MAC USB SOFTWARE REFERENCE

Purpose of this document:

This document outlines the structure and basic use of ACCES software support for its USB products on the Macintosh platform.

About the software:

The software produced by ACCES was developed under Mac OS X. At the time of this writing it was running on Version 10.4.10. The software runs on both Intel and PPC based Macs. The software is provided as three X-Code projects. These projects are AccesLoader, aiousb, and carbonUSBsample.

AccesLoader:

The primary purpose of AccesLoader is to demonstrate the loading of firmware on ACCES USB devices. Most ACCES USB products use a Cypress FX-2 chip. When the device is plugged into the computer it contains only enough information to tell the system what type of product it is. The firmware can then be loaded onto the card, and then the card is instructed to reenumerate.

AccesLoader will search the system for any ACCES USB devices that do not have their firmware loaded. If it finds any such devices, it will program the first one it finds and exits. If there are no devices attached it will wait for a device to be attached. Once that device is attached, it will be programmed and the program will exit.

aiousb:

Aiousb is a library that can be used with other X-Code projects to make use of ACCES USB products. Full source is provided. The library encapsulates the vendor requests used by ACCES USB devices. The functions exported by the library are discussed in detail here.

Sint8 queryDevices(void)

This function will check check for connected ACCES USB devices. This will only find devices that have already had their firmware uploaded and gone through the reenumeration process. The return value is the number of devices found. Valid device indexes are from 0 to 1 minus the return value. If the function returns 2, then valid device indexes are 0 and 1. Returns 0 or greater on success.

SInt8 queryDeviceInfo(SInt8 deviceIndex, UInt16 *productID, UInt16 nameSize, char *name, UInt8 *DIOBytes, UInt *counters)

This function will retrieve information about the device specified by deviceIndex.

SInt8 dioConfig (SInt8 deviceIndex, bool tristate, UInt8 outMask, UInt8 *pBytes)

This function is used to configure the direction of the ports on the USB-DIO family of products. The tristate parameter indicates weather or not the port should be made active after the configuration is complete. A true value means the ports are all tristated and will not be

active. The outMask parameter determines which ports are input and which are output. The least significant bit corresponds to port 0. If a bit is 0 then that port is input, and if it is 1 then that port is output. Set any bits that do not have a corresponding port to 0.

The pBytes parameter is a pointer to a buffer that holds information on what to be output on ports configured as output. Obviously, the data in this buffer is only relevant if the port is configured for output and the tristate parameter is false, but the buffer still must have a byte for every port.

SInt8 DIO_WriteAll (SInt8 deviceIndex, UInt8 *pData)

This function will write the data pointed to by pData to the card. The buffer must contain 1 byte for every possible output port on the card. This means that if you are using the USB-DIO-32 with only 1 port set to output, you must still send a 4 byte buffer. Any bytes corresponding to input ports on the card will simply be ignored by the device.

SInt8 DIO_Write8 (SInt8 deviceIndex, UInt8 byteIndex, UInt8 *data)

This function will write the byte pointed to by data to the port specified by byteIndex. If the specified port is set for input the data will be ignored by the device.

SInt8 DIO_Write1 (SInt8 deviceIndex, UInt8 bitIndex, bool *data)

This function will set a single bit on the device. The bitIndex is based on all the possible ports of the card. This means that a bit index of 8 will actually be bit 0 on port 1. If the value pointed to by data is true the bit will be set to 1, and if it is false the bit will be set to 0. If the bitIndex refers to a bit configured as input the write to the card will be ignored.

SInt8 DIO_ReadAll (SInt8 deviceIndex, UInt8 *pData)

This function will read all the ports available on a card. Any ports that are output will be read back. The pData parameter must point to a buffer of at least one byte per port on the device.

SInt8 DIO_Read8 (SInt8 deviceIndex, UInt8 *pData)

This function will read a single port from the card. If the port is configured for output then this function will act as a read back for that port.

SInt DIO_Read1 (SInt8 deviceIndex, UInt8 bitIndex, bool *data)

This function will read a single bit back from the device. The bitIndex is based on all the possible ports of the card. This means that a bitIndex of 8 will acutally be bit 0 on port 1. If the bitIndex refers to a port configured as output this will function as a read back.

