



DOBOT

通信协议

Dobot Magician 通信协议

版本：V1.1.3

发布日期：2018/11/16

深圳市越疆科技有限公司

版权所有 © 越疆科技有限公司2018。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本档容的部分或全部，并不得以任何形式传播。

免责声明

在法律允许的最大范围内，本手册所描述的产品（含其硬件、软件、固件等）均“按照现状”提供，可能存在瑕疵、错误或故障，越疆不提供任何形式的明示或默示保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证；亦不对使用本手册或使用本公司产品导致的任何特殊、附带、偶然或间接的损害进行赔偿。

在使用本产品前详细阅读本使用手册及网上发布的相关技术文档并了解相关信息，确保在充分了解机器人及其相关知识的前提下使用机械臂。越疆建议您在专业人员的指导下使用本手册。该手册所包含的所有安全方面的信息都不得视为Dobot的保证，即便遵循本手册及相关说明，使用过程中造成的危害或损失依然有可能发生。

本产品的使用者有责任确保遵循相关国家的切实可行的法律法规，确保在越疆机械臂的使用中不存在任何重大危险。

越疆科技有限公司

地址：深圳市南山区同富裕工业城三栋三楼

网址：<http://cn.dobot.cc/>

前言

目的

本文档适用于 Dobot Magician 产品上位机与 Dobot Magician 机械臂之间命令/数据交互的通信协议。

读者对象

本手册适用于：


- 客户工程师
- 销售工程师
- 安装调试工程师
- 技术支持工程师

修订记录

日期	原因
2016/09/22	创建文档
2016/11/18	增加通信参数说明
2016/11/21	增加 BLE 读/写 Characteristic UUID
2016/11/30	增加运动控制相关参数说明（JOG PTP CP ARC 等）
2016/12/20	获取指令队列剩余空间的补充
2017/05/05	修复执行回零命令的错误、增加导轨相关接口、增加 UID 端口、增加颜色传感器端口...
2017/5/25	获取 EIO 状态下发端口参数修改
2017/6/26	颜色传感器和光电开关、丢步检测功能添加 EIO 输入增加上拉输入和下拉输入
2017/9/6	I/O 复用顺序错误修复、I/O 触发条件错误修复、设置 EMotor 文本错误修复
2018/11/16	修复协议指令类型错误，代码错误

符号约定

在本手册中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害

符号	说明
 警告	表示有中度或低度潜在危害，如果不能避免，可能导致人员轻微伤害、机械臂毁坏等情况
 注意	表示有潜在风险，如果忽视这些文本，可能导致机械臂损坏、数据丢失或不可预知的结果
 说明	表示是正文的附加信息，是对正文的强调和补充

目 录

1. 通讯协议.....	1
1.1 通信参数.....	1
1.2 协议简介.....	1
1.2.1 协议特点.....	1
1.2.2 校验计算.....	2
1.2.3 协议分类.....	2
1.3 设备信息.....	3
1.3.1 设置和获取设备序列号 (Set/Get DeviceSN)	3
1.3.2 设置和获取设备名称 (Set/Get DeviceName)	4
1.3.3 获取设备版本号 (GetDeviceVersion)	5
1.3.4 设置和获取设备滑轨状态 (Set/Get DeviceWithL)	5
1.3.5 获取设备系统滴答时钟 (Get DeviceTime)	6
1.3.6 获取设备 UID (Get DeviceID)	6
1.4 实时位姿.....	7
1.4.1 获取实时位姿 (GetPose)	7
1.4.2 重设实时姿态 (ResetPose)	7
1.4.3 获取滑轨实时位姿 (GetPoseL)	8
1.5 报警功能.....	8
1.5.1 获取报警状态 (GetAlarmsState)	8
1.5.2 清除系统的所有报警 (ClearAllAlarmsState)	9
1.6 回零功能.....	9
1.6.1 设置和获取零位参数 (Set/Get HOMEParams)	10
1.6.2 执行回零命令 (SetHOMECmd)	11
1.6.3 执行和获取自动调平 (Set/Get AutoLeveling)	11
1.7 手持示教功能.....	13
1.7.1 设置和获取手持示教触发模式 (Set/Get HHTTrigMode)	13
1.7.2 设置和获取触发模式输出使能/禁止 (Set/Get HHTTrigOutputEnabled)	14
1.7.3 获取触发输出状态 (GetHHTTrigOutput)	14
1.8 末端执行器.....	15
1.8.1 设置和获取末端执行器参数 (Set/Get EndEffectorParams)	15
1.8.2 设置和获取激光输出 (Set/Get EndEffectorLaser)	16
1.8.3 设置和获取吸盘输出 (Set/Get EndEffectorSuctionCup)	17
1.8.4 设置和获取爪子输出 (Set/Get EndEffectorGripper)	18
1.9 JOG 功能	19
1.9.1 设置和获取关节点动参数 (Set/Get JOGJointParams)	19
1.9.2 设置和获取坐标轴点动参数 (Set/Get JOGCoordinateParams)	20
1.9.3 设置和获取点动公共参数 (Set/Get JOGCommonParams)	21
1.9.4 执行点动功能 (SetJOGCmd)	22
1.9.5 设置和获取滑轨 L 点动参数 (Set/Get JOGLParams)	23
1.10 再现(PTP)功能	25
1.10.1 设置和获取关节点位参数 (Set/Get PTPJointParams)	25

1.10.2 设置和获取坐标轴点位参数 (Set/Get PTPCoordinateParams)	26
1.10.3 设置和获取门型模式点位参数 (Set/Get PTPJumpParams)	27
1.10.4 设置和获取点位公共参数 (Set/Get PTPCommonParams)	28
1.10.5 执行点位功能 (SetPTPCmd)	29
1.10.6 设置和获取滑轨关节点位参数 (Set/Get PTPLParams)	30
1.10.7 执行带滑轨点位功能 (SetPTPWithLCmd)	31
1.10.8 设置和获取门型模式点位参数 2 (Set/Get PTPJump2Params)	32
1.10.9 执行平行输出点位功能 (SetPTPPOCmd)	34
1.10.10 执行滑轨平行输出点位功能 (SetPTPPOWithLCmd)	35
1.11 CP 功能.....	36
1.11.1 设置和获取连续轨迹功能参数 (Set/Get CPParams)	37
1.11.2 执行连续轨迹功能 (SetCPCmd)	38
1.11.3 执行连续轨迹灰度雕刻功能 (SetCPLECmd)	39
1.12 ARC 功能.....	39
1.12.1 设置和获取圆弧插补功能参数 (Set/Get ARCPParams)	39
1.12.2 执行圆弧插补功能 (SetARCCmd)	41
1.13 WAIT 功能.....	42
1.13.1 执行时间等待功能 (SetWAITCmd)	42
1.14 TRIG 功能	43
1.14.1 执行触发功能 (SetTRIGCmd)	43
1.15 EIO 功能.....	44
1.15.1 设置和读取 I/O 复用 (Set/Get IOMultiplexing)	44
1.15.2 设置和读取 I/O 输出电平 (Set/Get IODO)	45
1.15.3 设置和读取 PWM 输出 (Set/Get IOPWM)	46
1.15.4 读取 I/O 输入电平 (GetIODI)	47
1.15.5 读取 I/O 模数转换值 (GetIOADC)	48
1.15.6 设置扩展电机接口 (SetEMotor)	48
1.15.7 设置和读取颜色传感器 (Set/Get ColorSensor)	49
1.15.8 设置和读取红外开关 (Set/Get IRSwitch)	50
1.16 校准(CAL)功能.....	51
1.16.1 设置和读取角度传感器静态误差 (Set/Get AngleSensorStaticError)	51
1.17 WIFI 功能.....	52
1.17.1 设置和获取 WIFI 配置模式 (Set/Get WIFIConfigMode)	52
1.17.2 设置和获取 SSID (Set/Get WIFISSID)	53
1.17.3 设置和获取网络密码 (Set/Get WIFIPassword)	54
1.17.4 设置和获取 IP 地址 (Set/Get WIFIIPAddress)	55
1.17.5 设置和获取子网掩码 (Set/Get WIFINetmask)	56
1.17.6 设置和获取网关 (Set/Get WIFIGateway)	57
1.17.7 设置和获取 DNS (Set/Get WIFIDNS)	58
1.17.8 获取 WIFI 连接状态 (GetWIFIConnectStatus)	59
1.18 丢步功能.....	59
1.18.1 设置丢步检测功能 (SetLostStepValue)	59
1.18.2 执行丢步检测功能 (SetLostStepCmd)	59
1.19 队列执行控制命令.....	60

1.19.1 启动指令队列运行 (SetQueuedCmdStartExec)	60
1.19.2 停止指令队列运行 (SetQueuedCmdStopExec)	60
1.19.3 强制停止指令队列运行 (SetQueuedCmdForceStopExec)	61
1.19.4 启动指令队列下载 (SetQueuedCmdStartDownload)	61
1.19.5 完成指令队列下载 (SetQueuedCmdStopDownload)	62
1.19.6 清空指令队列 (SetQueuedCmdClear)	62
1.19.7 获取指令队列索引 (GetQueuedCmdCurrentIndex)	63
1.19.8 获取指令队列剩余空间 (GetQueuedCmdLeftSpace)	63

