QuickUSB™

User's Guide

Bitwise™ SYSTEMS

Bitwise Systems
5385 Hollister Ave, Suite 215
Santa Barbara, CA 93111

Voice	(805) 683-6469
Fax	(805) 683-6469
Toll Free	(800) 224-1633
Web Site	www.bitwisesys.com
Technical Support	support@bitwisesys.com

Version 1.30
March 11, 2003

Table of Contents

Getting Started

Thank you for choosing QuickUSB™, the quick Hi-Speed USB 2.0 connectivity solution. QuickUSB is designed to make adding general-purpose USB 2.0 connectivity to your product quick and easy.

The QuickUSB plug-in module is a complete, bus-powered USB 2.0 device that contains a wide variety of general-purpose I/O ports to facilitate connecting nearly any type of digital device to a PC using USB 2.0. The QuickUSB Library is an application programmer's interface (API) that provide a software interface to the I/O ports on the QuickUSB module. The QuickUSB Library CD contains the QuickUSB driver, DLLs and software development application examples for the following languages:

- Microsoft C/C++ version 6.0
- Borland C++ Builder version 5
- Microsoft Visual Basic 6.0
- Borland C++ Builder 5.0
- Borland Delphi 4.0

The QuickUSB library may be used with other software development languages. See the ?? section for more information.

Install the QuickUSB Library

The QuickUSB Library is a set of software development libraries and programs that enable you to use and write software to use the QuickUSB module. To install the QuickUSB Library, Insert the QuickUSB Library CD into the CD drive of the target computer and double-click the Setup.exe file. This will launch the installer program that will install the QuickUSB Library onto your computer.

Install the QuickUSB Driver

This step is required before you can communicate with the QuickUSB module. To install the QuickUSB driver:

1. Install the QuickUSB Library onto the computer you want to use the QuickUSB module on
2. Connect the QuickUSB module to the computer. When Windows asks where the driver is,

The QuickUSB driver is located in two places for your convenience. First, it is present in the root directory of the QuickUSB Library CD. This is in case you need to install the QuickUSB driver on a computer that doesn't need the whole QuickUSB Library installed. Next, the QuickUSB driver is installed in the \Program Files\Bitwise Systems\QuickUSB\Drivers of any computer that has the QuickUSB Library installed. This is the best place to install your driver from because sometimes Windows will need to reload the driver. By default, Windows looks at the last place the driver was installed from and if it is stored in a known location on the computer's hard drive, it will just reinstall it automatically and you won't have to insert a CD or answer any questions. If the driver is not located where it was last installed from, you will need to tell Windows where to find the driver.

Upgrading from a Previous Version

To upgrade your QuickUSB product, uninstall your current version by going to Control Panel, Add/Remove programs and Remove QuickUSB Library. Then, install the latest version of the QuickUSB Library that you downloaded from the web site. Next, upgrade the firmware in your QuickUSB module by using the QuickUSB Programmer to load firmware file with the same version number as the QuickUSB Library. Of course, after upgrading you will need to recompile any applications using the new QuickUSB.bas or QuickUSB.h files to use the new version of QuickUSB.dll and/or QuickUSB.lib.

Using the QuickUSB Module

Overview

Let's begin by explaining how the QuickUSB module is used to implement USB 2.0 peripherals. The basic premise behind the QuickUSB plug-in module is that electronic designers often need to connect their circuitry to a host computer in order to transfer data between their electronics and the host computer. One of the most attractive host computer interfaces is USB. However, USB tends to be a rather complex interface to implement both in hardware and software and for that reason, low and medium volume applications tend to avoid USB and go with an easier interface to implement.

The QuickUSB module offers a variety of hardware interfaces. There are four basic interface types; high-speed parallel, general-purpose parallel I/O ports, asynchronous serial ports and synchronous serial ports. These interfaces cover the vast majority of PC connectivity needs.

High-Speed Parallel Port

The high-speed parallel (HSP) port is an 8 or 16-bit port that transfers 512 byte blocks of data and increments an address bus synchronously with the 48MHz IFCLK. This type of interface is best suited to interface with high speed FPGA or DSP based applications where low latency and high throughput are the primary goals.

General Purpose Parallel I/O Ports

The QuickUSB module has three 8-bit general-purpose parallel I/O ports named A, C & E. The QuickUSB Library functions for these ports provide the capability to set the direction of each pin individually and send or receive from 1 to 64 bytes of port data in a single function call. Some of these ports are shared with the SPI and FPGA configuration functions. See SPI and FPGA configuration documentation to see which ports are affected.

Asynchronous Serial Ports

The QuickUSB module has two interrupt-driven RS-232 compatible (EIA/TIA-562) ports onboard. They have programmable baud rate and both ports must have the same baud rate setting. The parity setting is programmable to None, Odd or Even and the number of stop bits is fixed at 1.

