CH9329 芯片串口

通信协议

V1.2

文档变更记录

版本号	变更范围	变更内容	修改人
V1.0	文档建立	建立文档，初稿	TECH2
V1.1	文档修改	修改附录	TECH2
V1.2	文档修改	修改模拟鼠标动作	TECH2

CH9329 芯片有 3 串口通信种模式：

串口通信模式 0：协议传输模式(默认)；

串口通信模式 1：ASCII 模式；

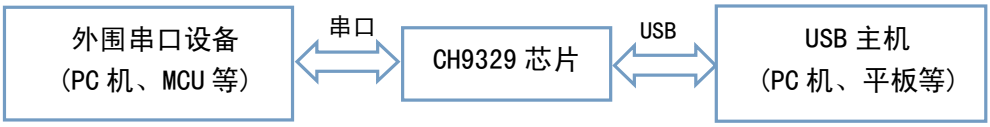
串口通信模式 2：透传模式。

CH9329 芯片默认工作在串口通信模式 0(协议传输模式)，本协议主要用于指定 CH9329 芯片工作在该模式下的串口通信协议。

任何模式下，芯片检测到该 SET 引脚为低电平后自动切换到“协议传输模式”，客户端串口设备可进行参数配置。因此，需要进行参数配置时，可以先设置 SET 引脚为低电平，然后再进行配置。

一、通信结构

外围串口设备(PC 机、MCU 或其它串口设备)与 CH9329 芯片之间的通信结构框图如下：



二、通信方式

外围串口设备(PC 机、MCU 或其它串口设备)与 CH9329 芯片的通信为主从方式，外围串口设备为主机，CH9329 芯片为从机。命令都是由外围串口设备发起，CH9329 芯片进行被动应答。外围串口设备在 500mS 内接收不到 CH9329 芯片的应答或者应答信息错误，则认为本次通信失败。

2.1. 帧格式说明

通信以帧为单位，即以数据包的形式发送，每帧数据都带有帧头字节、地址码、命令码、后续数据长度、后续数据以及累加和。如果 CH9329 芯片接收到错误帧，则返回错误应答帧或者将其直接丢弃。

以下将外围串口设备发起的通信帧称为“命令包”，CH9329 芯片返回的通信帧称为“应答包”。对于“命令包”，外围串口设备发送之后，需要等待 CH9329 芯片返回“应答包”，根据“应答包”来确定本次命令是否执行成功。如果返回错误状态或接收不到“应答包”，则需要根据情况进行重试或者出错处理。

注：以下所有描述的数据均为 16 进制格式。

命令包及应答包数据格式如下：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
2 个字节	1 个字节	1 个字节	1 个字节	N 个字节(0-64)	1 个字节

帧头：占 2 个字节，固定为 0x57、0xAB；

地址码：占 1 个字节，默认为 0x00，可接收任意地址码的命令包，如果芯片地址设置成 0x01---0xFE，则只能接收对应地址码或地址码为 0xFF 的命令包。0xFF 为广播包，芯片不需要进行应答；

命令码：占 1 个字节，外围串口设备发起的帧的命令码有效范围为：0x01---0x3F，CH9329 芯片发送正常应答包时的命令码为：原命令码 | 0x80；CH9329 芯片发送异常应答包时的命令码为：原命令码 | 0xC0；

