USB Serial Library

API Guide

Revision 1.0

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone (USA): 800.858.1810
Phone (Intnl): 408.943.2600
http://www.cypress.com

The information in this document is subject to change without notice and should not be construed as a commitment by Cypress. While reasonable precautions have been taken, Cypress assumes no responsibility for any errors that may appear in this document. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Cypress. Made in the U.S.A.

Disclaimer

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

License Agreement

Please read the license agreement during installation.

Table of Contents

1 Overview

This section provides an overview of the contents and functionality of the USB Serial API Library.

1.1 API Functionality

The USB Serial API Library provides functions to configure the Cypress USB Serial device and to perform read/write operations to various device blocks.

The Cypress USB Serial device is a USB Full Speed to Serial Peripheral bridge that supports:

- Two configurable UART / SPI / I2C interfaces.
- General Purpose IO pins
- JTAG controller
- Cap-sense buttons with LED and encoded GPIO indication.

The USB Serial API library provides a set of functions to configure the interfaces of the device, and to perform data transfers from/to the peripherals connected on the UART, SPI and I2C interfaces of the device.

The API library is supported on Windows, Linux and Mac-OS systems. The Windows version makes use of the CyUsb3 general purpose USB driver. The Linux and Mac-OS versions make use of the open source LibUsb library.

2 Constants

This section contains details of the all the constants that are part of Cypress USB Serial driver library.

2.1 CY_US_VERSION

Description

Version number for the device.

2.2 CY_US_VERSION_MAJOR

Description

Major version number for library.

2.3 CY_US_VERSION_MINOR

Description

Minor version number for library.

2.4 CY_US_VERSION_PATCH

Description

Patch version number for library.

2.5 CY_US_VERSION_BUILD

Description

Library build number.

2.6 CY_MAX_DEVICE_INTERFACE

Description

Maximum number of interfaces

2.7 CY_STRING_DESCRIPTOR_SIZE

Description

String descriptor size

3 Data Types

This section defines the data types that are used by Cypress USB Serial driver library.

3.1 CY_HANDLE

CyUSB Device Handle.

Description

The handle is used by application to communicate with USB serial device. The handle is obtained by calling CyOpen.

See Also

- CyOpen

3.2 CY_RETURN_STATUS

Enumeration defining return status of USB serial library APIs

Description

The enumeration CY_RETURN_STATUS holds the different return status of all the APIs supported by USB Serial library.

Members

Members	Description
`CY_SUCCESS = 0`	API returned successfully without any errors.
`CY_ERROR_ACCESS_DENIED`	Access of the API is denied for the application
`CY_ERROR_DRIVER_INIT_FAILED`	Driver initialisation failed
`CY_ERROR_DEVICE_INFO_FETCH_FAILED`	Device information fetch failed
`CY_ERROR_DRIVER_OPEN_FAILED`	Failed to open a device in the library
`CY_ERROR_INVALID_PARAMETER`	One or more parameters sent to the API was invalid
`CY_ERROR_REQUEST_FAILED`	Request sent to USB Serial device failed
`CY_ERROR_DOWNLOAD_FAILED`	Firmware download to the device failed
`CY_ERROR_FIRMWARE_INVALID_SIGNATURE`	Invalid Firmware signature in firmware file
`CY_ERROR_INVALID_FIRMWARE`	Invalid firmware
`CY_ERROR_DEVICE_NOT_FOUND`	Device disconnected
`CY_ERROR_IO_TIMEOUT`	Timed out while processing a user request
`CY_ERROR_PIPE_HALTED`	Pipe halted while trying to transfer data
`CY_ERROR_BUFFER_OVERFLOW`	OverFlow of buffer while trying to read/write data
`CY_ERROR_INVALID_HANDLE`	Device handle is invalid
`CY_ERROR_ALLOCATION_FAILED`	Error in Allocation of the resource inside the library

CY_ERROR_I2C_DEVICE_BUSY	I2C device busy
CY_ERROR_I2C_NAK_ERROR	I2C device NAK
CY_ERROR_I2C_ARBITRATION_ERROR	I2C bus arbitration error
CY_ERROR_I2C_BUS_ERROR	I2C bus error
CY_ERROR_I2C_BUS_BUSY	I2C bus is busy
CY_ERROR_I2C_STOP_BIT_SET	I2C master has sent a stop bit during a transaction
CY_ERROR_STATUS_MONITOR_EXIST	API Failed because the SPI/UART status monitor thread already exists

3.3 CY_DEVICE_CLASS

Enumeration defining list of USB device classes supported by USB Serial device.

Description

This is the list of USB device classes supported by USB Serial device.

Members

Members	Description
CY_CLASS_DISABLED = 0	None or the interface is disabled
CY_CLASS_CDC = 0x02	CDC ACM class
CY_CLASS_PHDC = 0x0F	PHDC class
CY_CLASS_VENDOR = 0xFF	VENDOR specific class

See Also

- CY_DEVICE_INFO
- CyGetDeviceInfo
- CyGetDeviceInfoVidPid

3.4 CY_DEVICE_INFO

This structure is used to hold information of the device connected to host.

Description

The structure holds the information about device currently connected to host. The information can be obtained by using CyGetDeviceInfo and CyGetDeviceInfoVidPid APIs.

The information includes VID, PID, number of interfaces, string descriptors, device type and device class supported by each interface. Device type is valid only if the interface is CY_CLASS_VENDOR.

