CH9329 chip serial port

letter of agreement

V1.0

Document change history

version number	Scope of change	change content	Modifier
V1.0	Document creation	Create document, first draft	TECH2

The CH9329 chip has 3 serial communication modes: Serial

communication mode 0: protocol transmission mode (default);

Serial communication mode 1: ASCII mode;

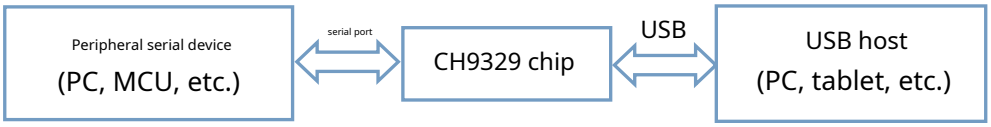
Serial communication mode 2: transparent transmission mode.

The CH9329 chip works in serial communication mode 0 (protocol transmission mode) by default. This protocol is mainly used to specify the serial communication protocol of the CH9329 chip working in this mode.

In any mode, the chip automatically switches to "protocol transmission mode" after detecting that the SET pin is low level. , client string The parameters of the port device can be configured. Therefore, when parameter configuration is required, you can first set the SET pin to low level and then configure.

1. Communication structure

The communication structure block diagram between the peripheral serial device (PC, MCU or other serial device) and the CH9329 chip is as follows:



2. Communication methods

The communication between peripheral serial device (PC, MCU or other serial device) and CH9329 chip is master-slave mode. The peripheral serial device is the host and the CH9329 chip is the slave. Commands are initiated by peripheral serial device, and the CH9329 chip responds passively. If the peripheral serial port device cannot receive the response from the CH9329 chip within 500mS or the response information is incorrect, the communication will be considered failed.

2.1. Frame format description

Communication is based on frames, that is, sent in the form of data packets. Each frame of data contains frame header bytes, address code, command code, subsequent data length, subsequent data, and cumulative sum. If the CH9329 chip receives an error frame, it returns an error response frame or discards it directly.

In the following, the communication frame initiated by the peripheral serial device is called "command packet" , the communication frame returned by the CH9329 chip is called "should "Answer."For the "command packet", after the peripheral serial device sends it, you need to wait for the CH9329 chip to return the "response packet" ,root Determine whether the command is executed successfully according to the "response packet". If an error status is returned or the "response packet" is not received , then it is necessary Retry or error handling is required depending on the situation.

Note: All the data described below are in hexadecimal format.

The data format of the command packet and response packet is as follows:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
2 bytes	1 byte	1 byte	1 byte	N bytes (0-64)	1 byte

Frame header: 2 bytes, fixed at 0x57, 0xAB;

Address code: 1 byte, default is 0x00, can receive command packets with any address code. If the chip address is set to 0x01---0xFE, it can only receive command packets with the corresponding address code or the address code 0xFF. 0xFF is a broadcast packet, and the chip does not need to respond;

Command code: occupies 1 byte. The valid range of the command code for the frame initiated by the peripheral serial device is: 0x01---0x3F. The command code when the CH9329 chip sends a normal response packet is: original command code | 0x80; the CH9329 chip sends abnormally The command code when responding to the packet is: original command code | 0xC0;

Subsequent data length: 1 byte, mainly used to record the length of the actual subsequent data of the packet, including only subsequent data

part, excluding frame header bytes, address codes, command codes and accumulated sum bytes;

Subsequent data: occupies N bytes, and the valid range of N is 0---64.

Accumulated sum: 1 byte, calculated as: SUM = HEAD+ADDR+CMD+LEN+DATA.