Synchronous Serial Ports

The QuickUSB module has one soft SPI port onboard. This port consumes port E, so if you use this port, port E will not be available for parallel I/O. Also, for SPI slave addresses from 2 through 9, port A pins 0 through 7 respectively are consumed as nSS (not Slave Select) lines for each address. The QuickUSB Library functions for SPI transfers provide the capability to send or receive from 1 to 64 bytes in a single function call. The bit-shifting algorithm is also programmable as LSbit first or MSbit first.

Using the QuickUSB Library

The QuickUSB™ Library API gives programmers a cohesive programming interface to the QuickUSB family of easy-to-use USB plug-in modules. The same QuickUSB Library will work for all QuickUSB products, so you write your software once and all your QuickUSB based products will work.

General

When a QuickUSB client application runs, there may be any number of QuickUSB modules connected to the system. Use the QuickUsbFindModules function to get a list of available devices, then parse out the individual device names. The general procedure to use a QuickUSB module is to open the module using QuickUsbOpen by passing in a device name obtained using the QuickUsbFindModules function and save the device id returned in ‘newDevId’.

Then, make library calls using the devId to refer to that device. When calls are complete, close the device with a call to QuickUsbClose(devId). This will free the device to be used by other threads and processes. It is generally a good idea to minimize the amount of time that a device is kept open so that other software can use the module.

*int QuickUsbFindModules(char *nameList, unsigned long bufferLength)*

Purpose

To scan the system for connected QuickUSB modules.

Parameters

nameList – A buffer in which to store a of QuickUSB module names found by the library. Must be large enough to contain all the device names + 1 character.
bufferLength – The length of the nameList buffer in characters.

Returns

Non-zero value on success, zero (0) on failure. This function returns a NULL (‘\0’ or CHR(0) or ‘□’) delimited list of QuickUSB module names found by the library in ‘nameList’. The final entry is designated by two consecutive NULL characters. After executing this function with one module connected, nameList will contain "QUSB-0□□". If there were two devices plugged in, nameList would contain "QUSB-0□QUSB-1□□".

Notes

This routine will only find closed devices.

Examples

<u>Visual Basic</u>

```
Dim result As Long
Dim devCount As Integer
Dim ctlCount As Integer
Dim devNameStr As String
Dim startIndex As Integer
Dim nullIndex As Integer
Static oldDevList As String

devList = Space(1024)                   ‘ Don’t forget to fill the string to allocate storage
result = QuickUsbFindModules(devList, 1024)

startIndex = 1                          ‘ Parse devList and put the names into a combo box
ModuleComboBox.Clear
Do While (Mid(devList, startIndex, 1) <> Chr(0))
  nullIndex = InStr(startIndex, devList, Chr(0), vbBinaryCompare)      ‘ Find the first null
  devNameStr = Mid(devList, startIndex, nullIndex - 1)                 ‘ Extract the name
  ModuleComboBox.AddItem devNameStr                                    ‘ Add it to the combobox
  startIndex = nullIndex + 1
  devCount = devCount + 1
Loop
```

<u>Visual C++</u>

```
char *str, nameList[512];
int result;
unsigned long length = 512;

result = QuickUsbFindModules(nameList, length);

// Parse the name list and put the names in a combo box
str = nameList;
while (*str != '\0') {
  m_moduleComboBox.AddString(str);
  str = str + strlen(str) + 1;
}
```

int QuickUsbOpen(HANDLE *hDevice, char *devName)

Purpose

Open a QuickUSB device for use by the library

Parameters

hDevice – A pointer to a HANDLE in which to place the new device ID. If successful, newDevId will contain a the new devId HANDLE.

devName – a NULL terminated character array containing the name of the device. Device names are of the form 'QUSB-X' where X is the device address (0-126) in decimal.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

Examples

Visual Basic

```
Dim hDevice As Long
Dim result As Long
Dim devName As String

' Open the device
result = QuickUsbOpen(hDevice, devName)
If (result = 0) Then
  ShowStatus "Cannot open Device:" & devName
  Exit Sub
End If
```

Visual C++

```
char devName[16];
int result;
int sel;
HANDLE hDevice;

// Get device name from the ComboBox control
sel = m_moduleComboBox.GetCurSel();
m_moduleComboBox.GetLBText(sel, devName);

// Open the device
result = QuickUsbOpen(&hDevice, devName);
if (result == FALSE) {
  return;
}
```

int QuickUsbClose(HANDLE hDevice);

Purpose

Close a QuickUSB device.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

Examples

Visual Basic

```
' Close the device when you're done
QuickUsbClose hDevice
```

Visual C++

```
// Close the QuickUSB module
result = QuickUsbClose(hDevice);
if (result == FALSE) {
   return;
}
```

*int QuickUsbGetStringDescriptor(HANDLE hDevice, unsigned char index, char *buffer, unsigned short length)*

Purpose

Returns the string descriptor for the selected QuickUSB module.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
index – The device descriptor string index (use 2 to get the QuickUSB Device ID string).
buffer – A pointer to a buffer in which to place the string descriptor.

length – The length of the buffer in bytes (should be at least 32).

