Dobot M1 Communication Protocol

Issue: V1.0

Date: 2018-11-10

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 M1 upper computer and Dobot M1 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
2018/11/10	The first release

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 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 bit	8 Bit
	Stop bit	1 Bit
	Parity bit	Void
UDP	IP	Route and other distribution
	COM port	54321

1.2 Protocol Introduction

Dobot M1 can be controlled by PC, achieving data transmission through certain communication protocols. The communication can be realized by USB-serial port, Local network (UDP).

Physical layer receives 8 bit 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 M1 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 M1 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 a N-bit number, the 2's complement is equal to 2 ^ N minus the number. In this protocol, assuming that 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 side, 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
- WAIT mode command
- TRIG trigger related command
- I/O control command and so on

The communication protocol function IDs based on the classification are shown in Table 2:

Table 2 Classification of functional items

Classification of functional items	Function ID area	Available 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
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**.

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 used 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 SetDevice 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 used 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 used 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 used 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 used to get device version, the issued command packet is shown in Table 11, and the returned command packet 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+4	2	0	0ss	uint8_t: typeIndex (see Program 1.1)	uint8_t: majorVersion	uint8_t: minorVersion	uint8_t: revision	Payload checksum

Program 1.1 typeIndex definition

```
typeIndex: Version type
enum FirmwareType{
    NO_SWITCH,
    DOBOT_SWITCH,
    PRINTING_SWITCH,
    DRIVER1_SWITCH,
    DRIVER2_SWITCH,
    DRIVER3_SWITCH,
    DRIVER4_SWITCH,
    DRIVER5_SWITCH,
    FPGA_SWITCH,
    SWITCH_FM_MAX
};
```


1.3.4 Get Hardware Version

This command is used to get hardware version, the issued command packet is shown in Table 13, and the returned command packet is shown in Table 14.

Table 13 The command packet of Get HardwareVersion

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	5	0	0	Empty	Payload checksum

Table 14 The returned command packet of Get HardwareVersion

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+132	5	0	0	HardwareInfo (see Program 1.2)	Payload checksum

Program 1.2 HardwareInfo definition

```
typedef struct tagHardwareInfo{
    char machineNum[11];
    char mainBoard[11];
    char driverRearArm[11];
    char driverFrontArm[11];
    char driverZArm[11];
    char driverRArm[11];
    char encoderRearArm[11];
    char encoderFrontArm[11];
    char encoderZArm[11];
    char encoderRArm[11];
    char brakeBoard[11];
    char endIOBoard[ss11];
}HardwareInfo;
```

1.4 Real-time Pose

1.4.1 Get Pose

This command is to get the real-time pose of the Dobot M1, the issued command packet is

shown in Table 15, and the returned command packet is shown in Table 16.

Table 15 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 16 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 1.3）	Payload checksum

Program 1.3 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;
```

1.4.2 Calibrate Dobot M1 (Reset Pose)

This command is to calibrate Dobot M1, the issued command packet is shown in Table 17, and the returned command packet is shown in Table 18.

Table 17 The command packet of ResetPose

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+9	11	1	0	ResetPoseParams (see Program 1.4)	Payload checksum

Table 18 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

Program 1.4 ResetPoseParams definition

