CP210x Virtual COM Port Interface

1. Introduction

This document describes the CP210x Virtual COM Port interface between the host and CP210x device for the purposes of creating custom drivers that interface with the CP210x. For more information on how to customize the CP210x hardware, see "AN721: CP21xx Device Customization Guide".

1.1. Related Documents

Universal Serial Bus Specification: version 2.0 (also referred to as the USB Specification). This specification is available at http://www.usb.org.

Universal Serial Bus Class Definitions for Communication Devices: version 1.1 (also referred to as the CDC Specification, where "CDC" stands for "Communication Device Class"). This specification is available at http://www.usb.org.

ANSI/TIA-602: Serial Asynchronous Automatic Dialing and Control - available at https://global.ihs.com.

1.2. Terms

end device: refers to the hardware connected to the CP210x UART pins.

host: refers to the PC driver and host controller.

2. Interface Architecture

The CP210x device implements one or more communications interfaces implemented using the Bulk protocol. These USB Bulk pipes transport data only and any status information must be obtained from the control pipe.

To configure and control the port, the host sends requests to the device via the control pipe. The host uses these requests to perform such functions as setting the baud rate, setting handshaking modes, and configuring special characters.

To send data from the host to the port, the host queues BULK-OUT packets to the endpoint of the interface associated with the port. The data from these packets will be loaded into the interface's data buffers and passed on to the external device.

To move data from the port to the host, the host issues IN requests to the port's data IN endpoint.

3. Flow Control

The Virtual COM Port protocol is designed to allow ports to perform the same handshaking that is performed by PCs when running Windows with a real serial port. To do this, the device is required to follow detailed rules for flow control.

The Virtual COM Port protocol provides full-duplex data streams. The transmit stream is used for data from the host to the serial port; the receive stream is used for data from the serial port to the host. Each stream has provisions for flow control.

- Transmit flow control allows the end device to ask the port to stop transmitting data. The CP210x will continue to accept OUT data from the host until its internal buffers are full, and then will NACK further OUT packets to the transmit endpoint. When the end device has accepted enough data from the CP210x buffers, the CP210x will accept more OUT packets from the host.
- Receive flow control allows the port to ask the end device to stop transmitting data to the host. The CP210x device holds the data received from the end device in buffers until the host asks for it. If the host does not ask for data in a timely fashion (usually because the application is busy), the CP210x's buffers may start to fill up, and the CP210x uses receive flow control to ask the end device to stop sending data. When the host catches up and empties the CP210x's buffers, the CP210x then uses receive flow control to ask the end device to resume sending data.

All flow-control handling is performed in the CP210x device. This is done for two reasons:

- To reduce overhead at the host.
- To eliminate latency for responding to flow-control events.

The host's only involvement with flow control is to set the device into the desired flow control mode.

3.1. XON/XOFF Flow Control

Two characters are defined by the "set special characters" mechanism (SET_CHARS): XON (normally 0x11, CTRL/Q), and XOFF (normally 0x13, CTRL/S). The same values are used for XON and XOFF for both transmit and receive flow control. XON and XOFF may be set to the same character, but usually are different.

The CP210x device follows "OS/2 rules:" if transmit flow control says "don't transmit", the port prevents any characters from being transmitted, including the flow control characters. The device normally stops transmitting whenever it sends XOFF for receive flow control. The port resumes transmitting only when it sends XON to allow the remote device to send data. This prevents a potential buffer overflow because the flow control characters cannot be transmitted while XOFF is in effect.

3.2. Flow Control Rules

- The CP210x uses a bit pattern similar to the ulHoldReasons mask from the Serial Status response (see Table 8) to control transmission. If any of the hold reasons are set, the CP210x will not send (except that certain characters, such as XON for receive flow control, can always be sent).
- If the CP210x sends an XOFF special character in order to stop reception, the device will transmitting until the device decides to send XON. While transmission is halted, the CP210x will set bit 4 in ulHoldReasons (see Table 8).
- If the CP210x receives an XOFF special character and is configured to XON/XOFF flow control mode, then the CP210x will stop transmitting characters. While stopped, the CP210x sets bit 3 in ulHoldReasons. When an XON is received, the CP210x clears bit 3 in ulHoldReasons and resumes transmission if ulHoldReasons [5..0] is now zero.
- If the CP210x is instructed to send a BREAK, it will stop sending normal characters while BREAK is asserted and set bit 5 of ulHoldReasons.
- If the CP210x is waiting for any of the modem control signals CTS, DSR or DCD, the CP210x sets the appropriate bits in ulHoldReasons when the corresponding signal is causing a delay; the device will clear those bits when the hold condition is removed, either because the flow-control programming was changed, or because the signal changed state externally.
- Immediate characters are not blocked by software flow control (either due to XOFF sent or due to XOFF received). However, they are held off by BREAK, CTS handshaking, DSR handshaking or DCD handshaking (if enabled and holding off traffic).
- After sending an XOFF, the CP210x sends XON to start receiving data only if output is not being held, or otherwise if either a sent XOFF or a received XOFF is the only reason that output is being held.

