USB Bus Interface Chip CH372

Datasheet (II): External Firmware Mode
Version: 1A
http://wch.cn

1. Additional Commands of External Firmware

Code	Command Name	Input Data	Output Data	Purpose
13H	SET_USB_ADDR	Address value		Set USB address
18H	SET_ENDP2	Working mode	(Wait for 4uS)	Set the receiver for USB endpoint 0
19H	SET_ENDP3	Working mode	(Wait for 4uS)	Set the transmitter for USB endpoint 0
1AH	SET_ENDP4	Working mode	(Wait for 4uS)	Set the receiver for USB endpoint 1
1BH	SET_ENDP5	Working mode	(Wait for 4uS)	Set the transmitter for USB endpoint 1
1CH	SET_ENDP6	Working mode	(Wait for 4uS)	Set the receiver for USB endpoint 2
1DH	SET_ENDP7	Working mode	(Wait for 4uS)	Set the transmitter for USB endpoint 2
0AH	GET_TOGGLE	Data 1AH	Sync Status	Get the current OUT transaction synchronization state
29H	WR_USB_DATA3	Data Length		Write the data block to upload buffer of USB endpoint 0
		Data stream		

If the input data is the working mode of transceiver for USB endpoint, refer to the following table.

Working mode byte	Name	Bit analysis description of working mode	
Bits 7-6	Synchronous trigger flag	If the bit 7 is 1, the bit 6 will be the new synchronous trigger flag:	
		00 or 01 = Keep the current synchronous trigger flag unchanged	
		10 = Synchronous trigger flag set to 0	11 = Synchronous trigger flag set to 1
Bits 5-4	(Reserved bit)	(Undefined, must be 0)	
Bits 3-0	Transaction response mode	1101 = (reserved mode, prohibited)	
		1110 = Device busy, return NAK	
		1111= Device error, return STALL	
		0000 ~ 1000= Device ready, For the transmitter for endpoint 0 and endpoint 1, this value also indicates the length of the data to be sent, 0000-1000 respectively indicate transmit lengths 0-8, ACK is returned for OUT and DATA is returned for IN	

1.1. Command SET_USB_ADDR

This command is used to set the USB device address. This command needs to input 1 data. In the external firmware mode, the external MCU must immediately write the USB device address assigned by the USB host into CH372 through this command after processing the USB standard device request SET_ADDRESS, so that CH372 can enable the new USB address to communicate with the USB host.

1.2. Command SET_ENDP2

This command is used to set the receiver for USB endpoint 0. This command needs to input 1 data to specify a new working mode. For example, if the USB device cannot process the data received in the control write operation in a timely manner, the transaction response of the receiver at the USB endpoint 0 can be set through this command to return NAK to the OUT transaction, so that the USB host waits and retransmits the data. The byte in the corresponding working mode is 0EH. Typically, this command is completed within 4uS.

1.3. Command SET_ENDP3

This command is used to set the transmitter for USB endpoint 0. This command needs to input 1 data to specify a new working mode. For example, if the USB device does not support the USB standard device request SET_INTERFACE, the transaction response mode of the transmitter at the endpoint 0 can be set through this command upon the receipt of the request, so as to return STALL to IN transaction. The byte in the corresponding working mode byte is 0FH. Typically, this command is completed within 4uS.

1.4. Command SET_ENDP4

This command is used to set the receiver for USB endpoint 1. This command needs to input 1 data to specify a new working mode. For example, if receiver for the endpoint 1 has an error, the transaction response mode of the receiver at the endpoint 1 can be set through this command, so as to return STALL to OUT transaction. The byte in the corresponding working mode byte is 0FH. Typically, this command is completed within 4uS.

1.5. Command SET_ENDP5

This command is used to set the transmitter for USB endpoint 1. This command needs to input 1 data to specify a new working mode. For example, if the transmitter for the endpoint 1 has no data to transmit for a while, the transaction response of the transmitter at the USB endpoint 1 can be set through this command to return NAK to the IN transaction, so that the USB host waits and receives the data again. The byte in the corresponding working mode is 0EH. Typically, this command is completed within 4uS.