```
typedef struct{
uint8_t manual;          //only set to 1
float frontAngle1        //frontAngle1 and frontAngle2 are the forearm angles where Dobot M1 moves to
                           the same point as the righty and lefty arm orientation respectively
float frontAngle2;
} ResetPoseParams
```

1.5 Alarm

1.5.1 Get Alarms State

This command is to get alarm status, the issued command packet is shown in Table 19, and the returned command packet is shown in Table 20.

Table 19 The command packet of GetAlarmsState

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

Table 20 The returned command packet of GetAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	11	0	0	uint8_t[32]: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. For details about alarms, please see *Dobot M1 Alarm Description*.

1.5.2 Clear All Alarms State

This command is to clear alarm status, the issued command packet is shown in Table 21, and the returned command packet is shown in Table 22.

Table 21 The command packet of ClearAllAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	20	1	0	Empty	Payload checksum

Table 22 The returned command packet of ClearAllAlarmsState

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	20	1	0	Empty	Payload checksum

1.6 Homing Function

This part is homing function, including setting the current position as the initial position and executing homing procedure with homing switch .The default initial position is (400,0,0,0).

1.6.1 Set Initial Position Command

This command is to set the current position as the initial position. Namely, set the current position to (400,0,0,0), 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 SetHOMECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	31	1	0	Empty	Payload checksum

Table 24 The returned command packet of SetHOMECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	31	1	0	Empty	Payload checksum

1.6.2 Set Homing With Switch

This command is to execute homing procedure with homing switch, the issued command packet is shown in Table 25, and the returned command packet is shown in Table 26.

Table 25 The command packet of SeHOMEWithSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xsAA	2+1	33	1	0 or 1	uint8_t :isResetPars	Payload checksum

- isResetPars is 0: When Dobot M1 moves to the homing position, the zero position stored in the Flash is assigned to system parameter. Generally, please set to 0.
- isResetPars is 1: When Dobot M1 moves to the homing position, the system parameter is updated to Flash. Only set to 1 for calibration before being shipped out.

Table 26 The returned command packet of SetHOMEWithSwitch

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueued=0 : 2+0; isQueued=1: 2+8	33	1	0 or 1	isQueued=0:Empty; isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

1.7 Hand Hold Teaching

1.7.1 Set/Get HHTTrigMode

This command is to set the hand-hold teaching mode, the issued command packet is shown in Table 27, and the returned command packet is shown in Table 28.

Table 27 The command packet of Set/Get HHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	40	1	0	HHTTrigMode（See Program 1.5）	Payload checksum

Table 28 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 in Table 29, and the returned command packet format is shown in Table 30.

Table 29 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 30 The returned command packet of GetHHTTrigMode

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	40	0	0	HHTTrigMode（See Program 1.5）	Payload checksum

Program 1.5 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 the status of the hand-hold teaching function, the issued command packet is shown in Table 31, and the returned command packet is shown in Table 32.

Table 31 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 32 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 33, and the returned command packet format is shown in Table 34.

Table 33 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 34 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 status, the issued command packet is shown in Table 35, and the returned command packet is shown in Table 36.

Table 35 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 36 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 ArmOrientation

1.8.1 Set/Get ArmOrientation

⚠NOTICE

This command is currently only applicable to SCARA models.

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

Table 37 The command packet of SetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	50	1	0 or 1	ArmOrientation（See Program 1.6）	Payload checksum

Table 38 The returned command packet of SetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueued=0: 2+0 isQueued=1:2+8	50	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

- This command is to get arm orientation, the issued command packet is shown in Table 39, and the returned command packet is shown in Table 40.

Table 39 The command packet of GetArmOrientation

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

Table 40 The returned command packet of GetArmOrientation

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+1	50	0	0	ArmOrientation（See Program 1.6)	Payload checksum

Program 1.6 ArmOrientation definition

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

1.9 EndEffectors

📖NOTE

These commands are only suitable for the matched laser engraving kit and 3D printing kit.

1.9.1 Set/Get EndEffectorParams

- This command is to set the offset of the end-effector, the issued command packet is shown in Table 41, and the returned command packet is shown in Table 42.

Table 41 The command packet of SetEndEffectorParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	60	1	0 or 1	EndEffectorParams（See Program 1.7）	Payload checksum

Table 42 The returned command packet of SetEndEffectorParams

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

- This command is to get the offset of the end-effector, the issued command packet is shown in Table 43, and the returned command packet is shown in Table 44.

Table 43 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 44 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 1.7）	Payload checksum

Program 1.7 EndEffectorParams definition