1. 通讯协议

1.1 通信参数

表格 1 通信参数说明

通信参数	详细参数	参数说明
USB 转串口	波特率	115200bps
	数据位	8 位
	停止位	1 位
	校验位	无
Wi-Fi	IP	路由等分配
	端口	8899
BLE	Service UUID	0003CDD0-0000-1000-8000-00805F9B0131
	读（指令）端口 Characteristic UUID	0003CDD1-0000-1000-8000-00805F9B0131
	写（指令）端口 Characteristic UUID	0003CDD2-0000-1000-8000-00805F9B0131
扩展串口 (TTL)	波特率	115200bps
	数据位	8 位
	停止位	1 位
	校验位	无

1.2 协议简介

Dobot Magician 机械臂可通过 PC/Android/iOS 客户端控制，这些设备与机械臂之间通过特定的通信协议实现数据传输，进而实现控制。目前 Dobot Magician 可以通过默认 USB 转串口、TTL 电平串口、Wi-Fi（UDP）控制。

物理层每次接收的数据是 8 位原始数据，需要制定通信协议来确定数据传输的开始与结束、以及校验数据的准确性。通信协议一般需要包括数据包的包头、数据负载、负载校验，来保证数据的准确传输。

1.2.1 协议特点

Dobot 的通信协议的特点如下：

- 协议指令不定长；
- 协议指令由包头、负载帧长、负载帧、校验组成；
- 指令分为立即指令和队列指令；
- 所有的通信都由主机主动发起，且对于所有通信指令，下位机都会返回数据（无论读写）；对于队列指令，其返回带有 64 位的执行索引值；
- 立即指令将立即被调用执行，所有的读操作都是立即指令；
- 队列指令将被放入下位机队列中，串行地执行。对于写（或设置）操作，其中运动类型的指令都应该是队列命令（比如回零、JOG、PTP 等）；
- 设置参数的指令既可以是立即指令，也可以是队列指令；
- 在主机向下位机发送队列命令前，应查询下位机命令队列的剩余空间（可查询一次，发送多条命令）；

- 立即指令总是立即执行完成；队列指令的执行完成情况可通过查询当前下位机正在执行的队列命令索引，与该条命令的索引（第4点中提到的命令中返回的）的比较得到；
- 命令中的参数使用小端模式。

1.2.2 校验计算

在 Dobot Magician 的通信协议中，发送端校验计算方法如下：

步骤 1 将负载帧（Payload）中的所有内容按照字节（8位）的方式逐字节相加，得到一个结果R（低8位）；

步骤 2 对结果R（低8位）求二补数，存入校验字节中。

说明

二补数：Two's complement。对于一个 N 位的数字，其二补数等于 2^N 减去该数。在本协议中，假设上述的结果 R 为 0x0A，其二补数也即上述的校验结果等于 $(2^8 - 0x0A) = (256 - 10) = 246 = 0xF6$ 。

在接收端，校验一帧数据是否正确的方法如下：

步骤 1 将负载帧（Payload）中的所有内容按照字节（8位）的方式逐字节相加，得到一个结果A（低8位）；

步骤 2 结果A（低8位）与校验字节相加，如果等于0，则说明校验正确。

1.2.3 协议分类

根据实现功能不同，又可分为以下几部分：

- 队列执行控制命令
- 设备信息相关命令
- 公共参数命令
- 回零功能命令
- 手持示教命令
- 点动模式命令
- PTP 模式命令
- CP 模式命令
- TRACK 模式命令
- WAIT 模式命令
- TRIG 触发相关命令
- IO 控制命令等

通过分类，将通信协议功能 ID 分成表格 2 中的几大项：

表格 2 功能项划分

功能项划分	Function ID 区间	可用 ID 个数
ProtocolFunctionDeviceInfoBase	[0, 10)	10
ProtocolFunctionPoseBase	[10, 20)	10
ProtocolFunctionALARMBase	[20, 30)	10
ProtocolFunctionHOMEBase	[30, 40)	10
ProtocolFunctionHHTBase	[40, 50)	10
ProtocolFunctionArmOrientationBase	[50, 60)	10

功能项划分	Function ID 区间	可用 ID 个数
ProtocolFunctionEndEffectorBase	[60, 70)	10
ProtocolFunctionJOGBase	[70, 80)	10
ProtocolFunctionPTPBase	[80, 90)	10
ProtocolFunctionCPBase	[90, 100)	10
ProtocolFunctionARCBBase	[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

说明

在下文的每条指令说明中，都带有 ID 说明；

下文中 Ctrl 字节中，rw 为 Ctrl 字节的第 0 位，isQueued 为 Ctrl 字节的第 1 位。

说明

当 isQueued=1 时，指令为队列指令，返回带有 64 位的执行索引值，所以指令长度为：2+8；当 isQueued=0 时，指令为非队列指令，无返回值。指令长度为：2+0。

1.3 设备信息

该部分命令用于设置设备 SN 号、设备名称和设备版本号，并通过命令读回设备当前信息。

1.3.1 设置和获取设备序列号（Set/Get DeviceSN）

- 设置设备序列号（SetDeviceSN），下发的指令包格式如表格 3 所示，返回指令包格式如表格 4 所示；

表格 3 设置设备序列号指令包

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

表格 4 设置设备序列号返回指令包

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

- 获取设备序列号（GetDeviceSN），下发的指令包格式如表格 5 所示，返回指令包格式如表格 6 所示。

表格 5 获取设备序列号指令包

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

表格 6 获取设备序列号返回指令包

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

1.3.2 设置和获取设备名称（Set/Get DeviceName）

- 设置设备名称（SetDeviceName），下发的指令包格式如表格 7 所示，返回指令包格式如表格 8 所示；

表格 7 设置设备名称指令包

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

表格 8 设置设备名称返回指令包

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

- 获取设备名称（GetDeviceName），下发的指令包格式如表格 9 所示，返回指令包格式如表格 10 所示。

表格 9 获取设备名称指令包

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

表格 10 获取设备名称返回指令包

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

1.3.3 获取设备版本号 (GetDeviceVersion)

获取设备版本号 (GetDeviceVersion)，下发的指令包格式如表格 11 所示，返回指令包格式如表格 12 所示。

表格 11 获取设备版本号指令包

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

表格 12 获取设备版本号返回指令包

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

1.3.4 设置和获取设备滑轨状态 (Set/Get DeviceWithL)

设置设备滑轨状态 (Set DeviceWithL)，下发的指令包格式如表格 13 所示，返回指令包格式如表格 14 所示。

表格 13 设置设备滑轨状态指令包

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

表格 14 设置设备滑轨状态返回指令包

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

- 获取设备滑轨状态（Set DeviceWithL），下发的指令包格式如表格 15 所示，返回指令包格式如表格 16 所示。

表格 15 获取设备滑轨状态指令包

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

表格 16 获取设备滑轨状态返回指令包

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

1.3.5 获取设备系统滴答时钟（Get DeviceTime）

- 获取设备系统滴答时钟（GetDeviceTime），下发的指令包格式如表格 17 所示，返回指令包格式如表格 18 所示。

表格 17 获取设备 UID 指令包

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

表格 18 获取设备 UID 返回指令包

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

1.3.6 获取设备 UID（Get DeviceID）

- 获取设备 UID（GetDeviceID），下发的指令包格式如表格 19 所示，返回指令包格式如表格 20 所示。

表格 19 获取设备 UID 指令包

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

表格 20 获取设备 UID 返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	5	0	0	uint32_t[3]: DeviceID	Payload checksum

1.4 实时位姿

实现设置初始姿态、获取实时位姿、运动学参数等功能。

1.4.1 获取实时位姿 (GetPose)

获取实时位姿 (GetPose)，下发的指令包格式如表格 21 所示，返回指令包格式如表格 22 所示。

表格 21 获取实时位姿指令包

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

表格 22 获取实时位姿返回指令包

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

程序 1 Pose 定义

```
typedef struct tagPose {
    float x;           //机械臂坐标系 x
    float y;           //机械臂坐标系 y
    float z;           //机械臂坐标系 z
    float r;           //机械臂坐标系 r
    float jointAngle[4]; //机械臂 4 轴(底座、大臂、小臂、末端)角度
} Pose;
```

1.4.2 重设实时姿态 (ResetPose)

重设实时姿态 (ResetPose)，下发的指令包格式如表格 23 所示，返回指令包格式如表格 24 所示。

表格 23 重设实时位姿指令包

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

表格 24 重设实时位姿返回指令包

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

说明

manual 为 0 时，自动重设姿态，不用传入 rearArmAngle 及 frontArmAngle；manual 为 1 时，传入 rearArmAngle（大臂角度）和 frontArmAngle（小臂角度）。