Members

Members	Description
`CY_VID_PID vidPid;`	VID and PID
`UCHAR numInterfaces;`	Number of interfaces supported
`UCHAR manufacturerName[CY_STRING_DESCRIPTOR_SIZE];`	Manufacturer name
`UCHAR productName[CY_STRING_DESCRIPTOR_SIZE];`	Product name
`UCHAR serialNum[CY_STRING_DESCRIPTOR_SIZE];`	Serial number
`UCHAR deviceFriendlyName[CY_STRING_DESCRIPTOR_SIZE];`	Device friendly name : Windows only
`CY_DEVICE_TYPE deviceType[CY_MAX_DEVICE_INTERFACE];`	Type of the device each interface has (Valid only for USB Serial Device) and interface in vendor class
`CY_DEVICE_CLASS deviceClass[CY_MAX_DEVICE_INTERFACE];`	Interface class of each interface
`CY_DEVICE_SERIAL_BLOCK deviceBlock;`	On Windows, each USB Serial device interface is associated with a separate driver instance. This variable represents the present driver interface instance that is associated with a serial block.

See Also

- CY_VID_PID
- CY_DEVICE_CLASS
- CY_DEVICE_TYPE
- CyGetDeviceInfo
- CyGetDeviceInfoVidPid

3.5 PCY_DEVICE_INFO

Pointer to CY_DEVICE_INFO structure.

Description

This type represents a pointer to a CY_DEVICE_INFO structure.

See Also

- CY_DEVICE_INFO

3.6 CY_DEVICE_SERIAL_BLOCK

This enumeration type defines the available device serial blocks.

Description

USB Serial device has up to two configurable serial blocks. UART, SPI, I2C or JTAG functionality can be configured and used in these serial block. Windows driver binds to a serial block rather than the entire device. So, it is essential to find out which serial block to which current communications are directed. These enumeration structure provides the possible SERIAL BLOCK Options.

This enumerated data type is a member of CY_DEVICE_INFO structure. This data type information doesn't apply for non-windows operating system.

Members

Members	Description
`SerialBlock_SCB0 = 0`	Serial Block Number 0
`SerialBlock_SCB1`	Serial Block Number 1
`SerialBlock_MFG`	Serial Block Manufacturing Interface.

See Also

- CY_DEVICE_INFO
- CyGetDeviceInfo
- CyGetDeviceInfoVidPid

3.7 CY_DEVICE_TYPE

Enumeration defining list of device types supported by USB Serial device in each interface.

Description

This is the list of device types supported by USB Serial device when the interface type is configured as CY_CLASS_VENDOR. The interface type can be queried from the device by using CyGetDeviceInfo and CyGetDeviceInfoVidPid APIs.

The member of CY_DEVICE_INFO structure contains the interface type.

Members

Members	Description
`CY_TYPE_DISABLED = 0`	Invalid device type or interface is not CY_CLASS_VENDOR
`CY_TYPE_UART`	Interface of device is of type UART
`CY_TYPE_SPI`	Interface of device is of type SPI
`CY_TYPE_I2C`	Interface of device is of type I2C
`CY_TYPE_JTAG`	Interface of device is of type JTAG
`CY_TYPE_MFG`	Interface of device is in Manufacturing mode

See Also

- CY_DEVICE_INFO

- CyGetDeviceInfo
- CyGetDeviceInfoVidPid

3.8 CY_VID_PID

This structure is used to hold VID and PID of USB device

Description

This Structure holds the VID and PID of a USB device.

Members

Members	Description
`UINT16 vid;`	Holds the VID of the device
`UINT16 pid;`	Holds the PID of the device

See Also

- CY_DEVICE_INFO
- CyGetDeviceInfoVidPid

3.9 PCY_VID_PID

Pointer to CY_VID_PID structure.

Description

This type represents a pointer to a CY_VID_PID structure.

See Also

- CY_VID_PID

3.10 CY_DATA_BUFFER

This structure is used to hold data buffer information.

Description

This structure is used by all the data transaction APIs in the library to perform read, write operations. Before using a variable of this structre users need to initialize various members appropriately.

Members

Members	Description
`UCHAR * buffer;`	Pointer to the buffer from where the data is read/written
`UINT32 length;`	Length of the buffer
`UINT32 transferCount;`	Number of bytes actually read/written

See Also

- CyUartRead
- CyUartWrite
- CyI2cRead
- CyI2cWrite
- CySpiReadWrite
- CyJtagWrite
- CyJtagRead

3.11 PCY_DATA_BUFFER

Pointer to CY_DATA_BUFFER structure.

Description

This type represents a pointer to a CY_DATA_BUFFER structure.

See Also

- CY_DATA_BUFFER

3.12 CY_LIBRARY_VERSION

This structure is used to hold version information of the library.

Description

This structure can be used to retrive the version information of the library.

Members

Members	Description
`UINT8 majorVersion;`	The major version of the library
`UINT8 minorVersion;`	The minor version of the library
`UINT16 patch;`	The patch number of the library
`UINT8 buildNumber;`	The build number of the library

See Also

- CyGetLibraryVersion

3.13 PCY_LIBRARY_VERSION

Pointer to CY_LIBRARY_VERSION structure.

Description

This type represents a pointer to a CY_LIBRARY_VERSION structure.

See Also

- CY_LIBRARY_VERSION

3.14 CY_FIRMWARE_VERSION

This structure is used to hold firmware version of the USB Serial device.

Description

This structure holds the version information of the USB serial device. It has major version, minor version, patch number and build number.

Members

Members	Description
`UINT8 majorVersion;`	Major version of the Firmware
`UINT8 minorVersion;`	Minor version of the Firmware
`UINT16 patchNumber;`	Patch Number of the Firmware
`UINT32 buildNumber;`	Build Number of the Firmware

See Also

- CyGetFirmwareVersion

3.15 PCY_FIRMWARE_VERSION

Pointer to CY_FIRMWARE_VERSION structure.