```
typedef struct tagEndEffectorParams {
float xBias;   X-axis Bias
float yBias;   Y-axis Bias
float zBias;   Z-axis Bias
} EndEffectorParams;
```

1.9.2 Set/Get EndEffectorLaser

- This command is to set the status of the laser, the issued command packet is shown in Table 45, and the returned command packet is shown in Table 46.

Table 45 The command packet of SetEndEffectorLaser

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

Table 46 The returned command packet of SetEndEffectorLaser

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

NOTE

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 47, and the returned command packet format is shown in Table 48.

Table 47 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 48 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

1.10 JOG

1.10.1 Set/Get JOGJointParams

- This command is to set the velocity and acceleration of the joints coordinate axes in jogging mode, the issued command packet is shown in Table 49, and the returned command is shown in Table 50.

Table 49 The command packet of SetJOGJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	70	1	0 or 1	JOGJointParams（See Program 1.8）	Payload checksum

Table 50 The returned command packet of SetJOGJointParams

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

	d=1: 2+8					

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 is shown in Table 51, and the returned command packet is shown in Table 52.

Table 51 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 52 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 1.8）	Payload checksum

Program 1.8 JOGJointParams definition

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

1.10.2 Set/Get JOGCoordinateParams

- This command is to set the velocity and acceleration of the Cartesian coordinate axes in jogging mode, the issued command packet is shown in Table 53, and the returned command packet is shown in Table 54.

Table 53 The command packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	1	0 or 1	JOGCoordinateParams	Payload

					（See Program 1.9）	checksum

Table 54 The returned command packet of SetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueued=0: 2+0; isQueued=1: 2+8	71	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	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 is shown in Table 55, and the returned command packet is shown in Table 56

Table 55 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 56 The returned command packet of GetJOGCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	71	0	0	JOGCoordinateParam （See Program 1.9）	Payload checksum

Program 1.9 JOGCoordinateParams definition

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

```
} JOGCoordinateParams;
```

1.10.3 Set/Get JOGCommonParams

- This command is to set the velocity ratio and acceleration ratio in jogging mode, the issued command packet is shown in Table 57, and the returned command packet is shown in Table 58.

Table 57 The command packet of SetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	1	0 or 1	JOGCommonParams（See Program 1.10）	Payload checksum

Table 58 The returned command packet of SetJOGCommonParams

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

- This command is to get the velocity and acceleration in jogging mode, the issued command packet is shown in Table 59, and the returned command packet is shown in Table 60.

Table 59 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 60 The returned command packet of GetJOGCommonParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	0	0	JOGCommonParams	Payload

					（See Program 1.10）	checksum

Program 1.10 JOGCommonParams definition

```
typedef struct tagJOGCommonParams {
float velocityRatio;   //Velocity ratio, which is suitable for joint coordinate axes and Cartesian coordinate axes
float accelerationRatio; //Acceleration ratio,  which is suitable for joint coordinate axes and Cartesian coordinate
                         axes
} JOGCommonParams;
```

1.10.4 Set JOGCmd

This command is to execute the jogging command, the issued command packet is shown in Table 61, and the returned command packet is shown in Table 62.

Table 61 The command packet of SetJOGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	73	1	0 or 1	JOGCmd（See Program 1.11）	Payload checksum

Table 62 The returned command packet of SetJOGCmd

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

Program 1.11 JOGCmd definition

```
typedef struct tagJOGCmd {
uint8_t isJoint;      //Jog mode 0:Jog in Cartesian coordinate system   1: Jog in joint coordinate system
uint8_t cmd;          //Jog command(Value range0~8)
}JOGCmd;
//The detailed description of JOGCmd
enum {
    IDEL,           //Invalid
    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-
};
```

1.11 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.

1.11.1 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 is shown in Table 63, and the returned command packet is shown in Table 64.

Table 63 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	1	0 or 1	PTPJointParams （See Program 1.12）	Payload checksum

Table 64 The returned command packet of SetPTPJointParams

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

- This command is to get the velocity and acceleration of the joint coordinate axes in PTP mode, the issued command packet is shown in Table 65, and the returned command packet

is shown in Table 66.

Table 65 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 66 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 1.12）	Payload checksum

Program 1.12 PTPJointParams definition

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

1.11.2 Set/Get PTPCoordinateParams

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

Table 67 The command packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	1	0 or 1	PTPCoordinateParams （See Program 1.13）	Payload checksum

Table 68 The returned command packet of SetPTPCoordinateParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueue	81	1	0 or 1	isQueued=0:Empty	Payload

	d=0: 2+0; isQueued=1: 2+8				isQueued=1:uint64_t:queuedCmdIndex	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 69, and the returned command packet is shown in Table 70.

Table 69 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 70 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 1.13）	Payload checksum

Program 1.13 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;
```