后续数据长度：占 1 个字节，主要用于记录该包实际后续数据的长度，仅包含后续数据部分，不包括帧头字节、地址码、命令码和累加和字节；

后续数据：占 N 个字节，N 有效范围为 0---64。

累加和：占 1 个字节，计算方式为： SUM = HEAD+ADDR+CMD+LEN+DATA。

2.2. 命令码说明

表 1-命令码表

命令名称	命名码	命令说明
CMD_GET_INFO	0x01	**获取芯片版本等信息** 通过该命令向芯片获取版本号、USB 枚举状态、键盘大小写指示灯状态等信息
CMD_SEND_KB_GENERAL_DATA	0x02	**发送 USB 键盘普通数据** 通过该命令向芯片发送普通键盘数据包，模拟普通按键按下或释放动作
CMD_SEND_KB_MEDIA_DATA	0x03	**发送 USB 键盘多媒体数据** 通过该命令向芯片发送多媒体键盘数据包，模拟多媒体按键按下或释放动作
CMD_SEND_MS_ABS_DATA	0x04	**发送 USB 绝对鼠标数据** 通过该命令向芯片发送绝对鼠标数据包，模拟绝对鼠标相关动作
CMD_SEND_MS_REL_DATA	0x05	**发送 USB 相对鼠标数据** 通过该命令向芯片发送相对鼠标数据包，模拟相对鼠标相关动作
CMD_SEND_MY_HID_DATA	0x06	**发送 USB 自定义 HID 设备数据** 通过该命令向芯片发送自定义 HID 类设备数据包
CMD_READ_MY_HID_DATA	0x87	**读取 USB 自定义 HID 设备数据** 通过该命令从芯片读取自定义 HID 类设备数据包 注：PC 机向芯片下传 1 包自定义 HID 数据包后，由芯片串口自动打包发送给外围串口设备
CMD_GET_PARA_CFG	0x08	**获取参数配置** 通过该命令向芯片获取当前参数配置信息
CMD_SET_PARA_CFG	0x09	**设置参数配置** 通过该命令向芯片设置当前参数配置信息
CMD_GET_USB_STRING	0x0A	**获取字符串描述符配置** 通过该命令向芯片获取当前所使

		用的 USB 字符串描述符配置
CMD_SET_USB_STRING	**0x0B**	**设置字符串描述符配置** 通过该命令向芯片设置当前所使用的 USB 字符串描述符配置
CMD_SET_DEFAULT_CFG	**0x0C**	**恢复出厂默认配置** 通过该命令将芯片的参数配置及字符串配置信息恢复到出厂默认设置
CMD_RESET	**0x0F**	**复位芯片** 通过该命令控制芯片进行软件复位控制

2.2.1. CMD_GET_INFO

通过该命令向芯片获取版本号、USB 枚举状态、键盘大小写指示灯状态等信息。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x01	0x00	无数据	0x03

该命令不带任何参数。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x81	0x08	8 个字节数据	0x??

返回的 8 个字节后续数据依次为：

(1)、1 个字节芯片版本号：如 0x30 表示 V1.0，如 0x31 表示 V1.1；

(2)、1 个字节 USB 枚举状态：

0x00 表示 USB 端未连接计算机或未识别；

0x01 表示 USB 端已连接计算机并识别成功；

(3)、1 个字节当前键盘大小指示灯状态信息；

位 0：键盘 NUM LOCK 指示灯状态，0：熄灭；1：点亮；

位 1：键盘 CAPS LOCK 指示灯状态，0：熄灭；1：点亮；

位 2：键盘 SCROLL LOCK 指示灯状态，0：熄灭；1：点亮；

位 7---3：无效；

(4)、5 个字节保留；

2.2.2. CMD_SEND_KB_GENERAL_DATA

通过该命令向芯片发送普通键盘数据包，模拟普通按键按下或释放动作。支持全键盘、组合键操作，可支持 8+6 个无冲突按键，其中 8 为 8 个控制键(左 Ctrl、右 Ctrl、左 Shift、右 Shift、左 Windows、右 Windows、左 Alt 和右 Alt)，6 为 6 个控制键之外的普通按键。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x02	8	8 个字节数据	0x??

该命令带 8 个字节的后续数据，后续数据为 USB 键盘普通按键的键值。

依次为：

(1)、第 1 个字节：1 个字节的控制键，每个位表示 1 个按键，具体如下：

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
右 Windows 键	右 Alt 键	右 Shift 键	右 Ctrl 键	左 Windows 键	左 Alt 键	左 Shift 键	左 Ctrl 键

(2)、第 2 个字节：1 个字节 0x00，该字节必须为 0x00；

(3)、第 3-8 个字节：6 个字节普通按键值，最多可以表示 6 个按键按下，如果无按键按下则填写 0x00；

具体键盘普通按键及对应的键码见附录 1-“CH9329 键码表”。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x82	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

以下举例进行说明：

举例 1：模拟先按下“A”键，再释放“A”键，则需要发送 2 个命令包为：

(1)、模拟按下“A”键：0x57、0xAB、0x00、0x02、0x08、0x00、0x00、0x04、0x00、0x00、0x00、0x00、0x00、0x10。

