



---

# USBXPRESS™ FOR WIN CE PROGRAMMER'S GUIDE

---

## Relevant Devices

This application note applies to the following devices:  
CP2101, CP2102

---

## 1. Introduction

The Silicon Laboratories USBXpress™ Development Kit provides a complete host and device software solution for interfacing Silicon Laboratories CP210x devices to the Universal Serial Bus (USB). No USB protocol or host device driver expertise is required. Instead, a simple, high-level Application Program Interface (API) for both the host software and device firmware is used to provide complete USB connectivity.

The USBXpress Development Kit includes a Windows device driver, and a host interface function library (host API) provided in the form of a Windows Dynamic Link Library (DLL). The included device drivers and installation files support MS Windows CE .NET 5.0.

## 2. Host API Functions

The host API is provided in the form of a Windows Dynamic Link Library (DLL). The host interface DLL communicates with the USB device via the provided device driver and the operating system's USB stack. The following is a list of the host API functions available:

<code>SI_GetNumDevices()</code>	- Returns the number of devices connected
<code>SI_Open()</code>	- Opens a device and returns a handle
<code>SI_Close()</code>	- Cancels pending IO and closes a device
<code>SI_Read()</code>	- Reads a block of data from a device
<code>SI_Write()</code>	- Writes a block of data to a device
<code>SI_FlushBuffers()</code>	- Flushes the TX and RX buffers for a device
<code>SI_SetTimeouts()</code>	- Sets read and write block timeouts
<code>SI_GetTimeouts()</code>	- Gets read and write block timeouts
<code>SI_CheckRXQueue()</code>	- Returns the number of bytes in a device's RX queue
<code>SI_SetBaudRate()</code>	- Sets the specified CP210x Baud Rate
<code>SI_SetBaudDivisor()</code>	- Sets the specified CP210x Baud Divisor Value
<code>SI_SetLineControl()</code>	- Sets the CP210x device Line Control
<code>SI_SetFlowControl()</code>	- Sets the CP210x device Flow Control
<code>SI_GetModemStatus()</code>	- Gets the CP210x device Modem Status
<code>SI_DeviceIOControl()</code>	- Interface for miscellaneous device control functions

In general, the user initiates communication with the target USB device(s) by making a call to *SI\_GetNumDevices*. This call will return the number of target devices.

To access a device, it must first be opened by a call to *SI\_Open* using an index determined from the call to *SI\_GetNumDevices*. The *SI\_Open* function will return a handle to the device that is used in all subsequent accesses. Data I/O is performed using the *SI\_Write* and *SI\_Read* functions. When I/O operations are complete, the device is closed by a call to *SI\_Close*.

Additional functions are provided to flush the transmit and receive buffers (*SI\_FlushBuffers*), set receive and transmit timeouts (*SI\_SetTimeouts*), check the receive buffer's status (*SI\_CheckRXQueue*) and miscellaneous device control (*SI\_DeviceIOControl*).

Functions are also available to set the baud rate (*SI\_SetBaudRate*); set the baud divisor (*SI\_SetBaudDivisor*); adjust the line control settings such as word length, stop bits and parity (*SI\_SetLineControl*); set hardware handshaking, software handshaking and modem control signals (*SI\_SetFlowControl*); and get modem status (*SI\_GetModemStatus*).

Each of these functions are described in detail in the following sections. Type definitions and constants are defined in Appendix A.

## 2.1. SI\_GetNumDevices

**Description:** This function returns the number of devices connected to the host.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_GetNumDevices (LPDWORD NumDevices)`

**Parameters:** 1. NumDevices—Address of a DWORD variable that will contain the number of devices connected on return.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_DEVICE\_NOT\_FOUND or  
SI\_INVALID\_PARAMETER

## 2.2. SI\_Open

**Description:** Opens a device (using device number as returned by *SI\_GetNumDevices*) and returns a handle which will be used for subsequent accesses.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_Open (DWORD DeviceNum, HANDLE *Handle)`

**Parameters:** 1. DeviceNum—Device index. 0 for first device, 1 for 2nd, etc.  
2. Handle—Pointer to a variable where the handle to the device will be stored. This handle will be used by all subsequent accesses to the device.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_DEVICE\_NOT\_FOUND or  
SI\_INVALID\_HANDLE or  
SI\_INVALID\_PARAMETER

## 2.3. SI\_Close

**Description:** Closes an open device using the handle provided by *SI\_Open*.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_Close (HANDLE Handle)`

**Parameters:** 1. Handle—Handle to the device to close as returned by *SI\_Open*.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_INVALID\_HANDLE

## 2.4. SI\_Read

**Description:** Reads the specified number of bytes into the specified buffer and retrieves the number of bytes read. Given valid input parameters, this function is blocking until the specified number of bytes become available or a timeout occurs (see section 2.7. (*SI\_SetTimeouts*)).