1.11.3 Set/Get PTPJumpParams

- This command is to set the lifting height and the maximum lifting height in JUMP mode, the issued command packet is shown in Table 71, and the returned command packet is shown in Table 72.

Table 71 The command packet of SetPTPJumpParams

Header	Len	Payload			Checksum
		ID	Ctrl	Params	

			rw	isQueued		
0xAA 0xAA	2+8	82	1	0 or 1	PTPJumpParams（See Program 1.14）	Payload checksum

Table 72 The returned command packet of SetPTPJumpParams

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

- This command is to get the lifting height and the maximum lifting height in JUMP mode, the issued command packet is shown in Table 73, and the returned command packet is shown in Table 74.

Table 73 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 74 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 1.14）	Payload checksum

Program 1.14 PTPJumpParams definition

```
typedef struct tagPTPJumpParams {
float jumpHeight;          //Lifting height in Jump mode
float zLimit;          //Maximum lifting height in Jump mode
} PTPJumpParams;
```

1.11.4 Set/Get PTPCommonParams

- This command is to set the velocity ratio and the acceleration ratio in PTP mode, the issued command packet is shown in Table 75, and the returned command packet is shown in Table 76.

Table 75 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	1	0 or 1	PTPCommonParams（See Program 1.15）	Payload checksum

Table 76 The returned command packet of SetPTPJointParams

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

- This command is to get the velocity ratio and the acceleration ratio in PTP mode, the issued command packet is shown in Table 77, and the returned command packet is shown in Table 78.

Table 77 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 78 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 1.15）	Payload checksum

Program 1.15 PTPCommonParams definition

```
typedef struct tagPTPCommonParams {
float velocityRatio;            //Velocity ratio in PTP mode, which is suitable for joint coordinate axes and
                                    Cartesian coordinate axes
float accelerationRatio;        //Acceleration ratio in PTP mode, which is suitable for joint coordinate axes and
                                    Cartesian coordinate axes
} PTPCommonParams;
```

1.11.5 Set PTPCmd

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

Table 79 The command packet of SetPTPJointParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	84	1	0 or 1	PTPCmd（See Program 1.16）	Payload checksum

Table 80 The returned command packet of SetPTPJointParams

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

Program 1.16 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
};
```

1.12 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.

1.12.1 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 is shown in Table 81, and the returned command packet is shown in Table 82.

Table 81 The command packet of SetCPParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	1	0 or 1	CPParams（See Program 1.17）	Payload checksum

Table 82 The returned command packet of SetCPParams

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

	2+8					

- This command is to get the motion parameters in CP mode, the issued command packet is shown in Table 83, and the returned command packet is shown in Table 84.

Table 83 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 84 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 1.17）	Payload checksum

Program 1.17 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;
```

1.12.2 Set CPCmd

This command is to execute the CP command, the issued command packet is shown in Table 85, and the returned command packet is shown in Table 86.

Table 85 The command packet of SetCPCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	91	1	0 or 1	CPCmd（See Program 1.18）	Payload checksum

Table 86 The returned command packet of SetCPCmd

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

Program 1.18 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
    }
} CPCmd;
```

1.12.3 Set CPLECmd

This command is to execute CP command with laser engraving commands, the issued command is shown in Table 87, and the returned command is shown in Table 88.

Table 87 The command packet of SetCPLECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	92	1	0 or 1	CPCmd（See Program 1.19）	Payload checksum

Table 88 The returned command packet of SetCPLECmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueued=0: 2+0; isQueue	92	1	ss0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

	d=1: 2+8					

Program 1.19 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 {
        float velocity; //Reserved
        float power;    // Laser power 0~100
    }
} CPCmd;
```

1.13 ARC

1.13.1 Set/Get ARCParams

- This command is to set the velocity and acceleration in ARC mode, the issued command packet is shown in Table 89, and the returned command packet is shown in Table 90.