1.6. Command SET_ENDP6

This command is used to set the receiver for USB endpoint 2. This command needs to input 1 data to specify a new working mode. For example, if a USB standard device request SET_CONFIG or CLEAR_FEATURE and ENDPOINT_HALT for the receiver of the endpoint 2 are received, the synchronous trigger flag of the receiver at the endpoint 2 must be set to 0 through this command. The byte in the corresponding working mode byte is 80H. Typically, this command is completed within 4uS.

1.7. Command SET_ENDP7

This command is used to set the transmitter for USB endpoint 2. This command needs to input 1 data to specify a new working mode. For example, if the transmitter for the endpoint 2 has no data to transmit for a while, the transaction response of the transmitter at the USB endpoint 2 can be set through this command to return NAK to the IN transaction, so that the USB host waits and receives the data again. The byte in the corresponding working mode is 0EH. Typically, this command is completed within 4uS.

1.8. Command GET_TOGGLE

This command is used to get the current OUT transaction synchronous state. This command needs to input 1 data 1AH, and the output data is the synchronous state. The bit 4 is 1, indicating that the current OUT transaction is synchronous, and the bit 4 is 0, indicating that the current OUT transaction is not synchronous. In the control write operation, if CH372 requests the OUT transaction interrupt successfully to the external MCU, MCU shall determine whether the current OUT transaction is synchronous through this command. If not, it shall be ignored.

1.9. Command WR_USB_DATA3

This command is used to write data block to the upload buffer for USB endpoint 0. The input data firstly written is data block length, namely, the number of bytes of subsequent data streams. The effective value for the length of the data block is 0 to 8. If the length is not 0, MCU must write the subsequent data to CH372 one by one. For example, this command can be used to return the first 8 bytes of the USB descriptor to the USB host, and then execute the command for many times to return the subsequent data of the USB descriptor.

2. Description of External Firmware

2.1. Endpoint 0

The receive buffer and transmit buffer at the endpoint 0 are 8 bytes respectively. SETUP transaction and OUT transaction use the same receive buffer, but use different transaction response modes.

After successfully completing the SETUP transaction of the endpoint 0, CH372 will automatically set the synchronous trigger flag of the receiver and transmitter for the endpoint 0 to 1, and then notify the external MCU in the interrupt mode to read SETUP data and process it.

When successfully completing the OUT transaction of the endpoint 0, CH372 will automatically trigger the synchronous trigger flag of the receiver for the endpoint 0 from 0 to 1 and from 1 to 0.

When successfully completing the IN transaction of the endpoint 0, CH372 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint 0 from 0 to 1 and from 1 to 0.

For read operation control, as CH372 automatically sets the synchronous trigger flag to 1 after completing the SETUP transaction, the first set of data sent by CH372 is DATA1 by default, then DATA0, then DATA1, and so on. In general, the external MCU only needs to prepare the data and send it without considering the synchronous trigger flag.

As CH372 does not analyze the synchronous trigger flag when completing the OUT transaction of the endpoint 0, it will notify the external MCU in the interrupt mode whether the data is synchronous or not, so for the write operation control, MCU can determine whether the current OUT transaction is synchronous or not and then process it through the command GET_TOGGLE.

The command SET_ENDP2 has no effect on SETUP transaction response mode. The receiver of the endpoint 0 will return NAK for SETUP transaction if the USB buffer is not released, and return ACK if the USB buffer is released.

If the command WR_USB_DATA3 is executed after the command SET_ENDP3, the transmitter of the endpoint 0 will automatically set the transaction response to return DATA for IN transaction, and the data length will determined by the command WR_USB_DATA3.

If the command SET_ENDP3 is executed after the command WR_USB_DATA3, the transmitter of the endpoint 0 will keep the data unchanged, but the transaction will be processed in the transaction response mode set by the command SET_ENDP3. If the transaction response mode is set to return DATA to IN, the transmission length will be redetermined by the command SET_ENDP3.

2.2. Endpoint 1

The receive buffer and transmit buffer at the endpoint 1 are 8 bytes respectively.

When successfully completing the OUT transaction of the endpoint 1, CH372 will automatically trigger the synchronous trigger flag of the receiver for the endpoint 1 from 0 to 1 and from 1 to 0.

When successfully completing the IN transaction of the endpoint 1, CH372 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint 1 from 0 to 1 and from 1 to 0.

CH372 will automatically analyze the synchronous trigger flag when completing the OUT transaction of the endpoint 1. If the data is not synchronous, the external MCU will not be notified in the interrupt mode, and the external MCU will only receive the OUT transaction interrupt of data synchronization.