Note: If timeouts are set to zero this function is not blocking and will immediately return the current number of bytes available in the device driver buffer. This may be less than the requested number of bytes (zero bytes if there is no data available) so make sure to check the read size return values in this scenario.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_Read (HANDLE Handle, LPVOID Buffer, DWORD NumBytesToRead, DWORD *NumBytesReturned)`

**Parameters:**

1. Handle—Handle to the device to read as returned by *SI\_Open*.
2. Buffer—Address of a character buffer to be filled with read data.
3. NumBytesToRead—Number of bytes to read from the device into the buffer (0 - 16KBytes).
4. NumBytesReturned—Address of a DWORD which will contain the number of bytes actually read into the buffer on return.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_READ\_ERROR or  
SI\_INVALID\_REQUEST\_LENGTH or  
SI\_INVALID\_PARAMETER or  
SI\_RX\_QUEUE\_NOT\_READY or  
SI\_INVALID\_HANDLE

## 2.5. SI\_Write

**Description:** Writes the specified number of bytes from the specified buffer to the device. Given valid parameters, this function is blocking until the write is successful, fails, or a timeout occurs. The write is successful when the device has accepted all of the data. If the write fails or a timeout occurs, SI\_WRITE\_ERROR is returned.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_Write (HANDLE Handle, LPVOID Buffer, DWORD NumBytesToWrite, DWORD *NumBytesWritten)`

**Parameters:**

1. Handle—Handle to the device to write as returned by *SI\_Open*.
2. Buffer—Address of a character buffer of data to be sent to the device.
3. NumBytesToWrite—Number of bytes to write to the device (0 - 8Kbytes).
4. NumBytesWritten—Address of a DWORD which will contain the number of bytes actually written to the device.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_WRITE\_ERROR or  
SI\_INVALID\_REQUEST\_LENGTH or  
SI\_INVALID\_PARAMETER or  
SI\_INVALID\_HANDLE

## 2.6. SI\_FlushBuffers

**Description:** Flushes the receive buffer in the USB stack and the transmit buffer in the device.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_FlushBuffers (HANDLE Handle, BYTE FlushTransmit, BYTE FlushReceive)`

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. FlushTransmit—Set to a non-zero value to flush the transmit buffer.
3. FlushReceive—Set to a non-zero value to flush the receive buffer.

**Return Value:** `SI_STATUS = SI_SUCCESS` or  
`SI_DEVICE_IO_FAILED` or  
`SI_INVALID_HANDLE`

## 2.7. SI\_SetTimeouts

**Description:** Sets the read and write timeouts.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_SetTimeouts (DWORD ReadTimeout, DWORD WriteTimeout)`

**Parameters:**

1. ReadTimeout—*SI\_Read* operation timeout (in milliseconds).
2. WriteTimeout—*SI\_Write* operation timeout (in milliseconds).

**Return Value:** `SI_STATUS = SI_SUCCESS`

## 2.8. SI\_GetTimeouts

**Description:** Returns the current read and write timeouts.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_GetTimeouts (LPDWORD ReadTimeout, LPDWORD WriteTimeout)`

**Parameters:**

1. ReadTimeout—*SI\_Read* operation timeout (in milliseconds).
2. WriteTimeout—*SI\_Write* operation timeout (in milliseconds).

**Return Value:** `SI_STATUS = SI_SUCCESS` or  
`SI_INVALID_PARAMETER`

## 2.9. SI\_CheckRXQueue

**Description:** Returns the number of bytes in the receive queue and a status value that indicates if an overrun (SI\_QUEUE\_OVERRUN) has occurred and if the RX queue is ready (SI\_QUEUE\_READY) for reading.