4. CP210x Hardware Descriptors

The CP210x device descriptor contains zero in the bDeviceClass, bDeviceSubClass, and bDeviceProtocol fields. All other fields are set as defined in Section 9.6.1 "Device" of the USB Specification.

Table 1. CP210x Default Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of the descriptor in bytes.
1	bDescriptorType	1	0x01	Device descriptor type code.
2	bcdUSB	2	0x0110 or 0x0200	Claimed version compliance, where the LSB is the low order and the MSB is the high order (0x0100 is version 1.0).
4	bDeviceClass	1	0x00	Indicates that this device is of no particular device class.
5	bDeviceSubClass	1	0x00	Inidicates that there's no particular subclass.
6	bDeviceProtocol	1	0x00	No device-specific protocol.
7	bMaxPacketSize	1	0x40	Indicates the maximum packet size for Endpoint0.
8	idVendor	2	0x10C4	Manufacturer's Vendor ID.
10	idProduct	2	0xEA60 or 0xEA70	Manufacturer's Product ID.
12	bcdDevice	2	0x0100	Product version in BCD.
14	iManufacturer	1	0x01	Manufacturer string descriptor index.
15	iProduct	1	0x02	Product string descriptor index.
16	iSerialNumber	1	0x03 or 0x05	Serial string descriptor index.
17	bNumConfigurations	1	0x01	Number of possible configurations.

Table 2. CP210x Default Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of the descriptor in bytes.
1	bDescriptorType	1	0x02	Configuration descriptor type code.
2	wTotalLength	2	0x0020 or 0x0037	Total length of data returned for configuration 0 (stored in little-endian order).
4	bNumInterfaces	1	0x01, 0x02, or 0x04	Number of interfaces in this configuration (CP2105 has two interfaces; CP2108 has four interfaces).
5	bConfigurationValue	1	0x01	Indicates the value to be used in a Set Configuration packet to activate this configuration.
6	iConfiguration	1	0x00	Index of the string descriptor describing this configuration.
7	bmAttributes	1	0x80	Indicates the characteristics of the device when this configuration is selected: bit 7 set: Device is bus powered. bit 6 set: Device is self powered. bit 5-0: reserved.
8	MaxPower	1	0x32	Maximum power consumption in 2mA units. For self-powered devices, this field is zero.

Table 3. CP210x Default Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of the descriptor in bytes.
1	bDescriptorType	1	0x04	Interface descriptor type code.
2	bInterfaceNumber	1	0x00, 0x01, 0x02, or 0x03	The interface described by this descriptor (CP2105 has two interfaces; CP2108 has four interfaces).
3	bAlternateSetting	1	0x00	Indicates the alternate setting which enables the interface to operate as described.
4	bNumEndpoints	1	0x02	This interface has 2 Endpoints.
5	bInterfaceClass	1	0xFF	Indicates that this is a vendor-specific interface.
6	bInterfaceSubClass	1	0x00	No particular subclass.
7	bInterfaceProtocol	1	0x00	The interface uses no specific protocol.
8	iInterface	1	0x02, 0x03, or 0x04	Index of the string descriptor describing this interface.

Table 4. CP210x Default IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of the descriptor in bytes.
1	bDescriptorType	1	0x05	Endpoint descriptor type code.
2	bEndpointAddress	1	0x81 or 0x82	IN Endpoint (either Endpoint 1 or Endpoint 2).
3	bmAttributes	1	0x02	Bulk Endpoint.
4	wMaxPacketSize	2	0x0040 or 0x0020	The maximum packet size is 64 or 32 bytes.
5	bInterval	1	0x00	Not used for Bulk Endpoints.