2.2. Command code description

Table 1-Command code table

Command name	naming code	Command description
CMD_GET_INFO	**0x01**	**Get information such as chip version** Get the version number from the chip through this command. USB enumeration status, keyboard case indicators Status and other information
CMD_SEND_KB_GENERAL_DATA	**0x02**	**Send USB keyboard normal data** Send a normal keyboard to the chip through this command Data packet, simulates normal key press or release action
CMD_SEND_KB_MEDIA_DATA	**0x03**	**Send USB keyboard multimedia data** Send multimedia keys to the chip through this command Disk data packet, simulate multimedia button press or release action
CMD_SEND_MS_ABS_DATA	**0x04**	**Send USB absolute mouse data** Send the absolute mouse to the chip through this command Data package, simulates absolute mouse related actions
CMD_SEND_MS_REL_DATA	**0x05**	**Send USB relative mouse data** Send the relative mouse to the chip through this command Data packet, simulate relative mouse related actions
CMD_SEND_MY_HID_DATA	**0x06**	**Send USB custom HID device data** Send a custom HID to the chip through this command class device packet
CMD_READ_MY_HID_DATA	**0x87**	**Read USB custom HID device data** Read the custom HID from the chip through this command class device packet Note: After the PC downloads 1 customized HID data packet to the chip, it will be automatically opened by the chip serial port. Packet sent to peripheral serial device
CMD_GET_PARA_CFG	**0x08**	**Get parameter configuration** Get the current parameters from the chip through this command Configuration information
CMD_SET_PARA_CFG	**0x09**	**Set parameter configuration** Set the current parameters to the chip through this command Configuration information
CMD_GET_USB_STRING	**0x0A**	**Get string descriptor configuration** Use this command to obtain the currently used USB string descriptor configuration used
CMD_SET_USB_STRING	**0x0B**	**Set string descriptor configuration**

		Use this command to set the currently used USB string descriptor configuration used
CMD_SET_DEFAULT_CFG	**0x0C**	**Restore factory default configuration** Use this command to configure the chip parameters and Restore string configuration information to factory default set up
CMD_RESET	**0x0F**	**reset chip** Use this command to control the chip to perform software replication. bit control

2.2.1, CMD_GET_INFO

Use this command to obtain the version number, USB enumeration status, keyboard case indicator status and other information from the chip. Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x01	0x00	no data	0x03

This command takes no parameters.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x81	0x08	8 bytes of data	0x??

The returned 8 bytes of subsequent data are as follows:

(1), 1 byte chip version number: such as 0x30 represents V1.0, such as 0x31 represents V1.1;

(2), 1 byte USB enumeration status:

0x00 means the USB end is not connected to the computer or not recognized; 0x01 means the USB end is connected to the computer and recognized successfully;

(3), 1 byte of current keyboard size indicator light status information;

Bit 0: keyboard NUM LOCK indicator light status, 0: off; 1: on; Bit 1: keyboard CAPS LOCK indicator light status, 0: off; 1: on; Bit 2: keyboard SCROLL LOCK indicator light status, 0: Off; 1: On; Bit 7---3: Invalid;

(4), 5 bytes reserved;

2.2.2, CMD_SEND_KB_GENERAL_DATA Use this command to send ordinary keyboard data packets to the chip to simulate ordinary key press or release actions. Supports full keyboard and key combination operations, and can support 8+6 conflict-free keys, 8 of which are 8 control keys (left Ctrl, right Ctrl, left Shift, right Shift, left Windows, right Windows, left Alt and right Alt) , 6 is the normal keys other than the 6 control keys.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x02	8	8 bytes of data	0x??

This command carries 8 bytes of subsequent data, which are the key values of ordinary keys on the USB keyboard.

as followed:

(1), 1st byte: 1 byte control key, each bit represents 1 key, the details are as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
right Windows key	right Alt key	right Shift key	right Ctrl key	Left Windows key	Left Alt key	Left Shift key	Left Ctrl key

(2), 2nd byte: 1 byte 0x00, this byte must be 0x00;

(3), Bytes 3-8: 6 bytes of ordinary button values, which can represent up to 6 button presses. If no button is pressed, Fill in 0x00 below;

For the specific common keyboard keys and corresponding key codes, see Appendix 1 - "CH9329 Key Code Table".

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x82	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

The following examples illustrate:

Example 1: To simulate pressing the "A" key first and then releasing the "A" key, you need to send 2 command packets:

(1) Simulate pressing the "A" key: 0x57, 0xAB, 0x00, 0x02, 0x08,0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00,0x10.

(2) Simulate releasing the "A" key: 0x57, 0xAB, 0x00, 0x02, 0x08,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,0x0C.

