CH9329 Chip Serial Communication Protocol

V1.2

Document Change Log

version number	Scope of change	Changes	modifier
V1.0	Document Creation	Creating documents, first drafts	TECH2
V1.1	Document Modification	Revision of the Appendix	TECH2
V1.2	Document Modification	Modify Analog Mouse Action	TECH2

The CH9329 chip has 3 modes of serial communication:

Serial communication mode 0: Protocol transmission mode (default);

Serial communication mode 1: ASCII mode;

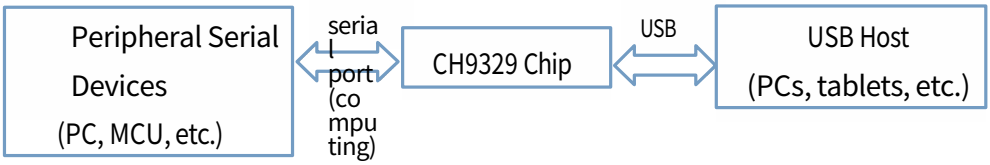
Serial communication mode 2: Pass-through mode.

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

In any mode, the chip detects that the SET pin is low and automatically switches to the "protocol transmission mode", and the client serial device can carry out parameter configuration. Therefore, when you need to configure parameters, you can set the SET pin to low level first and then configure.

I. Communications structure

The communication structure between peripheral serial devices (PC, MCU or other serial devices) and CH9329 chip is shown below:



II. Means of communication

The communication between peripheral serial devices (PC, MCU or other serial devices) and CH9329 chip is in master-slave mode, the peripheral serial device is the host and CH9329 chip is the slave. The commands are initiated by the peripheral serial device and the CH9329 chip responds passively. If the peripheral serial device does not receive the response from CH9329 chip within 500mS or the response information is wrong, the communication will be considered as failed.

2.1. Description of the frame format

Communication is in frames, i.e., sent in packets, each frame with a header byte, address code, command code, subsequent data length, subsequent data, and a cumulative sum. If the CH9329 chip receives an error frame, it returns an error answer frame or simply discards it.

The communication frame initiated by the peripheral serial device is called "Command Packet", and the communication frame returned by CH9329 chip is called "Answer Packet". For the "command packet", after the peripheral serial device sends it, it needs to wait for the CH9329 chip to return the "answer packet", and then determine whether the command is executed successfully or not according to the "answer packet". According to the "answer packet", we can determine whether this command is executed successfully or not. If it returns an error status or does not receive the "answer packet", then you need to retry or error processing according to the situation.

Note: All data described below is in hexadecimal format.

The command packet and answer packet data formats are as follows:

header	address code	command code	Length of follow-up data	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: occupies 2 bytes, fixed at 0x57, 0xAB;

Address code: 1 byte, default is 0x00, can receive command packet with any address code, if the chip address is set to 0x01---0xFE, it can only receive the command packet with the corresponding address code or the address code of 0xFF. 0xFF is a broadcasting packet, the chip doesn't need to answer;

Command code: occupies 1 byte, the valid range of the command code of the frame initiated by the peripheral serial device is 0x01---0x3F, the command code when the CH9329 chip sends a normal answer packet is: original command code | 0x80; the command code when the CH9329 chip sends an abnormal answer packet is: original command code | 0xC0;

Follow-up data length: 1 byte, mainly used to record the length of the actual follow-up data of the packet, only contains follow-up data part, excluding the frame header byte, address code, command code and accumulator byte;

subsequent data: occupies N bytes, N valid range is 0---64.

Cumulative sum: Takes up 1 byte and is calculated as follows: SUM = HEAD+ADDR+CMD+LEN+DATA.

2.2. Command Code Description

Table 1-Command Code List

command name	naming code	Command Description
CMD_GET_INFO	0x01	**Obtaining chip version and other information** Get the version number from the chip with this command, USB Enumeration Status, Keyboard Case Indicator Status and other information
CMD_SEND_KB_GENERAL_DATA	0x02	**Send USB keyboard general data** This command sends a normal keyboard packet to the chip to simulate a normal key press or release movements
CMD_SEND_KB_MEDIA_DATA	0x03	**Send USB keyboard multimedia data** Send multimedia keyboard packets to the chip with this command to simulate multimedia key presses or Release Action
CMD_SEND_MS_ABS_DATA	0x04	**Send USB absolute mouse data** Send the absolute mouse to the chip with this command Data packs that simulate absolute mouse-related actions
CMD_SEND_MS_REL_DATA	0x05	**Send USB relative mouse data** Send the relative mouse to the chip with this command Data packets that simulate relative mouse-related actions
CMD_SEND_MY_HID_DATA	0x06	**Send USB Custom HID Device Data** This command sends a custom HID to the chip class device packet
CMD_READ_MY_HID_DATA	0x87	**Read USB custom HID device data** This command reads a custom HID from the chip.