Table 5. CP210x Default OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of the descriptor in bytes.
1	bDescriptorType	1	0x05	Endpoint descriptor type code.
2	bEndpointAddress	1	0x01 or 0x02	OUT Endpoint (either Endpoint 1 or Endpoint 2).
3	bmAttributes	1	0x02	Bulk Endpoint.
4	wMaxPacketSize	2	0x0040 or 0x0020	The maximum packet size is 64 or 32 bytes.
5	bInterval	1	0x00	Not used for Bulk Endpoints.

5. CP210x Control Commands

The CP210x commands in Table 6 follow the standard USB Setup packet structure and are defined in addition to the messages defined as part of Chapter 9 of the USB Specification. The interface number can be obtained from the CP210x descriptors upon enumeration and driver load. This section discusses each of these commands.

Table 6. Interface Commands

Name	Code	Description	Page
IFC_ENABLE	0x00	Enable or disable the interface.	8
SET_BAUDDIV	0x01	Set the baud rate divisor.	8
GET_BAUDDIV	0x02	Get the baud rate divisor.	8
SET_LINE_CTL	0x03	Set the line control.	9
GET_LINE_CTL	0x04	Get the line control.	10
SET_BREAK	0x05	Set a BREAK.	14
IMM_CHAR	0x06	Send character out of order.	14
SET_MHS	0x07	Set modem handshaking.	10
GET_MDMSTS	0x08	Get modem status.	11
SET_XON	0x09	Emulate XON.	11
SET_XOFF	0x0A	Emulate XOFF.	12
SET_EVENTMASK	0x0B	Set the event mask.	12
GET_EVENTMASK	0x0C	Get the event mask.	13
GET_EVENTSTATE	0x16	Get the event state.	13
SET_RECEIVE	0x17	Set receiver max timeout.	14
GET_RECEIVE	0x18	Get receiver max timeout.	14
SET_CHAR	0x0D	Set special character individually.	15
GET_CHARS	0x0E	Get special characters.	15
GET_PROPS	0x0F	Get properties.	15
GET_COMM_STATUS	0x10	Get the serial status.	14
RESET	0x11	Reset.	8
PURGE	0x12	Purge.	16
SET_FLOW	0x13	Set flow control.	11
GET_FLOW	0x14	Get flow control.	11
EMBED_EVENTS	0x15	Control embedding of events in the data stream.	17
GET_BAUDRATE	0x1D	Get the baud rate.	9
SET_BAUDRATE	0x1E	Set the baud rate.	8
SET_CHARS	0x19	Set special characters.	15
VENDOR_SPECIFIC	0xFF	Vendor specific command.	18

5.1. IFC_ENABLE (0x00)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	IFC_ENABLE	Boolean	interface	0	None

This command enables or disables the specified CP210x interface. If wValue is 0x0001, the interface is enabled and serial data can be transmitted. If wValue is zero, the interface is disabled.

Host drivers should enable the interface before sending any of the other commands specified by this document.

For maximum compatibility with Windows, the CP210x will not change its baud rate or line control parameters on enable or disable. However, the CP210x may lower DTR.

5.2. RESET (0x11)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	RESET	0	interface	0	None

This command is in place for compatibility reasons. The CP210x device will ignore this command.

5.3. SET_BAUDDIV (0x01)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_BAUDDIV	baud rate divisor	interface	0	None

This command sets the baud rate for the specified interface according to wValue. The value is determined by dividing the desired baud rate into 3.6864 MHz. This command does not support the CP2102N device.

5.4. GET_BAUDDIV (0x02)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_BAUDDIV	0	interface	2	baud rate divisor

This command gets the baud rate for the specified interface. The value is determined by multiplying the baud rate divisor returned by 3.6864 MHz. This command does not support the CP2102N device.

5.5. SET_BAUDRATE (0x1E)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_BAUDRATE	0	interface	4	baud rate

This command sets the baud rate for the specified interface, according to a 4-byte value specified in the Data phase of the control request.

5.6. GET_BAUDRATE (0x1D)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_BAUDRATE	0	interface	4	baud rate

This command gets the baud rate for the CP210x interface.

5.7. SET_LINE_CTL (0x03)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_LINE_CTL	line control	interface	0	None

This command adjusts the line control settings for the selected CP210x interface, according to the value of wValue. The settings will only take effect if the selection is valid for the interface (see the specific CP210x data sheet for more details). If an invalid setting is selected, the CP210x will issue a USB procedural stall. The settings are as follows:

bits 3-0: Stop bits:

0 = 1 stop bit

1 = 1.5 stop bits

2 = 2 stop bits

other values reserved.

bits 7-4: Parity setting:

0 = none.