Returns

Non-zero value on success, zero (0) on failure. This function returns a NULL terminated string in the buffer if successful.

Notes

None.

Settings

The QuickUSB module has certain settings that control the behavior of the module. These functions manipulate those settings in order to customize the module's behavior for your particular needs. A list of settings follows:

Addr	Name	Saved	Description	Values
1	SETTING_WORDWIDE	No	High-speed port data width	1 = 16 bits, 0 = 8 bits
2	SETTING_DATAADDRESS	No	Data bus starting address and flags to enable and disable 1) the address bus and 2) the feature that autoincrements the bus address after each data transaction.	Bits 0-8: address Bit 14: 0 = enable address bus, 1 = disable address bus Bit 15: 0 = increment address bus, 1 = don't increment address bus
3	SETTING_FIFO_CONFIG	No	Sets the FIFO configuration	See register definition for IFCONFIG, CY7C68013 TRM, pg15-14. Master Mode = 0xFA Slave Mode = 0x03
4	SETTING_FPGATYPE	No	Sets the FPGA configuration scheme	0 = Altera PS (passive serial) mode
5	SETTING_CPUCONFIG	No	Sets the CPU configuration	See register definition for CPUCS, CY7C68013 TRM, pg15-13

Table 1 - QuickUSB Settings

Settings that have a yes in the saved column are saved across power cycles in non-volatile memory. At the present time, there are no non-volatile memory settings.

*int QuickUsbReadSetting(HANDLE hDevice, unsigned short address, unsigned short *setting);*

Purpose

To read QuickUSB module settings.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The setting address (number)
*setting – The value of the addressed setting if successful.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None.

int QuickUsbWriteSetting(HANDLE hDevice, unsigned short address, unsigned short setting);

Purpose

To write QuickUSB module settings.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The setting address (number)
setting – The value of the addressed setting.

Returns

A non-zero value if successful, 0 otherwise.

Notes

See QuickUsbReadSetting

FPGA Configuration

The QuickUSB Plug-In module can configure SRAM-based FPGA devices over the USB. At the present time the only supported FPGA configuration mode is Altera® Passive Serial mode. It is the default FPGA type. For more information on changing the FPGA type, see the FPGATYPE setting of the 'Settings' section of this document.

To configure an FPGA, follow these steps:

1. Call QuickUsbStartFpgaConfiguration. This resets the FPGA and starts the configuration process.
2. Open the RBF file and read in 64 bytes into a buffer.
3. Call QuickUsbWriteFpgaData and pass in the data from the file. Repeat this process until the entire file is sent. Of course, the last block will probably not be 64 bytes long. Also, if you need to restart the configuration process for some reason, you can call restart at step 1 at any time. Be sure to close the RBF file when you're done.
4. Call QuickUsbIsFpgaConfigured. If the 'isConfigured' parameter gets set to '1', the FPGA was correctly configured. If not, try again starting at step 1.

Altera FPGAs

Altera Passive Serial mode configuration uses port E. The QuickUSB module provides the DCLK, DATA0, nCE, nCONFIG, nSTATUS and CONF_DONE signals required to configure Altera devices in passive-serial mode.

The QuickUSB module uses 3.3V I/O, so make sure your device can handle 3.3V on the configuration pins. Then, connect the like-named signals to the Altera device as shown in the 'QuickUSB Target Interface' document available from www.quickusb.com or in the Doc directory of the QuickUSB Library. You must also be sure to add pull-up resistors (normally 1k) to the open-drain configuration lines (CONF_DONE, nSTATUS & nCONFIG). The QuickUSB can transfer an unlimited number of 64 byte blocks of configuration data, so multiple devices can be configured using QuickUSB. If more than one device must be configured over the interface, the devices should be 'daisy-chained'.

In addition, consult Altera Application Note 116, 'Configuring SRAM-Based LUT Devices'. It is on the Altera web site. In particular, the section 'PS Configuration with a Microprocessor' section of this document specifies the circuitry needed to configure your Altera device with a microcontroller (the QuickUSB module).