		USB String Descriptor Configuration for
CMD_SET_USB_STRING	**0x0B**	**Setting the String Descriptor Configuration** This command sets the currently enabled USB String Descriptor Configuration for
CMD_SET_DEFAULT_CFG	**0x0C**	**Restore Factory Default Configuration** This command restores the chip's parameter configuration and string configuration information to the factory defaults. set up
CMD_RESET	**0x0F**	**reset chip** This command is used to control the chip to perform software replication. bit control

2.2.1. CMD_GET_INFO

Get the version number, USB enumeration status, keyboard case indicator status and other information from the chip with this command. Peripheral Serial Devices → Chip:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x01	0x00	No data available	0x03

This command comes

with no parameters.

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

serial devices:

The 8 bytes of subsequent data returned are in order:

(1) The 1-byte chip version number: e.g. 0x30 means V1.0, e.g. 0x31 means V1.1;

(2) 1 byte USB Enumeration status.

0x00 Indicates that the USB terminal is not connected to the computer or is not recognized;

0x01 Indicates that the USB side is connected to the computer and recognized successfully;

(3) The current keypad size indicator status information is 1 byte;

Bit 0: Keypad NUM LOCK indicator status, 0: off; 1: lit;
Bit 1: Keypad CAPS LOCK indicator status, 0: off; 1: on;
Bit 2: Keypad SCROLL LOCK indicator status, 0: off; 1: on; Bit 7---3: invalid;

(4) The following five bytes are reserved;

2.2.2. CMD_SEND_KB_GENERAL_DATA

The command sends a normal keyboard packet to the chip to simulate a normal key press or release. Supports full keyboard and key combination operation, and can support 8+6 non-conflicting keys, of which 8 are the 8 control keys (Left Ctrl, Right Ctrl, Left Shift, Right Shift, Left Windows, Right Windows, Left Alt, and Right Alt), and 6 are the normal keys other than the 6 control keys.

Peripheral serial device → chip:

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

The command is followed by 8 bytes of data, which is the key value of the normal keys of the USB keyboard.

In order:

(1) The first byte: 1 byte of control keys, each bit represents 1 key as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
right (-hand) Windows (computer) linchpin	right (-hand) Alt linchpin	right (-hand) Shift linchpin	right (-hand) Ctrl linchpin	queer Windows (computer) linchpin	queer Alt linchpin	queer Shift linchpin	queer Ctrl linchpin

(2) The second byte: 1 byte 0x00, which must be 0x00;

(3) If there are no keys pressed, the value of the keypad will be displayed in the first 8 bytes of the byte list.

If you want to use the following, fill in 0x00;

See Appendix 1-"CH9329 KeyCodeTable"for specific keypad general keys and their corresponding key codes.

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current

command execution status. The following is an

example:

Example 1: To simulate pressing the "A" key and then releasing the "A" key, 2 command packets need to be sent:

(1) The following are the analog presses of the "A" key: 0x57, 0xAB, 0x00, 0x02, 0x08, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10.

(2) The following keys are used to release the "A" key: 0x57, 0xAB, 0x00, 0x02, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C.

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

(2), Analog release all keys: 0x57, 0xAB, 0x00, 0x02, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C.

2.2.3. CMD_SEND_KB_MEDIA_DATA

Send multimedia keyboard packets to the chip through this command to

simulate multimedia key press or release actions. Peripheral Serial Device → Chip:

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

The command is followed by 2 bytes of data, which are the key values of the multimedia keys of the USB keyboard.

See Appendix 1-"CH9329 KeyCodeTable"for specific keypad general keys and their corresponding key codes.

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status. The following is an example:

Example 1: To simulate pressing and releasing the multimedia's "No Tone" key, send 2
A command package for:

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

(2) The "No Tone" keys for analog release of multimedia: 0x57, 0xAB, 0x00, 0x03, 0x04, 0x02, 0x00, 0x00, 0x00, 0x00, 0x0B.

2.2.4. CMD_SEND_MS_ABS_DATA

The command sends absolute mouse packets to the chip to simulate absolute mouse-related actions (including left, center and right button presses and releases, scroll wheel scrolling up and down, up and down, left and right movements).