1 = odd.

2 = even.

3 = mark.

4 = space.

other values reserved.

bits 15-8: Word length, legal values are 5, 6, 7 and 8.

5.8. GET_LINE_CTL (0x04)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_LINE_CTL	0	interface	2	line control

This command gets the line control settings for the selected CP210x interface. The settings are as follows:

bits 3-0: Stop bits:

0 = 1 stop bit.

1 = 1.5 stop bits.

2 = 2 stop bits.

other values reserved.

bits 7-4: Parity setting:

0 = none.

1 = odd.

2 = even.

3 = mark.

4 = space.

other values reserved.

bits 15-8: Word length, legal values are 5, 6, 7 and 8.

5.9. SET_MHS (0x07)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_MHS	bit mask	interface	0	None

This command sets the modem handshaking states for the selected CP210x interface according to the value of wValue. DTR and RTS values can be set only if the current handshaking state of the interface allows direct control of the modem control lines.

bit 0: DTR state.

bit 1: RTS state.

bits 2–7: reserved.

bit 8: DTR mask, if clear, DTR will not be changed.

bit 9: RTS mask, if clear, RTS will not be changed.

bits 10–15: reserved.

5.10. GET_MDMSTS (0x08)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_MDMSTS	0	interface	1	modem status

This command returns the current states of the RS-232 modem control lines for the specified CP210x interface. The modem control line status byte is defined as follows:

bit 0: DTR state (as set by host or by handshaking logic in CP210x).

bit 1: RTS state (as set by host or by handshaking logic in CP210x).

bits 2–3: reserved.

bit 4: CTS state (as set by end device).

bit 5: DSR state (as set by end device).

bit 6: RI state (as set by end device).

bit 7: DCD state (as set by end device).

5.11. SET_FLOW (0x13)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_FLOW	0	interface	0x0010	See Table 9

This command sets the flow control state of the specified CP210x interface.

5.12. GET_FLOW (0x14)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_FLOW	0	interface	0x0010	See Table 9

This command gets the flow control state of the specified CP210x interface.

5.13. SET_XON (0x09)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_XON	0	interface	0	None

Emulate the receipt of XON (transmit flow control). If the CP210x interface was in XOFF state and if hardware handshaking permits, the interface will resume transmitting data to the external device.

If the CP210x interface is not waiting for XON, this command has no effect.

5.14. SET_XOFF (0x0A)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_XOFF	0	interface	0	None

Emulate the receipt of XOFF (transmit flow control) from the end device. The CP210x interface will stop sending characters to the end device as soon as possible and will not resume until an XON is received from the end device, or else SET_XON is received from the host. The interface may continue to accept BULK OUT packets if there is enough buffer memory available inside the device.

If the CP210x interface is already stopped, this command has no effect.

5.15. SET_EVENTMASK (0x0B)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_EVENTMASK	wait mask	interface	0	None

wValue represents the wait mask as follows:

bit 0: RI trailing edge occurred.

bit 1: unused.

bit 2: The receive buffer is 80% full.

bits 3–7: unused.

bit 8: Character received.

bit 9: Special character received.

bit 10: The transmit queue is empty.

bit 11: CTS state changed.

bit 12: DSR state changed.

bit 13: DCD state changed.

bit 14: Line break received.

bits 15: A line-status error occurred.

5.16. GET_EVENTMASK (0x0C)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_EVENTMASK	0	interface	2	wait mask

Data represents the wait mask as follows:

bit 0: RI trailing edge occurred.

bit 1: unused.

bit 2: The receive buffer is 80% full.

bits 3–7: unused.

bit 8: Character received.

bit 9: Special character received.

bit 10: The transmit queue is empty.

bit 11: CTS state changed.

bit 12: DSR state changed.

bit 13: DCD state changed.

bit 14: Line break received.

bits 15: A line-status error occurred.

5.17. GET_EVENTSTATE (0x16)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_EVENTSTATE	0	interface	2	event state

Data represents the event state as follows (this command also clears the events):

bit 0: RI trailing edge occurred.

bit 1: unused.

bit 2: The receive buffer is 80% full.

bits 3–7: unused.

bit 8: Character received.

bit 9: Special character received.

bit 10: The transmit queue is empty.

bit 11: CTS state changed.

bit 12: DSR state changed.

bit 13: DCD state changed.

bit 14: Line break received.

bits 15: A line-status error occurred.