Description

This type represents a pointer to a CY_FIRMWARE_VERSION structure.

See Also

- CY_FIRMWARE_VERSION

3.16 CY_CALLBACK_EVENTS

Enumeration defining UART/SPI transfer error or status bit maps.

Description

Enumeration lists the bit maps that are used to report error or status during UART/SPI transfer.

Members

Members	Description
`CY_UART_CTS_BIT = 0x01`	CTS pin notification bit
`CY_UART_DSR_BIT = 0x02`	State of transmission carrier. This signal corresponds to V.24 signal 106 and RS-232 signal DSR.
`CY_UART_BREAK_BIT = 0x04`	State of break detection mechanism of the device
`CY_UART_RING_SIGNAL_BIT = 0x08`	State of ring signal detection of the device
`CY_UART_FRAME_ERROR_BIT = 0x10`	A framing error has occurred
`CY_UART_PARITY_ERROR_BIT = 0x20`	A parity error has occured
`CY_UART_DATA_OVERRUN_BIT = 0x40`	Received data has been discarded due to overrun in the device
`CY_UART_DCD_BIT = 0x100`	State of receiver carrier detection mechanism of device. This signal corresponds to V.24 signal 109 and RS-232 signal DCD
`CY_SPI_TX_UNDERFLOW_BIT = 0x200`	Notification sent when SPI fifo is empty
`CY_SPI_BUS_ERROR_BIT = 0x400`	Spi bus error has been detected
`CY_ERROR_EVENT_FAILED_BIT = 0x800`	Event thread failed

See Also

- CySetEventNotification

3.17 CY_EVENT_NOTIFICATION_CB_FN

Function pointer for getting async error/success notification on UART/SPI

Description

This function pointer that will be passed to CySetEventNotification and get a callback with a 2 byte value bit map that reports error/events triggered during UART/SPI transaction. The bit map is defined in

CY_CALLBACK_EVENTS.

See Also

- CY_CALLBACK_EVENTS

3.18 CY_UART_CONFIG

Structure holds configuration of UART module of USB Serial device.

Description

This structure defines parameters used for configuring the UART module. CySetUartConfig and CyGetUartConfig APIs are used to configure and retrieve the UART configuration information.

Members

Members	Description
`CY_UART_BAUD_RATE baudRate;`	Baud rate as defined in CY_UART_BAUD_RATE
`UINT8 dataWidth;`	Data width: valid values 7 or 8
`CY_UART_STOP_BIT stopBits;`	Number of stop bits to be used 1 or 2
`CY_UART_PARITY_MODE parityMode;`	UART parity mode as defined in CY_UART_PARITY_MODE
`BOOL isDropOnRxErrors;`	Whether to ignore framing as well as parity errors and receive data

See Also

- CySetUartConfig
- CyGetUartConfig

3.19 PCY_UART_CONFIG

Pointer to CY_UART_CONFIG structure.

Description

This type represents a pointer to a CY_UART_CONFIG structure.

See Also

- CY_UART_CONFIG

3.20 CY_UART_BAUD_RATE

Enumeration defines UART baud rates supported by UART module of USB Serial device.

Description

The enumeration lists the various baud rates supported by the UART when it is in UART vendor mode.

Members

Members	Description
`CY_UART_BAUD_300 = 300`	Baud rate of 300.
`CY_UART_BAUD_600 = 600`	Baud rate of 600.
`CY_UART_BAUD_1200 = 1200`	Baud rate of 1200.
`CY_UART_BAUD_2400 = 2400`	Baud rate of 2400.
`CY_UART_BAUD_4800 = 4800`	Baud rate of 4800.
`CY_UART_BAUD_9600 = 9600`	Baud rate of 9600.
`CY_UART_BAUD_14400 = 14400`	Baud rate of 14400.
`CY_UART_BAUD_19200 = 19200`	Baud rate of 19200.
`CY_UART_BAUD_38400 = 38400`	Baud rate of 38400.
`CY_UART_BAUD_56000 = 56000`	Baud rate of 56000.
`CY_UART_BAUD_57600 = 57600`	Baud rate of 57600.
`CY_UART_BAUD_115200 = 115200`	Baud rate of 115200.
`CY_UART_BAUD_230400 = 230400`	Baud rate of 230400.
`CY_UART_BAUD_460800 = 460800`	Baud rate of 460800.
`CY_UART_BAUD_921600 = 921600`	Baud rate of 921600.
`CY_UART_BAUD_1000000 = 1000000`	Baud rate of 1000000.
`CY_UART_BAUD_3000000 = 3000000`	Baud rate of 3000000.

See Also

- CY_UART_CONFIG
- CySetUartConfig
- CyGetUartConfig

3.21 CY_UART_PARITY_MODE

Enumeration defines the different parity modes supported by UART module of USB Serial device.

Description

This enumeration defines the different parity modes of USB Serial UART module. It supports odd, even, mark and

space parity modes.

Members

Members	Description
`CY_DATA_PARITY_DISABLE = 0`	Data parity disabled
`CY_DATA_PARITY_ODD`	Odd Parity
`CY_DATA_PARITY_EVEN`	Even Parity
`CY_DATA_PARITY_MARK`	Mark parity
`CY_DATA_PARITY_SPACE`	Space parity

See Also

- CY_UART_CONFIG
- CySetUartConfig
- CyGetUartConfig

3.22 CY_UART_STOP_BIT

Enumeration defines the different stop bit values supported by UART module of USB Serial device.