Peripheral serial device → chip:

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

The command carries 7 bytes of subsequent data. The 7 bytes of subsequent data are the data packets of USB Absolute Mouse, in order:

(1) The first byte: must be 0x02;

(2) The second byte: 1 byte of the mouse button value, the lowest 3 bits represent 1 button per bit, as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0		center	right (i.e. right side)	Left

0

chemical bond key (on a piano or computer keyboard) key (on a piano or computer keyboard)

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

(2) The X-axis coordinate value is the first byte of the low byte and the second byte of the high byte;

(3) The low byte comes first and the high byte comes second;

(4) 7th byte: byte number of teeth of the scroll wheel, if it is 0, it means that there is no scrolling action;

0x01---0x7F, indicates upward scrolling, unit: number of teeth;

0x81 - 0xFF for downward scrolling, unit: number of teeth;

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status.

Note: The default analog absolute mouse resolution of the chip is 4096 * 4096, and the peripheral serial device downlinks the XY absolute value.
When you do this, you need to calculate the value according to your own screen resolution first, and then download the calculated value.

For example, if the current screen resolution is: X_MAX(1280) * Y_MAX(768), you need to move to the point (100, 120), and you need to do the following calculation:

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

Examples are given below for illustration:

Example 1: To simulate pressing the left mouse button first and then releasing the left mouse button, you need to send 2 command packets as follows: (1), press the left mouse button: 0x57. 0xAB, 0x00, 0x04, 0x07, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10.

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

Example 2: Assuming the screen resolution is: 1280*768, control the mouse to move to the (100,100) position first, and then move to the (968,500) position, then you need to send 2 packets of commands as:

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

Calculate position X1 = (100 * 4096) / 1280 = 320 = 0x140

Calculate position Y1 = (100 * 4096) / 768= 533 = 0x215

Send command packets as 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x00, 0x40, 0x01, 0x15, 0x02, 0x00, 0x67.

(2), move to position (968,500).

Calculate position X1 = (968 * 4096) / 1280 = 3097 = 0xC19

Calculate position Y1 = (500 * 4096) / 768= 2667 = 0xA6B

Send command packets as 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x00, 0x19, 0x0C, 0x6B, 0x0A, 0x00, 0xA9.

2.2.5. CMD_SEND_MS_REL_DATA

The command sends relative mouse packets to the chip to simulate relative mouse-related actions (including left, center and right button press and release, scroll wheel up and down, up and down, left and right movement).

Peripheral serial device → chip:

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

The command carries 5 bytes of follow-up data, which are USB packets relative to the mouse, in that order:

(1) The first byte: must be 0x01;

(2) The second byte: 1 byte of the mouse button value, the lowest 3 bits represent 1 button per bit, as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	middle button	right click	left click

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

(3) The third byte: 1 byte X-direction (horizontal coordinate, left/right direction) travel distance;

A. No movement: Byte 3 = 0x00, then it means no movement in the X-axis direction;

B. Move right: 0x01 <= byte 3 <= 0x7F; move pixel point = byte 3;

C. Move left: 0x80 <= byte 3 <= 0xFF; move pixel point = 0x100 - byte 3;

(4) A, No movement: Byte 4 = 0x00, then it means no movement in the Y direction;

B. Move down: 0x01 <= byte 4 <= 0x7F; move pixel point = byte 4;

C. Move up: 0x80 <= byte 4 <= 0xFF; move pixel point = 0x100 - byte 4;

(5) The fifth byte: 1 byte of the number of rolling teeth of the wheel.

0x01---0x7F, indicating the screen scrolls upward, in teeth;

0x81---0xFF, indicates the screen scrolls down, in teeth;

Calculation of the distance moved by scrolling down:

For example, if the byte is 0x81, the actual distance traveled = 0x100-0x81 = 127 pixels;

For example, if the byte is 0xFF, the actual distance traveled = 0x100-0xFF = 1 pixel.

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status. The following is an example:

Example 1: To simulate pressing the left mouse button first and then releasing the left mouse button, you need to send 2 packets of commands:

(1) Press the left mouse button: 0x57, 0xAB, 0x00, 0x05, 0x05, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0E.

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

Example 2: To control the mouse to move 3 pixels to the left and then 5 pixels down, you need to send 2 packets of commands:

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

(2) The following table describes how to move the program down 5 pixels: 0x57, 0xAB, 0x00, 0x05, 0x05, 0x01, 0x00, 0x00, 0x05, 0x00, 0x12.

2.2.6. CMD_SEND_MY_HID_DATA

Send a custom HID class device packet to the chip with this command. Peripheral Serial Device → Chip:

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

The command is followed by N bytes of data, which is the HID packet you want to upload via USB, N is valid.

The range is: 0-64;

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status.

