addr[4];
} WIFIGateway;
```

2.18.7 Set/Get WIFIDNS

1. SetWIFIDNS, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure ;

Figure 152 The instruction packet of SetWIFIDNS

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

Figure 153 The returned instruction packet of SetWIFIDNS

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

						checksum

2. GetWIFIDNS, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 154 The instruction packet of GetWIFIDNS

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

Figure 152 The returned instruction packet of GetWIFIDNS

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

Program 33 WIFIDNS Definition

```
typedef struct tagWIFIDNS {
    uint8_t addr[4];
} WIFIDNS;
```

2.18.8 GetWIFIConnectStatus

GetWIFIConnectStatus, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 156 The instruction packet of GetWIFIConnectStatus

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

Figure 157 The returned instruction packet of GetWIFIConnectStatus

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

2.19 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).

2.19.1 SetQueuedCmdStartExec

SetQueuedCmdStartExec, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 158 The instruction packet of SetQueuedCmdStartExec

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

Figure 159 The returned instruction packet of SetQueuedCmdStartExec

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

2.19.2 SetQueuedCmdStopExec

SetQueuedCmdStopExec, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 160 The instruction packet of SetQueuedCmdStopExec

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

Figure 161 The returned instruction packet of SetQueuedCmdStopExec

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

2.19.3 SetQueuedCmdForceStopExec

SetQueuedCmdForceStopExec, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 162 The instruction packet of SetQueuedCmdForceStopExec,

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

Figure 163 The returned instruction packet of SetQueuedCmdForceStopExec,

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

2.19.4 SetQueuedCmdStartDownload

Start commands quene download（SetQueuedCmdStartDownload）, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 164 The instruction 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

Figure 165 The returned instruction 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.

2.19.5 SetQueuedCmdStopDownload

SetQueuedCmdStopDownload, the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 166 The instruction packet of SetQueuedCmdStopDownload

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

Figure 167 The returned instruction packet of SetQueuedCmdStopDownload

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

2.19.6 SetQueuedCmdClear

Clear quene commands（SetQueuedCmdClear），the issued instruction packet format is shown in Figure , and the returned instruction packet format is shown in Figure .

Figure 168 The instruction packet of SetQueuedCmdClear

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

Figure 169 The returned instruction packet of SetQueuedCmdClear

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

2.19.7 GetQueuedCmdCurrentIndex

GetQueuedCmdCurrentIndex, the issued instruction packet format is shown in Figure 170, and the returned instruction packet format is shown in Figure 171.

Figure 170 The instruction packet of GetQueuedCmdCurrentIndex

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

Figure 171 The returned instruction 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: In Dobot controller instruction queue mechanism, there is a 64-bit internal count index.

The counter is automatically incremented each time the controller executes a command. With this internal index, you can check how many queue instructions the controller has executed, and the instructions that are currently executing (indicating the progress of the run).

2.19.8 GetQueuedCmdLeftSpace

GetQueuedCmdLeftSpace, the issued instruction packet format is shown in Figure 172, and the returned instruction packet format is shown in Figure 173.

Figure 172 The instruction packet of GetQueuedCmdLeftSpace

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	247	0	0	Empty	Payload checksum

Figure173 The instruction packet of GetQueuedCmdLeftSpace

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	247	0	0	uint32_t:leftSpace	Payload checksum

Notice: In the Dobot controller instruction queue mechanism, there is an instruction queue. When sending a queue instruction, the remaining space of the instruction queue should be queried. If it is not zero, the queue instruction can be sent to the Dobot controller.

Shenzhen Yuejiang Technology Co.,Ltd

Zip Code：510630

Website：www.dobot.cc

Tel：(0755)38730916

Address：4F, A8, Tanglang Industrial Area, Taoyuan Street, Nanshan District, Shenzhen, China