Members

Members	Description
`CY_UART_ONE_STOP_BIT = 1`	One stop bit
`CY_UART_TWO_STOP_BIT`	Two stop bits

See Also

- CY_UART_CONFIG
- CySetUartConfig
- CyGetUartConfig

3.23 CY_FLOW_CONTROL_MODES

Enumeration defines flow control modes supported by UART module of USB Serial device.

Description

The list provides the various flow control modes supported by USB Serial device.

Members

Members	Description
`CY_UART_FLOW_CONTROL_DISABLE = 0`	Disable Flow control

CY_UART_FLOW_CONTROL_DSR	Enable DSR mode of flow control
CY_UART_FLOW_CONTROL_RTS_CTS	Enable RTS CTS mode of flow control
CY_UART_FLOW_CONTROL_ALL	Enable RTS CTS and DSR flow control

See Also

- CyUartSetHwFlowControl
- CyUartGetHwFlowControl

3.24 CY_I2C_CONFIG

This structure is used to store configuration of I2C module.

Description

The structure contains parameters that are used in configuring I2C module of Cypress USB Serial device. CyGetI2cConfig and CySetI2cConfig APIs can be used to retrieve and configure I2C module respectively.

Members

Members	Description
UINT32 frequency;	I2C clock frequency 1KHz to 400KHz
UINT8 slaveAddress;	Slave address of the I2C module, when it is configured as slave
BOOL isMaster;	Set to True if USB Serial is I2C Master, False if USB Serial is I2C slave.
BOOL isClockStretch;	true: Stretch clock in case of no data availability (valid only for slave mode). false: Do not use clock stretching.

See Also

- CyGetI2cConfig
- CySetI2cConfig

3.25 PCY_I2C_CONFIG

Pointer to CY_I2C_CONFIG structure.

Description

This type represents a pointer to a CY_I2C_CONFIG structure.

See Also

- CY_I2C_CONFIG

3.26 CY_I2C_DATA_CONFIG

This structure is used to configure each I2C data transaction.

Description

This structure defines parameters that are used for configuring I2C module during each data transaction. Which includes setting slave address (when device is in I2C slave mode), stopbit (to enable or disable) and Nak bit (to enable or disable).

Members

Members	Description
`UCHAR slaveAddress;`	Slave address the master will communicate with
`BOOL isStopBit;`	Set when stop bit is used
`BOOL isNakBit;`	Set when I2C master wants to NAK the slave after read. Applicable only when doing I2C read

See Also

- CyI2cWrite
- CyI2cRead

3.27 PCY_I2C_DATA_CONFIG

Pointer to CY_I2C_DATA_CONFIG structure.

Description

This type represents a pointer to a CY_I2C_DATA_CONFIG structure.

See Also

- CY_I2C_DATA_CONFIG

3.28 CY_SPI_CONFIG

This structure is used to configure the SPI module of USB Serial device.

Description

This structure defines configuration parameters that are used for configuring the SPI module .

Members

Members	Description
`UINT32 frequency;`	SPI clock frequency. IMPORTANT: The frequency range supported by SPI module is 1000(1KHz) to 3000000(3MHz)
`UCHAR dataWidth;`	Data width in bits. The valid values are from 4 to 16.
`CY_SPI_PROTOCOL protocol;`	SPI protocol to be used as defined in CY_SPI_PROTOCOL
`BOOL isMsbFirst;`	false -> least significant bit is sent out first. true -> most significant bit is sent out first
`BOOL isMaster;`	false --> Slave mode selected true --> Master mode selected
`BOOL isContinuousMode;`	true - Slave select line is not toggled for every word, and stays asserted during the entire transaction. false- Slave select line is toggled after each word of data is transferred.
`BOOL isSelectPrecede;`	Valid only when protocol is CY_SPI_TI. true - The start pulse precedes the first data false - The start pulse is in sync with first data.
`BOOL isCpha;`	false - Clock phase is 0 true - Clock phase is 1.
`BOOL isCpol;`	false - Clock polarity is 0 true - Clock polarity is 1.

See Also

- CY_SPI_PROTOCOL
- CyGetSpiConfig
- CySetSpiConfig

3.29 PCY_SPI_CONFIG

Pointer to CY_SPI_CONFIG structure.

Description

This type represents a pointer to a CY_SPI_CONFIG structure.

See Also

- CY_SPI_CONFIG

3.30 CY_SPI_PROTOCOL

Enumeration defining SPI protocol types supported by USB Serial SPI module.

Description

These are the different protocols supported by USB-Serial SPI module.

Members

Members	Description
`CY_SPI_MOTOROLA = 0`	SPI transfers follow the Motorola definition of SPI protocol. Refer to the Cypress USB Serial device datasheet for details.
`CY_SPI_TI`	SPI transfers follow the Texas Instruments definition of SPI protocol. Refer to the Cypress USB Serial device datasheet for details.
`CY_SPI_NS`	SPI transfers follow the National Semiconductors definition of SPI protocol. Refer to the Cypress USB Serial device datasheet for details.

See Also

- CY_SPI_CONFIG
- CyGetSpiConfig
- CySetSpiConfig

4 USB Initialization API

This section has all the APIs that handle device initialization and fetching information about the device connected.

4.1 CyLibraryInit

This API is used to initialize the library.

Description

The API is used to initialize the underlying libusb library and is expected to be called when the application is being started.

Returns

- CY_SUCCESS on success
- CY_ERROR_DRIVER_INIT_FAILED on failure (Failure could be because of not calling CyLibraryExit previously)

See Also

- CyOpen
- CyLibraryExit

4.2 CyLibraryExit

This API is used to free the library.

Description

The API is used to free the library and should be called when exiting the application.