2.2.7. CMD_READ_MY_HID_DATA

This command is used to read the customized HID device packet from the chip. 1 packet of customized HID packet is transmitted from the PC to the chip, and then automatically packaged by the serial port of the chip and sent to the peripheral serial devices.

Chip → Peripheral serial devices:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x87	N	N bytes of data	0x?

The command is followed by N bytes of data, which is the HID packet transmitted by USB, and the valid range of N is:

0-64;

Note: This command is sent to the peripheral serial device by the chip actively and does not require the peripheral serial device to answer.

2.2.8. CMD_GET_PARA_CFG

This command is used to get the current parameter configuration information from the chip, and the specific parameters are described in the following return data.

Peripheral serial device → chip:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x08	0	not have	0x?

This command does not carry any parameter data.

Chip → Peripheral serial devices:

header	address code	command code	Length of follow-up data	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 are returned:

(1) 0x00: software set working mode 0, standard USB keyboard

(normal+multimedia)+USB mouse (absolute mouse+mouse). relative to the mouse);

0x01: Software set working mode 1, Standard USB keyboard (normal);

0x02: Software set working mode 2, Standard USB mouse (Absolute Mouse + Relative Mouse);

0x03: Software-set operating mode 3, standard USB custom HID class device;

0x80: working mode 0 set by hardware pin, standard USB keyboard (normal + multimedia) + USB mouse (absolute mouse + relative mouse); current MODE1 pin is high, MODE0 pin is high;

0x81: Hardware pin set working mode 1, standard USB keyboard (normal); MODE1 pin is currently high, MODE0 pin is low;

0x82: working mode 2 set by hardware pin, standard USB mouse (absolute mouse + relative mouse); current MODE1 pin is low and the MODE0 pin is high;

0x83: Hardware pin set working mode 3, standard USB custom HID class device; MODE1 pin is currently low, MODE0 pin is low;

(2) 0x00: Serial communication mode 0 set by software, protocol transmission mode;

0x01: Serial communication mode 1 set by software, ASCII mode;

0x02: Serial communication mode 2 set by software, pass-through mode;

0x80: Serial communication mode 0 set by hardware pin, protocol transfer mode; current CFG1 pin is high, CFG0 pin is high;

0x81: Serial communication mode 1 set by hardware pin, ASCII mode; current CFG1 pin is high, CFG0 pin is low;

0x82: Serial communication mode 2, pass-through mode, set by hardware pin; current CFG1 pin is low, CFG0 pin is high;

(3) The valid range is 0x00 - 0xFF, and the default is 0x00;

(4) The default baud rate is 0x00002580, i.e. the baud rate is 9600bps;

(5) The following two bytes are reserved;

(6) The default value is 3 in mS and the valid range is 0x0000 - 0xFFFF. That is, if the chip does not receive the next byte for more than 3mS, it indicates the end of this packet;

(7) The VID and PID of the 4-byte chip USB, the default chip VID is 0x1A86, PID is 0xE129, and the PID is different in different working modes;

(8) The valid range for the 2-byte chip USB keyboard upload interval (valid only in ASCII mode) is 0x0000 - 0xFFFF, default is 0, unit is mS, i.e. the chip uploads the next packet immediately after the first 1 packet;

(9) The valid range is 0x0000 - 0xFFFF, the default is 1, the unit is mS, that is, 1mS after the chip uploads the key press packet, the chip uploads the key release packet;

(10) Valid range is 0x00 - 0x01, 0x00 means no auto carriage return, 0x01 means auto carriage return at the end of this packet;

(11) The 8-byte chip USB keyboard enter character (only valid in ASCII mode), 4 bytes in a group, a total of 2 groups, i.e., you can set up 2 different types of enter characters, the default encountered ASCII value is 0x0D to enter;

(12) The first 4 bytes are the start character of the filter, the last 4 bytes are the start character of the filter, and the last 4 bytes are the start character of the filter. byte is the end character of the filter;

(13) The USB String Enable Flag is a 1-byte chip USB String Enable Flag that is used to enable the USB String.

Bit 7: A value of 0 disables; a value of 1 enables custom string descriptors;

Bit 6-3: Reserved;

Bit 2: A value of 0 disables; a value of 1 enables custom vendor string descriptors;

Bit 1: A value of 0 disables; a value of 1 enables custom product string descriptors;

Bit 0: A 0 means disabled; a 1 means enable custom sequence number string descriptor;

(14) The 1-byte chip USB keyboard fast upload flag (only valid in ASCII mode), valid range is 0x00 - 0x01, 0x00 means USB keyboard upload speed is normal, 0x01 means enable USB keyboard fast upload mode, after enabling fast upload mode, after uploading 1

character, do not send the release key packet, and continue to upload the next character until all characters have been uploaded and then upload the next character. It will continue to upload the next character until all characters have been uploaded.

