CH9329 Chip Serial Port Communication Protocol

V1.2

Document Change Record

Version No.	Scope of change	Change content	Modifier
V1.0	Document establishment	Establish a document, a first draft	TECH2
V1.1	Document modification	Modify the appendix	TECH2
V1.2	Document modification	Modify simulated mouse action	TECH2

The CH9329 chip has three serial communication modes:

Serial communication mode 0: Protocol transfer mode (default)

Serial communication mode 1: ASCII mode

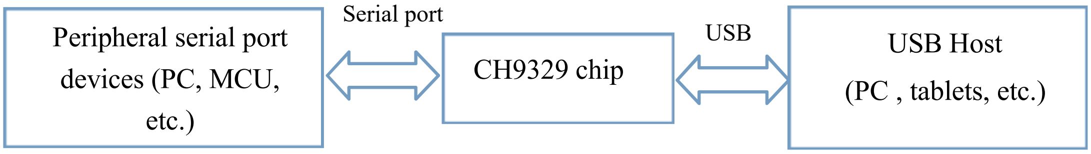
Serial communication mode 2: Transparent transmission mode.

The CH9329 chip works in the serial communication mode 0 (protocol transmission mode) by default. This protocol is mainly used to specify the serial communication protocol in which the CH9329 chip works 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 port device can configure the parameters. Therefore, when you need to configure parameters, you can first set the SET pin to low level, and then configure it.

1. Communication Structure

The communication structure between peripheral serial port devices (PC, MCU or other serial port devices) and CH9329 chip is as follows:



2. Communication Mode

The communication between the peripheral serial port devices (PC, MCU or other serial port devices) and the CH9329 chip is the main slave mode, the peripheral serial port devices are the host and the CH9329 chip is the slave. Commands are initiated by peripheral serial devices, and the CH9329 chip responds passively. If the peripheral serial port device does not receive the reply from the CH9329 chip in the 500mS or the response message is wrong, the communication is considered to have failed.

2.1 Frame Format Description

Communication is in frames, that is, in the form of data packets, each frame of data with a frame header byte, address code, command code, subsequent data length, subsequent data, and cumulative sum. If the CH9329 chip receives the error frame, it returns the error response frame or discards it directly.

The communication frame initiated by the peripheral serial port device is called the "command packet", and the communication frame returned by the CH9329 chip is called the "response packet". For the "command packet", after the peripheral serial port device sends it, it needs to wait for the CH9329 chip to return the "reply packet" and determine whether the command is executed successfully according to the "reply packet". If you return an error status or do not receive an "reply packet", you need to retry or error handling as appropriate.

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

The data formats of the command package and reply package are as follows:

Frame header	Address code	Command code	Subsequent data length	Subsequent 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 as 0x57 and 0xAB

Address code: Occupies 1 byte, the default is 0x00, can receive any address code of the command packet, if the chip address is set to 0x01---0xFE, then can only receive the corresponding address code or address code is 0xFF command package. 0xFF is a broadcast packet, and the chip does not need to reply.

Command code: 1 byte. The valid range of the command code for the frame initiated by the peripheral serial device is: 0x01While 0x3F dint CH9329 chip sends normal response packet, the command code is: original command code | when 0x80bot CH9329 chip sends abnormal response packet, the command code is: original command code | 0xC0

Subsequent data length: 1 byte, mainly used to record the actual subsequent data length of the packet, including only the subsequent data part, excluding frame header bytes, address codes, command codes and cumulative bytes

Follow-up data: Account for N bytes, and the N effective range is 0-64.