5.18. SET_RECEIVE (0x17)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_RECEIVE	Receiver Max Timeout	interface	0x0010	None

Sets the Receiver Max Timeout setting (microseconds) for the CP2102N (CP2102N only).

5.19. GET_RECEIVE (0x18)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_RECEIVE	Receiver Max Timeout	interface	0x0010	None

Gets the Receiver Max Timeout setting (microseconds) from the CP2102N (CP2102N only).

5.20. SET_BREAK (0x05)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_BREAK	break state	interface	0	None

If wValue is 0x0001, then a BREAK is transmitted. If wValue is 0x000, then BREAK is reset. This is not necessarily synchronized with queued transmit data. This command is not supported on the second CP2105 interface. This command is supported on all four CP2108 interfaces.

5.21. GET_COMM_STATUS (0x10)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_COMM_STATUS	0	interface	0x0013	See Table 8

This command will return the serial status of the CP210x interface specified by wIndex. The returned Serial Status structure has information on the amount of data in the CP210x's transmit and receive queues and other transmission information.

5.22. IMM_CHAR (0x06)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	IMM_CHAR	char	interface	0	None

The character specified by wValue (which must be in the range 0x00-0xFF) is queued to be transmitted as soon as possible (bypassing any characters waiting in the queue).

The CP210x will return an error if the value of wValue is out of range.

Because the default pipe is a shared resource, the CP210x will not delay completion of this request. If IMM_CHAR is received and the device cannot buffer the character immediately, the CP210x will return an error indication.

5.23. SET_CHAR (0x0D)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_CHAR	special char and index value	interface	0	None

The individual special character values are set using wValue as follows:

bits 0–7: Index of the character to set:

0 = EofChar

1 = ErrorChar

2 = BreakChar

3 = EventChar

4 = XonChar

5 = XoffChar

bit 8–15: Character value to set for the specified index.

5.24. SET_CHARS (0x19)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	SET_CHARS	0	interface	0x0006	See Table 12

The special character values are set using the six-byte array passed as Data. It is an error for the host to send anything other than six bytes, and the CP210x will return an error if this occurs.

5.25. GET_CHARS (0x0E)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_CHARS	0	interface	0x0006	See Table 12

The special character values are returned using the six-byte array passed as Data.

5.26. GET_PROPS (0x0F)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	GET_PROPS	0	interface	0x0100	See Table 7

This command causes the interface to respond with a communication properties response that describes the capabilities of this interface. The details of this response are shown in Table 7.

5.27. PURGE (0x12)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	PURGE	mask; see below	interface	0	None

This command causes the CP210x to purge the selected transmit or receive queues, based on the value of the mask. The bit meanings are as follows:

bit 0: Clear the transmit queue.

bit 1: Clear the receive queue.

bit 2: Clear the transmit queue.

bit 3: Clear the receive queue.

Note: The PURGE (0x12) control command of CP2102N not only clears TX/RX buffers but also clears the configuration by SET_FLOW (0x13). Therefore, if SET_FLOW (0x13) is called before PURGE, all of the configuration will be clear.

5.28. EMBED_EVENTS (0x15)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	EMBED_EVENTS	0 or escape character	interface	0	None

This command selects or deselects event insertion mode, depending on the value of wValue. If wValue is zero, then the CP210x-to-host data stream consists only of data from the serial port. If wValue is non-zero, then events are multiplexed with data. Events are marked by a special character selected by wValue (which must be in the range 0x01-0xFF), followed by one or more bytes of qualifying code. The CP210x will check wValue and return an error indication if wValue is out of range.

Event insertion mode is disabled by USB device reset or interface disable (IFC_ENABLE with wValue == 0x0000).

The CP210x will mark events by interleaving special characters in the data stream transmitted to the PC. Events are marked as follows (<ESCCHAR> is the value selected by wValue):

<ESCCHAR> <0x00>: Indicates that the <ESCCHAR> itself appeared in the input stream.

<ESCCHAR> <0x01> <lsrval> <dataval>: Indicates that a line status register change occurred when a data byte was waiting to be received. <lsrval> is the new line status register value; <dataval> is the next data character.