(2)、模拟释放“A”键：0x57、0xAB、0x00、0x02、0x08、0x00、0x00、0x00、0x00、0x00、0x00、0x00、0x00、0x0C。

举例 2：模拟先同时按下“左 Shift”+“A”键，再释放，则需要发送 2 个命令包为：

(1)、模拟同时按下“左 Shift”+“A”键：0x57、0xAB、0x00、0x02、0x08、0x02、0x00、0x04、0x00、0x00、0x00、0x00、0x00、0x12。

(2)、模拟释放所有键：0x57、0xAB、0x00、0x02、0x08、0x00、0x00、0x00、0x00、0x00、0x00、0x00、0x00、0x0C。

2.2.3. CMD_SEND_KB_MEDIA_DATA

通过该命令向芯片发送多媒体键盘数据包，模拟多媒体按键按下或释放动作。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x03	2	2 个字节数据	0x??

该命令带 2 个字节的后续数据，后续数据为 USB 键盘多媒体按键的键值。
具体键盘普通按键及对应的键码见附录 1-“CH9329 键码表”。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x83	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

以下举例进行说明：
举例 1：模拟先按下多媒体的“禁音”键，再释放多媒体的“禁音”键，则需要发送 2 个命令包为：
(1)、按下多媒体的“禁音”键：0x57、0xAB、0x00、0x03、0x04、0x02、0x04、0x00、0x00、0x0F。
(2)、模拟释放多媒体的“禁音”键：0x57、0xAB、0x00、0x03、0x04、0x02、0x00、0x00、0x00、0x0B。

2.2.4. CMD_SEND_MS_ABS_DATA

通过该命令向芯片发送绝对鼠标数据包，模拟绝对鼠标相关动作(包括左中右键按下与释放、滚轮上下滚动、上下左右移动)。
外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x04	7	7 个字节数据	0x??

该命令带 7 个字节后续数据，7 个字节后续数据为 USB 绝对鼠标的数据包，依次为：
(1)、第 1 个字节：必须为 0x02；
(2)、第 2 个字节：1 个字节的鼠标按键值，最低 3 位每位表示 1 个按键，具体如下：

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	中键	右键	左键

BIT2---BIT0：为 1 表示该键按下，为 0 表示该键释放或未按下。
(2)、第 3-4 个字节：2 个字节 X 轴坐标值，低字节在前，高字节在后；
(3)、第 5-6 个字节：2 个字节 Y 轴坐标值，低字节在前，高字节在后；
(4)、第 7 个字节： 1 个字节滚轮滚动齿数，
如果为 0，则表示滚动无动作；
0x01---0x7F，表示向上滚动，单位：齿数；
0x81---0xFF，表示向下滚动，单位：齿数；

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM

0x57、0xAB	0x00	0x84	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

注意：芯片默认模拟的绝对鼠标分辨率为 4096 * 4096，外围串口设备下传 XY 绝对值时，需要先根据自身屏幕分辨率进行计算，再下传计算后的值。

比如当前屏幕分辨率为：X_MAX(1280) * Y_MAX(768)，则需要移动到点(100，120)，需要进行如下计算：

X_Cur = (4096 * 100) / X_MAX;

Y_Cur = (4096 * 120) / Y_MAX;