Also, you must convert your FPGA configuration file to .RBF (raw binary format-one bit) format to configure it using QuickUSB. If you are configuring multiple devices, they must be daisy-chained (consult AN116) and the configuration files combined in the conversion process into a single RBF file.

int QuickUsbStartFpgaConfiguration(HANDLE hDevice)

Purpose

Start the process of FPGA configuration. If the FPGA is in the process of being configured, the process will restart. If the FPGA is already configured, it will be reset and reconfigured.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

*int QuickUsbWriteFpgaData(HANDLE hDevice, char *data, unsigned long length)*

Purpose

Sends FPGA configuration data to the FPGA using the QuickUSB FPGA configuration port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
data – A pointer to the FPGA configuration data.
length – The length of the data in bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

FPGA configuration must be preceded by calling QuickUsbStartFpgaConfiguration.

*int QuickUsbIsFpgaConfigured(HANDLE hDevice, unsigned short *isConfigured)*

Purpose

Check to see if the FPGA is correctly configured.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
isConfigured – A pointer to an unsigned short in which to write the configuration status of the FPGA connected to the QuickUSB FPGA configuration port. 1 = the FPGA is configured (CONF_DONE = ‘1’), 0 = the FPGA is not configured (CONF_DONE = ‘0’).

Returns

A non-zero value if successful, 0 otherwise.

Notes

The value of the CONF_DONE is written to isConfigured.

High-Speed Parallel Port Commands

The high-speed parallel port (HSPP) is an 8 or 16-bit port that transfers data synchronously with the 48MHz IFCLK. In addition, there is a 9-bit address bus which optionally increments each time a data element is read or written. The address bus can be programmed not to increment also for multiple writes to the same address. The address bus can also be disabled so the address bus bits can be reused as general purpose I/O. See the DATAADDRESS setting in the 'Settings' section of this manual for more information

High Speed Parallel Port Modes

There are two modes of HSPP operation, master and slave. The HSPP mode is selected by changing the FIFO_MODE setting and may be changed at any time. But typically, your hardware will be configured for either master or slave mode and the requirements of your application will determine which mode is best for you.

HSPP Master Mode (default)

In HSP master mode, the QuickUSB module controls all aspects of the HSPP and the host PC initiates all data transfers through the QuickUSB module.

In master mode, all HSPP transfers are executed synchronously with IFCLK and are controlled by two signals, REN and WEN. REN indicates read operations and WEN indicates write operations. If the address bus is enabled, the HSPP functions set the address bus to the passed address value and increment it once for each data element transferred. The WORDWIDE setting controls the data element width. If WORDWIDE is 1, the transfers are 16-bits wide and if 0, 8-bits wide. For more information about WORDWIDE, see the WORDWIDE setting in the 'Settings' section of this document.

In master mode, the QuickUSB module uses the HSPP as a two channel bi-directional parallel port. The two channels are called command (CMD) and data (DATA).

Command Transfers
Command transfers are low-speed transfers that use the data bus (FD) and the address bus (GPIFADR) to read and write data to and from registers in the target hardware. The QuickUsbReadCommand and QuickUsbWriteCommand functions are used to read and write registers in the peripheral. They transfer data one element at a time with the CMD_DATA line set high ('1').

Data Transfers
Data transfers are high-speed block-oriented data transfers that use the data bus (FD) and optionally the address bus (GPIFADR) to read and write data to either a FIFO or a memory in the target hardware. The QuickUsbReadData and QuickUsbWriteData functions are used to perform high-speed data transfers. They transfer data in blocks of 512 bytes (256 words, depending on the WORDWIDE setting) with the CMD_DATA line set low ('0').

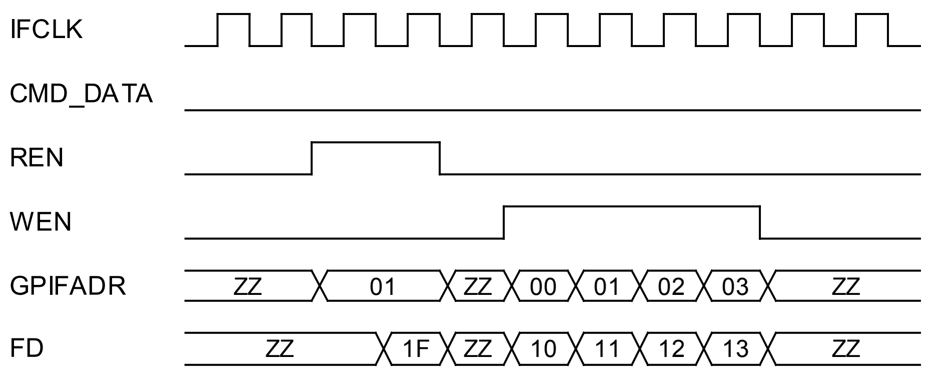


Figure 1 –Master Mode Command and Data Transfer Waveforms

HSPP Slave Mode

In addition, the HSP may also be operated in 'slave FIFO' mode. In this mode, external logic controls the QuickUSB FIFOs. This way an external microcontroller or FPGA can load the QuickUSB FIFOs and the data is then automatically transferred between the USB and the FIFOs. Slave FIFO mode is controlled by the FIFO_MODE setting.