1.4.3 获取滑轨实时位姿（GetPoseL）

获取滑轨实时位姿（GetPoseL），下发的指令包格式如表格 25 所示，返回指令包格式如表格 26 所示。

表格 25 获取滑轨实时位姿指令包

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

表格 26 获取滑轨实时位姿返回指令包

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

1.5 报警功能

1.5.1 获取报警状态（GetAlarmsState）

获取报警状态（GetAlarmsState），下发的指令包格式如表格 27 所示，返回指令包格式如表格 28 所示。

表格 27 获取系统报警状态指令包

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

表格 28 获取系统报警状态返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	20	0	0	uint8_t[16]:alarmsState	Payload checksum

说明

数组 alarmsState 中的每一个字节可以标识 8 个报警项的报警状态，且 MSB（Most Significant Bit）在高位，LSB（Least Significant Bit）在低位。每个位的具体含义请参考报警说明相关文档。

1.5.2 清除系统的所有报警（ClearAllAlarmsState）

清除系统的所有报警（ClearAllAlarmsState），下发的指令包格式如表格 29 所示，返回指令包格式如表格 30 所示。

表格 29 清除系统报警状态指令包

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

表格 30 清除系统报警状态返回指令包

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

1.6 回零功能

该部分为回零位功能，包括设置零位参数、获取零位参数、设置回零位命令。机械臂默认零位为(0°, 45°, 45°, 0°)所对应的坐标值，用户可根据自己需求调用 SetHOMEParams 重新设置零位坐标。执行回零命令后，机械臂会运动到设置的零位位置。



注意

回零过程中机械臂指示灯为蓝色闪烁，运动到零位附近时会进行微调，蜂鸣器响并且指示灯变绿色后表明回零命令执行成功。

1.6.1 设置和获取零位参数（Set/Get HOMEParams）

- 设置零位参数（SetHOMEParams），下发的指令包格式如表格 31 所示，返回指令包格式如表格 32 所示；

表格 31 设置零位参数指令包

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

表格 32 设置零位参数返回指令包

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

- 获取零位参数（GetHOMEParams），下发的指令包格式如表格 33 所示，返回指令包格式如表格 34 所示。

表格 33 获取零位参数指令包

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

表格 34 获取零位参数返回指令包

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

					程序 2)	
--	--	--	--	--	-------	--

程序 2 HOMEParams 定义

```
typedef struct tagHOMEParams {
    float x;           //机械臂坐标系 x
    float y;           //机械臂坐标系 y
    float z;           //机械臂坐标系 z
    float r;           //机械臂坐标系 r
} HOMEParams;
```

1.6.2 执行回零命令（SetHOMECmd）

执行回零（SetHOMECmd），下发的指令包格式如表格 35 所示，返回指令包格式如表格 36 所示。

表格 35 执行回零指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	31	1	1 or 0	HOMECmd（见程序 3）	Payload checksum

表格 36 执行回零返回指令包

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

程序 3 HOMECmd 定义

```
typedef struct tagHOMECmd {
    uint32_t reserved; // 预留未来使用
} HOMECmd;
```

1.6.3 执行和获取自动调平（Set/Get AutoLeveling）

- 执行自动调平（Set AutoLeveling），下发的指令包格式如，所示表格 37，返回指令包格式如表格 38 所示。

表格 37 设置零位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	30	1	1 or 0	AutoLeveling（见程序 4）	Payload checksum

表格 38 设置零位参数返回指令包

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

- 获取自动调平结果（GetAutoLeveling），下发的指令包格式如表格 39 所示，返回指令包格式如表格 40 所示。

表格 39 获取零位参数指令包

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

表格 40 获取零位参数返回指令包

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

程序 4 AutoLevelingParams 定义

```
typedef struct tagAutoLevelingParams {
    uint8_t IsAutoleveling;    //调平执行标记
    float Accuracy;           //回零精度
} AutoLevelingParams;
```

1.7 手持示教功能

手持示教功能的指令，用于手持示教相关的命令配置与信息获取，包括使能/禁止手持示教模式、获取手持示教使能信息、获取手持示教是否有新的点增加。

1.7.1 设置和获取手持示教触发模式（Set/Get HHTTrigMode）

- 设置手持示教时的存点触发模式（SetHHTTrigMode），下发的指令包格式如表格 41 所示返回指令包格式如表格 42 所示。

表格 41 设置手持示教触发模式指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	40	1	0	HHTTrigMode（见程序 5）	Payload checksum

表格 42 设置手持示教触发模式返回指令包

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

- 获取手持示教时的存点触发模式（GetHHTTrigMode），下发的指令包格式如表格 43 所示，返回指令包格式如表格 44 所示。

表格 43 获取手持示教触发模式指令包

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

表格 44 获取手持示教触发模式返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	40	1	0	HHTTrigMode（见程序 5）	Payload checksum

程序 5 HHTTrigMode 的定义

```
typedef enum tagHHTTrigMode {
```

```

TriggeredOnKeyReleased,      //按键释放时更新

TriggeredOnPeriodicInterval  //定时更新

} HHTTrigMode;

```

1.7.2 设置和获取触发模式输出使能/禁止 (Set/Get HHTTrigOutputEnabled)

- 设置手持示教触发输出使能状态 (SetHHTTrigOutputEnabled)，下发的指令包格式如表格 45 所示，返回指令包格式如表格 46 所示；

表格 45 设置触发模式输出使能/禁止指令包

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

表格 46 设置触发模式输出使能/禁止返回指令包

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

- 获取触发输出使能/禁止状态 (GetHHTTrigOutputEnabled)，下发的指令包格式如表格 47 所示，返回指令包格式如表格 48 所示。

表格 47 获取触发模式输出使能/禁止指令包

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

表格 48 获取触发模式输出使能/禁止返回指令包

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

1.7.3 获取触发输出状态 (GetHHTTrigOutput)

获取触发输出状态 (GetHHTTrigOutput)，下发的指令包格式如表格 49 所示，返回指令包格式如表格 50 所示。

表格 49 获取触发输出状态指令包

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

表格 50 获取触发输出状态返回指令包

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

1.8 末端执行器

1.8.1 设置和获取末端执行器参数 (Set/Get EndEffectorParams)

- 设置末端执行器参数 (SetEndEffectorParams)，下发的指令包格式如表格 51 所示，返回指令包格式如表格 52 所示。

表格 51 设置末端执行器参数指令包

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

表格 52 设置末端执行器参数返回指令包

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

- 获取末端执行器参数 (GetEndEffectorParams)，下发的指令包格式如所示，返回指令包格式如表格 54 所示。

表格 53 获取末端执行器参数指令包

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

表格 54 获取末端执行器参数返回指令包

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

程序 6 EndEffectorParams 定义

```
typedef struct tagEndEffectorParams {
    float xBias;           //末端 x 轴方向长度
    float yBias;           //末端 y 轴方向长度
    float zBias;           //末端 z 轴方向长度
} EndEffectorParams;
```

1.8.2 设置和获取激光输出（Set/Get EndEffectorLaser）

- 设置激光开关（SetEndEffectorLaser），下发的指令包格式如表格 55 所示，返回指令包格式如表格 56 所示；

表格 55 设置激光开关指令包

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

表格 56 设置激光开关返回指令包

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

说明

控制是否使能（isCtrlEnabled），激光是否开启（isOn）。

- 获取激光开关状态（GetEndEffectorLaser），下发的指令包格式如表格 57 所示，返回指令包格式如表格 58 所示。

表格 57 获取激光开关状态指令包

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

表格 58 获取激光开关状态指令包

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

说明

控制是否使能 (isCtrlEnabled)，激光是否开启 (isOn)。

1.8.3 设置和获取吸盘输出 (Set/Get EndEffectorSuctionCup)

- 设置吸盘吸放 (SetEndEffectorSuctionCup)，下发的指令包格式如表格 59 所示，返回指令包格式如表格 60 所示；

表格 59 设置吸盘吸放指令包

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

表格 60 设置吸盘吸放返回指令包

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

说明

控制是否使能 (isCtrlEnabled)，吸盘是否吸住 (isSucked)。

- 获取吸盘吸放状态 (GetEndEffectorSuctionCup)，下发的指令包格式如表格 61 所示，返回指令包格式如表格 62 所示。

表格 61 获取吸盘吸放状态指令包

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

表格 62 获取吸盘吸放状态返回指令包

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

说明

控制是否使能 (isCtrlEnabled)，吸盘是否吸住 (isSucked)。

1.8.4 设置和获取爪子输出 (Set/Get EndEffectorGripper)