以下举例进行说明：

例如 1：模拟先按下鼠标“左”键，再释放鼠标“左”键，则需要发送 2 个命令包为：

(1)、按下鼠标“左”键：0x57、0xAB、0x00、0x04、0x07、0x02、0x01、0x00、0x00、0x00、0x00、0x00、0x10。

(2)、释放鼠标“左”键：0x57、0xAB、0x00、0x04、0x07、0x02、0x00、0x00、0x00、0x00、0x00、0x00、0x0F。

例如 2：假设屏幕分辨率为：1280*768，控制鼠标先移动到(100，100)位置，再移动到(968,500)位置，则需要发送 2 个命令包为：

(1)、移动到(100，100)位置：

计算位置 X1 = (100 * 4096) / 1280 = 320 = 0x140

计算位置 Y1 = (100 * 4096) / 768 = 533 = 0x215

发送命令包为：0x57、0xAB、0x00、0x04、0x07、0x02、0x00、0x40、0x01、0x15、0x02、0x00、0x67。

(2)、移动到(968,500)位置：

计算位置 X1 = (968 * 4096) / 1280 = 3097 = 0xC19

计算位置 Y1 = (500 * 4096) / 768 = 2667 = 0xA6B

发送命令包为：0x57、0xAB、0x00、0x04、0x07、0x02、0x00、0x19、0x0C、0x6B、0x0A、0x00、0xA9。

2.2.5. CMD_SEND_MS_REL_DATA

通过该命令向芯片发送相对鼠标数据包，模拟相对鼠标相关动作(包括左中右键按下与释放、滚轮上下滚动、上下左右移动)。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x05	5	5 个字节数据	0x??

该命令带 5 个字节后续数据，后续数据为 USB 相对鼠标的数据包，依次为：

(1)、第 1 个字节：必须为 0x01；

(2)、第 2 个字节：1 个字节的鼠标按键值，最低 3 位每位表示 1 个按键，具体如下：

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	中键	右键	左键

BIT2---BIT0：为 1 表示该键按下，为 0 表示该键释放或未按下。

(3)、第 3 个字节：1 个字节 X 方向(横坐标，左右方向)移动距离；

A、不动：字节 3 = 0x00，则表示在 X 轴方向不移动；

B、向右移动：0x01 <= 字节 3 <= 0x7F； 移动像素点 = 字节 3；

C、向左移动：0x80 <= 字节 3 <= 0xFF； 移动像素点 = 0x100 - 字节 3；

(4)、第 4 个字节：1 个字节 Y 方向(纵坐标，上下方向)移动距离；

A、不动：字节 4 = 0x00，则表示在 Y 轴方向不移动；

B、向下移动：0x01 <= 字节 4 <= 0x7F； 移动像素点 = 字节 4；

C、向上移动：0x80 <= 字节 4 <= 0xFF； 移动像素点 = 0x100 - 字节 4；

(5)、第 5 个字节： 1 个字节滚轮滚动齿数，

0x01---0x7F，表示屏幕向上滚动，单位：齿数；

0x81---0xFF，表示屏幕向下滚动，单位：齿数；

向下滚动移动的距离计算方法：

例如该字节为 0x81，实际移动距离=0x100-0x81=127 个像素；

例如该字节为 0xFF，实际移动距离=0x100-0xFF=1 个像素。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x85	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

以下举例进行说明：

例如 1：模拟先按下鼠标“左”键，再释放鼠标“左”键，则需要发送 2 个命令包为：

(1)、按下鼠标“左”键：0x57、0xAB、0x00、0x05、0x05、0x01、0x01、0x00、0x00、0x00、0x0E。

(2)、释放鼠标“左”键：0x57、0xAB、0x00、0x05、0x05、0x01、0x00、0x00、0x00、0x00、0x0D。

例如 2：控制鼠标先向左移动 3 个像素点，再向下移动 5 个像素点，则需要发送 2 个命令包为：

(1)、先向左移动 3 个像素点:0x57、0xAB、0x00、0x05、0x05、0x01、0x00、0xFD、0x00、0x00、0x0A。

(2)、向下移动 5 个像素点：0x57、0xAB、0x00、0x05、0x05、0x01、0x00、0x00、0x05、0x00、0x12。

2.2.6. CMD_SEND_MY_HID_DATA

通过该命令向芯片发送自定义 HID 类设备数据包。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x06	N	N 个字节数据	0x??

该命令带 N 个字节的后续数据，后续数据就是希望通过 USB 上传的 HID 数据包，N 有效

范围为：0-64；

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x86	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

2.2.7. CMD_READ_MY_HID_DATA

通过该命令从芯片读取自定义 HID 类设备数据包。PC 机向芯片下传 1 包自定义 HID 数据包后，由芯片串口自动打包发送给外围串口设备。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x87	N	N 个字节数据	0x??

该命令带 N 个字节的后续数据，后续数据就是 USB 下传的 HID 数据包，N 有效范围为：0-64；

注意：该命令由芯片主动发送给外围串口设备，不需要外围串口设备进行应答。

2.2.8. CMD_GET_PARA_CFG

通过该命令向芯片获取当前参数配置信息,具体参数见下面返回数据说明。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x08	0	无	0x??