Cumulative 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, USB enumeration status, keyboard case indicator status and other information from the chip through this command.
CMD_SEND_KB_GENERAL_DATA	**0x02**	**Send USB keyboard normal data** Through this command, ordinary keyboard packets are sent to the chip to simulate the action of pressing or releasing ordinary keys.
CMD_SEND_KB_MEDIA_DATA	**0x03**	**Send USB keyboard multimedia data** Send multimedia keyboard packets to the chip through this command to simulate the action of pressing or releasing multimedia keys.
CMD_SEND_MS_ABS_DATA	**0x04**	**Send USB absolute mouse data** Through this command, absolute mouse packets are sent to the chip to simulate absolute mouse-related actions.
CMD_SEND_MS_REL_DATA	**0x05**	**Send USB relative mouse data** The relative mouse packet is sent to the chip through this command to simulate the relative mouse related action.
CMD_SEND_MY_HID_DATA	**0x06**	**Send USB custom HID device data** Send custom HID device packets to the chip through this command
CMD_READ_MY_HID_DATA	**0x87**	**Read USB custom HID device data** Read custom HID device packets from the chip through this command *Note: After the PC sends a custom HID packet to the chip, the chip serial port automatically packages and sends it to the peripheral serial port devices.*

CMD_GET_PARA_CFG	**0x08**	**Get parameter configuration** Get the current parameter configuration information from the chip through this command
CMD_SET_PARA_CFG	**0x09**	**Set parameter configuration** Use this command to set the current parameter configuration information to the chip
CMD_GET_USB_STRING	**0x0A**	**Get string descriptor configuration** Get the current USB string descriptor configuration from the chip with this command.
CMD_SET_USB_STRING	**0x0B**	**Set string descriptor configuration** Use this command to set the currently used USB string descriptor configuration to the chip
CMD_SET_DEFAULT_CFG	**0x0C**	**Restore the factory default configuration** Restore the parameter configuration and string configuration information of the chip to the factory default settings through this command
CMD_RESET	**0x0F**	**Reset chip** Software reset control through the command control chip

2.2.1 CMD_GET_INFO

Through this command, we get the version number, USB enumeration status, keyboard case indicator status and other information from the chip.

Chip → Peripheral serial port device chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x01	0x00	Innumerable data	0x03

This command takes no arguments.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x81	0x08	8 bytes data	0x??

The 8 bytes of subsequent data returned are as follows:

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

(2) 1 byte USB enumeration status:
0x00 indicates that the computer is not connected to the USB or is not recognized;
0x01 indicates that the USB side is connected to the computer and recognized successfully;

(3) 1 byte current keyboard size indicator status information
Bit 0: Keyboard NUMLOCK LED status, 0: Off; 1: On.

Bit 1: Keyboard CAPSLOCK LED status, 0: Off; 1: On.
Bit 2: Keyboard SCROLLLOCK LED status, 0: Off; 1: On.
Bit 7-3: Invalid.

(4) 5 bytes reserved.

2.2.2 CMD_SEND_KB_GENERAL_DATA

Through this command, ordinary keyboard packets are sent to the chip to simulate the action of pressing or releasing ordinary keys. Support full keyboard, key combination operation, can support 8 conflict-free keys, of which 8 are 8 control keys (left Ctrl, right Ctrl, left Shift, right Shift, left Windows, right Windows, left Alt and right Alt), 6 are normal keys other than 6 control keys.

Peripheral serial port device → Chip:

Frame header	**Address code**	**Command code**	**Subsequent data length**	**Subsequent data**	**Cumulative sum**
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x02	8	8 bytes data	0x??

The command takes 8 bytes of subsequent data, which is the key value of the normal key of the USB keyboard. The order is as follows:

(1) 1st byte: A control key of 1 byte, each bit represents a key, as follows:

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

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

(3) 3-8 bytes: 6 bytes of normal key value, which can indicate up to 6 key presses. If no key is pressed, fill in 0x00. For specific keyboard common keys and corresponding key codes, see Appendix 1-"CH9329 key Code Table".

Chip → Peripheral serial port devices

Frame header	**Address code**	**Command code**	**Subsequent data length**	**Subsequent data**	**Cumulative sum**
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x82	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

The following examples are given to illustrate:

Example 1: If the analog presses the "A" key and then releases the "A" key, two command packages need to be sent:

(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: If the analog presses the "left Shift" + "A" key at the same time, and then releases it, two command packages need to be sent:

(1) Simulate pressing the "left Shift" + "A" key: 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

Through this command, a multimedia keyboard packet is sent to the chip to simulate the action of pressing or releasing the multimedia key.