Example 2: To simulate pressing the "Left Shift" + "A" keys at the same time and then release them, you need to send 2 command packets: (1), to simulate pressing the "Left Shift" + "A" keys at the same time: 0x57, 0xAB, 0x00, 0x02, 0x08,0x02, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00,0x12.

(2) Simulate releasing all keys: 0x57, 0xAB, 0x00, 0x02, 0x08,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,0x0C.

2.2.3, CMD_SEND_KB_MEDIA_DATA Use this command to send multimedia keyboard data packets to the chip to simulate the pressing or releasing of multimedia keys. Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x03	2	2 bytes of data	0x??

This command carries 2 bytes of subsequent data, which are the key values of the USB keyboard multimedia keys. For the specific common keyboard keys and corresponding key codes, see Appendix 1 - "CH9329 Key Code Table".

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x83	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

The following examples illustrate:

Example 1: To simulate first pressing the multimedia "silence" button and then releasing the multimedia "silence" button, two command packets need to be sent:

(1) Press the "silence" button of the multimedia: 0x57, 0xAB, 0x00, 0x03, 0x04,0x02, 0x04, 0x00, 0x00,0x0F.

(2) Simulate the "silence" key that releases multimedia: 0x57, 0xAB, 0x00, 0x03, 0x04,0x02, 0x00, 0x00, 0x00,0x0B.

2.2.4, CMD_SEND_MS_ABS_DATA Use this command to send an absolute mouse data packet to the chip to simulate absolute mouse related actions (including pressing and releasing the left, middle and right buttons, rolling the wheel up and down, and moving up, down, left, and right).

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x04	7	7 bytes of data	0x??

This command carries 7 bytes of subsequent data, and the 7 bytes of subsequent data are USB absolute mouse data packets, in order: (1), the first byte: must be 0x02;

(2) The second byte: 1 byte of mouse button value. Each of the lowest 3 bits represents 1 button. The details are as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	middle key	right key	Left key

BIT2---BIT0: 1 means the key is pressed, 0 means the key is released or not pressed. (2),

3rd-4th bytes: 2-byte X-axis coordinate value, low byte first, high byte last;

(3), 5th-6th bytes: 2-byte Y-axis coordinate value, low byte first, high byte last;

(4). The 7th byte: 1 byte, the number of rolling teeth of the roller. If it

is 0, it means there is no rolling action;

0x01---0x7F, means scrolling up, unit: number of teeth;

0x81---0xFF, means scrolling down, unit: number of teeth;

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x84	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

Note: The default simulated absolute mouse resolution of the chip is 4096 * 4096. When the peripheral serial port device downloads the XY absolute value, it needs to first calculate it based on its own screen resolution, and then download the calculated value.

For example, if the current screen resolution is:

X_Cur = (4096 * 100) / X_MAX; Y_Cur = (4096 * 120) / Y_MAX;

The following examples illustrate:

For example 1: To simulate pressing the "left" mouse button first, and then releasing the "left" mouse button, you need to send 2 command packets:

(1) Press the "left" button of the mouse: 0x57, 0xAB, 0x00, 0x04, 0x07,0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,0x10.

(2) Release the "left" mouse button: 0x57, 0xAB, 0x00, 0x04, 0x07,0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,0x0F.

For example 2: Assume the screen resolution is: 1280*768, and control the mouse to first move to the (100, 100) position, and then move to the (968,500) position, you need to send 2 command packets:

(1), move to (100, 100) position:

Calculate position X1 = (100 * 4096) / 1280 = 320 = 0x140 Calculate position Y1 = (100 * 4096) / 768 = 533 = 0x215 Send command packets: 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x00, 0x40, 0x01, 0x15, 0x02, 0x00,0x67.

(2). Move to position (968,500):

Calculated position X1 = (968 * 4096) / 1280 = 3097 = 0xC19 Calculated position Y1 = (500 * 4096) / 768 = 2667 = 0xA6B The command packet sent is: 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x00, 0x19, 0x0C, 0x6B, 0x0A, 0x00,0xA9.

2.2.5, CMD_SEND_MS_REL_DATA Use this command to send relative mouse data packets to the chip to simulate relative mouse related actions (including pressing and releasing the left, middle and right buttons, rolling the wheel up and down, and moving up, down, left, and right).

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x05	5	5 bytes of data	0x??