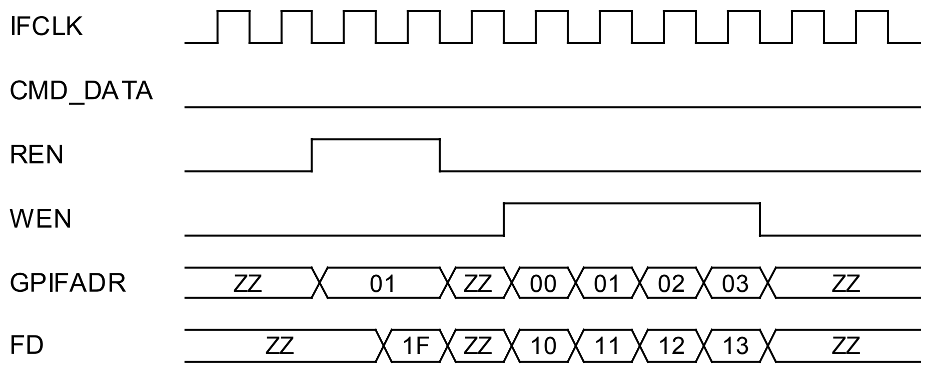


Figure 2 - Slave Mode Data Transfer Waveforms

*int QuickUsbReadCommand(HANDLE hDevice, unsigned short address, unsigned char *data, unsigned short *length)*

Purpose

Read a block of command values from the high-speed parallel port using the QuickUSB module.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – An unsigned short address. This address is the starting value of the HSP address bus. If address bit 14 is set (1), then the address bus will not be driven.
If address bit 15 is set (1), then the address will not be incremented after each read.
data – A pointer to a block of commands to read from the high-speed parallel port.
length – A pointer to an unsigned short containing the number of bytes to read from the high-speed parallel port on input and the number of bytes actually read on return. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

Commands are transferred over the high-speed parallel port with the CMD_DATA line set to ‘1’.

*int QuickUsbWriteCommand(HANDLE hDevice, unsigned short address, unsigned char * data, unsigned short length)*

Purpose

Write a block of command values to the high-speed parallel port using the QuickUSB module.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – An unsigned short address. This address is the starting value of the HSP address bus. If address bit 14 is set (1), then the address bus will not be driven.
If address bit 15 is set (1), then the address will not be incremented after each write.
data – A pointer to a block of commands to write to the high-speed parallel port.
length – An unsigned short containing the number of bytes to write to the high-speed parallel port. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

Commands are transferred over the high-speed parallel port with the CMD_DATA line set to ‘1’.

*int QuickUsbReadData(HANDLE hDevice, unsigned char *inData, unsigned long *length)*

Purpose

Read a block of data values from the high-speed parallel port using the QuickUSB module.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
inData – A pointer to a block of data values to read from the high-speed parallel port.
length – A pointer to an unsigned short containing the number of bytes to read from the high-speed parallel port on input and the number of bytes actually read on return.

Returns

A non-zero value if successful, 0 otherwise.

Notes

- In master mode, data values are transferred over the high-speed parallel port with the CMD_DATA line set to ‘0’.
- In slave mode, the USB will hang and wait for ‘length’ transfers to occur before returning. If the target interface does not write length transfers to the slave FIFOs, the transfer will hang the system and a reboot may be required to restore the USB stack to an operational state.
- The address bus behavior for data transfers is defined by the DATAADDRESS setting.

*int QuickUsbWriteData(HANDLE hDevice, unsigned char *outData, unsigned long length)*

Purpose

Write a block of data values to the high-speed parallel port using the QuickUSB module.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
inData – A pointer to a block of data values to write to the high-speed parallel port.
length – An unsigned short containing the number of bytes to write to the high-speed parallel port.

Returns

A non-zero value if successful, 0 otherwise.

Notes

- In master mode, data values are transferred over the high-speed parallel port with the CMD_DATA line set to ‘0’.
- In slave mode, the USB will hang and wait for ‘length’ transfers to occur before returning. If the target interface does not read length transfers from the slave FIFOs, the transfer will hang the system and a reboot may be required to restore the USB stack to an operational state.
- The address bus behavior for data transfers is defined by the DATAADDRESS setting.