If the command WR_USB_DATA5 is executed after the command SET_ENDP5, the transmitter of the endpoint 1 will automatically set the transaction response to return DATA for IN transaction, and the data length will determined by the command WR_USB_DATA5.

If the command SET_ENDP5 is executed after the command WR_USB_DATA5, the transmitter of the endpoint 1 will keep the data unchanged, but the transaction will be processed in the transaction response mode set by the command SET_ENDP5. If the transaction response mode is set to return DATA to IN, the transmission length will be redetermined by the command SET_ENDP5.

2.3. Endpoint 2

The receive buffer and transmit buffer at the endpoint 2 are 64 bytes respectively.

When successfully completing the OUT transaction of the endpoint 2, CH372 will automatically trigger the synchronous trigger flag of the receiver for the endpoint 2 from 0 to 1 and from 1 to 0.

When successfully completing the IN transaction of the endpoint 2, CH372 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint 2 from 0 to 1 and from 1 to 0.

CH372 will automatically analyze the synchronous trigger flag when completing the OUT transaction of the endpoint 2. If the data is not synchronous, the external MCU will not be notified in the interrupt mode, and the external MCU will only receive the OUT transaction interrupt of data synchronization.

If the command WR_USB_DATA7 is executed after the command SET_ENDP7, the transmitter of the endpoint 2 will automatically set the transaction response to return DATA for IN transaction, and the data length will determined by the command WR_USB_DATA7.

If the command SET_ENDP7 is executed after the command WR_USB_DATA7, the transmitter of the endpoint 2 will keep the data and length unchanged, but the transaction will be processed in the transaction response mode set by the command SET_ENDP7.

2.4. External Firmware Reference Flow

MCU source program in external firmware mode is provided in the CH372 evaluation board data. The following procedure is used for the external MCU to handle USB standard device request for reference.

(I) After MCU is started, it first initializes CH372 as a USB device mode using external firmware, and then sets the interrupt.

(II) When receiving the interrupt, MCU uses the command GET_STATUS to get the interrupt status. Analysis and processing are as follows:

(1) If OUT for the endpoint 2 or 1 is successful, the command RD_USB_DATA will be used to read the data and notify the main program to process it.
(2) If IN for the endpoint 2 or 1 is successful, the command UNLOCK_USB will be used to release the buffer and notify the main program to continue.
(3) If SETUP for the endpoint 0 is successful, the command RD_USB_DATA will be used to read the

data. Analysis and processing are as follows:

① If it is a USB request CLEAR_FEATURE, analyze and process according to FEATURE in the request and the endpoint number. For ENDPOINT_HALT, the command SET_ENDP can be used.

② If it is a USB request GET_DESCRIPTOR, use the command WR_USB_DATA3 to return the first 8 bytes of the descriptor, and save the USB request command and the current descriptor count for subsequent return.

③ If it is a USB request SET_ADDRESS, save the USB request command.

④ If it is a USB request SET_CONFIG, save the set value, and notify the main program whether the USB initialization is successful or not.

⑤ If it is a USB request GET_CONFIG, use the command WR_USB_DATA3 to return the current configuration value.

⑥ If it is a USB request GET_INTERFACE, use the command WR_USB_DATA3 to return the current interface value.

⑦ If it is a USB request GET_STATUS, use the command WR_USB_DATA3 to return the current status value.

⑧ Other USB requests are handled as needed. If it is not supported, use the command SET_ENDP2 or 3 to set the response to STALL.

(4) If OUT for the endpoint 0 is successful, the command RD_USB_DATA will be used to read the data, and the data can be discarded.

(5) If IN for the endpoint 0 is successful, process as follows according to the saved USB request command:

① If it is a USB request GET_DESCRIPTOR, use the command WR_USB_DATA3 to continue to return the remaining descriptors.

② If it is a USB request SET_ADDRESS, use the command SET_USB_ADDR to set the USB address.

③ Any USB request includes the above request. Use the command UNLOCK_USB before the interrupt exits.

(6) If USB bus is reset, clear the configuration value, etc. CH372 will automatically clear the USB address and the synchronous trigger flag.

(7) Any CH372 interrupt must correspond to a unique command UNLOCK_USB or RD_USB_DATA.