This command carries 5 bytes of subsequent data, which are USB relative mouse data packets, in order: (1), the first byte: must be 0x01;

(2) The second byte: 1 byte of mouse button value. Each of the lowest 3 bits represents 1 button. The details are as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	middle key	right key	Left key

BIT2---BIT0: 1 means the key is pressed, 0 means the key is released or not pressed. (3), 3rd byte: 1 byte X direction (vertical coordinate, up and down direction) movement distance; A. Not moving: Byte 3 = 0x00, which means no movement in the X axis direction;

B. Move right: 0x01 <= Byte 3 <= 0x7F; Move pixel = Byte 3; C. Move left: 0x80 <= Byte 3 <= 0xFF; Move pixel = 0x00 - Byte 3;

(4). The 4th byte: 1 byte moving distance in the Y direction (ordinate, up and down direction); A. Not moving: Byte 4 = 0x00, which means no movement in the Y axis direction;

B. Move right: 0x01 <= Byte 4 <= 0x7F; Move pixel = Byte 4; C. Move left: 0x80 <= Byte 4 <= 0xFF; Move pixel = 0x00 - Byte 4;

(5), 5th byte: 1 byte number of scroll wheel teeth, 0x01---0x7F, indicating that the screen scrolls upward, unit: number of teeth; 0x81---0xFF, indicating that the screen scrolls downward, unit: number of teeth;

How to calculate the distance of scrolling down:

For example, if the byte is 0x81, the actual moving distance = 0x100-0x81 = 127 pixels; For example, if the byte is 0xFF, the actual moving distance = 0x100-0xFF = 1 pixel.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x85	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

The following examples illustrate:

For example 1: To simulate pressing the "left" mouse button first, and then releasing the "left" mouse button, you need to send 2 command packets: (1), press the "left" mouse button: 0x57, 0xAB, 0x00, 0x05, 0x05、0x01, 0x01, 0x00, 0x00, 0x00,0x0E.

(2) Release the "left" mouse button: 0x57, 0xAB, 0x00, 0x05, 0x05,0x01, 0x00, 0x00, 0x00, 0x00,0x0D.

For example 2: Control the mouse to move 3 pixels to the left first, and then move 5 pixels down. You need to send 2 command packets:

(1), first move 3 pixels to the left: 0x57, 0xAB, 0x00, 0x05, 0x05,0x01, 0x00, 0xFD, 0x00, 0x00,0x0A.

(2), move down 5 pixels: 0x57, 0xAB, 0x00, 0x05, 0x05,0x01, 0x00, 0x00, 0x05, 0x00,0x12.

2.2.6, CMD_SEND_MY_HID_DATA Use this command to send a custom HID class device data packet to the chip. Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x06	N	N bytes of data	0x??

This command carries N bytes of subsequent data. The subsequent data is the HID data packet that you want to upload through USB. The valid range of N is: 0-64;

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x86	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

2.2.7, CMD_READ_MY_HID_DATA

Use this command to read custom HID class device data packets from the chip. After the PC downloads a custom HID data packet to the chip, the chip serial port automatically packages it and sends it to the peripheral serial port device.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum

HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x87	N	N bytes of data	0x??

This command carries N bytes of subsequent data. The subsequent data is the HID data packet transmitted from USB. The valid range of N is: 0-64;

Note: This command is actively sent by the chip to the peripheral serial port device and does not require a response from the peripheral serial port device.

2.2.8, CMD_GET_PARA_CFG

Use this command to obtain the current parameter configuration information from the chip. For specific parameters, see the return data description below.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x08	0	none	0x??

This command does not take any parameter data.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x88	50	50 bytes of data	0x??