该命令不带任何参数数据。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x88	50	50 个字节数据	0x??

返回的 50 个字节后续数据为：

(1)、1 个字节芯片工作模式：有效值为 0x00-0x03、0x80---0x83，默认为 0x80；

0x00：软件设置的工作模式 0，标准 USB 键盘(普通+多媒体)+USB 鼠标(绝对鼠标+相对鼠标)；

0x01：软件设置的工作模式 1，标准 USB 键盘(普通)；

0x02：软件设置的工作模式 2，标准 USB 鼠标(绝对鼠标+相对鼠标)；

0x03：软件设置的工作模式 3，标准 USB 自定义 HID 类设备；

0x80：硬件引脚设置的工作模式 0，标准 USB 键盘(普通+多媒体)+USB 鼠标(绝对鼠标+相对鼠标)；当前 MODE1 引脚为高电平，MODE0 引脚为高电平；

0x81：硬件引脚设置的工作模式 1，标准 USB 键盘(普通)；当前 MODE1 引脚为高电平，MODE0 引脚为低电平；

0x82:硬件引脚设置的工作模式 2,标准 USB 鼠标(绝对鼠标+相对鼠标)；当前 MODE1 引脚为低电平，MODE0 引脚为高电平；

0x83：硬件引脚设置的工作模式 3，标准 USB 自定义 HID 类设备；当前 MODE1 引脚为低电平，MODE0 引脚为低电平；

(2)、1 个字节芯片串口通信模式，有效值为 0x00-0x02、0x80---0x82，默认为 0x80；

0x00：软件设置的串口通信模式 0，协议传输模式；

0x01：软件设置的串口通信模式 1，ASCII 模式；

0x02：软件设置的串口通信模式 2，透传模式；

0x80：硬件引脚设置的串口通信模式 0，协议传输模式；当前 CFG1 引脚为高电平，CFG0 引脚为高电平；

0x81:硬件引脚设置的串口通信模式 1,ASCII 模式；当前 CFG1 引脚为高电平,CFG0 引脚为低电平；

0x82：硬件引脚设置的串口通信模式 2，透传模式；当前 CFG1 引脚为低电平，CFG0 引脚为高电平；

(3)、1 个字节芯片串口通信地址，有效范围为 0x00--0xFF，默认为 0x00；

(4)、4 个字节芯片串口通信波特率，高字节在前，默认为 0x00002580，即波特率为 9600bps；

(5)、2 个字节保留；

(6)、2 个字节芯片串口通信包间隔，有效范围为 0x0000--0xFFFF，默认为 3，单位为 mS，即芯片如果超过 3mS 未接收到下一个字节则表示本包结束；

(7)、4 个字节芯片 USB 的 VID 和 PID，默认芯片 VID 为 0x1A86，PID 为 0xE129，不同工作模式下，PID 有所区别；

(8)、2 个字节芯片 USB 键盘上传时间间隔(仅 ASCII 模式有效)，有效范围为 0x0000--0xFFFF，默认为 0，单位为 mS，即芯片上传完前 1 包数据之后立马上传下一包数据；

(9)、2 个字节芯片 USB 键盘释放延时时间(仅 ASCII 模式有效)，有效范围为 0x0000--0xFFFF，默认为 1，单位为 mS，即芯片上传完按键按下数据包之后 1mS 后上传按键释放数据包；

(10)、1 个字节芯片 USB 键盘自动回车标志(仅 ASCII 模式有效),有效范围为 0x00--0x01，0x00 表示不自动进行回车，0x01 表示本包结束后自动进行回车；

(11)、8 个字节芯片 USB 键盘回车符(仅 ASCII 模式有效)，4 个字节一组，共 2 组，即可以设置 2 种不同的回车符，默认遇到 ASCII 值为 0x0D 时进行回车；