Peripheral serial port device → chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x82	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

The following examples are given to illustrate:

Example 1: If the simulation first presses the multimedia "mute" key, and then releases the multimedia "mute" key, two command packages need to be sent:

(1) Press the "mute" key of multimedia: 0x57, 0xAB, 0x00, 0x03, 0x04, 0x02, 0x04, 0x00, 0x00, 0x0F

(2) Simulate the release of the multimedia "mute" key: 0x57, 0xAB, 0x00, 0x03, 0x04, 0x02, 0x00, 0x00, 0x00, 0x0B

2.2.4 CMD_SEND_MS_ABS_DATA

Through this command, absolute mouse packets are sent to the chip to simulate absolute mouse-related actions (including pressing and releasing the left, middle and right keys, scrolling the wheel up and down, moving up and down, left and right).

Peripheral serial port device → chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x04	7	7 bytes data	0x??

The command takes 7 bytes of follow-up data, and the 7-byte follow-up data is the packet of USB absolute mouse, which is in turn:

(1) 1st byte: Must be 0x02;

(2) 2nd byte: a mouse key value of 1 byte, with a minimum of 3 bits each representing a key, as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	Middle key	Right key	Left key

BIT2---BIT0: 1 indicates that the key is pressed, and 0 indicates that the key is released or not pressed.

(2) 3-4 bytes: 2-byte X-axis coordinate value, low byte first, high byte last

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

(4) 7th byte: Number of rolling teeth of 1 byte roller

If 0, it indicates scrolling has no action

0x01---0x7F, which indicates scrolling up, in number of teeth

0x81---0xFF, which indicates to scroll down, in number of teeth

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x84	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

Note: The absolute mouse resolution of the chip is 4096mm 4096 by default. When the peripheral serial port equipment downloads the absolute value of XY, it needs to be calculated according to its own screen resolution, 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 (100120) and calculate as follows:

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

The following examples are given to illustrate:

For example 1: If the simulation first presses the "left" key of the mouse, and then releases the "left" key of the mouse, you need to send two command packages:

(1) Press the left mouse key: 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10
(2) Release the left mouse key: 0x57, 0xAB, 0x00, 0x04, 0x07, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F

For example 2: If the screen resolution is 1280mm / 768 and the mouse is moved to (100100) and then to (968500), two command packages need to be sent:

(1) Move to position (100, 100):
Calculate position X1 = (100 * 4096) / 1280 = 320 = 0x140
Calculate position Y1 = (100 * 4096) / 768 = 533 = 0x215
Send the command package as follows: 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 the command package as follows: 0x57, 0Xab, 0x00, 0x04, 0x07, 0x02, 0x00, 0x19, 0x0C, 0x6B, 0x0A, 0x00, 0xA9

2.2.5 CMD_SEND_MS_REL_DATA

Through this command, relative mouse packets are sent to the chip to simulate relative mouse-related actions (Including pressing and releasing the left, middle and right keys, scrolling the wheel up and down, moving up and down, left and right).

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM

0x57, 0xAB	0x00	0x05	5	5 bytes data	0x??

The command takes 5 bytes of subsequent data, and the subsequent data is the packet of USB relative to the mouse, in turn:

(1) 1st byte: Must be 0x01

(2) 2nd byte: A mouse key value of 1 byte, with a minimum of 3 bits each representing a key, as follows:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	Middle key	Right key	Left key

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

(3) 3rd byte: 1 byte X direction (Abscissa, left and right direction) moving distance
A, do not move: byte 3=0x00, indicates it does not move in the X-axis.
B, move to the right: 0x01 < = byte 3 < = 0x7F; move pixel = byte 3
C, move to the left: 0x80 < = byte 3 < = 0xFF; move pixel = 0x100-byte 3