**Supported Devices:** CP2101/2

**Prototype:** SI\_STATUS SI\_CheckRXQueue (HANDLE Handle, LPDWORD NumBytesInQueue, LPDWORD QueueStatus)

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. NumBytesInQueue—Address of a DWORD variable that contains the number of bytes currently in the receive queue on return.
3. QueueStatus—Address of a DWORD variable that contains the SI\_RX\_COMPLETE flag or the SI\_RX\_OVERRUN flag if an RX overrun occurred.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_DEVICE\_IO\_FAILED or  
SI\_INVALID\_HANDLE or  
SI\_INVALID\_PARAMETER

## 2.10. SI\_SetBaudRate

**Description:** Sets the Baud Rate. Refer to the device data sheet for a list of Baud Rates supported by the device.

**Supported Devices:** CP2101/2

**Prototype:** SI\_STATUS SI\_SetBaudRate (HANDLE Handle, DWORD dwBaudRate)

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. dwBaudRate—A DWORD value specifying the Baud Rate to set.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_INVALID\_BAUDRATE or  
SI\_INVALID\_HANDLE

## 2.11. SI\_SetBaudDivisor

**Description:** Sets the Baud Rate directly by using a specific divisor value. This function is obsolete; use *SI\_SetBaudRate* instead.

**Supported Devices:** CP2101/2

**Prototype:** SI\_STATUS SI\_GetBaudDivisor (HANDLE Handle, WORD wBaudDivisor)

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. wBaudDivisor—A WORD value specifying the Baud Divisor to set.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_INVALID\_BAUDRATE or  
SI\_INVALID\_HANDLE

## 2.12. SI\_SetLineControl

**Description:** Adjusts the line control settings: word length, stop bits and parity. Refer to the device data sheet for valid line control settings.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_SetLineControl (HANDLE Handle, WORD wLineControl)`

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. wLineControl—A WORD variable that contains the desired line control settings. Possible input settings are:

**bits 0-3      Number of Stop bits**

0 :	1 stop bit;
1 :	1.5 stop bits;
2 :	2 stop bits

**bits 4-7      Parity**

0 :	None
1 :	Odd
2 :	Even
3 :	Mark
4 :	Space

**bits 8-15      Number of bits per word**  
5, 6, 7, or 8

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_DEVICE\_IO\_FAILED or  
SI\_INVALID\_HANDLE or  
SI\_INVALID\_PARAMETER

## 2.13. SI\_SetFlowControl

**Description:** Adjusts the following flow control settings: set hardware handshaking, software handshaking and modem control signals. See Appendix A for pin characteristic definitions.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_SetFlowControl (HANDLE Handle, BYTE bCTS_MaskCode, BYTE bRTS_MaskCode, BYTE bDTR_MaskCode, BYTE bDSRMaskCode, BYTE bDCD_MaskCode, BYTE bFlowXonXoff)`

- Parameters:**
1. Handle—Handle to the device as returned by *SI\_Open*.
  2. bCTS\_MaskCode—The CTS pin characteristic must be:  
SI\_STATUS\_INPUT or  
SI\_HANDSHAKE\_LINE.
  3. bRTS\_MaskCod—The RTS pin characteristic must be:  
SI\_HELD\_ACTIVE,  
SI\_HELD\_INACTIVE,  
SI\_FIRMWARE\_CONTROLLED or  
SI\_TRANSMIT\_ACTIVE\_SIGNAL.
  4. bDTR\_MaskCode—The DTR pin characteristic must be:  
SI\_HELD\_INACTIVE,  
SI\_HELD\_ACTIVE or  
SI\_FIRMWARE\_CONTROLLED.
  5. bDSR\_MaskCode—The DSR pin characteristic must be:  
SI\_STATUS\_INPUT or  
SI\_HANDSHAKE\_LINE.
  6. bDCD\_MaskCode—The DCD pin charactericstic must be:  
SI\_STATUS\_INPUT or  
SI\_HANDSHAKE\_LINE.

