

M5Stack Chain Key Protocol							All packets start with 0xAA 0x55 and end with 0x55 0xAA					V1 (Version)
指令集	Byte	0	1	2	3	4	5	6	7	8	9	10
		Length_low	Length_high	Index	Cmd	Data1	Data2	Data3	Data4	Data5	Data6	Data7
指令集	指令包格式	0x05+Num*3	0x00	Index_id	0x20	Index	Num	R	G	B	...	CRC
	应答包格式	0x04	0x00	Index_id	0x20	Operation_status	CRC					
设置RGB值	指令详解	<p>(1) 功能说明: 设置RGB值。            (2) 输入参数: Index_id (设备下标ID)、Index (RGB灯起始坐标)、Num (要设置的RGB数量)、RGB值            (3) 返回参数: Operation_status            (4) 指令代码: 0x20            注1: Operation_status 操作状态            0: 操作失败            1: 操作成功            注2: Index是RGB灯起始坐标, 从0开始, Key设备有两个RGB灯, 在定义上两个RGB是同色的, 所以Index = 0, Num = 1</p>										
获取RGB值	指令包格式	0x05	0x00	Index_id	0x21	Index	Num	CRC				
	应答包格式	0x04+Num*3/0x04	0x00	Index_id	0x21	Operation_status	R	G	B	...	CRC	
获取RGB值	指令详解	<p>(1) 功能说明: 获取RGB值。            (2) 输入参数: Index_id (设备下标ID)、Index (RGB灯起始坐标)、Num (要读取的RGB灯数量)            (3) 返回参数: RGB值            (4) 指令代码: 0x21            注1: Operation_status 操作状态            0: 操作失败            1: 操作成功            注2: Index是RGB灯起始坐标, 从0开始, Key设备有两个RGB灯, 在定义上两个RGB是同色的, 所以Index = 0, Num = 1            注3: 操作失败返回数据包不包含RGB值</p>										
设置RGB灯亮度	指令包格式	0x05	0x00	Index_id	0x22	Light	Save_to_flash	CRC				
	应答包格式	0x04	0x00	Index_id	0x22	Operation_status	CRC					
设置RGB灯亮度	指令详解	<p>(1) 功能说明: 设置RGB灯亮度            (2) 输入参数: Index_id (设备下标ID)、Light (RGB亮度, 0~100, 默认40)、Save_to_flash (是否保存到内部Flash)            (3) 返回参数: Operation_status            (4) 指令代码: 0x22            注1: Operation_status 操作状态            0: 操作失败            1: 操作成功            注2: Save_to_flash 是否保存到内部Flash            0: 不保存            1: 保存            注3: 设置成功即刻生效            注4: 保存至内部 Flash 需要擦除页面, 过程较为耗时 (约20ms), 期间会关闭串口中断, 且频繁操作会影响设备寿命, 如需要频繁调整亮度, 请避免每次都保存至内部 Flash。</p>										
获取RGB灯亮度	指令包格式	0x03	0x00	Index_id	0x23	CRC						
	应答包格式	0x04	0x00	Index_id	0x23	Light	CRC					
获取RGB灯亮度	指令详解	<p>(1) 功能说明: 获取RGB灯亮度。            (2) 输入参数: Index_id (设备下标ID)            (3) 返回参数: Light (RGB亮度)            (4) 指令代码: 0x23</p>										
按键按压	指令包格式	None										
	应答包格式	0x05	0x00	Index_id	0xE0	Status	0x00	CRC				
按键按压	指令详解	<p>(1) 功能说明: chain设备按下, 主动发送            (2) 输入参数: Index_id (设备下标ID)            (3) 返回参数: Status(按键触发状态)            (4) 指令代码: 0xE0            注1: Status 按键状态            0: 单击            1: 双击            2: 长按            注2: 该数据包仅在主动上报模式下发送。数据包发送后, RGB灯将在用户设置的灯效基础上叠加效果:            • 单击叠加蓝色 (G)            • 双击叠加蓝色 (B)            • 长按叠加红色 (R)            每次叠加的颜色通道值为255, 并在1秒内以10ms为周期逐步衰减, 最终叠加值为0, 恢复为用户设置的颜色效果。            举例: 用户将RGB灯设置为紫色 (R=128, G=0, B=128), 亮度设置为50%, 经过亮度压缩后, 实际显示为: R=64, G=0, B=64。            • 当用户单击时, G通道增加255: 显示为 R=64, G=255, B=64, 在1秒内叠加G值逐渐衰减至0, 最终回到 R=64, G=0, B=64。            • 当用户双击时, B通道增加255: 显示为 R=64, G=0, B=255, 在1秒内叠加B值逐渐衰减至0, 最终恢复为 R=64, G=0, B=64。            • 当用户长按时, R通道增加255: 原R=64, 叠加后超过255按255处理, 叠加后显示为 R=255, G=0, B=64, 在1秒内叠加R值逐渐衰减至0, 最终恢复为 R=64, G=0, B=64。</p>										
查询按键状态	指令包格式	0x03	0x00	Index_id	E1	CRC						
	应答包格式	0x04	0x00	Index_id	E1	Status	CRC					
查询按键状态	指令详解	<p>(1) 功能说明: 查询按键状态            (2) 输入参数: Index_id (设备下标ID)            (3) 返回参数: Status            (4) 指令代码: 0xE1            注1: Status 按键状态            0: 未按压            1: 按压</p>										
设置按键时间间隔	指令包格式	0x05	0x00	Index_id	E2	Double	Long	CRC				
	应答包格式	0x04	0x00	Index_id	E2	Operation_status	CRC					