General-Purpose Parallel I/O

There are five (5) 8-bit general-purpose parallel I/O ports on the QuickUSB module named A-E. Of those, some may be used for other purposes. Port E is used for the FPGA configuration and SPI functions, so if either of these functions are used in your design, some of the pins on port E will be consumed.

Other than that, the general-purpose I/O ports are just like I/O ports you would find on a microcontroller. The functions provided by the QuickUSB Library give you the capability to read or set the ports on a byte wide basis.

*int QuickUsbReadPortDir(HANDLE hDevice, unsigned short address, unsigned char *data)*

Purpose

Read the data direction of each data port bit for the specified port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The port address. Ports are addressed 0 to 4 corresponding to port A-E.
data – A pointer to a byte in which to place the data direction bit values. Each bit in data corresponds to data bits of the specified port. A data direction bit value of 0=input and 1=output (e.g. 0x03 means that bits 0 and 1 are outputs and bits 2-7 are inputs).

Returns

A non-zero value if successful, 0 otherwise. Also returns the data direction bits in data if successful. Otherwise value is not modified.

Notes

None

int QuickUsbWritePortDir(HANDLE hDevice, unsigned short address, unsigned char data)

Purpose

Set the data direction of each data port bit for the specified port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The port address. Ports are addressed 0 to 4 corresponding to port A-E.
data – A byte which contains the data direction bit values. Each bit in data corresponds to data bits of the specified port. A data direction bit value of 0=input and 1=output.

Returns

A non-zero value if successful, 0 otherwise. Also returns the data direction bits in data if successful. Otherwise value is not modified.

Notes

None

*int QuickUsbReadPort(HANDLE hDevice, unsigned char address, unsigned char *data, unsigned short *length)*

Purpose

Read a series of bytes from the specified port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.

address – The port address. Ports are addressed 0 to 4 corresponding to port A-E.

data – A pointer to an array of bytes in which to place the data. This buffer must be at least 'length' bytes long.

length – A pointer to the number of bytes to read from the port. The bytes are read sequentially. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise. Also returns the read value in value if successful. Otherwise value is not modified.

Notes

None

*int QuickUsbWritePort(HANDLE hDevice, unsigned char address, unsigned char *data, unsigned short *length)*

Purpose

None

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.

address – The port address. Ports are addressed 0 to 4 corresponding to port A-E.

data – A pointer to an array of bytes to send out the port. This buffer must be at least 'length' bytes long.

length – A pointer to the number of bytes to write to the port. The bytes are written sequentially. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

RS-232

The RS-232 ports are interrupt-driven for transmit and receive. This means that your code on the PC doesn't have to continuously poll the RS-232 ports of the QuickUSB for fear of missing a character. The QuickUSB buffers are 128 bytes deep on both the transmit and receive side, so your software just needs to service these buffers often enough to make sure they don't overflow.

int QuickUsbSetRs232BaudRate(HANDLE hDevice, unsigned long baudRate)

Purpose

Set the baud rate for both serial ports. Baud rates are programmable up to 230kbps. This function sets the baud rate of both serial ports. It is not possible to set the baud rate of each serial port independently.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
baudRate – A long integer containing the baud rate.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

*int QuickUsbReadRS232 (HANDLE hDevice, unsigned char portNum, char *inputString, unsigned short *length)*

Purpose

Read a block of characters from the interrupt receive buffer of the specified QuickUSB serial port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
portNum – The serial port number. Serial port 0 (P1) = 0, serial port 1 (P2) = 1.
length – A pointer to the number of characters to read. Set to the number of characters actually read on return.

Returns

A non-zero value and the port value in setting if successful, 0 otherwise.

Notes

The interrupt buffer is 128 bytes deep. If length is set to more than 128, the routine will hang and wait for the specified number of characters to be read from the port before returning.

*int QuickUsbWriteRS232 (HANDLE hDevice, unsigned char portNum, char *outputString, unsigned short length)*

Purpose

Write a block of characters to the interrupt transmit buffer of the specified QuickUSB serial port.

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
portNum – The serial port number. Serial port 0 (P1) = 0, serial port 1 (P2) = 1.
length – The number of characters to read.

Returns

A non-zero value if successful, 0 otherwise.

Notes

The interrupt buffer is 128 bytes deep. If length is set to more than 128, the routine will hang and wait for the specified number of characters to be sent to the port before returning.

I^{2}C-Compatible Port

The QuickUSB I^{2}C-compatible port is a master-only bus controller. The bus runs at approximately 100Khz. Address 1 is reserved.

*int QuickUsbReadI2c (HANDLE hDevice, unsigned short address, char *data, unsigned short length)*

Purpose

None

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The device address.
data – A pointer to a buffer in which to place the data.
length – The length of the data buffer in bytes. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise. Also, length is set the actual number of bytes read.