Returns

- CY_SUCCESS on success
- CY_ERROR_REQUEST_FAILED on failure

See Also

- CyOpen
- CyClose
- CyLibraryInit

4.3 CyGetListofDevices

This API retrieves number of USB devices connected to the host.

Description

This API retrieves the number of devices connected to the host. In Windows family of operating systems the API retrieves only the number of devices that are attached to CyUSB3.SYS driver. For other operating systems, it retrieves the total number of USB devices present on the bus. It includes both USB Serial device as well as other devices.

Parameters

Parameters	Description
`UINT8* numDevices`	Number of Devices connected

Returns

- CY_SUCCESS on success
- CY_ERROR_DEVICE_NOT_FOUND if there are no devices attached.
- CY_ERROR_REQUEST_FAILED if library was not initialized.

See Also

- CyGetDeviceInfo
- CyGetDeviceInfoVidPid
- CyOpen
- CyClose

4.4 CyGetDeviceInfo

This API retrieves the device information of a USB device.

Description

This API retrieves information about a device connected to host. In order to get the device information on particular device user needs to provide the device number. To identify the device of interest, the application needs to loop through all devices connected and obtain the information.

Parameters

Parameters	Description
`UINT8 deviceNumber`	Device number of the device of interest
`CY_DEVICE_INFO * deviceInfo`	Info of device returned

Returns

- CY_SUCCESS on success
- CY_ERROR_REQUEST_FAILED if library is not initialized (Only for Linux and Mac).
- CY_ERROR_INVALID_PARAMETER if the input parameters are invalid.
- CY_ERROR_DEVICE_INFO_FETCH_FAILED if failed to fetch device information.

- CY_ERROR_ACCESS_DENIED if access is denied by operating system.
- CY_ERROR_DEVICE_NOT_FOUND if specified device number is invalid.

See Also

- CY_DEVICE_INFO
- CY_DEVICE_TYPE
- CY_DEVICE_CLASS
- CyGetListofDevices
- CyGetDeviceInfoVidPid
- CyOpen
- CyClose

4.5 CyGetDeviceInfoVidPid

This API is used to retrieve the information of all devices with specified Vendor ID and Product ID.

Description

For a given VID and PID, the API returns deviceIdList and deviceInfoList. The deviceIdList contains the device numbers of all the devices with specified VID and PID. Using deviceInfoList application can identify the device of interest. Information that is provided includes interface number, string descriptor, deviceType and deviceClass.

Parameters

Parameters	Description
`CY_VID_PID vidPid`	VID and PID of device of interest
`UINT8 * deviceIdList`	Array of device ID's returned
`CY_DEVICE_INFO * deviceInfoList`	Array of pointers to device info list
`UINT8 * deviceCount`	Count of devices with specified VID PID
`UINT8 infoListLength`	Total length of the deviceInfoList allocated (Size of deviceInfoList array)

Returns

- CY_SUCCESS on success else error codes as defined in the enumeration CY_RETURN_STATUS.
- CY_ERROR_REQUEST_FAILED on if library is not initialized (Only for Linux and Mac)
- CY_ERROR_INVALID_PARAMETER if the input parameters are invalid.
- CY_ERROR_DEVICE_INFO_FETCH_FAILED if failed to fetch device information.
- CY_ERROR_ACCESS_DENIED if access is denied by operating system.
- CY_ERROR_DEVICE_NOT_FOUND if specified device number is invalid.

See Also

- CY_DEVICE_INFO
- CY_DEVICE_CLASS
- CY_DEVICE_TYPE

- CyGetListofDevices
- CyGetDeviceInfo
- CyOpen
- CyClose

4.6 CyOpen

This API is used to open the USB Serial device.

Description

This API is used to open USB Serial device based on the device number.

Parameters

Parameters	Description
`UINT8 deviceNumber`	Device number of device that needs to be opened
`UINT8 interfaceNum`	Interface Number
`CY_HANDLE * handle`	Handle returned by the API

Returns

- CY_SUCCESS on success
- CY_ERROR_REQUEST_FAILED on if library is not initialized (Only for Linux and Mac)
- CY_ERROR_INVALID_PARAMETER if the input parameters are invalid.
- CY_ERROR_DRIVER_OPEN_FAILED if open was unsuccessful.
- CY_ERROR_ACCESS_DENIED if access is denied by operating system.
- CY_ERROR_ALLOCATION_FAILED if memory allocation was failed.
- CY_ERROR_DEVICE_NOT_FOUND if specified device number is invalid.

See Also

- CyGetListofDevices
- CyGetDeviceInfoVidPid
- CyGetDeviceInfo
- CyClose

4.7 CyClose

This API closes the specified device handle and releases all resources associated with it.

Description

This API closes the device handle and releases all the resources allocated internally in the library. This API should be invoked using a valid device handle and upon successful return of CyOpen.

Parameters

Parameters	Description
`CY_HANDLE handle`	Handle of the device that needs to be closed

Returns

- CY_SUCCESS on success.
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if handle is invalid in case of Windows.
- CY_ERROR_REQUEST_FAILED on error in case of library being not initialized (Only for Linux and Mac).

See Also

- CyOpen

4.8 CyCyclePort

This API is used to power cycle the host port.

Description

This API will power cycle the upstream port. It will reenumerate the device after the power cycle.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if handle is invalid in case of Windows.
- CY_ERROR_REQUEST_FAILED on error if request was failed by driver.

See Also

- CyResetDevice

5 Common APIs

These APIs provide an interface for accessing GPIO pins, error status notification on UART/SPI, getting library and firmware version and signature.

5.1 CyGetLibraryVersion

This API retrieves the version of USB Serial library.