The 50 bytes of subsequent data returned are:

(1), 1-byte chip working mode: valid values are 0x00-0x03, 0x80---0x83, the default is 0x80; 0x00: working mode 0 set by the software, standard USB keyboard (normal + multimedia) + USB mouse (Absolute mouse + relative mouse);

0x01: Working mode 1 set by the software, standard USB keyboard (normal); 0x02: Working mode 2 set by the software, standard USB mouse (absolute mouse + relative mouse); 0x03: Working mode 3 set by the software, standard USB custom HID Class device; 0x80: Working mode 0 of hardware pin setting, standard USB keyboard (normal + multimedia) + USB mouse (absolute mouse + relative mouse); currently the MODE1 pin is high level, and the MODE0 pin is high level;

0x81: Working mode 1 of hardware pin setting, standard USB keyboard (normal); currently the MODE1 pin is high level and the MODE0 pin is low level;

0x82: Working mode 2 of hardware pin setting, standard USB mouse (absolute mouse + relative mouse); currently the MODE1 pin is low level and the MODE0 pin is high level;

0x83: Working mode 3 of hardware pin setting, standard USB custom HID class device; currently the MODE1 pin is low level and the MODE0 pin is low level;

(2), 1-byte chip serial port communication mode, valid values are 0x00-0x02, 0x80---0x82, the default is 0x80; 0x00: Serial port communication mode 0 set by the software, protocol transmission mode;

0x01: Serial communication mode 1 set by software, ASCII mode; 0x02: Serial communication mode 2 set by software, transparent transmission mode; 0x80: Serial communication mode 0 set by hardware pin, protocol transmission mode; the current CFG1 pin is high power flat, the CFG0 pin is high;

0x81: Serial communication mode 1 of hardware pin setting, ASCII mode; currently the CFG1 pin is high level and the CFG0 pin is low level;

0x82: Serial communication mode 2 of hardware pin setting, transparent transmission mode; currently the CFG1 pin is low level and the CFG0 pin is high level;

(3), 1 byte chip serial port communication address, the valid range is 0x00--0xFF, the default is 0x00;

(4), 4-byte chip serial port communication baud rate, high byte first, The default is 0x00002580, which means the baud rate is 9600bps;

(5), 2 bytes reserved;

(6), 2-byte chip serial port communication packet interval, the valid range is 0x0000--0xFFFF, the default is 3, the unit is mS, That is, if the chip does not receive the next byte for more than 3mS, it means the end of this packet;

(7) VID and PID of 4-byte chip USB. The default chip VID is 0x1A86 and the PID is 0xE129. Different working In operation mode, the PID is different;

(8), 2-byte chip USB keyboard upload time interval (only valid in ASCII mode), the valid range is 0x0000--0xFFFF, the default is 0, the unit is mS, that is, the chip uploads the next packet of data immediately after uploading the first packet of data;

(9), 2-byte chip USB keyboard release delay time (only valid in ASCII mode), the valid range is 0x0000--0xFFFF, the default is 1, the unit is mS, that is, the chip uploads the key release data packet 1ms after it uploads the key press data packet;

(10), 1-byte chip USB keyboard automatic carriage return mark (only valid in ASCII mode), The valid range is 0x00--0x01, 0x00 means no automatic carriage return, 0x01 means automatic carriage return after the package ends;

(11), 8-byte chip USB keyboard carriage return character (only valid in ASCII mode), 4 bytes in one group, 2 groups in total, that's it You can set 2 different carriage return characters. By default, carriage return is performed when the ASCII value is 0x0D;

(12), 8-byte chip USB keyboard filtering start and end strings, the first 4 bytes are the filtering start characters, the last 4 Bytes are the filter end characters;

(13), 1-byte chip USB string enable flag,

Bit 7: 0 means disabled; 1 means custom string descriptor is enabled; Bit

6-3: Reserved;

Bit 2: 0 means disabled; 1 means customized manufacturer string descriptor is enabled; Bit

1: 0 means disabled; 1 means customized product string descriptor is enabled; Bit 0: 0

means disabled; 1 means enabling the custom serial number string descriptor;

(14), 1-byte chip USB keyboard fast upload flag (only valid in ASCII mode), The valid range is 0x00--0x01, 0x00 means the USB keyboard upload speed is normal, 0x01 means the USB keyboard fast upload mode is enabled. After the fast upload mode is enabled, the release key packet will not be sent after uploading 1 character, and the next character will continue to be uploaded until all characters are uploaded. Send release button packet.

(15), 12 bytes reserved;

2.2.9, CMD_SET_PARA_CFG

Use this command to set the current parameter configuration information to the chip. For the specific parameter format, see the description of the previous command. Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x09	50	50 bytes of data	0x??