Notes

None

*int QuickUsbWriteI2c(HANDLE hDevice, unsigned short address, char *data, unsigned short length)*

Purpose

None

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
address – The device address.
data – A pointer to the data to send.
length – The length of the data buffer in bytes. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

SPI-Compatible Port

The QuickUSB module implements a ‘soft’ SPI port using pins on port E and optionally port A. These routines support from 1 to 10 devices with individual slave select lines for each device. The slave select lines are active low and are shown in the ‘QuickUSB Target Interface’ document found on www.quickusb.com or in the Doc directory of the QuickUSB Library. The signals are MOSI, SCK, MISO and nSS0-9. Data is shifted in and out LSbit to Msbit. This bus runs at approximately 500Kbps.

*int QuickUsbReadSpi (HANDLE hDevice, char *data, unsigned short *length)*

Purpose

None

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
data – A pointer to a buffer in which to place the received data.
length – A pointer to the number of bytes to read. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise. Also the data buffer is fulled with the received data and the length is set to the number of bytes actually received on success. Both data and length are left unchanged if the function failed.

Notes

None

*int QuickUsbWriteSpi (HANDLE hDevice, char *data, unsigned short length)*

Purpose

None

Parameters

hDevice – A HANDLE which was returned from a call to QuickUsbOpen.
data – A pointer to the data to send
length – The number of bytes to send. The maximum length is 64 bytes.

Returns

A non-zero value if successful, 0 otherwise.

Notes

None

Examples

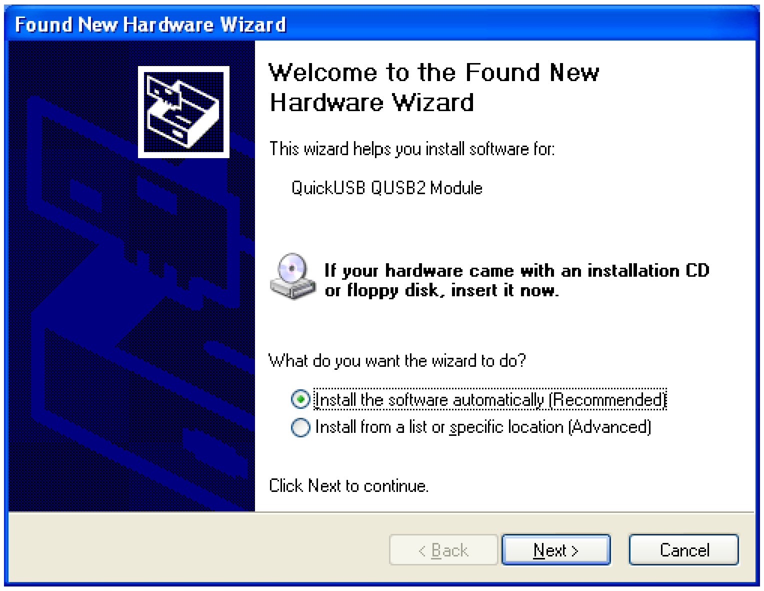
Appendix A

Install the QuickUSB Library

The QuickUSB Library is a set of software development libraries and programs that enable you to use and write software to use the QuickUSB module. To install the QuickUSB Library, Insert the QuickUSB Library CD into the CD drive of the target computer and double-click the Setup.exe file. This will launch the installer program that will install the QuickUSB Library onto your computer.

Install the QuickUSB Driver

Next, connect the QuickUSB module to your computer's USB connector. You should see an installation message on the computer screen shortly after connecting the QuickUSB module. The following screen shots are from Windows XP. The dialog boxes you get may look different if you are using a different operating system.


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

QuickUSB QUSB2 Module

If your hardware came with an installation CD or floppy disk, insert it now.

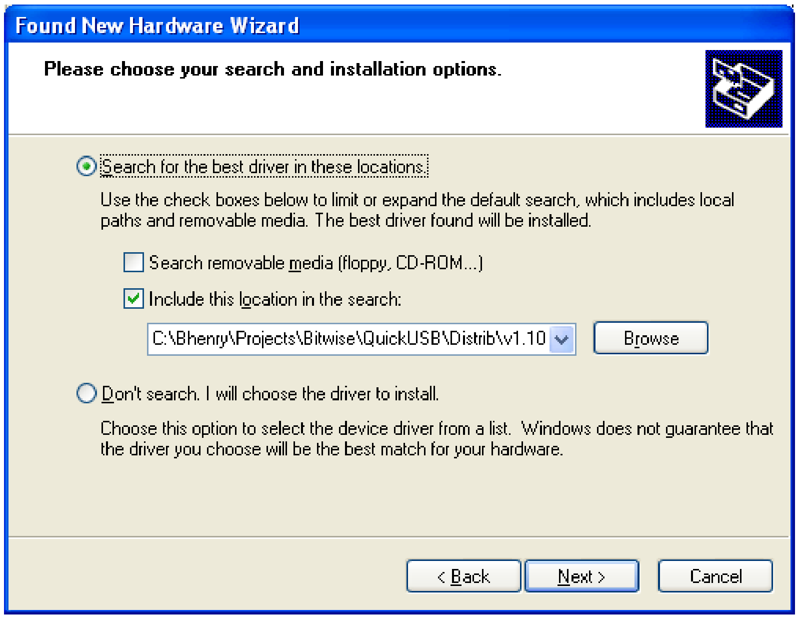
What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