- 设置爪子抓住/释放 (SetEndEffectorGripper)，下发的指令包格式如表格 63 所示，返回指令包格式如表格 64 所示。

表格 63 设置爪子抓住/释放指令包

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

表格 64 设置爪子抓住/释放返回指令包

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

说明

控制是否使能 (isCtrlEnabled)，爪子是否抓住 (isGripped)。

- 获取爪子夹住状态 (SetEndEffectorGripper)，下发的指令包格式如表格 65 所示，返回指令包格式如表格 66 所示。

表格 65 获取爪子夹住状态指令包

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

表格 66 获取爪子夹住状态返回指令包

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

说明

控制是否使能（isCtrlEnabled），爪子是否抓住（isGripped）。

1.9 JOG 功能

设置/获取其中包括关节参数、坐标系参数，点动公共参数和执行点动功能。

1.9.1 设置和获取关节点动参数（Set/Get JOGJointParams）

- 设置关节点动参数（SetJOGJointParams），下发的指令包格式如表格 67 所示，返回指令包格式如表格 68 所示。

表格 67 设置关节点动参数指令包

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

表格 68 设置关节点动参数返回指令包

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

说明

在示教的关节运动中，需要设定关节的速度和加速度参数，这一组指令都是与此相关的指令在关节运动的时候需要预先设定好。此指令中会设置四个关节的速度和加速度。

- 获取关节点动参数（GetJOGJointParams），下发的指令包格式如表格 69 所示。返回指令包格式如表格 70。

表格 69 获取关节点动参数指令包

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

表格 70 获取关节点动参数返回指令包

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

程序 7 JOGJointParams 定义

```
typedef struct tagJOGJointParams{  
    float velocity[4];      //4 轴关节速度  
    float acceleration[4];  //4 轴关节加速度  
}JOGJointParams;
```

1.9.2 设置和获取坐标轴点动参数（Set/Get JOGCoordinateParams）

- 设置坐标系参数（SetJOGCoordinateParams），下发的指令包格式如表格 71 所示，返回指令包格式如表格 72 所示；

表格 71 设置坐标系参数指令包

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

表格 72 设置坐标系参数返回指令包

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

说明

此指令与单关节运动参数指令类似不同的是，本条指令设置的是坐标系的参数，分别为 X、Y、Z、R 轴的速度和加速度。

- 获取坐标轴点动参数（GetJOGCoordinateParams），下发的指令包格式如表格 73 所示，返回指令包格式如表格 74 所示。

表格 73 获取坐标轴点动参数指令包

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

表格 74 获取坐标轴点动参数返回指令包

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

程序 8 JOGCoordinateParams 定义

```
typedef struct tagJOGCoordinateParams {
    float velocity[4];      //4 轴坐标轴 (x,y,z,r) 速度
    float acceleration[4];  //4 轴坐标轴 (x,y,z,r) 加速度
} JOGCoordinateParams;
```

1.9.3 设置和获取点动公共参数（Set/Get JOGCommonParams）

- 设置点动公共参数（SetJOGCommonParams），下发的指令包格式如返回指令包格式如表格 75 所示，返回指令包格式如表格 76 所示。

表格 75 设置点动公共参数指令包

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

表格 76 设置点动公共参数返回指令包

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

- 获取点动公共参数（GetJOGCommonParams），下发的指令包格式如返回指令包格式如表格 77 所示，返回指令包格式如表格 78 所示。

表格 77 获取点动公共参数指令包

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

表格 78 获取点动公共参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	72	0	0	JOGCommonParams (见程序 9)	Payload checksum

程序 9 JOGCommonParams 定义

```
typedef struct tagJOGCommonParams {
    float velocityRatio;    //速度比例，关节点动和坐标轴点动共用
    float accelerationRatio; //加速度比例，关节点动和坐标轴点动共用
} JOGCommonParams;
```

1.9.4 执行点动功能（SetJOGCmd）

执行点动功能，下发的指令包格式如表格 79所示，返回指令包格式如表格 80所示。

表格 79 执行点动功能指令包

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

表格 80 执行点动功能返回指令包

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

程序 10 JOGCmd 定义

```
typedef struct tagJOGCmd {  
    uint8_t isJoint;    //点动方式 0—坐标轴点动 1—关节点动  
    uint8_t cmd;        //点动命令（取值范围 0~8）  
}JOGCmd;  
  
//点动命令详细说明  
enum {  
    IDEL,                //无效状态  
    AP_DOWN,            // X+/Joint1+  
    AN_DOWN,            // X-/Joint1-  
    BP_DOWN,            // Y+/Joint2+  
    BN_DOWN,            // Y-/Joint2-  
    CP_DOWN,            // Z+/Joint3+  
    CN_DOWN,            // Z-/Joint3-  
    DP_DOWN,            // R+/Joint4+  
    DN_DOWN             // R-/Joint4-  
};
```

1.9.5 设置和获取滑轨 L 点动参数（Set/Get JOGLParams）

- 设置滑轨 L 点动参数（SetJOGLParams），下发的指令包格式如表格 81 所示，返回指令包格式如表格 82 所示；

表格 81 设置滑轨 L 点动参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	74	1	1 or 0	JOGLParams（见程序 11）	Payload checksum

表格 82 设置滑轨 L 点动参数返回指令包

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

说明

在示教的关节运动中，需要设定关节的速度和加速度参数，这一组指令都是与此相关的指令在关节运动的时候需要预先设定好。此指令中会设置四个关节的速度和加速度。

- 获取滑轨点动参数（GetJOGLParams），下发的指令包格式如表格 83 所示，返回指令包格式如表格 84 所示。

表格 83 获取滑轨 L 点动参数指令包

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

表格 84 获取滑轨 L 点动参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	74	0	0	JOGLParams（见程序 11）	Payload checksum

程序 11 JOGLParams 定义

```
typedef struct tagJOGLParams{
    float velocity;        //滑轨关节速度
    float acceleration;    //滑轨关节加速度
}JOGLParams;
```

1.10 再现(PTP)功能

再现功能的指令，用于再现相关的运动设置和配置。其中包括关节参数、坐标系参数，比例参数和其他相关的参数。

1.10.1 设置和获取关节点位参数（Set/Get PTPJointParams）

这两条命令用于设置和获取再现速度参数，包括单关节再现速度加速度以及笛卡尔坐标系下的速度加速度，此命令设置的速度仅适用于再现运动，对于 JOG 功能无效。

- 设置关节点位参数（SetPTPJointParams），用于控制再现运动的速度实现运动的快慢控制。下发的指令包格式如表格 85 所示，返回指令包格式如表格 86 所示。

表格 85 设置关节点位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	1	1 or 0	PTPJointParams（见程序 12）	Payload checksum

表格 86 设置关节点位参数返回指令包

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

- 获取关节点位参数（GetPTPJointParams），下发的指令包格式如表格 87 所示，返回指令包格式如表格 88 所示。

表格 87 获取关节点位参数指令包

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

表格 88 获取关节点位参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+32	80	0	0	PTPJointParams（见程序12）	Payload checksum

程序 12 PTPJointParams 定义

```
typedef struct tagPTPJointParams {
    float velocity[4];      //PTP 模式下 4 轴关节速度
    float acceleration[4];  //PTP 模式下 4 轴关节加速度
} PTPJointParams;
```

1.10.2 设置和获取坐标轴点位参数（Set/Get PTPCoordinateParams）

- 设置坐标轴点位参数（SetPTPCoordinateParams），下发的指令包格式如表格 89 所示，返回指令包格式如表格 90 所示；

表格 89 设置坐标轴点位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	1	1 or 0	PTPCoordinateParams（见程序13）	Payload checksum

表格 90 设置坐标轴点位参数返回指令包

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

- 获取坐标轴点位参数（GetPTPCoordinateParams），下发的指令包格式如表格 91 所示，返回指令包格式如表格 92 所示。

表格 91 获取坐标轴点位参数指令包

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

表格 92 获取坐标轴点位参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	81	0	0	PTPCoordinateParams（见程序13）	Payload checksum

程序 13 PTPCoordinateParams 定义

```
typedef struct tagPTPCoordinateParams {
    float xyzVelocity;      //PTP 模式下 xyz 3 轴坐标轴速度
    float rVelocity;        //PTP 模式下末端速度
    float xyzAcceleration;  //PTP 模式下 xyz 3 轴坐标轴加速度
    float rAcceleration;    //PTP 模式下末端加速度
} PTPCoordinateParams;
```

1.10.3 设置和获取门型模式点位参数（Set/Get PTPJumpParams）

- 设置门型模式点位参数（SetPTPJumpParams），下发的指令包格式如表格 93 所示，返回指令包格式如表格 94 所示。

表格 93 设置门型模式点位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	82	1	1 or 0	PTPJumpParams（见程序 14）	Payload checksum

表格 94 设置门型模式点位参数返回指令包

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

- 获取门型模式点位参数（GetPTPJumpParams），下发的指令包格式如表格 95 所示，返回指令包格式如表格 96 所示。

表格 95 获取门型模式点位参数指令包

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

表格 96 获取门型模式点位参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	82	0	0	PTPJumpParams（见程序 14）	Payload checksum

程序 14 PTPJumpParams 定义

```
typedef struct tagPTPJumpParams {
    float jumpHeight;           //门型模式运动抬升距离
    float zLimit;              //门型模式运动最大抬升高度限制
} PTPJumpParams;
```

1.10.4 设置和获取点位公共参数 (Set/Get PTPCommonParams)

- 设置点位公共参数 (SetPTPCommonParams)，下发的指令包格式如表格 97 所示，返回指令包格式如表格 98 所示；

表格 97 设置点位公共参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	1	1 or 0	PTPCommonParams (见程序 15)	Payload checksum