This command carries 50 bytes of subsequent data. For specific data format, see "**CMD_GET_PARA_CFG**"Return of the command.

Note:

(1) When setting the chip working mode, the valid range is: 0x00-0x03;

(2) When setting the chip serial port communication mode, the valid range is: 0x00-0x02;

(3) After all parameters are set, they will be enabled the next time you power on.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum

HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x89	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

2.2.10, CMD_GET_USB_STRING Use this command to obtain the currently used USB string descriptor configuration from the chip.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0A	1	1 byte data	0x??

This command takes 1 byte of parameters, in order:

(1), 1-byte string type, 0x00 represents the manufacturer string descriptor; 0x01 represents the product string descriptor; 0x02 represents the serial number string descriptor;

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8A	2+N	2+N bytes of data	0x??

The returned 2+N bytes of subsequent data are as follows:

(1), 1 byte string type;

(2), 1 byte string length, the valid range is 0---23;

(3), N bytes current string descriptor, the valid range of N is: 1-23;

2.2.11, CMD_SET_USB_STRING Use this command to set the currently used USB string descriptor configuration to the chip.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0B	2+N	2+N bytes of data	0x??

This command takes 2+N bytes of parameters, in order:

(1), 1-byte string type, 0x00 represents the manufacturer string descriptor; 0x01 represents the product string descriptor; 0x02 represents the serial number string descriptor;

(2), 1 byte string length, the valid range is 0---23;

(3), N byte string descriptors, the valid range of N is: 1-23;

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8B	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

2.2.12, CMD_SET_DEFAULT_CFG Use this command to restore the chip's parameter configuration and string configuration information to the factory default settings.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0C	0	none	0x??

This command takes no parameters.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8C	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

2.2.13, CMD_RESET

Use this command to control the chip for software reset control.

Peripheral serial device-chip:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0F	0	none	0x??

This command takes no parameters.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8F	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

2.3. Error response packet

If the command packet received by the chip has problems such as command code errors, verification errors, or execution failures, it needs to respond with an error response packet. The error response packet contains 1 byte of subsequent data, which is the command execution status.

Chip-peripheral serial device:

Frame header	address code	command code	Subsequent data length	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0xC?	1	1 byte data	0x??

The 1-byte subsequent data returned is: current command execution status.

Returned command code = original command code | 0xC0;

Table 2-Command execution status is as follows

Status name	status code	Status description
DEF_CMD_SUCCESS	0x00	Command execution successful
DEF_CMD_ERR_TIMEOUT	0xE1	The serial port receives one byte and times out.

DEF_CMD_ERR_HEAD	0xE2	Serial port receiving packet header byte error
DEF_CMD_ERR_CMD	0xE3	Serial port receiving command code error
DEF_CMD_ERR_SUM	0xE4	Accumulation and test values do not match
DEF_CMD_ERR_PARA	0xE5	Parameter error
DEF_CMD_ERR_OPERATE	0xE6	Frame is normal, execution failed

Appendix 1 - "CH9329 Key Code Table"

1. Ordinary keys and corresponding key code tables:

serial number	symbol		HID Page	HID Code	serial number	symbol		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	twenty one	58	Ctrl(L)		07	E0
6	%	5	07	twenty two	60	Alt(L)		07	E2
7	^	6	07	twenty three	61	Ctrl(L)		07	2C
8	&	7	07	twenty four	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	htK		07	4E
19	E		07	08	89	→		07	4F
20	R		07	15	90	Num Lock		07	53
twenty one	T		07	17	91	7	Home	07	5F
twenty two	Y		07	1C	92	4	←	07	5C
twenty three	U		07	18	93	1	End	07	59
twenty four	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	htK	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			

serial number	symbol	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. Multimedia buttons and corresponding key code tables:

For the ACPI key, there are 2 bytes in total, the first byte is the REPORT ID, fixed at 0x01, and the second byte is the ACPI key code.

Byte number	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: key pressed 0: key released								

For other multimedia keys, it occupies 4 bytes. The first byte is the REPORT ID, which is fixed at 0x02. The second byte is

The 4th byte is the multimedia key value.

Byte number	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	Email
4	Rewind	Record	Minimize	My Computer	Screen Save	Calculator	Explorer	Media
1: key pressed 0: key released								