(12)、8 个字节芯片 USB 键盘过滤开始、结束字符串，前 4 个字节为过滤开始字符，后 4 个字节为过滤结束字符；

(13)、1 个字节芯片 USB 字符串使能标志，

位 7：为 0 表示禁止；为 1 表示使能自定义字符串描述符；

位 6-3：保留；

位 2：为 0 表示禁止；为 1 表示使能自定义厂商字符串描述符；

位 1：为 0 表示禁止；为 1 表示使能自定义产品字符串描述符；

位 0：为 0 表示禁止；为 1 表示使能自定义序列号字符串描述符；

(14)、1 个字节芯片 USB 键盘快速上传标志(仅 ASCII 模式有效),有效范围为 0x00--0x01，0x00 表示 USB 键盘上传速度正常，0x01 表示使能 USB 键盘快速上传模式，使能快速上传模式后，上传完 1 个字符后不发送释放按键包，继续上传下一个字符，只到所有字符上传完毕再

发送释放按键包。

(15)、12 个字节保留；

2.2.9. CMD_SET_PARA_CFG

通过该命令向芯片设置当前参数配置信息,具体参数格式见上一条指令描述。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x09	50	50 个字节数据	0x??

该命令带 50 个字节的后续数据，具体数据格式见“**CMD_GET_PARA_CFG**”指令的返回。

注意：

(1)、芯片工作模式设置时，有效范围为：0x00-0x03；

(2)、芯片串口通信模式设置时，有效范围为：0x00-0x02；

(3)、所有参数设置后，下一次上电启用。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x89	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

2.2.10. CMD_GET_USB_STRING

通过该命令向芯片获取当前所使用的 USB 字符串描述符配置。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x0A	1	1 个字节数据	0x??

该命令带 1 个字节的参数，依次为：

(1)、1 个字节字符串类型，0x00 表示厂商字符串描述符；0x01 表示产品字符串描述符；0x02 表示序列号字符串描述符；

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x8A	2+N	2+N 个字节数据	0x??

返回的 2+N 个字节后续数据，依次为：

(1)、1 个字节字符串类型；

(2)、1 个字节字符串长度，有效范围为 0---23；

(3)、N 个字节当前字符串描述符，N 有效范围为：1-23；

2.2.11. CMD_SET_USB_STRING

通过该命令向芯片设置当前所使用的 USB 字符串描述符配置。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x0B	2+N	2+N 个字节数据	0x??

该命令带 2+N 个字节的参数，依次为：

(1)、1 个字节字符串类型，0x00 表示厂商字符串描述符；0x01 表示产品字符串描述符；0x02 表示序列号字符串描述符；

(2)、1 个字节字符串长度，有效范围为 0---23；

(3)、N 个字节字符串描述符，N 有效范围为：1-23；

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x8B	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

2.2.12. CMD_SET_DEFAULT_CFG

通过该命令将芯片的参数配置及字符串配置信息恢复到出厂默认设置。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x0C	0	无	0x??

该命令不带任何参数。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x8C	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

2.2.13. CMD_RESET

通过该命令控制芯片进行软件复位控制。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x0F	0	无	0x??

该命令不带任何参数。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x8F	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

2.3. 错误应答包

芯片接收到的命令包如果存在命令码错误、校验错误或者执行失败等问题时，则需要通过错误应答包进行应答。错误应答包包含 1 个字节后续数据，该数据即为命令执行状态。

芯片→外围串口设备：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0xC?	1	1 个字节数据	0x??

返回的 1 个字节后续数据为：当前命令执行状态。

返回的命令码 = 原命令码 | 0xC0;

表 2-命令执行状态如下表

状态名称	状态码	状态说明
DEF_CMD_SUCCESS	0x00	命令执行成功
DEF_CMD_ERR_TIMEOUT	0xE1	串口接收一个字节超时
DEF_CMD_ERR_HEAD	0xE2	串口接收包头字节出错
DEF_CMD_ERR_CMD	0xE3	串口接收命令码错误
DEF_CMD_ERR_SUM	0xE4	累加和检验值不匹配
DEF_CMD_ERR_PARA	0xE5	参数错误
DEF_CMD_ERR_OPERATE	0xE6	帧正常，执行失败

附录 1-“CH9329 键码表”