Table 89 The command packet of SetARCParams

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	1	0 or 1	ARCParams（See Program 1.20）	Payload checksum

Table 90 The returned command packet of SetARCParams

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

- This command is to get the velocity and acceleration in ARC mode, the issued command packet is shown in Table 91, and the returned command packet is shown in Table 92.

Table 91 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 92 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 1.20）	Payload checksum

Program 1.20 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;
```

1.13.2 Set ARCCmd

This command is to execute the ARC command, the issued command packet is shown in Table 93, and the returned command packet is shown in Table 94.

Table 93 The command packet of SetARCCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	101	1	0 or 1	ARCCmd（See Program 1.21）	Payload checksum

Table 94 The returned command packet of SetARCCmd

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

	isQueued=1:2+8					

Program 1.21 ARCCmd definition

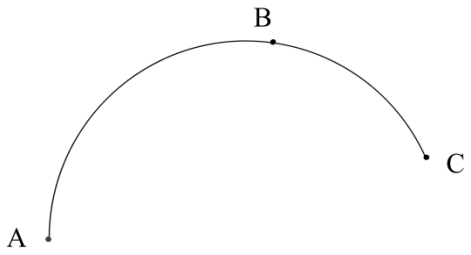
```
typedef struct tagARCCmd {
struct{
float x;
float y;
float z;
float r;
} cirPoint;         //Any point

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

📖NOTE

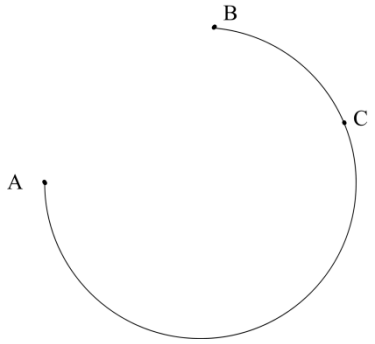
The trajectory of ARC mode is an arc, which is determined by three points (the current point, any point and the end point on the arc

ARC trajectory is 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.



1.13.3 Set CircleCmd

This command is to execute the ARC command, the issued command packet is shown in Table 93, and the returned command packet is shown in Table 94.

Table 95 The command packet of SetARCCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	102	1	0 or 1	CircleCmd（See Program 1.22）	Payload checksum

Table 96 The returned command packet of SetARCCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	isQueued=0: 2+0; isQueued=1: 2+8	102	1	0 or 1	isQueued=0:Empty isQueued=1:uint64_t:queuedCmdIndex	Payload checksum

Program 1.22 CircleCmd definition

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

```
struct {
        float x;
        float y;
        float z;
        float r;
} toPoint;              // End point
        uint32_t count;
}CircleCmd
```

1.14 WAIT

1.14.1 Set WAITCmd

This command is to execute the waiting command, the issued command packet is shown in Table 97, and the returned command packet sis shown in Table 98.

Table 97 The command packet of SetWAITCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	110	1	0 or 1	WAITCmd (See Program 1.23)	Payload checksum

Table 98 The returned command packet of SetWAITCmd

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

Program 1.23 SetWAITCmd definition

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

1.15 TRIG

1.15.1 Set TRIGCmd

This command is to execute the triggering command, the issued command packet is shown in Table 99, and the returned command packet is shown in Table 100.