<ESCCHAR> <0x02> <lsrval>: Indicates that a line status register change occurred when no data byte was waiting to be received. <lsrval> is the new line status register value.

<ESCCHAR> <0x03> <msrval>: Indicates that a modem status register change has occurred. <msrval> is the new value.

The Line Status Register value is encoded as follows:

bit 0: Data ready, Only set when the port is encoding an event with both <lsrval> and <dataval> bytes (that is, if the sequence is <ESCCHAR> <0x01> <lsrval> <dataval>).

bit 1: Hardware Overrun, A receiver hardware overrun occurred.

bit 2: Parity Error, A parity error occurred.

bit 3: Framing error, A framing error occurred.

bit 4: Break, A break was detected.

bits 5–7: reserved (always 000).

The Modem Status Register value is encoded as follows:

bit 0: Delta CTS, Set if there has been a change in CTS.

bit 1: Delta DSR, Set if there has been a change in DSR.

bit 2: Trailing edge RI, Set if the falling edge of RI has occurred.

bit 3: Delta DCD, Set if there has been a change in DCD.

bit 4: CTS, Current state of CTS.

bit 5: DSR, Current state of DSR.

bit 6: RI, Current state of RI.

bit 7: DCD, Current state of DCD.

5.29. VENDOR_SPECIFIC (0xFF)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
See below	VENDOR_SPECIFIC	See below	See below	See below	See below

This command is used for any vendor specific commands. These are as follows:

5.29.1. WRITE_LATCH (CP2102N, CP2103, CP2104)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	VENDOR_SPECIFIC	0x37E1	write latch value	0	None

The write latch value that is supplied in wIndex is represented as follows:

bits 0–7: Mask of the latch state (in bits 8-15) to write, where bit 0 is GPIO0, bit 1 is GPIO1, etc. up tp GPIOn where n is the total number of GPIO pins the interface supports.

bits 8–15: Latch state to write, where bit 8 is GPIO0, bit 9 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

5.29.2. WRITE_LATCH (CP2105)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	VENDOR_SPECIFIC	0x37E1	interface	2	write latch value

The write latch value that is supplied in the Data phase is represented as follows:

bits 0–7: Mask of the latch state (in bits 8-15) to write, where bit 0 is GPIO0, bit 1 is GPIO1, etc. up tp GPIOn where n is the total number of GPIO pins the interface supports.

bits 8–15: Latch state to write, where bit 8 is GPIO0, bit 9 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

5.29.3. WRITE_LATCH (CP2108)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000001b	VENDOR_SPECIFIC	0x37E1	0	4	write latch value

The write latch value that is supplied in the Data phase is represented as follows:

bits 0–15: Mask of the latch state (in bits 16-31) to write, where bit 0 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

bits 16–31: Latch state to write, where bit 16 is GPIO0, bit 7 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

5.29.4. READ_LATCH (CP2102N, CP2103, CP2104)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	VENDOR_SPECIFIC	0x00C2	0	1	read latch value

The read latch value that is returned is represented as follows:

bits 0–7: Current latch state, where bit 0 is GPIO0, bit 1 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

5.29.5. READ_LATCH (CP2105)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	VENDOR_SPECIFIC	0x00C2	interface	1	read latch value

The read latch value that is returned is represented as follows:

bits 0–7: Current latch state, where bit 0 is GPIO0, bit 1 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

5.29.6. READ_LATCH (CP2108)

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000001b	VENDOR_SPECIFIC	0x00C2	0	2	read latch value

The read latch value that is returned is represented as follows:

bits 0–15: Current latch state, where bit 0 is GPIO0, bit 1 is GPIO1, etc. up to GPIOn where n is the total number of GPIO pins the interface supports.

6. Control Formats

This section describes the additional data structures that are transmitted or received in conjunction with the commands discussed in "5. CP210x Control Commands" on page 7.

Table 7. Communication Properties Response

Offset	Field	Size	Value	Description
0	wLength	2	Number: 0x????	Size of structure in bytes. This must reflect the total available size, even if the host requests fewer bytes.
2	bcdVersion	2	BCD: 0x0100	Version of response, in BCD: 0x0100 is Version 1.00.
4	ulServiceMask	4	Number: 0x00000001	Service provider identifier; 1 for compatibility with NT serial.sys.
8	reserved	4	0	reserved
12	ulMaxTxQueue	4	Number	Maximum transmit queue size.
16	ulMaxRxQueue	4	Number	Maximum receive queue size.
20	ulMaBaud	4	Number	Maximum baud rate.
24	ulProvSubType	4	Code: 0 1 6	Indicates kind of device: Unspecified RS-232 Modem or TA All other values are reserved.
28	ulProvCapabilities	4	BitMask	Capabilities Mask. Bits are: 0: DTR/DSR support 1: RTS/CTS support 2: DCD support 3: Can check parity 4: XON/XOFF support 5: Can set XON/XOFF characters 6: reserved 7: reserved 8: Can set special characters 9: Supports 16-bit mode (always 0) All other bits are reserved.