**Return Value:** SI\_STATUS = SI\_SUCCESS or  
SI\_DEVICE\_IO\_FAILED or  
SI\_INVALID\_HANDLE or  
SI\_INVALID\_PARAMETER

## 2.14. SI\_GetModem Status

**Description:** Gets the Modem Status from the device. This includes the modem pin states.

**Supported Devices:** CP2101/2

**Prototype:** `SI_STATUS SI_GetModemStatus (HANDLE Handle, PBYTE ModemStatus)`

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. lpbModemStatus—Address of a BYTE variable that contains the current states of the RS-232 modem control lines. The byte is defined as follows:

bit 0	DTR State
bit 1	RTS State
bit 4	CTS State
bit 5	DSR State
bit 6	RI State
bit 7	DCD State

**Return Value:** `SI_STATUS` = `SI_SUCCESS` or  
`SI_DEVICE_IO_FAILED` or  
`SI_INVALID_HANDLE` or  
`SI_INVALID_PARAMETER`

## 2.15. SI\_DeviceIOControl

**Description:** Interface for any miscellaneous device control functions. A separate call to *SI\_DeviceIOControl* is required for each input or output operation. A single call cannot be used to perform both an input and output operation simultaneously.

**Supported Devices:**

**Prototype:** `SI_STATUS SI_DeviceIOControl (HANDLE Handle, DWORD IOControlCode, LPVOID InBuffer, DWORD BytesToRead, LPVOID OutBuffer, DWORD BytesToWrite, LPDWORD BytesSucceeded)`

**Parameters:**

1. Handle—Handle to the device as returned by *SI\_Open*.
2. IOControlCode—Code to select control function.
3. InBuffer—Pointer to input data buffer.
4. BytesToRead—Number of bytes to be read into InBuffer.
5. OutBuffer—Pointer to output data buffer.
6. BytesToWrite—Number of bytes to write from OutBuffer.
7. BytesSucceeded—Address of a DWORD variable that will contain the number of bytes read by a input operation or the number of bytes written by a output operation on return.

**Return Value:** `SI_STATUS` = `SI_SUCCESS` or  
`SI_DEVICE_IO_FAILED` or  
`SI_INVALID_HANDLE`



## APPENDIX A—TYPE DEFINITIONS (FROM C++ HEADER FILE SiUSB.H)

```
// GetProductString function flags
#define SI_RETURN_SERIAL_NUMBER      0x00
#define SI_RETURN_DESCRIPTION        0x01

// Return codes
#define SI_SUCCESS                    0x00
#define SI_DEVICE_NOT_FOUND          0xFF
#define SI_INVALID_HANDLE            0x01
#define SI_READ_ERROR                0x02
#define SI_RX_QUEUE_NOT_READY        0x03
#define SI_WRITE_ERROR               0x04
#define SI_INVALID_PARAMETER          0x06
#define SI_INVALID_REQUEST_LENGTH    0x07
#define SI_DEVICE_IO_FAILED          0x08
#define SI_INVALID_BAUDRATE          0x09

// RX Queue status flags
#define SI_RX_NO_OVERRUN             0x00
#define SI_RX_OVERRUN               0x01
#define SI_RX_READY                  0x02

// Buffer size limits
#define SI_MAX_DEVICE_STRLEN         256
#define SI_MAX_READ_SIZE             4096*16
#define SI_MAX_WRITE_SIZE            4096

// Type definitions
typedef int    SI_STATUS;
typedef char   SI_DEVICE_STRING[SI_MAX_DEVICE_STRLEN];

// Input and Output pin Characteristics
#define SI_HELD_INACTIVE             0x00
#define SI_HELD_ACTIVE               0x01
#define SI_FIRMWARE_CONTROLLED       0x02
#define SI_RECEIVE_FLOW_CONTROL      0x02
#define SI_TRANSMIT_ACTIVE_SIGNAL    0x03

#define SI_STATUS_INPUT              0x00
#define SI_HANDSHAKE_LINE            0x01

//Common variable type definitions used
typedef unsigned long    DWORD;
typedef int              BOOL;
typedef unsigned char    BYTE;
typedef unsigned short   WORD;
typedef BYTE near        *PBYTE;
typedef DWORD near       *PDWORD;
typedef DWORD far        *LPDWORD;
typedef void far         *LPVOID;
```

## CONTACT INFORMATION

Silicon Laboratories Inc.  
4635 Boston Lane  
Austin, TX 78735  
Tel: 1+(512) 416-8500  
Fax: 1+(512) 416-9669  
Toll Free: 1+(877) 444-3032  
Email: [MCUinfo@silabs.com](mailto:MCUinfo@silabs.com)  
Internet: [www.silabs.com](http://www.silabs.com)

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories, Silicon Labs, and USBXpress are trademarks of Silicon Laboratories Inc.

Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.