Figure 3 - New Hardware Wizard

Select the lower item 'Install from a list or specific location (Advanced). Select 'Next'.


Found New Hardware Wizard

Please choose your search and installation options.

(●) Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

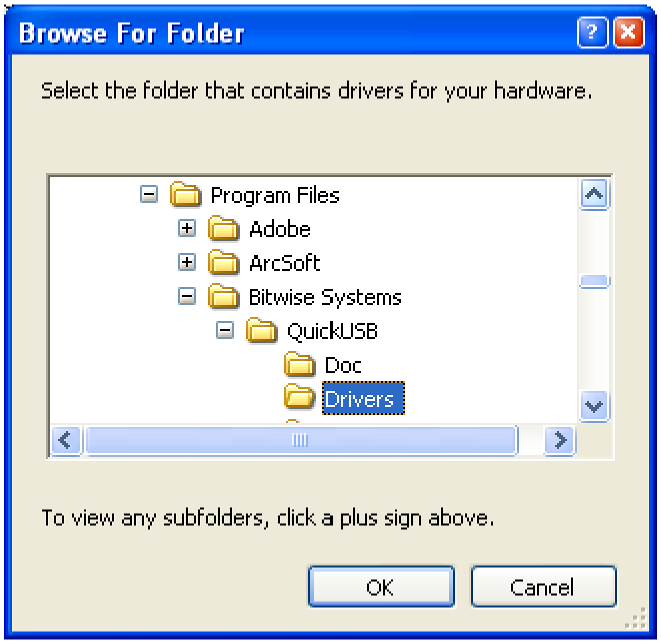
C:\Bhenry\Projects\Bitwise\QuickUSB\Distrib\v1.10

( ) Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Figure 4 - Dialog box to select the QuickUSB INF file location

Next, select 'Search for the best driver in these locations', then check 'Include this location in the search'. Select the 'Browse' button and select the folder "C:\Program Files\Bitwise Systems\QuickUSB"

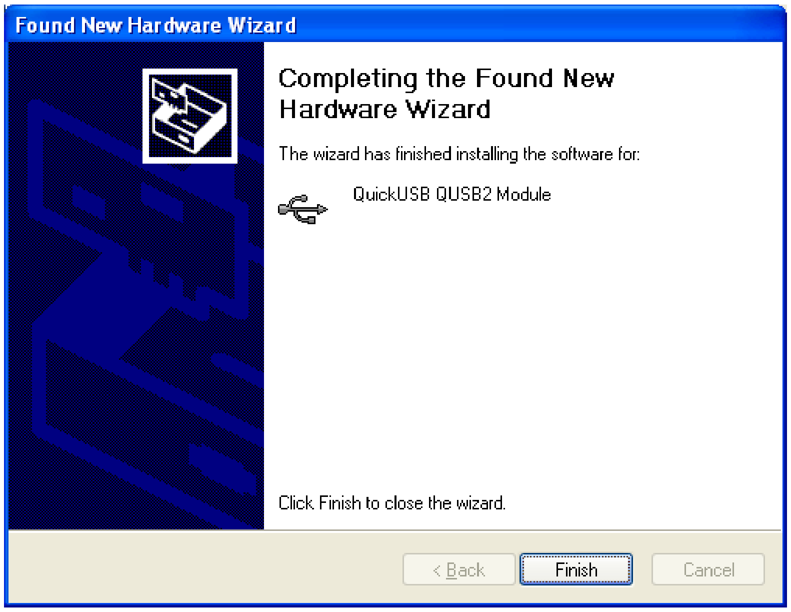

Browse For Folder

Select the folder that contains drivers for your hardware.

To view any subfolders, click a plus sign above.

Figure 5 - Browse for folder Dialog Box

When you select the correct folder the browse dialog box will look like this.



Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

QuickUSB QUSB2 Module

Click Finish to close the wizard.

Figure 6 - Found new hardware wizard completion dialog box

When the QuickUSB driver is installation is complete, you will see the following dialog box. Once the driver installation is complete, you can run the QuickUSB Library demonstration software.