(4) 4th byte: 1 byte Y direction (ordinate, up and down) moving distance
A, do not move: byte 4=0x00, indicates it does not move in the Y-axis.
B, move down: 0x01 < = byte 4 < = 0x7F; move pixel = byte 4
C, move up: 0x80 < = byte 4 < = 0xFF; move pixel = 0x100-byte 4

(5) 5th byte: Number of rolling teeth of 1 byte roller
0x01---0x7F, indicating that the screen scrolls up, in number of teeth
0x81---0xFF, indicating that the screen scrolls down, in number of teeth
The method for calculating the distance of scrolling down:
For example, the byte is 0x81, and the actual moving distance = 0x100-0x81=127 pixels
For example, the byte is 0xFF, and the actual moving distance = 0x100-0xFF=1 pixels.

Chip → Peripheral serial port devices:

Frame header	**Address code**	**Command code**	**Subsequent data length**	**Subsequent data**	**Cumulative sum**
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x85	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

The following examples are given to illustrate:

For example, 1: If the simulation first presses the "left" key of the mouse, and then releases the "left" key of the mouse, you need to send two command packages:

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

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

For example, 2: If you control the mouse to move 3 pixels to the left and then 5 pixels down, you need to send 2 command packages:

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

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

2.2.6 CMD_SEND_MY_HID_DATA

Send custom HID device packets to the chip through this command.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x06	N	N bytes data	0x??

This command carries N bytes of follow-up data, which is the HID packet you want to upload via USB, and the valid range of N is 0-64.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x86	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

2.2.7 CMD_READ_MY_HID_DATA

Use this command to read custom HID device packets from the chip. After the PC sends one packet of custom HID packet to the chip, the chip serial port automatically packages and sends it to the peripheral serial port equipment.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x87	N	N byte data	0x??

This command takes N bytes of subsequent data, which is the HID packet downloaded by USB. The N valid range is 0-64.

Note: This command is actively sent by the chip to the peripheral serial device and does not need to be answered by the peripheral serial device.

2.2.8 CMD_GET_PARA_CFG

Get the current parameter configuration information from the chip through this command. For more information, please see the returned data below.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent 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 port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM

0x57, 0xAB	0x00	0x88	50	50 bytes data	0x??

The 50 bytes of subsequent data returned is:

(1) 1-byte chip working mode: valid values are 0x00-0x03, 0x80---0x83. Default is 0x80.

0x00: Working mode 0 of the software settings, standard USB keyboard (normal + multimedia) + USB mouse (absolute mouse + relative mouse)

0x01: Working mode 1 of software settings, standard USB keyboard (normal)

0x02: Working mode 2 of software settings, standard USB mouse (absolute mouse + relative mouse)

0x03: Working mode 3 of software settings, standard USB custom HID class devices

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

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

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

0x83: Working mode 3 of hardware pin settings, standard USB custom HID devices; current MODE1 pin is low, MODE0 pin is low

(2) 1-byte chip serial communication mode. Valid values are 0x00-0x02, 0x80---0x82. Default is 0x80.

0x00: Serial communication mode 0 and protocol transfer mode set by the software

0x01: Serial communication mode set by software 1Magi ASCII mode

0x02: Serial communication mode 2, transparent mode set by software

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

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

0x82: Serial communication mode 2 set by hardware pin, transparent mode; current CFG1 pin is low level, CFG0 pin is high level

(3) 1 byte serial communication address of chip. Valid range is 0x00--0xFF. Default is 0x00.

(4) 4-byte chip serial communication baud rate, high byte comes first, default is 0x00002580, that is, baud rate is 9600bps

(5) 2 bytes reserved

(6) 2-byte chip serial communication packet interval, valid range is 0x0000--0xFFFF, default is 3, unit is mS, that is, if the chip exceeds 3mS and does not receive the next byte, the packet ends

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

(8) 2-byte chip USB keyboard upload interval (valid only in ASCII mode). Valid range is 0x0000--0xFFFF, default is 0, 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 (valid only in ASCII mode). Valid range is 0x0000--0xFFFF, default is 1, unit is mS, that is, after the chip uploads and presses the data packet, 1mS uploads the key to release the data packet.