Description

This API retrieves the version of USB Serial library.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`PCY_LIBRARY_VERSION version`	Library version of the current library

Returns

- CY_SUCCESS

See Also

- CyGetFirmwareVersion

5.2 CyGetFirmwareVersion

This API retrieves the firmware version of the USB Serial device.

Description

This API retrieves the firmware version of the USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`PCY_FIRMWARE_VERSION firmwareVersion`	Firmware version.

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.

- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

See Also

- CyGetLibraryVersion

5.3 CyGetSignature

This API retrieves the signature of the device firmware.

Description

This API retrieves the signature of the device firmware.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`UCHAR * pSignature`	Signature returned

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

5.4 CySetEventNotification

This API is used to register a callback for error/event notifications during UART/SPI data transfers.

Description

The API is used to register a callback for error/event notifications while doing data transfer on UART or SPI. A callback will be issued based on the error/events sent by the device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_EVENT_NOTIFICATION_CB_FN notificationCbFn`	Callback function pointer

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_STATUS_MONITOR_EXIST if notification callback is already registered.

See Also

- CY_CALLBACK_EVENTS
- CY_EVENT_NOTIFICATION_CB_FN
- CyAbortEventNotification

5.5 CyAbortEventNotification

The API is used to unregister the event callback.

Description

The API is used to unregister the event callback.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_REQUEST_FAILED if API is called before registering callback.

See Also

- CySetEventNotification

5.6 CyGetGpioValue

This API retrieves the value of a GPIO.

Description

This API retrieves the value of a GPIO.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`UINT8 gpioNumber`	GPIO number
`UINT8 * value`	Current state of the GPIO

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

See Also

- CySetGpioValue

5.7 CySetGpioValue

This API sets the value of a GPIO.

Description

This API is used to set the value of a GPIO. It can only set the value of a GPIO that is configured as an output.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`UINT8 gpioNumber`	GPIO number
`UINT8 value`	Value that needs to be set

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED on error when request is failed by USB Serial device.

See Also

- CyGetGpioValue

5.8 CyResetDevice

This API resets the device by sending a vendor request.

Description

The API will reset the device by sending a vendor request to the firmware. The device will be re-enumerated.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

See Also

- CyCyclePort

5.9 CyReadUserFlash

The API reads from the flash address specified.

Description

Read from user flash area.The total space available is 512 bytes. The flash area address offset is from 0x0000 to 0x00200 and should be read page wise (page size is 128 bytes). On return transferCount parameter in CY_DATA_BUFFER will specify the number of bytes actually read.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER * readBuffer`	data buffer containing buffer address, length to read
`UINT32 flashAddress`	Address from which the data is read
`UINT32 timeout`	Timeout value of the API

Returns

- CY_SUCCESS on success

- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

See Also

- CyProgUserFlash

5.10 CyProgUserFlash

The API writes to the user flash area on the USB Serial device.

Description

The API programs user flash area. The total space available is 512 bytes. The flash area address offset is from 0x0000 to 0x00200 and should be written page wise (page size is 128 bytes). On return, transferCount parameter in CY_DATA_BUFFER will specify the number of bytes actually programmed.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER * progBuffer`	Data buffer containing buffer address, length to write
`UINT32 flashAddress`	Address to the data is written
`UINT32 timeout`	Timeout value of the API

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device.

See Also

- CyReadUserFlash

6 UART API

APIs used to communicate with UART module of the USB Serial device. These APIs provide support for configuration, data transfer and flow control.

6.1 CySetUartConfig

This API sets the UART configuration of USB Serial device.

Description

This API sets the UART configuration of USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_UART_CONFIG * uartConfig`	UART configuration value

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CY_UART_CONFIG
- CyGetUartConfig

6.2 CyGetUartConfig

This API retrieves the UART configuration from the USB Serial device.

Description

This API retrieves the UART configuration from the USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_UART_CONFIG * uartConfig`	UART configuration value read back

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CY_UART_CONFIG
- CySetUartConfig

6.3 CyUartRead

This API reads data from UART device.

Description

This API is used to read data from UART device. User needs to initialize the readBuffer with buffer pointer, number of bytes to read before invoking this API. On return the transferCount parameter in CY_DATA_BUFFER will contain the number of bytes read.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER* readBuffer`	Read buffer details
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success.
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if input parameters were invalid.
- CY_ERROR_REQUEST_FAILED if the device type is not UART.
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_PIPE_HALTED if pipe was stalled during data transfer.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.
- CY_ERROR_ALLOCATION_FAILED if transaction transmit buffer allocation was failed (Only in Windows).

See Also

- CY_DATA_BUFFER
- CyUartWrite

6.4 CyUartWrite

This API writes the data to UART device.

Description

This API writes the data to UART device. User need to initialize the writeBuffer with buffer pointer, number of bytes to write before invoking the API. On return the transferCount parameter in CY_DATA_BUFFER will contain the number of bytes written.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER* writeBuffer`	Write buffer details
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success.
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if input parameters were invalid.
- CY_ERROR_REQUEST_FAILED if the device type is not UART.
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_PIPE_HALTED if pipe was stalled during data transfer.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.
- CY_ERROR_ALLOCATION_FAILED if transaction transmit buffer allocation was failed (Only in Windows).

See Also

- CY_DATA_BUFFER
- CyUartRead

6.5 CyUartSetDtr

This API sets DTR signal in UART.

Description

This API used set the DTR. It sets the DTR pin to logical low.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartClearRts
- CyUartSetRts
- CyUartClearDtr

6.6 CyUartClearDtr

This API can be used to clear DTR signal in UART.

Description

This API can be used clear the DTR. It sets the DTR pin to logical high.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartSetRts
- CyUartSetDtr
- CyUartClearRts

6.7 CyUartSetRts

This API sets RTS signal in UART module.