表格 98 设置点位公共参数返回指令包

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

- 获取点位公共参数 (GetPTPCommonParams)，下发的指令包格式如表格 99 所示，返回指令包格式如表格 100 所示。

表格 99 获取点位公共参数指令包

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

表格 100 获取点位公共参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	83	0	0	PTPCCommonParams (见程序 15)	Payload checksum

程序 15 PTPCommonParams 定义

```
typedef struct tagPTPCCommonParams {
    float velocityRatio;    //PTP 模式速度比例，关节和坐标轴模式共用
    float accelerationRatio; //PTP 模式加速度比例，关节和坐标轴模式共用
} PTPCommonParams;
```

1.10.5 执行点位功能（SetPTPCmd）

执行点位功能（PTPCmd），下发的指令包格式如表格 101所示，返回指令包格式如表格 102所示。

表格 101 执行点位功能指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+17	84	1	1 or 0	PTPCmd（见程序 16）	Payload checksum

表格 102 执行点位功能返回指令包

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

程序 16 PTPCmd 定义

```
typedef struct tagPTPCmd {
    uint8_t ptpMode; //PTP 模式 (取值范围 0~8)

    float x;          //x,y,z,r 为 ptpMode 运动方式的参数，可为坐标、
                    //关节角度、或者坐标/角度增量

    float y;
    float z;
    float r;
} PTPCmd;
```

其中，ptpMode 取值如下：

```
enum {
    JUMP_XYZ,      //门型运动，参数为目标点坐标
    MOVJ_XYZ,      //关节运动，参数为目标点坐标
    MOVL_XYZ,      //直线运动，参数为目标点坐标
    JUMP_ANGLE,    //门型运动，参数为目标点关节角度
    MOVJ_ANGLE,    //关节运动，参数为目标点关节角度
    MOVL_ANGLE,    //直线运动，参数为目标点关节角度
    MOVJ_INC,      //关节运动增量模式，参数为目标点关节角度增量
    MOVL_INC,      //直线运动增量模式，参数为目标点坐标增量
    MOVJ_XYZ_INC,  //关节运动增量模式，参数为目标点坐标增量
    JUMP_MOVL_XYZ, //门型运动，平移时运动模式为 MOVL
};
```

1.10.6 设置和获取滑轨关节位参数（Set/Get PTPLParams）

这两条命令用于设置和获取滑轨的再现速度参数，包括速度加速度以及直线速度和加速度，此命令设置的速度仅适用于再现运动，对于示教运动是不起作用的。

- 设置滑轨关节位参数（SetPTPLParams），用于控制再现运动的速度实现运动的快慢控制。下发的指令包格式如表格 103 所示，返回指令包格式如表格 104 所示。

表格 103 设置滑轨关节位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	85	1	1 or 0	PTPLParams（见程序 17 ）	Payload checksum

表格 104 设置关节点位参数返回指令包

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

- 获取关节点位参数（GetPTPJointParams），下发的指令包格式如表格 105 所示，返回指令包格式如表格 106 所示。

表格 105 获取关节点位参数指令包

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

表格 106 获取关节点位参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8	85	0	0	PTPJointParams（见程序 17 ）	Payload checksum

程序 17 PTPJointParams 定义

```
typedef struct tagPTPJointParams {
    float velocity;          //PTP 模式下 4 轴关节速度
    float acceleration;      //PTP 模式下 4 轴关节加速度
} PTPLParams;
```

1.10.7 执行带滑轨点位功能（SetPTPWithLCmd）

执行带滑轨点位功能（SetPTPWithLCmd），下发的指令包格式如表格 107 所示，返回指令包格式如表格 108 所示。

表格 107 执行点位功能指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+21	86	1	1 or 0	PTPWithLCmd（见程序 18）	Payload checksum

表格 108 执行点位功能返回指令包

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

程序 18 PTPCmd 定义

```
typedef struct tagPTPWithLCmd {
    uint8_t ptpMode; //PTP 模式 (取值范围 0~9)
    float x;          //x,y,z,r 为 ptpMode 运动方式的参数，可为坐标
                     //关节角度、或者坐标/角度增量
    float y;
    float z;
    float r;
    float l;          滑轨运动距离
} PTPWithLCmd;
```

其中，ptpMode 取值如下：

```
enum {
    JUMP_XYZ,          //JUMP 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVJ_XYZ,          //MOVJ 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVL_XYZ,          //MOVL 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    JUMP_ANGLE,        //JUMP 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVJ_ANGLE,        //MOVJ 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVL_ANGLE,        //MOVL 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVJ_INC,          //MOVJ 模式，(x,y,z,r) 为关节坐标系下的坐标增量
    MOVL_INC,          //MOVL 模式，(x,y,z,r) 为笛卡尔坐标系下的坐标增量
    MOVJ_XYZ_INC,      //MOVJ 模式，(x,y,z,r) 为笛卡尔坐标系下的坐标增量
    JUMP_MOVL_XYZ,     //JUMP 模式，平时运动模式为 MOVL。(x,y,z,r) 为笛卡尔坐标系下的坐标增量
};
```

1.10.8 设置和获取门型模式点位参数 2 (Set/Get PTPJump2Params)

设置和获取滑轨门型移动参数 2，把抬升高度分成开始和结束，可以设置不同高度。

- 设置滑轨门型模式点位参数（SetPTPJumpLParams），下发的指令包格式如表格 109 所示，返回指令包格式如表格 110 所示；

表格 109 设置门型模式点位参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	87	1	1 or 0	PTPJump2Params（见 程序 19）	Payload checksum

表格 110 设置门型模式点位参数返回指令包

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

- 获取门型模式点位参数（GetPTPJumpParams），下发的指令包格式如表格 111 所示，返回指令包格式如表格 112 所示。

表格 111 获取门型模式点位参数指令包

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

表格 112 获取门型模式点位参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+12	87	0	0	PTPJump2Params （见程序19）	Payload checksum

程序 19 PTPJumpParams 定义

```
typedef struct tagPTPJump2Params {
    float startJumpHeight;    //门型模式开始抬升高度
    float endJumpHeight;      //门型模式结束抬升高度
}
```



```
float zLimit; //门型模式运动最大抬升高度限制
} PTPJump2Params;
```

1.10.9 执行平行输出点位功能（SetPTPPOCmd）

执行平行输出点位功能（SetPTPPOCmd），主要实现的是在运行PTP命令中间某一个时刻时可以对I/O进行操作，而不必在运动完成之后才能通过EIO的命令来控制I/O输出状态。

下发的指令包格式如表格 113所示，返回指令包格式如表格 114所示。

表格 113 执行平行输出点位功能指令包

Header	Len	Payload						Checksum
		ID	Ctrl		Params			
			rw	isQueued				
0xAA 0xAA	2+18+4*count	88	1	1 or 0	PTPCmd (见程序 20)	uint8_t count	POCmd (count) (见程序 21)	Payload checksum

表格 114 执行平行输出点位功能返回指令包

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

程序 20 PTPCmd 定义

```
typedef struct tagPTPCmd {
    uint8_t ptpMode; //PTP 模式 (取值范围 0~9)
    float x;          //x,y,z,r 为 ptpMode 运动方式的参数，可为坐标
    float y;          //关节角度、或者坐标/角度增量
    float z;
    float r;
} PTPPOCmd;
```

其中，ptpMode 取值如下：

```
enum {
    JUMP_XYZ,      //JUMP 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVJ_XYZ,      //MOVJ 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVL_XYZ,      //MOVL 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
```

```

JUMP_ANGLE,    //JUMP 模式, (x,y,z,r) 为关节坐标系下的目标点坐标
MOVJ_ANGLE,    //MOVJ 模式, (x,y,z,r) 为关节坐标系下的目标点坐标
MOVL_ANGLE,    //MOVL 模式, (x,y,z,r) 为关节坐标系下的目标点坐标
MOVJ_INC,      //MOVJ 模式, (x,y,z,r) 为关节坐标系下的坐标增量
MOVL_INC,      //MOVL 模式, (x,y,z,r) 为笛卡尔坐标系下的坐标增量
MOVJ_XYZ_INC,  //MOVJ 模式, (x,y,z,r) 为笛卡尔坐标系下的坐标增量
JUMP_MOVL_XYZ, //JUMP 模式, 平移时运动模式为 MOVL。(x,y,z,r) 为笛卡尔坐标系下的坐标增量
};

```

程序 21 POCmd 定义

结构体 POCmd 定义如下:

```

typedef struct tagPOCmd{
    uint8_t ratio;    //运动完成百分比
    uint16_t address //EIO 编号
    uint8_t level //输出状态
}POCmd;

```

1.10.10 执行滑轨平行输出点位功能 (SetPTPPOWithLCmd)

执行滑轨平行输出点位功能 (SetPTPPOWithLCmd), 主要实现的是在运行PTP命令中间某一个时刻时可以对I/O进行操作, 而不必在运动完成之后才能通过EIO的命令来控制I/O输出状态。