(10) 1 byte chip USB keyboard auto enter flag (valid only in ASCII mode). Valid range: 0x00Murray 0x01d0x00 indicates no automatic enter. 0x01 indicates to enter automatically after the end of this package.

(11) 8-byte chip USB keyboard carriage return (valid only in ASCII mode), 4 bytes in a group, a total of 2 groups, that is, you can set two different carriage returns. Enter when the ASCII value is 0x0D by default.

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

(13) 1 byte chip USB string enable flag

Bit 7: 0 for forbidden; 1 for enable custom string descriptor

Bit 6-3: Reserved

Bit 2: 0 indicates forbidden; 1 indicates enable custom vendor string descriptor

Bit 1: 0 indicates forbidden; 1 indicates enable custom product string descriptor

Bit 0: 0 indicates forbidden; 1 indicates enable 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 indicates USB keyboard upload speed is normal, 0x01 indicates enable USB keyboard fast upload mode, after enabling fast upload mode, after uploading one character, it will not send the release key packet, and continue to upload the next character, and only send the release key packet after all characters have been uploaded.

(15) 12 bytes reserved

2.2.9 CMD_SET_PARA_CFG

The current parameter configuration information is set to the chip through this command, and the specific parameter format is described in the previous instruction.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x09	50	50 bytes data	0x??

This command takes 50 bytes of subsequent data, which is formatted in the return of the "CMD_GET_PARA_CFG" instruction.

Note:

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

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

(3) After all the parameters are set, the next power-up will be enabled.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x89	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

2.2.10 CMD_GET_USB_STRING

Through this command, the USB string descriptor configuration currently used is obtained from the chip.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0A	1	1 byte data	0x??

The command takes a 1-byte argument, in turn:

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

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8A	2+N	2+N bytes data	0x??

The 2 bytes of subsequent data returned is as follows:

(1) 1 byte string type

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

(3) N bytes of the current string descriptor, N valid range: 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 port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0B	2+N	2+N bytes data	0x??

The command takes an argument of 2 bytes, which in turn is:

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

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

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

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x88	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

2.2.12 CMD_SET_DEFAULT_CFG

Through this command, the parameter configuration and string configuration information of the chip are restored to the factory default settings.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0C	0	None	0x??

This command takes no arguments.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum

HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8C	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

2.2.13 CMD_SET_DEFAULT_CFG

The software reset control is carried out through the command control chip.

Peripheral serial port devices → Chip:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x0F	0	None	0x??

This command takes no arguments.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0x8F	1	1 byte data	0x??

The 1 byte of subsequent data returned is the current command execution status.

2.3 Error Response Packet

If there are some problems in the command packet received by the chip, such as command code error, verification error or execution failure, it needs to be answered by the error response packet. The error response packet contains 1 byte of subsequent data, which is the command execution status.

Chip → Peripheral serial port devices:

Frame header	Address code	Command code	Subsequent data length	Subsequent data	Cumulative sum
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57, 0xAB	0x00	0XC?	1	1 byte data	0x??

The 1 byte of subsequent data returned is the 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	Command executed successfully
DEF_CMD_ERR_TIMEOUT	0xE1	Serial port receives one byte timeout
DEF_CMD_ERR_HEAD	0xE2	Error receiving packet header byte through serial port
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 normal, execution failed

Appendix 1-"CH9329 Key Code Table"

1. Ordinary keys and the corresponding key code table:

No.	Symbol		HID Page	HID Code	No.	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	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 KeyboardOnly * 5 _ 105 KeyboardOnly					*B _ 107 KeyboardOnly *J _ 109 Keyboard Only			

No.	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 keys and corresponding key codes table:

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

Byte	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 press 0: Key release								

For other multimedia keys, it accounts for 4 bytes, the first byte is REPORTID, fixed as 0x02, and the second to the fourth byte is the multimedia key value.

Byte	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 press 0: Key release								