SInt CTR_8254Mode (SInt8 deviceIndex, UInt8 blockIndex, UInt8 counterIndex, UInt8 mode)

This function will set the mode for one of the counters on the device.[1] The blockIndex specifies which 8254 chip is being referred to. The counterIndex refers to the counter within that chip that is being programmed. Valid values for counterIndex are 0, 1, and 2. Valid values for the mode are 0 through 5 inclusive. For more information on the 8254 chip please refer to your product's manual.

SInt8 CTR_8254ModeLoad (SInt8 deviceIndex, UInt8 blockIndex, UInt8 counterIndex, UInt8 mode, UInt16 value)

This function will set the mode and load a count value for one of the counters on the device. The blockIndex specifies which 8254 chip is being referred to. The counterIndex refers to the counter within that chip that is being programmed. Valid values for counterIndex are 0, 1, and 2. Valid values for the mode are 0 through 5 inclusive. For more information on the 8254 chip please refer to your product's manual.

SInt8 CTR_8254Read(SInt8 deviceIndex, UInt8 blockIndex, UInt8 counterIndex, UInt16 *value)

This function will read the current value from one of the counters on the device. The blockIndex specifies which 8254 chip is being referred to. The counterIndex refers to the counter within that chip that is being read. Valid values for counterIndex are 0, 1, and 2. Valid values for the mode are 0 through 5 inclusive. For more information on the 8254 chip please refer to your product's manual.

SInt8 CTR_8254ReadAll (SInt8 deviceIndex, UInt16 *data)

This function will read the current value from all counters on the device. The data buffer must be at least 2 * counters bytes long. Keep in mind that each 8254 counter chip contains three individual counters.

SInt8 EEPromWrite (SInt8 deviceIndex, UInt16 address, UInt8 *data, SInt8 length)

This function will write data to the user accessible region of the EEPROM. The user accessible addresses are from 1E00(h) to 2000(h) inclusive. Attempts to write to an address outside this range will result in an error return value. The address protection is a function of the library and not of the device. If you write your own USB code, be sure to write only within this address space. It is possible to permanently ruin your device by writing to the wrong address.

SInt8 EEPromRead (SInt8 deviceIndex, UInt16 address, UInt8 *data, SInt8 length)

This function will read data from the user accessible region of the EEPROM. The user accessible addresses are from 1E00(h) to 2000(h) inclusive. Attempts to read from an address outside this range will result in an error return value. The address protection is a function of the library and not the device. If you write your own USB code, be sure to read only within this address space.

SInt8 DacWriteOne (SInt8 deviceIndex, UInt16 data, SInt8 channel)

This function will write the value in data to the selected channel.

1 Each 8254 counter chip actually contains 3 counters. In the case of the USB-DIO-32 the counters come with a pre-wired scheme that is shown in the product manual's block diagram.

SInt8 DacWriteAll(SInt8 deviceIndex, UInt8 *data)

This function will write to multiple DAC channels. The data buffer must be at least 17 bytes long. The first byte is a bit mask indicating which channels should be updated. The least significant bit corresponds to the first channel, and a 1 indicates the channel should be updated. The rest of the buffer is the data to be written. Keep in mind that each channel requires 2 bytes for its value. Data for channels that are not being updated is ignored.

SInt8 CTR_8254ReadAllLatchedCounts (SInt8 deviceIndex, UInt8 *data)

This function is used for frequency measurement. The "Vendor Requests.pdf" file contains a more in depth discussion of the use of this function. The 31st byte of data will contain the number of times the board read frequency data since the last time this function was called. Ideally this value will always be 1. This function is intended to be used only with the USB-CTR-15

SInt8 CTR_8254SelectGateSource (SInt8 deviceIndex, int counter)

This function is used for frequency measurement. It selects the counter to be used as a gate source, and it is intended to be used only with the USB-CTR-15.

carbonUSBsample:

This sample demonstrates how to call aiousb functions in a GUI program. It has limited functionality.

Tips:

- All the boards with any sort of digital input and output use the DIO functions. Even the cards that use isolated inputs and relay outputs instead of flexible DIO.
- The output bytes for the USB-IIRO-16 series start at 0 with the input bytes starting at 2.
- Except for queryDevices all functions return 0 on success. The queryDevices function returns 0 or greater.