Description

This API is used to set the RTS pin to logical low..

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartClearRts
- CyUartSetDtr
- CyUartClearDtr

6.8 CyUartClearRts

This API can be used to clear RTS signal in UART module.

Description

This API used clear the RTS. It sets the RTS pin to logical high.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.

- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartSetRts
- CyUartSetDtr
- CyUartClearDtr

6.9 CyUartSetHwFlowControl

This API enables hardware flow control.

Description

This API enables hardware flow control.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_FLOW_CONTROL_MODES mode`	Flow control mode

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE on error if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER on error if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT on error if request was timed out.
- CY_ERROR_REQUEST_FAILED on error if request was failed by device or if device type is not UART.

See Also

- CyUartGetHwFlowControl

6.10 CyUartGetHwFlowControl

This API retrieves the current hardware flow control status.

Description

This API retrieves the current hardware flow control status.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

`CY_FLOW_CONTROL_MODES * mode`	Flow control mode

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartSetHwFlowControl

6.11 CyUartSetBreak

This API can be used to set break timeout value .

Description

This API can be used to set break timeout value .

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`UINT16 timeout`	Break timeout value in milliseconds

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not UART.

See Also

- CyUartSetHwFlowControl

7 I2C API

These set of APIs provide an interface to configure I2C module and do read/write on the I2C device connected to USB Serial device.

7.1 CySetI2cConfig

This API configures the I2C module of USB Serial device.

Description

This API configures the I2C module of USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_I2C_CONFIG * i2cConfig`	I2C configuration value

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not I2C.

See Also

- CY_I2C_CONFIG
- CyGetI2cConfig

7.2 CyGetI2cConfig

This API retrieves the configuration of I2C module of USB Serial device.

Description

This API retrieves the configuration of I2C module of USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_I2C_CONFIG * i2cConfig`	I2C configuration value read back

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not I2C.

See Also

- CY_I2C_CONFIG
- CySetI2cConfig

7.3 CyI2cRead

This API reads data from the USB Serial I2C module.

Description

This API provides an interface to read data from the I2C device connected to USB Serial.

The readBuffer parameter needs to be initialized with buffer pointer, number of bytes to be read before invoking the API. On return, the transferCount field will contain the number of bytes read back from device.
CY_I2C_DATA_CONFIG structure specifies parameters such as setting stop bit, NAK and slave address of the I2C device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_I2C_DATA_CONFIG * dataConfig`	I2C data config
`CY_DATA_BUFFER * readBuffer`	Read buffer details
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success.
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if input parameters were invalid.
- CY_ERROR_REQUEST_FAILED if the device type is not I2C
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.
- CY_ERROR_ALLOCATION_FAILED if transaction transmit buffer allocation was failed (Only in Windows).
- CY_ERROR_I2C_DEVICE_BUSY if I2C device was busy processing previous request.

- CY_ERROR_I2C_NAK_ERROR if request was nacked by I2C device.
- CY_ERROR_I2C_ARBITRATION_ERROR if a I2C bus arbitration error occured.
- CY_ERROR_I2C_BUS_ERROR if there was any I2C bus error while an on going transaction.
- CY_ERROR_I2C_STOP_BIT_SET if stop bit was set by I2C master.

See Also

- CY_DATA_BUFFER
- CY_I2C_DATA_CONFIG
- CyI2cWrite

7.4 CyI2cWrite

This API writes data to USB Serial I2C module .

Description

This API provides an interface to write data to the I2C device connected to USB Serial. The writeBuffer parameter needs to be initialized with buffer pointer, number of bytes to be written before invoking the API. On return, transferCount field contains number of bytes actually written to the device. CY_I2C_DATA_CONFIG structure specifies parameter such as setting stop bit, Nak and slave address of the I2C device being communicated when USB Serial is master.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_I2C_DATA_CONFIG * dataConfig`	I2C Slave address
`CY_DATA_BUFFER * writeBuffer`	Write buffer details
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success.
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if input parameters were invalid.
- CY_ERROR_REQUEST_FAILED if the device type is not I2C
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_PIPE_HALTED if pipe was stalled during data transfer.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.
- CY_ERROR_ALLOCATION_FAILED if transaction transmit buffer allocation was failed (Only in Windows).
- CY_ERROR_I2C_DEVICE_BUSY if I2C device was busy processing previous request.
- CY_ERROR_I2C_NAK_ERROR if request was nacked by I2C device.
- CY_ERROR_I2C_ARBITRATION_ERROR if a I2C bus arbitration error occured.

- CY_ERROR_I2C_BUS_ERROR if there was any I2C bus error while an on going transaction.
- CY_ERROR_I2C_STOP_BIT_SET if stop bit was set by I2C master.

See Also

- CY_DATA_BUFFER
- CY_I2C_DATA_CONFIG
- CyI2cRead

7.5 CyI2cReset

This API resets the I2C module in USB Serial device.

Description

This API resets the I2C module whenever there is an error in data transaction.

If resetMode = 0 the I2C read module will be reset. If resetMode = 1 the I2C write module will be reset.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`BOOL resetMode`	Reset mode

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not I2C.

See Also

- CyI2cRead
- CyI2cWrite

8 SPI API

These set of APIs provide an interface to configure SPI module and perform read/write operations with the SPI device connected to USB Serial device.

8.1 CySetSpiConfig

This API sets the configuration of the SPI module on USB Serial device.

Description; This API sets the configuration of the SPI module in USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_SPI_CONFIG * spiConfig`	SPI configuration structure value

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not SPI.