设置按键时间间隔	指令详解	<p>(1) 功能说明: 设置按键数据间隔            (2) 输入参数: Index_id (设备下标ID)、Double (双击的时间间隔)、Long (长按的时间间隔)            (3) 返回参数: Operation_status            (4) 指令代码: 0xE2            注1: Double: 双击时间间隔 (默认1) (Double+1)*100ms            0-9 (100ms~1000ms)            注2: Long: 长按时间间隔 (默认0) (Long+3)*1s            0-7 (3s~10s)            注3: Operation_status 操作状态            0: 操作失败            1: 操作成功</p>										
获取按键时间间隔	指令包格式	0x03	0x00	Index_id	E3	CRC						
	应答包格式	0x05	0x00	Index_id	E3	Double	Long	CRC				
获取按键时间间隔	指令详解	<p>(1) 功能说明: 获取按键数据间隔            (2) 输入参数: Index_id (设备下标ID)            (3) 返回参数: Double、Long            (4) 指令代码: 0xE3            注1: Double 双击时间间隔            0-9 (100ms~100ms)            注2: Long: 长按时间间隔            0-7 (3s~10s)</p>										

设置按键模式	指令包格式	0x04	0x00	Index_id	E4	Mode	CRC												
	应答包格式	0x04	0x00	Index_id	E4	Operation_status	CRC												
	指令详解	(1) 功能说明: 设置按键模式 (2) 输入参数: Index_id (设备下标ID)、Mode (按键模式) (3) 返回参数: Operation_status (4) 指令代码: 0xE4 注1: Mode (默认1) 0: 非主动上报 1: 主动上报 注2: Operation_status 操作状态 0: 操作失败 1: 操作成功																	
获取按键模式	指令包格式	0x03	0x00	Index_id	E5	CRC													
	应答包格式	0x04	0x00	Index_id	E5	Mode	CRC												
	指令详解	(1) 功能说明: 获取按键模式 (2) 输入参数: Index_id (设备下标ID) (3) 返回参数: Mode (4) 指令代码: 0xE5 注1: Mode (默认1) 0: 非主动上报 1: 主动上报																	
查询设备唯一UID	指令包格式	0x04	0x00	Index_id	0xF8	UID_Type	CRC												
	应答包格式	0x04+4/12	0x00	Index_id	0xF8	Operation_status	UID (多字节)	CRC											
	指令详解	(1) 功能说明: 查询设备唯一UID (2) 输入参数: Index_id (设备下标ID)、UID_Type (3) 返回参数: Operation_status、UID (4) 指令代码: 0xF8 注1: Operation_status 操作状态 0: 操作失败 1: 操作成功 注2: UID_Type UID的类型 0: 4byte UID 1: 12byte UID																	
查询升级程序版本号	指令包格式	0x03	0x00	Index_id	0xF9	CRC													
	应答包格式	0x04	0x00	Index_id	0xF9	Bootloader_version	CRC												
	指令详解	(1) 功能说明: 查询升级程序版本号。 (2) 输入参数: Index_id (设备下标ID) (3) 返回参数: Bootloader_version (4) 指令代码: 0xF9																	
查询设备软件版本号	指令包格式	0x03	0x00	Index_id	0xFA	CRC													
	应答包格式	0x04	0x00	Index_id	0xFA	Firmware_version	CRC												
	指令详解	(1) 功能说明: 查询设备软件版本号。 (2) 输入参数: Index_id (设备下标ID) (3) 返回参数: Firmware_version (4) 指令代码: 0xFA																	
查询设备类型	指令包格式	0x03	0x00	Index_id	0xFB	CRC													
	应答包格式	0x05	0x00	Index_id	0xFB	Device_type_low	Device_type_high	CRC											
	指令详解	(1) 功能说明: 查询设备类型。 (2) 输入参数: Index_id (设备下标ID) (3) 返回参数: Device type (4) 指令代码: 0xFB 注1: Device_type = (uint16_t)(Device_type_high << 8   Device_type_low) 注2: Key的设备类型码是0x003																	
枚举请求	指令包格式	None																	
	应答包格式	0x03	0x00	0xFF	0xFC	CRC													
	指令详解	(1) 功能说明: 枚举请求, chain链路变更未端设备发送, 以及设备上电发送, 通知主机更新链路设备状态。 (2) 输入参数: none (3) 返回参数: none (4) 指令代码: 0xFC																	
心跳包	指令包格式	0x03	0x00	0xFF	0xFD	CRC													
	应答包格式	0x03	0x00	0xFF	0xFD	CRC													
	指令详解	(1) 功能说明: 心跳包, chain设备之间定时通信, 可以自发现自己是不是未端设备, 主机也可以通过心跳包来判断是否有chain设备连接。 (2) 输入参数: none (3) 返回参数: none (4) 指令代码: 0xFD																	
枚举	指令包格式	0x04	0x00	0xFF	0xFE	Send_num	CRC												
	应答包格式	0x04	0x00	0xFF	0xFE	Receive_num	CRC												
	指令详解	(1) 功能说明: 枚举获取联级设备的个数。 (2) 输入参数: Send_num (默认0, 用于记录设备个数) (3) 返回参数: Receive_num (数值代表设备个数) (4) 指令代码: 0xFE																	
<p>注1: 数据包最大长度是 256 byte, 串口 注2: 数据长度是Index_id到CRC, 包括Index_id以及CRC不包括数据长度本身 注3: CRC 计算时, 需要排除包头、包尾、长度和 CRC 字段本身, 仅对其余数据求和 注4: 串口通讯波特率115200, 8位数据位, 1停止位, 无校验位</p> <pre>uint8_t calculateCRC(const uint8_t *buffer, uint16_t size) {     uint8_t crc8 = 0;     for (uint8_t i = 4; i &lt; (size - 3); i++) {         crc8 += buffer[i];     }     return crc8; }</pre>																			