下发的指令包格式如表格 115所示, 返回指令包格式如表格 116所示。

表格 115 执行滑轨平行输出点位功能指令包

Header	Len	Payload						Checksum
		ID	Ctrl		Params			
			rw	isQueued				
0xAA 0xAA	2+22+4*count	89	1	1 or 0	PTPCmd (见程序 22)	uint8_t count	POCmd (count)	Payload checksum

表格 116 执行滑轨平行输出点位功能返回指令包

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

程序 22 PTPCmd 和 POCmd 定义

```
typedef struct tagPTPPOCmd {
    uint8_t ptpMode; //PTP 模式 (取值范围 0~9)

    float x;          //x,y,z,r,为 ptpMode 运动方式的参数，可为坐标、
                    //关节角度、或者坐标/角度增量

    float y;
    float z;
    float r;
    float l;          //滑轨运动距离
} PTPPOCmd;

//其中，ptpMode 取值如下：

enum {
    JUMP_XYZ,        //JUMP 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVJ_XYZ,        //MOVJ 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    MOVL_XYZ,        //MOVL 模式，(x,y,z,r) 为笛卡尔坐标系下的目标点坐标
    JUMP_ANGLE,      //JUMP 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVJ_ANGLE,      //MOVJ 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVL_ANGLE,      //MOVL 模式，(x,y,z,r) 为关节坐标系下的目标点坐标
    MOVJ_INC,        //MOVJ 模式，(x,y,z,r) 为关节坐标系下的坐标增量
    MOVL_INC,        //MOVL 模式，(x,y,z,r) 为笛卡尔坐标系下的坐标增量
    MOVJ_XYZ_INC,    //MOVJ 模式，(x,y,z,r) 为笛卡尔坐标系下的坐标增量
    JUMP_MOVL_XYZ,   //JUMP 模式，平移时运动模式为 MOVL。(x,y,z,r) 为笛卡尔坐标系下的坐标增量
};

结构体 POCmd 如下：

typedef struct tagPOCmd{
    Uint8_t ratio;    //运动完成百分比

    Uint16_t address //EIO 编号

    Uint8_t level //输出状态
}POCmd;
```

1.11 CP 功能

连续轨迹功能的指令，用于连续轨迹相关的运动设置和配置。其中包括关节参数、坐标系参数、功能设置参数等。连续轨迹功能和上位机 Dobot CP 对应，可以实现写字、画画、激光雕刻等需要连续轨迹的功能。

1.11.1 设置和获取连续轨迹功能参数（Set/Get CPParams）

这两条命令用于设置和获取连续轨迹参数，包括预设加速度、关节速度及加速度，此命令设置的速度仅适用于连续轨迹运动。

- 设置连续轨迹运动参数（SetCPParams），用于控制连续轨迹运动的速度实现运动的快慢控制，下发的指令包格式如表格 117 所示，返回指令包格式如表格 118 所示；

表格 117 设置连续轨迹运动参数指令包

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

表格 118 设置连续轨迹运动参数返回指令包

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

- 获取连续运动轨迹参数（GetCPParams），下发的指令包格式如表格 119 所示，返回指令包格式如表格 120 所示。

表格 119 获取连续运动轨迹参数指令包

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

表格 120 获取连续运动轨迹参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+13	90	0	0	CPPParams（见程序 23 ）	Payload checksum

程序 23 CPParams 定义

```
typedef struct tagCPParams {
    float planAcc;           //规划加速度最大值
```

```

float junctionVel;    //拐角加速度最大值

union {
    float acc;        //实际加速度最大值，非实时模式下使用
    float period;     //插补周期，实时模式下使用
};

uint8_t realTimeTrack; //0—非实时模式； 1—实时模式
} CPPParams;

```

1.11.2 执行连续轨迹功能（SetCPCmd）

执行连续轨迹功能（SetCPCmd），下发的指令包格式如表格 121 所示，返回指令包格式如表格 122 所示。

表格 121 执行连续轨迹功能指令包

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

表格 122 执行连续轨迹功能返回指令包

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

程序 24 CPCmd 定义

```

typedef struct tagCPCmd {
    uint8_t cpMode;    //CP 模式 0-相对模式 1-绝对模式
    float x;           //x 坐标增量 / x 轴坐标
    float y;           //y 坐标增量 / y 轴坐标
    float z;           //z 坐标增量 / z 轴坐标
    union {
        float velocity; // Reserved
        float power;    //激光功率
    }
} CPCmd;

```

1.11.3 执行连续轨迹灰度雕刻功能（SetCPLECmd）

执行连续轨迹灰度雕刻功能（SetCPLECmd），下发的指令包格式如表格 123所示，返回指令包格式如表格 124所示。

表格 123 执行连续轨迹功能指令包

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

表格 124 执行连续轨迹功能返回指令包

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

程序 25 CPCmd 定义

```
typedef struct tagCPCmd {
    uint8_t cpMode;        //CP 模式 0-相对模式 1-绝对模式
    float x;               //x 坐标增量(相对模式) / x 轴坐标(绝对模式)
    float y;               //y 坐标增量(相对模式) / y 轴坐标(绝对模式)
    float z;               //z 坐标增量(相对模式) / z 轴坐标(绝对模式)
    union {
        float velocity;    //预留
        float power;       //激光功率 0~100
    }
} CPCmd;
```

1.12 ARC 功能

1.12.1 设置和获取圆弧插补功能参数（Set/Get ARCPParams）

- 设置圆弧插补功能参数（SetARCPParams），下发的指令包格式如表格 125 所示，返回指令包格式如表格 126 所示。

表格 125 设置圆弧插补功能参数指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	1	1 or 0	ARCPParams（见程序 26）	Payload checksum

表格 126 设置圆弧插补功能参数返回指令包

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

- 设置圆弧插补功能参数 (GetARCPParams)，下发的指令包格式如表格 127 所示，返回指令包格式如表格 128 所示。

表格 127 获取圆弧插补功能参数指令包

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

表格 128 获取圆弧插补功能参数返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+16	100	0	0	ARCPParams（程序 26）	Payload checksum

程序 26 ARCPParams 定义

```
typedef struct tagARCPParams {
    float xyzVelocity;      //圆弧运动 xyz 三坐标轴速度
    float rVelocity;        //圆弧运动末端旋转速度
    float xyzAcceleration; //圆弧运动 xyz 三坐标轴加速度
    float rAcceleration;    //圆弧运动末端旋转加速度
} ARCPParams;
```

1.12.2 执行圆弧插补功能（SetARCCmd）

执行圆弧插补功能（SetARCCmd），下发的指令包格式如表格 129 所示，返回指令包格式如表格 130 所示。

表格 129 执行圆弧插补功能指令包

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

表格 130 执行圆弧插补功能返回指令包

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

程序 27 ARCCmd 定义

```
typedef struct tagARCCmd {
    struct {
        float x;
        float y;
        float z;
        float r;
    } cirPoint;    //圆弧上任一点坐标

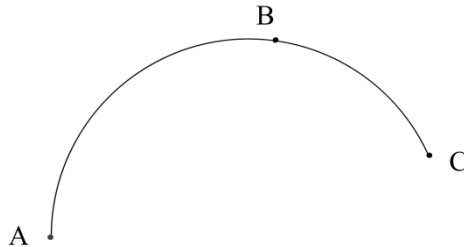
    struct {
        float x;
        float y;
        float z;
        float r;
    } toPoint;    //圆弧结束点坐标
} ARCCmd;
```

说明

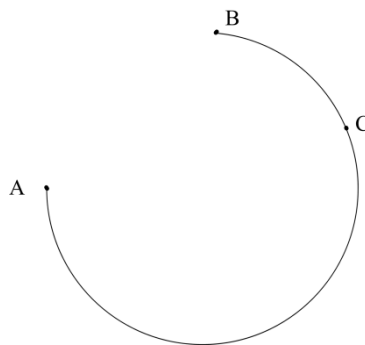
- 圆弧轨迹是空间的圆弧，由当前点、圆弧上任一点和圆弧结束点三点共同确定的；
- 圆弧总是从起点经过圆弧上一点再到结束点。

圆弧轨迹示例：

- A 为当前点， B 为圆弧上任一点， C 为结束点；



- A 为当前点， C 为圆弧上任一点， B 为结束点。



1.13 WAIT 功能

1.13.1 执行时间等待功能（SetWAITCmd）

执行时间等待功能（SetWAITCmd），下发的指令包格式如表格 131所示，返回指令包格式如表格 132所示。

表格 131 执行时间等待功能指令包

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

表格 132 执行时间等待功能返回指令包

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

程序 28 WAITCmd 定义

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

1.14 TRIG 功能

1.14.1 执行触发功能（SetTRIGCmd）

执行触发功能（SetTRIGCmd），下发的指令包格式如表格 133所示，返回指令包格式如表格 134所示。

表格 133 执行触发功能指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	120	1	1 or 0	TRIGCmd（见程序 29）	Payload checksum

表格 134 执行触发功能返回指令包

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

程序 29 TrigCmd 定义

```
typedef struct tagTRIGCmd {
    uint8_t address;    //I/O 地址。取值范围：1~20
    uint8_t mode;        //触发模式。0：电平触发。1：A/D 触发
    uint8_t condition;   //触发条件。电平：0，等于。1，不等于
                        //A/D：0，小于。1，小于等于
                        //2，大于等于。3，大于
    uint16_t threshold;  //触发阈值：电平，0 或 1。A/D：0~4095
} TRIGCmd;
```