Table 7. Communication Properties Response (Continued)

Offset	Field	Size	Value	Description
32	ulSettableParams	4	BitMask	Settable parameters mask. Bits are: 0: Can set parity type 1: Can set baud 2: Can set number of data bits 3: Can set stop-bits 4: Can set handshaking 5: Can set parity checking 6: Can set carrier-detect checking All other bits are reserved.
36	ulSettableBaud	4	BitMask	Settable baud rates mask. Bits are: 0: 75 baud 1: 110 baud 2: 134.5 baud 3: 150 baud 4: 300 baud 5: 600 baud 6: 1200 baud 7: 1800 baud 8: 2400 baud 9: 4800 baud 10: 7200 baud 11: 9600 baud 12: 14,400 baud 13: 19,200 baud 14: 38,400 baud 15: 56,000 baud 16: 128,000 baud 17: 115,200 baud 18: 57,600 baud 19-27: reserved 28: the CP210x supports additional baud rates other than those defined by bits 0-18. (There is no way to determine what these baud rates are, other than by trying to select them.) 29-31: reserved

Table 7. Communication Properties Response (Continued)

Offset	Field	Size	Value	Description
40	wSettableData	2	BitMask	Capabilities mask for permissible data bit settings: 0: 5 data bits 1: 6 data bits 2: 7 data bits 3: 8 data bits 4: 16 data bits 5: 16 data bits, extended 6-15: reserved
44	ulCurrentTx-Queue	4	Number	Current size of the transmit queue (allocated).
48	ulCurrentRx-Queue	4	Number	Current size of the receive queue (allocated).
52	Reserved	4	BitMask	Reserved
56	Reserved	4	BitMask	Reserved
60	uniProvName	15	“SILABS USB Vx.y”	Unicode string identifying the vendor of the device. The last three characters indicate the version.

Table 8. Serial Status Response

Offset	Field	Size	Value	Description
0	ulErrors	4	BitMask	Defines the current error status: bit 0: break bit 1: framing error bit 2: hardware overrun bit 3: queue overrun bit 4: parity error bits 5-31: reserved.
4	ulHoldReasons	4	BitMask	Reason(s) CP210x is holding: Transmit: bit 0: waiting for CTS bit 1:waiting for DSR bit 2: waiting for DCD bit 3: waiting for XON bit 4: XOFF sent, waiting bit 5: waiting on BREAK Receive: bit 6: waiting for DSR bits 7-31: reserved.
8	ulAmountInInQueue	4	Number	Number of bytes waiting in the input queue.
12	ulAmountInOutQueue	4	Number	Number of bytes waiting in hte output queue.
16	bEofReceived	1	Boolean	Always zero.
17	bWaitForImmediate	1	Boolean	0x01 if waiting for an immediate transmission to be sent.
18	bReserved	1	Zero	Reserved for future use.

Table 9. Flow Control State Setting/Response

Offset	Field	Size	Value	Description
0	ulControlHandshake	4	Code	Control handshake: see Table 10 for more information.
4	ulFlowReplace	4	Code	Control handshake: see Table 11 for more information.
8	ulXonLimit	4	Number	Threshold for sending XON. When the available space rises above this amount, XON will be sent (if in auto-receive mode).
12	ulXoffLimit	4	Number	Threshold for sending XOFF. When available space drops below this amount, XOFF will be sent (if in auto receive mode).