Table 99 The command packet of SetTRIGCmd

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	120	1	0 or 1	TRIGCmd （See Program 1.24）	Payload checksum

Table 100 The returned command packet of SetTRIGCmd

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

Program 1.24 SetTRIGCmd definition

```
typedef struct tagTrigCmd {
uint8_t address;                // EIO address: If mode is set to 0, the value range is 1 to 24. If mode is set to 1,
                                   the value range is 1 to 6
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;
typedef enum tagIOCondition {
TRIGInputIOEqual,
TRIGInputIONotEqual
} IOCondition;
```



```
typedef enum tagADCCondition {
TRIGADCLT,          //Lower than
TRIGADCLE,          //Lower than or Equal
TRIGADCGE,          //Greater than or Equal
TRIGADCGT           //Greater than
} ADCCondition;
```

1.16 EIO

In the Dobot M1controller, the addresses of the I/O interfaces are unified. Here, you can see as follows:

- High-low level output.
- Read High-low level output.
- Read analog-digital conversion value output.

For more details, please see *Dobot M1 User Guide*.

1.16.1 Set/Get IODO

- This command is to set the I/O output, the issued command packet is shown in Table 101 and the returned command packet is shown in Table 102.

Table 101 The command packet of SetIODO

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	131	1	0 or 1	IODO（See Program 1.25）	Payload checksum

Table 102 The returned command packet of SetIODO

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

- This command is to get the I/O output, the issued command packet is shown in Table 103, and the returned command packet is shown in Table 104.

Table 103 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 104 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 1.25）	Payload checksum

Program 1.25 IODO definition

```
typedef struct tagIODO {
uint8_t address;          //I/O address: 1~22
uint8_t level;           // 0: Low level; 1：High level
} IODO;ss
```

1.16.2 Get IODI

This command is to get the I/O input, the issued command packet format is shown in Table 105, and the returned command packet format is shown in Table 105.

Table 105 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 106 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 1.26）	Payload checksum

Program 1.26 IODI definition

```
typedef struct tagIODI {
uint8_t address;          //I/O address: 1~24
uint8_t level;            //0: Low level; 1:High level
}IODI;
```

1.16.3 Get IOADC

This command is to get the A/D input, 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 GetIOADC

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

Table 108 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 1.27）	Payload checksum

Program 1.27 IOADC definition

```
typedef struct tagIOADC{
uint8_t address;          //I/O address: 1~6
uint16_t value;           //Input value: 0~4095
}IOADC;
```

1.17 Command Queue Controlling

These 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).

1.17.1 Set QueuedCmdStartExec

This command is to start to query command queue periodically, the issued command packet is shown in Table 109, and the returned command packet is shown in Table 110

Table 109 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 110 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

1.17.2 Set QueuedCmdStopExec

This command is to stop to query command queue and to execute command, the issued command packet is shown in Table 111, and the returned command packet is shown in Table 112.

Table 111 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 112 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

1.17.3 Set QueuedCmdForceStopExec

This command is to stop to query command queue and to execute command forcedly, the issued command packet is shown in Table 113, and the returned command packet is shown in Table 114.

Table 113 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 114 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

1.17.4 Set QueuedCmdStartDownload

This command is to download commands, 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 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 116 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 M1controller supports storing commands in the external Flash of the controller, which can then be executed in the offline mode.

1.17.5 Set QueuedCmdStopDownload

This command is to stop to download commands, 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 SetQueuedCmdStopDownload

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+0	244	1	0	Empty	Payload checksum

Table 118 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

1.17.6 Set QueuedCmdClear

This command is to clear command queue，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 SetQueuedCmdClear

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

Table 120 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

1.17.7 GetQueuedCmdCurrentIndex

This command is to get command index, the issued command packet is shown in Table 121, and the returned command packet is shown in Table 122.

Table 121 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 122 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 counter in Dobot M1 controller. When the controller executes a command, the counter will automatically increment. With this internal index, you can get how many commands the controller has executed, and which command is being executed currently.

1.17.8 Get QueuedCmdLeftSpace

This command is to, the issued command packet format is shown in Table 123, and the returned command packet format is shown in Table 124.

NOTICE

When sending a queue command, the remaining space of the command queue should be queried. If it is not zero, the queue command can be sent to the Dobot M1 controller.

Table 123 The command packet of GetQueuedCmdLeftSpace

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

Table 124 The command packet of GetQueuedCmdLeftSpace

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