1.15 EIO 功能

EIO (Extended I/O), 即扩展 I/O。在 Dobot 的产品中, 其控制器中带有 EIO 并且绝大部分都有复用功能。

EIO 具体编址信息请参考 EIO 说明相关文档。

1.15.1 设置和读取 I/O 复用 (Set/Get IOMultiplexing)

- 设置 I/O 口复用 (SetIOMultiplexing), 下发的指令包格式如表格 135 所示, 返回指令包格式如表格 136 所示。

表格 135 设置 I/O 口复用指令包

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

表格 136 设置 I/O 口复用返回指令包

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

- 读取复用 I/O (SetIOMultiplexing), 下发的指令包格式如表格 137 所示, 返回指令包格式如表格 138 所示。

表格 137 读取复用 I/O 指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	130	0	0	uint8_t address	Payload checksum

表格 138 读取复用 I/O 返回指令包

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

程序 30 IOMultiplexing 定义

```
typedef struct tagIOMultiplexing {
    uint8_t address;           //EIO 地址（取值范围 1~20）
    uint8_t multiplex;         //EIO 功能
} IOMultiplexing;
```

其中 multiplex 支持的取值如下所示如程序 31：

程序 31 IOFunction 定义

```
typedef enum tagIOFunction {
    IOFunctionDummy,          //不配置功能
    IOFunctionDO,              //IO 输出
    IOFunctionPWM,             //PWM 输出
    IOFunctionDI,              //IO 输入
    IOFunctionADC,             //AD 输入
    IOFunctionDIPU,            //上拉输入
    IOFunctionDIPD             //下拉输入
} IOFunction;
```

1.15.2 设置和读取 I/O 输出电平（Set/Get IODO）

- 设置 I/O 口输出电平（SetIODO），下发的指令包格式如表格 139 所示，返回指令包格式如表格 140 所示。

表格 139 设置 I/O 口输出电平指令包

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

表格 140 设置 I/O 口输出电平返回指令包

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

- 读取 I/O 输出电平（GetIODO），下发的指令包格式如表格 141 所示，返回指令包格式如表格 142 所示。

表格 141 读取 I/O 口输出电平指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	131	0	0	uint8_t address	Payload checksum

表格 142 读取 I/O 口输出电平返回指令包

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

程序 32 IODO 定义

```
typedef struct tagIODO {
    uint8_t address;      //EIO 地址（取值范围 1~20）
    uint8_t level;        //输出电平 0-低电平 1-高电平
} IODO;
```

1.15.3 设置和读取 PWM 输出 (Set/Get IOPWM)

- 设置 I/O PWM 输出 (SetIOPWM)，下发的指令包格式如表格 143 所示，返回指令包格式如表格 144 所示；

表格 143 设置 I/O 口 PWM 输出指令包

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

表格 144 设置 I/O 口 PWM 输出返回指令包

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

- 读取 I/O PWM (GetIOPWM)，下发的指令包格式如表格 145 所示，返回指令包格式如表格 146。

表格 145 读取 I/O 口 PWM 指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	132	0	0	uint8_t address	Payload checksum

表格 146 读取 I/O 口 PWM 返回指令包

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

程序 33 IOPWM 定义

```
typedef struct tagIOPWM {
    uint8_t address;        //EIO 地址（取值范围 1~20）
    float frequency;        //PWM 频率 10HZ~1MHz
    float dutyCycle;        //PWM 占空比 0~100
} IOPWM;
```

1.15.4 读取 I/O 输入电平（GetIODI）

读取 I/O 输入电平（GetIODI），下发的指令包格式如表格 147 所示，返回指令包格式如表格 148 所示。

表格 147 读取 I/O 输出电平指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	133	0	0	uint8_t address	Payload checksum

表格 148 读取 I/O 输出电平返回指令包

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

程序 34 IODI 定义

```
typedef struct tagIODI {
    uint8_t address;        //EIO 地址（取值范围 1~20）
}
```

```
uint8_t level;           //输入 IO 电平 0-低电平 1-高电平
}IODI;
```

1.15.5 读取 I/O 模数转换值 (GetIOADC)

读取 I/O 模数转换值 (GetIOADC)，下发的指令包格式如表格 149 所示，返回指令包格式如表格 150 所示。

表格 149 读取 I/O 模数转换值指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	134	0	0	uint8_t address	Payload checksum

表格 150 读取 I/O 模数转换值返回指令包

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

程序 35 IOADC 定义

```
typedef struct tagIOADC{
    uint8_t address;           //EIO 地址 (取值范围 1~20)
    uint16_t value;           //输入 ADC 值, 范围 0~4095
}IOADC;
```

1.15.6 设置扩展电机接口 (SetEMotor)

设置扩展电机接口 (SetEMotor)，下发的指令包格式如表格 151 所示，返回指令包格式如表格 152 所示；

表格 151 设置 I/O 口输出电平指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+6	135	1	1 or 0	EMotor（见程序 36）	Payload checksum

表格 152 设置 I/O 口输出电平返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		

0xAA 0xAA	2+8 or 2+0	135	1	1 or 0	uint64_t:queuedCmdIndex or Empty	Payload checksum
-----------	------------------	-----	---	--------	-------------------------------------	---------------------

程序 36 EMotor 定义

```
typedef struct tagEMotor{
    uint8_t index;          //取值范围 0/1 0-Stepper1 1-Stepper2
    uint8_t insEnabled;     //电机控制使能
    float speed;            //电机控制速度(脉冲个数每秒)
}EMotor;
```

1.15.7 设置和读取颜色传感器 (Set/Get ColorSensor)

- 设置颜色传感器 (SetColorSensor)，下发的指令包格式如表格 153 所示，返回指令包格式如表格 154 所示；

表格 153 设置颜色传感器指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	137	1	1 or 0	uint8_t : isEnabled; uint8_t: Port(见程序 37)	Payload checksum

表格 154 设置颜色传感器返回指令包

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

- 读取颜色传感器 (GetColorSensor)，下发的指令包格式如表格 155 所示，返回指令包格式如表格 156。

表格 155 读取颜色传感器指令包

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

表格 156 读取颜色传感器返回指令包

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

程序 37 Port 定义

```
typedef struct tagPort {
    uint8_t PORT_GP1;
    uint8_t PORT_GP2;
    uint8_t PORT_GP4;
    uint8_t PORT_GP5;
} Port;
```

程序 38 Color 定义

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

1.15.8 设置和读取红外开关 (Set/Get IRSwitch)

- 设置红外开关 (Set IRSwitch)，下发的指令包格式如表格 157 所示，返回指令包格式如表格 158 所示。

表格 157 设置红外开关指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+2	138	1	1 or 0	uint8_t : isEnabled; uint8_t IRPort (程序 39)	Payload checksum

表格 158 设置红外开关返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+8 or 2+0	138	1	1 or 0	int64_t:queuedCmdIndex or Empty	Payload checksum

- 读取红外开关（Get IRSwitch），下发的指令包格式如表格 159 所示，返回指令包格式如表格 160 所示。

表格 159 读取红外开关指令包

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

表格 160 读取红外开关返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+1	137	0	0	uint8_t state	Payload checksum

程序 39 IRPort 定义

```
typedef enum tagIRPort {
    uint8_t PORT_GP1;
    uint8_t PORT_GP2;
    uint8_t PORT_GP4;
    uint8_t PORT_GP5;
} IRPort;
```

1.16 校准(CAL)功能

由于角度传感器焊接、机器状态等原因，大小臂上的角度传感器可能存在一个静态偏差。我们可以通过各种手段（如调平、与标准源比较），得到此静态误差，并通过此 API 写入到设备中。

1.16.1 设置和读取角度传感器静态误差（Set/Get AngleSensorStaticError）

- 设置角度传感器静态误差（SetAngleSensorStaticError），下发的指令包格式如表格 161 所示，返回指令包格式如表格 162。

表格 161 设置角度传感器静态误差指令包

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

表格 162 设置角度传感器静态误差返回指令包

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

- 读取角度传感器静态误差（GetAngleSensorStaticError），下发的指令包格式如表格 163 所示，返回指令包格式如表格 164 所示。

表格 163 获取角度传感器静态误差指令包

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

表格 164 获取角度传感器静态误差返回指令包

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

1.17 WIFI 功能

1.17.1 设置和获取 WIFI 配置模式（Set/Get WiFiConfigMode）

- 设置 WIFI 配置模式（SetWiFiConfigMode），下发的指令包格式如表格 165 所示，返回指令包格式如表格 166 所示。

表格 165 设置 WIFI 配置模式指令包

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

表格 166 设置 WIFI 配置模式返回指令包

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

- 获取当前 WIFI 是否配置模式 (GetWIFIConfigMode)，下发的指令包格式如表格 167 所示，返回指令包格式如表格 168 所示。

表格 167 获取当前 WIFI 是否配置模式指令包

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

表格 168 获取当前 WIFI 是否配置模式返回指令包

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

1.17.2 设置和获取 SSID (Set/Get WIFISSID)