See Also

- CY_SPI_CONFIG
- CyGetSpiConfig

8.2 CyGetSpiConfig

This API retrieves the configuration of SPI module of USB Serial device.

Description

This API retrieves the configuration of SPI module of USB Serial device.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_SPI_CONFIG * spiConfig`	SPI configuration structure value read back

Returns

- CY_SUCCESS on success

- CY_ERROR_INVALID_HANDLE if handle is invalid.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if the request is timed out.
- CY_ERROR_REQUEST_FAILED when request is failed by USB Serial device or when device type is not SPI.

See Also

- CY_SPI_CONFIG
- CySetSpiConfig

8.3 CySpiReadWrite

This API reads and writes data to SPI device connected to USB Serial device.

Description

This API provides an interface to do data transfer with the SPI slave/master connected to USB Serial device. To perform read only operation, pass NULL as argument for writeBuffer and to perform write only operation pass NULL as an argument for readBuffer. On return, the transferCount field will contain the number of bytes read and/or written.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER* readBuffer`	Read data buffer
`CY_DATA_BUFFER* writeBuffer`	Write data buffer
`UINT32 timeout`	Time out value of the API

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_REQUEST_FAILED if the device type is not SPI or when libusb reported unknown error in case of Linux/Mac.
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_PIPE_HALTED if pipe was stalled during data transfer.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.

See Also

- CY_DATA_BUFFER
- CyGetSpiConfig
- CySetSpiConfig

9 JTAG API

These set of APIs can be used to enable or disable JTAG module on the USB Serial device. Once the JTAG is enabled, read and write operations can be performed. When JTAG is enabled other modules in the USB Serial device cannot be used.

9.1 CyJtagEnable

This API enables JTAG module.

Description

This API enables JTAG module in USB Serial device and the function disables all other functionality till CyJtagDisable is invoked.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if request was timed out.
- CY_ERROR_REQUEST_FAILED if request was failed by device or if device type is not JTAG.

See Also

- CyJtagDisable

9.2 CyJtagDisable

This API disables JTAG module.

Description

This API disables Jtag interface in USB Serial device. This API must be invoked before exiting the application if CyJtagEnable was previously invoked.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_IO_TIMEOUT if request was timed out.
- CY_ERROR_REQUEST_FAILED if request was failed by device or if device type is not JTAG.

See Also

- CyJtagEnable

9.3 CyJtagRead

This API reads data from JTAG device.

Description

This API provides an interface to read data from JTAG device. The readBuffer need to be initialized with buffer and length of data to be written before invoking the API. Upon return, transferCount field in CY_DATA_BUFFER structure is updated with actual number of bytes read.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER * readBuffer`	Read buffer parameters
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_REQUEST_FAILED if device type is not JTAG or when encountered unknown libusb errors in Linux/Mac.
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.
- CY_ERROR_BUFFER_OVERFLOW if data received from USB Serial device is more than requested.

See Also

- CY_DATA_BUFFER
- CyJtagWrite
- CyJtagEnable

9.4 CyJtagWrite

This API can be used to write data to JTAG module.

Description

This API provides an interface to write data to JTAG device connected to USB Serial device. The writeBuffer need to be initialized with buffer and length of data to be written before invoking the API. Upon return, transferCount field in CY_DATA_BUFFER is updated with actual number of bytes written.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`CY_DATA_BUFFER * writeBuffer`	Write buffer details
`UINT32 timeout`	API timeout value

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle is invalid in case of Linux/Mac.
- CY_ERROR_INVALID_PARAMETER if specified parameters are invalid or out of range.
- CY_ERROR_REQUEST_FAILED if device type is not JTAG or when encountered unknown libusb errors in Linux/Mac.
- CY_ERROR_PIPE_HALTED if there was any pipe error during transaction.
- CY_ERROR_IO_TIMEOUT if transfer was timed out.
- CY_ERROR_DEVICE_NOT_FOUND if device was disconnected.

See Also

- CY_DATA_BUFFER
- CyJtagRead
- CyJtagEnable

10 PHDC API

Set of PHDC class request APIs. The PHDC class requests include set, clear feature and PHDC get status.

10.1 CyPhdcSetFeature

This API sends a PHDC set feature command.

Description

This API sends a PHDC set feature command.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle was invalid.
- CY_ERROR_IO_TIMEOUT if request timed out.
- CY_ERROR_REQUEST_FAILED if request was failed by device.

See Also

- CyPhdcClrFeature
- CyPhdcGetStatus

10.2 CyPhdcClrFeature

This API sends a PHDC clear feature command.

Description

This API sends a PHDC clear feature command.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle was invalid.
- CY_ERROR_IO_TIMEOUT if request timed out.

- CY_ERROR_REQUEST_FAILED if request was failed by device.

See Also

- CyPhdcSetFeature
- CyPhdcGetStatus

10.3 CyPhdcGetStatus

This API retrieves the endpoint status of PHDC transaction.

Description

The API retrieves the status of PHDC transaction. It returns 2 bytes of data pending bit map which is defined as per PHDC spec.

Parameters

Parameters	Description
`CY_HANDLE handle`	Valid device handle
`UINT16 * dataStatus`	Data pending status bit map

Returns

- CY_SUCCESS on success
- CY_ERROR_INVALID_HANDLE if handle was invalid.
- CY_ERROR_IO_TIMEOUT if request timed out.
- CY_ERROR_REQUEST_FAILED if request was failed by device.

See Also

- CyPhdcClrFeature
- CyPhdcSetFeature

11 Symbol Reference

11.1 Functions

The following table lists functions in this documentation.

11.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

11.3 Types

The following table lists types in this documentation.

11.4 Macros

The following table lists macros in this documentation.

Index

O

P

S

U