Table 10. Bits in ulControlHandshake

Bit	Name	Size	Value	Description
0-1	SERIAL_DTR_MASK	2	Code	This field controls the state of the DTR output for this interface. The following binary values are defined: 00: DTR is held inactive. 01: DTR is held active. 10: DTR is controlled by the CP210x device. 11: Reserved
2	Reserved	1	Flag	Reserved—must always be zero.
3	SERIAL_CTS_HANDSHAKE	1	Flag	Controls how the CP210x interprets CTS from the end device: 0: CTS is simply a status input. 1: CTS is a handshake line.
4	SERIAL_DSR_HANDSHAKE	1	Flag	Controls how the CP210x interprets DSR: 0: DSR is simply a status input. 1: DSR is a handshake line.
5	SERIAL_DCD_HANDSHAKE	1	Flag	Controls how the CP210x interprets DCD from the end device: 0: DCD is simply a status input. 1: DCD is a handshake line.
6	SERIAL_DSR_SENSITIVITY	1	Flag	Controls whether DSR controls input data reception: 0: DSR is simply a status input. 1: DSR low discards data
7-31	Reserved	25		Reserved

Table 11. Bits in ulFlowReplace

Bit	Name	Size	Value	Description
0	SERIAL_AUTO_TRANSMIT	1	Flag	Controls whether the CP210x acts on XON/XOFF characters received from the end device. 0: No XON/XOFF processing. 1: XON/XOFF start/stop output to serial port.
1	SERIAL_AUTO_RECEIVE	1	Flag	Controls whether the CP210x will try to transmit XON/XOFF in order to start/stop the reception of data from an endl device. If set, XOFF (as defined by the SET_CHARS command) will be sent by the CP210x to the end device when the CP210x's buffers are more than 80% full (or as selected by the XOFF threshold). XON will be sent when the interface's buffers drop below the XON threshold, as long as it is fine to do so.
2	SERIAL_ERROR_CHAR	1	Flag	Controls how CP210x handles characters that are received with errors: 0: The character is discarded. 1: The character is discarded, and the ERROR special-character is inserted. The ERROR character must be programmed by the host using the SET_CHARS messages (Section 5.24).
3	SERIAL_NULL_STRIPPING	1	Flag	If set, any NULL characters received by the CP210x from the end device will be discarded and will not be passed to the host. If clear, NULL characters are treated as data.
4	SERIAL_BREAK_CHAR	1	Flag	If set, a received break condition causes the CP210x to insert a BREAK special character (section 5.20) in the receive data stream. If clear, BREAK does not affect the input data stream. In either case, a received break always causes the appropriate bit of the error mask to be set.
5	Reserved	1	Reserved	Reserved
6–7	SERIAL_RTS_MASK	2	Code	This field controls the RTS line. 00: RTS is statically inactive. 01: RTS is statically active. 10: RTS is used for receive flow control. 11: RTS is transmit active signal.

Table 11. Bits in ulFlowReplace (Continued)

Bit	Name	Size	Value	Description
8–30	Reserved	23	zero	Reserved—must be written as zero.
31	SERIAL_XOFF_CONTINUE	1	Flag	If set, then the CP210x will send XON/XOFF receive flow control characters to the end device, even if the end device has sent XOFF to suspend output and has not yet sent XON to resume.

Table 12. Special Characters Response

Offset	Field	Size	Value	Description
0	bEofChar	1	Number	The character that indicates EOF (on the input).
1	bErrorChar	1	Number	The character that should be inserted in the input stream when an error occurs.
2	bBreakChar	1	Number	The character that should be inserted in the input stream when a break is detected.
3	bEventChar	1	Number	The special character that causes bit 2 of the event-occurred mask to be set whenever it is received.
4	bXonChar	1	Number	The character used for XON.
5	bXoffChar	1	Number	The character used for XOFF.

7. Revision History

Revision 0.4

July, 2024

- Updated to correct the API WRITE_LATCH, READ_LATCH command for CP2108 product in “5.29. VENDOR_SPECIFIC (0xFF)” .

Revision 0.3

August, 2022

- Updated to correct the application note reference to "AN721: CP21xx Device Customization Guide" in "1. Introduction" on page 1.
- Updated to correct Website link reference to "https://global.ihs.com" for ANSI/TIA-602 document in "1.1. Related Documents" on page 1.
- Added more information to SET_BAUDDIV() and GET_BAUDDIV() commands: "both commands are not supports for the CP2102N device"
- Added more information to GET_EVENTSTATE() command: "this command also clears the events"
- Added more information to PURGE() command: "the PURGE (0x12) control command of CP2102N not only clears TX/RX buffers but also clears the configuration by SET_FLOW (0x13), therefore, if SET_FLOW (0x13) is called before PURGE, all the configuration will be clear."
- Added “7. Revision History” .

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com