- 设置 SSID (SetWIFISSID)，下发的指令包格式如表格 169 所示，返回指令包如表格 170 所示。

表格 169 设置 SSID 指令包

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

表格 170 设置 SSID 返回指令包

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

- 获取当前设置 SSID (GetWIFISSID)，下发的指令包格式如表格 171 所示，返回指令包格式如表格 172 所示。

表格 171 获取当前设置 SSID 指令包

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

表格 172 获取当前设置 SSID 返回指令包

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

1.17.3 设置和获取网络密码 (Set/Get WIFIPassword)

- 设置网络密码 (SetWIFIPassword)，下发的指令包格式如表格 173 所示，返回指令包格式如表格 174 所示；

表格 173 设置网络密码指令包

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

表格 174 设置网络密码返回指令包

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

- 获取当前设置网络密码 (GetWIFIPassword)，下发的指令包格式如表格 175 所示，返回指令包格式如表格 176 所示。

表格 175 获取当前设置网络密码指令包

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

表格 176 获取当前设置网络密码返回指令包

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

1.17.4 设置和获取 IP 地址（Set/Get WIFIIPAddress）

- 设置 IP（SetWIFIIPAddress），下发的指令包格式如表格 177 所示，返回指令如表格 178 所示。

表格 177 设置 IP 指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	153	1	0	WIFIIPAddress（见程序 40）	Payload checksum

表格 178 设置 IP 返回指令包

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

- 获取当前设置 IP 地址（GetWIFIIPAddress），下发的指令包格式如表格 179 所示，返回指令如表格 180 所示。

表格 179 获取当前设置 IP 地址指令包

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

表格 180 获取当前设置 IP 地址指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+5	153	0	0	WIFIIPAddress（见程序 40）	Payload checksum

程序 40 WIFIIPAddress 定义

```
typedef struct tagWIFIIPAddress {
    uint8_t dhcp; //是否开启 DHCP, 0: 关闭, 1: 开启
    uint8_t addr[4]; //IP 地址分成四段, 每段取值范围为: 0~255
} WIFIIPAddress;
```

1.17.5 设置和获取子网掩码（Set/Get WIFINetmask）

- 设置子网掩码（SetWIFINetmask），下发的指令包格式如表格 181 所示，返回指令包格式如表格 182 所示。

表格 181 设置子网掩码指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	154	1	0	WIFINetmask（见 程序 41）	Payload checksum

表格 182 设置子网掩码返回指令包

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

- 获取当前设置子网掩码（GetWIFINetmask），下发的指令包格式如表格 183 所示，返回指令包格式如表格 184 所示。

表格 183 获取当前设置子网掩码指令包

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

表格 184 获取当前设置子网掩码返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	154	0	0	WIFINetmask（见 程序 41）	Payload checksum

程序 41 WIFINetmask 定义

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

1.17.6 设置和获取网关 (Set/Get WiFiGateway)

- 设置网关 (SetWiFiGateway)，下发的指令包格式如表格 185 所示，返回指令包格式如表格 186 所示；

表格 185 设置网关指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	155	1	0	WIFIGateway（见 程序 42）	Payload checksum

表格 186 设置网关返回指令包

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

- 获取当前设置网关 (GetWiFiGateway)，下发的指令包格式如表格 187 所示，返回指令包格式如表格 188 所示。

表格 187 获取当前设置网关指令包

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

表格 188 获取当前设置网关返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	155	0	0	WiFiGateway（见 程序 42）	Payload checksum

程序 42 WiFiGateway 定义

```
typedef struct tagWiFiGateway {
    uint8_t addr[4];
} WiFiGateway;
```


1.17.7 设置和获取 DNS (Set/Get WIFIDNS)

- 获取当前设置 DNS (GetWIFIDNS)，下发的指令包格式如表格 189 所示，返回指令包格式如表格 190 所示。

表格 189 设置 DNS 指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	156	1	0	WIFIDNS（见程序 43）	Payload checksum

表格 190 设置 DNS 返回指令包

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

- 获取当前设置 DNS (GetWIFIDNS)，下发的指令包格式如表格 191 所示，返回指令包格式如表格 192 所示。

表格 191 获取当前设置 DNS 指令包

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

表格 192 获取当前设置 DNS 返回指令包

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+4	156	0	0	WIFIDNS（见程序 43）	Payload checksum

程序 43 WIFIDNS 定义

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

1.17.8 获取 WIFI 连接状态（GetWIFIConnectStatus）

获取当前 WI-FI 模块的连接状态（GetWIFIConnectStatus），下发的指令包格式如表格 193 所示，返回指令包格式如表格 194 所示。

表格 193 获取当前 WI-FI 模块的连接状态指令包

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

表格 194 获取当前 WI-FI 模块的连接状态返回指令包

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

1.18 丢步功能

1.18.1 设置丢步检测功能（SetLostStepValue）

- 设置丢步检测偏离值（Set LostStepValue），下发的指令包格式如表格 195 所示，返回指令包格式如表格 196 所示；

表格 195 设置丢步检测幅度指令包

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

表格 196 设置 WIFI 配置模式返回指令包

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

1.18.2 执行丢步检测功能（SetLostStepCmd）

- 执行丢步检测功能（Set LostStepCmd），下发的指令包格式如表格 197 所示，返回指令包格式如表格 198 所示。

表格 197 设置执行丢步检测功能指令包

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

表格 198 设置执行丢步检测功能返回指令包

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

1.19 队列执行控制命令

队列执行控制命令主要是用于设置队列命令执行的相关参数，包括命令执行模式（在线/离线）、队列命令缓冲器当前状态、队列命令执行状态（TRUE/FALSE）、队列命令执行控制（START/PAUSE/STOP）。

1.19.1 启动指令队列运行（SetQueuedCmdStartExec）

启动指令队列运行（SetQueuedCmdStartExec），下发的指令包格式如表格 199 所示，返回指令包格式如表格 200 所示。

表格 199 启动指令队列运行指令包

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

表格 200 启动指令队列运行返回指令包

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

1.19.2 停止指令队列运行（SetQueuedCmdStopExec）

停止指令队列运行（SetQueuedCmdStopExec），下发的指令包格式如表格 201 所示，返回指令包格式如表格 202 所示。

回指令包格式如表格 202 所示。

表格 201 停止指令队列运行指令包

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

表格 202 停止指令队列运行返回指令包

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

1.19.3 强制停止指令队列运行 (SetQueuedCmdForceStopExec)

强制停止指令队列运行 (SetQueuedCmdForceStopExec)，下发的指令包格式如表格 203 所示，返回指令包格式如表格 204 所示。

表格 203 强制停止指令队列运行指令包

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

表格 204 强制停止指令队列运行返回指令包

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

1.19.4 启动指令队列下载 (SetQueuedCmdStartDownload)

启动指令队列下载 (SetQueuedCmdStartDownload)，下发的指令包格式如表格 205 所示，返回指令包格式如表格 206 所示。

表格 205 启动指令队列下载指令包

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

表格 206 启动指令队列下载返回指令包

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

说明

Dobot 的控制器支持将指令存储到控制器外部 Flash 中，而后可以通过控制器上的按键触发执行，也即脱机运行功能。

1.19.5 完成指令队列下载（SetQueuedCmdStopDownload）

完成指令队列下载（SetQueuedCmdStopDownload），下发的指令包格式如表格 207 所示，返回指令包格式如表格 208 所示。

表格 207 完成指令队列下载指令包

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

表格 208 完成指令队列下载返回指令包

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

1.19.6 清空指令队列（SetQueuedCmdClear）

清空指令队列（SetQueuedCmdClear），下发的指令包格式如表格 209 所示，返回指令包格式如表格 210 所示。

表格 209 清空指令队列指令包

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

表格 210 清空指令队列返回指令包

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

1.19.7 获取指令队列索引 (GetQueuedCmdCurrentIndex)

获取指令队列索引 (GetQueuedCmdCurrentIndex), 下发的指令包格式如表格 211 所示, 返回指令包格式如表格 212 所示。

表格 211 获取指令队列索引指令包

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

表格 212 获取指令队列索引返回指令包

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

说明

在 Dobot 控制器指令队列机制中, 有一个 64 位内部计数索引。当控制器每执行完一条命令式, 该计数器自动加一。通过该内部索引, 可以查询控制器已经执行了多少条队列指令, 以及当前执行到哪条指令 (指示运行进度时)。

1.19.8 获取指令队列剩余空间 (GetQueuedCmdLeftSpace)

获取指令队列剩余空间 (GetQueuedCmdLeftSpace), 下发的指令包格式如表格 213 所示, 返回指令包格式如表格 214 所示。

表格 213 获取指令队列剩余空间指令包

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

表格 214 获取指令队列剩余空间指令包

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

说明

在 Dobot 控制器指令队列机制中，有一个指令队列。在发送队列指令时，应先查询指令队列的剩余空间，若非零，才能向 Dobot 控制器发送队列指令。