Send a release key packet.

(15) The following table shows the number of bytes reserved for the first 12 bytes of the program;

2.2.9. CMD_SET_PARA_CFG

This command sets the current parameter configuration information to the chip, and the specific parameter format is described in the previous command. Peripheral serial device → chip:

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

This command has 50 bytes of follow-up data, and the specific data format is shown in the return of the "**CMD_GET_PARA_CFG**" command.

Attention:

(1) The valid range is 0x00-0x03 for chip operating mode setting;

(2) The valid range is: 0x00-0x02; (3), after all parameters are set, the next power-up is enabled.

Chip → Peripheral serial devices:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x89	1	1 byte of data	0x?

The 1-byte follow-up data returned is the current command execution status.

2.2.10. CMD_GET_USB_STRING

Use this command to get the USB string descriptor configuration currently in use from the chip.

Peripheral serial device → chip:

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

The command takes 1-byte arguments, in order:

(1) The following are some examples of the type of string used to describe a product: 0x00 for a vendor string descriptor; 0x01 for a product string descriptor; 0x00 for a vendor string descriptor; 0x01 for a product string descriptor;

0x02 Indicates the sequence number string descriptor;

Chip → Peripheral serial devices:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM

0x57, 0xAB	0x00	0x8A	2+N	2+N bytes of data	0x?

The 2+N bytes of subsequent data returned, in that order:

(1) The type of the string is 1-byte;

(2) The length of the string is 1 byte, and the valid range is 0 to 23;

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

2.2.11. CMD_SET_USB_STRING

This command sets the USB string descriptor configuration currently used to the chip.

Peripheral serial device → chip:

header	address code	command code	Length of follow-up data	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 that order:

(1) The following are some examples of the type of string used to describe a product: 0x00 for a vendor string descriptor; 0x01 for a product string descriptor; 0x00 for a vendor string descriptor; 0x01 for a product string descriptor;

0x02 Indicates the sequence number string descriptor;

(2) The length of the string is 1 byte, and the valid range is 0 to 23;

(3) N byte string descriptor, N valid range: 1-23;

Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status.

2.2.12. CMD_SET_DEFAULT_CFG

This command restores the parameter configuration and string configuration information of the chip to the factory default settings.

Peripheral serial device → chip:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0C	0	not have	0x?

This command comes with no parameters. Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status.

2.2.13. CMD_RESET

This command controls the software reset control of the chip.

Peripheral serial device → chip:

header	address code	command code	Length of follow-up data	Follow-up data	cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0F	0	not have	0x?

This command comes with no parameters. Chip → Peripheral serial devices:

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

The 1-byte follow-up data returned is the current command execution status.

2.3. Error Response Packet

If a command packet received by the chip has a command code error, checksum error, or execution failure, it needs to be passed through the
The answer is given in an error packet. The error response packet contains 1 byte of subsequent data, which is the command execution status. Chip → Peripheral serial device:

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

The 1-byte follow-up data returned is: current command execution status. Returned command code = original command code | 0xC0;

Table 2-The execution status of the command is as follows

Status Name	status code	Status Description
DEF_CMD_SUCCESS	0x00	The command was executed successfully.
DEF_CMD_ERR_TIMEOUT	0xE1	Serial port receives a byte timeout
DEF_CMD_ERR_HEAD	0xE2	Error receiving packet header bytes on serial port
DEF_CMD_ERR_CMD	0xE3	Serial port receive command code error
DEF_CMD_ERR_SUM	0xE4	Mismatch between cumulative and test values
DEF_CMD_ERR_PARA	0xE5	parameter error
DEF_CMD_ERR_OPERATE	0xE6	Normal frame, execution failed

Appendix 1-"CH9329 Keycode Table"

1. Common keys and the corresponding keycode table:

serial number	notation		HID page (on a website)	HID Code	serial number	notation		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			

serial number	notation	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 keys and corresponding keycode table:

For the ACPI key, there are 2 bytes, the first byte is the REPORT ID, fixed at 0x01, the second byte is the ACPI

Keycode.

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 release								

For other multimedia keys, it takes up 4 bytes, the first byte is the REPORT ID, which is fixed to 0x02, the second byte

to the 4th byte are multimedia key values.

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	E-Mail
4	Rewind	Record	Minimize	My Computer	Screen Save	Calculator	Explorer	Media
1:Key pressed 0:key release								