1、普通按键及对应的键码表：

序号	符号		HID Page	HID Code	序号	符号		HID Page	HID Code
1	~	`	07	35	54	>	.	07	37
2	!	1	07	1E	55	?	/	07	38
3	@	2	07	1F	56	Keycode56 (*BJ)		07	87
4	#	3	07	20	57	Shift (R)		07	E5
5	$	4	07	21	58	Ctrl (L)		07	E0
6	%	5	07	22	60	Alt (L)		07	E2
7	^	6	07	23	61	Space		07	2C
8	&	7	07	24	62	Alt (R)		07	E6
9	*	8	07	25	64	Ctrl (R)		07	E4
10	(	9	07	26	75	Insert		07	49
11	)	0	07	27	76	Delete		07	4C
12	_	-	07	2D	79	Left Arrow		07	50
13	+	=	07	2E	80	Home		07	4A
14	Keycode14 (*J)		07	89	81	End		07	4D
15	Back Space		07	2A	83	↑		07	52
16	Tab		07	2B	84	↓		07	51
17	Q		07	14	85	PgUp		07	4B
18	W		07	1A	86	PgDn		07	4E
19	E		07	08	89	→		07	4F
20	R		07	15	90	Num Lock		07	53
21	T		07	17	91	7	Home	07	5F
22	Y		07	1C	92	4	←	07	5C
23	U		07	18	93	1	End	07	59
24	I		07	0C	95	/		07	54
25	O		07	12	96	8	↑	07	60
26	P		07	13	97	5		07	5D
27	{	[	07	2F	98	2	↓	07	5A
28	}	]	07	30	99	0	Ins	07	62
29	Keycode29 (*4)		07	31	100	*		07	55
30	Caps Lock		07	39	101	9	PgUp	07	61
31	A		07	04	102	6	→	07	5E
32	S		07	16	103	3	PgDn	07	5B
33	D		07	07	104	.	Del	07	63
34	F		07	09	105	-		07	56
35	G		07	0A	106	+		07	57
36	H		07	0B	107	Keycode107 (*B)		07	85

37	J	07	0D	108	Enter_R	07	58
38	K	07	0E	110	ESC	07	29
39	L	07	0F	112	F1	07	3A
40	: ;	07	33	113	F2	07	3B
41	“ ‘	07	34	114	F3	07	3C
42	Keycode42 (*5BJ)	07	32	115	F4	07	3D
43	Enter_L	07	28	116	F5	07	3E
44	Shift (L)	07	E1	117	F6	07	3F
45	Keycode45 (*5B)	07	64	118	F7	07	40
46	Z	07	1D	119	F8	07	41
47	X	07	1B	120	F9	07	42
48	C	07	06	121	F10	07	43
49	V	07	19	122	F11	07	44
50	B	07	05	123	F12	07	45
51	N	07	11	124	Print Screen	07	46
52	M	07	10	125	Scroll Lock	07	47
53	<	07	36	126	Pause	07	48
* 4 _ 104 Keyboard Only * 5 _ 105 Keyboard Only				*B _ 107 Keyboard Only *J _ 109 Keyboard Only			

序号	符号	HID Page	HID Code
131 (*J)	Japanese J131	07	8B
132 (*J)	Japanese J132	07	8A
133 (*J)	Japanese J133	07	88
150	KoreaKC-L, Key_Hangul	07	90
151	Korea KC-R, Key_Hanja	07	91
ACPI	Power	01	81
ACPI	Sleep	01	82
ACPI	Wake-up	01	83
Windows Key	L_WIN	07	E3
Windows Key	R_WIN	07	E7
Windows Key	APP	07	65

2、多媒体按键及对应的键码表：

对于 ACPI 键，共 2 个字节，第 1 个字节为 REPORT ID，固定为 0x01，第 2 个字节为 ACPI 键码。

字节号	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	00000001b							
2	00000b					Wake-up	Sleep	Power
1:键按下 0:键释放								

对于其它多媒体键，占 4 个字节，第 1 个字节为 REPORT ID，固定为 0x02，第 2 个字节至第 4 个字节为多媒体键值。

字节号	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	00000010b							
2	Eject	CD Stop	Prev. Track	Next Track	Play/Pause	Mute	Volume-	Volume+
3	Refresh	WWW Stop	WWW Forward	WWW Back	WWW Home	WWW Favorites	WWW Search	E-Mail
4	Rewind	Record	Minimize	My Computer	Screen Save	Calculator	Explorer	Media
1:键按下 0:键释放								