Overview

Often times software will need to be able to distinguish between two QuickUSB Devices. When QuickUSB devices are connected to a computer, the QuickUSB Driver and Library assign each device a unique name such as 'QUSB-0' and 'QUSB-1', but these names do not reveal any information unique to each device. If the devices are connected to the computer in the reverse order then their names flip. This application note describes how you can acquire information about QuickUSB devices that is unique to each device so that your software may always know which QuickUSB to communicate with when multiple devices are connected to a computer. Additionally, this application note shows how you can retrieve specific information about the capabilities of the connected device.

Serial Number

The easiest way to distinguish multiple QuickUSB devices is to read out their serial numbers. Every QuickUSB device has a serial number, and if you are using a QuickUSB Module or have configured your hardware with iChipPack licenses then your device has been assigned a globally unique serial number. This serial number may be changed using the QuickUSB Customizer.

Reading the serial number from all connected devices is quite simple. First you call QuickUsbFindModules() to get a list of all QuickUSB devices currently connected to the system. Next, you loop through each device, open it, read out its serial number, and then close it. You may see an example of how to do this under the Reading Serial Numbers section of this document.

Custom EEPROM Data

Using only serial numbers to distinguish between QuickUSB devices has its limits. For example, if you release multiple revisions or variants of a product that uses QuickUSB, or even multiple products that use QuickUSB, then how can your software determine what QuickUSB product it is communicating with, its hardware revision, and the devices capabilities? The answer is to use EEPROM memory in QuickUSB to store information about your product. Every QuickUSB device, whether it is a QuickUSB Module or a custom design that incorporates the QuickUSB circuit, has an EEPROM that stores the QuickUSB firmware. The upper 2 KB of that EEPROM are reserved for user storage and are accessible with the QuickUSB Storage API functions QuickUsbReadStorage() and QuickUsbWriteStorage().

To uniquely identify information about your product you must first design a memory structure that contains information about any type of product you may design and its capabilities. Useful information could include a product ID, a revision number, and hardware capability information. You then store this information in the EEPROM of each QuickUSB product you design. Then, identifying products in software is easy. First, you must call QuickUsbFindModules() to get a list of all QuickUSB devices currently connected to the system. Next, you loop through each device, open it, read out the product information memory structure from the EEPROM, and parse it for relevant information about the product. Of course, when you are done communicating with the device make sure to close it. You may see an example of how to do this under the Accessing Custom EEPROM Data section of this document.

Examples

Reading Serial Numbers

This C++ example demonstrates how to display the serial number of every QuickUSB device connected to the computer.

```
#include "QuickUSB.h"
#include <iostream>

using namespace std;

// Main Program
int main(int argc, char **argv) {
    QCHAR nameList[1024], *name;
    QCHAR serial[32];
    QHANDLE hDevice;
    QRESULT ok;
    QULONG lastError;

    // Get a list of the connected QuickUSB devices
    ok = QuickUsbFindModules(nameList, 1024);
    if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
    }

    // Loop though all devices
    name = nameList;
    do {
        // Open the device
        ok = QuickUsbOpen(&hDevice, name);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Read the device's serial number
        ok = QuickUsbGetStringDescriptor(hDevice, QUICKUSB_SERIAL, serial, 32);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Write the device information to the console
        cout << name << ": " << serial << endl;

        // Close the device
        ok = QuickUsbClose(hDevice);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Move to the next device in the list
        name += strlen(name) + 1;
    } while (*name != 0);

    // Return successfully
    return 0;

}
```

Accessing Custom EEPROM Data

This C++ example demonstrates how one might create a product information memory structure and read it out of the QuickUSB EEPROM to identify QuickUSB devices. Keep in mind that uninitialized or blank EEPROM data contains the value of 0xFF, not 0x00.

In order for this code to produce meaningful results, you will have to first fill out and then write a *ProductInfo* object to the EEPROM of your QuickUSB devices. This example is broken into two portions: the first is how to write product information to the EEPROM and the second is how to retrieve that product information.

Writing Product Information from the EEPROM

```
#include "QuickUSB.h"
#include <iostream>

using namespace std;

// The Product Information Memory Structure
struct ProductInfo {
    QWORD productID;
    QBYTE hardwareRev;
    QULONG capabilites;
    // ...additional information
    QBYTE checksum;
};

// Main Program
int main(int argc, char **argv) {
    QCHAR nameList[1024], *name;
    QCHAR serial[32];
    QHANDLE hDevice;
    QRESULT ok;
    QULONG lastError;
    ProductInfo info;

    // Get a list of the connected QuickUSB devices
    ok = QuickUsbFindModules(nameList, 1024);
    if (!ok) {
        ok = QuickUsbGetLastError(&lastError);
        cout << "Error: " << lastError << endl;
        return 1;
    }

    // Loop though all devices
    name = nameList;
    do {
        // Open the device
        ok = QuickUsbOpen(&hDevice, name);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Fill out the product information for this device
        info.productID = 1;     // Our first product
        info.hardwareRev = 2;   // Rev 2
        info.capabilites = 0;   // No capabilities yet
        info.checksum = 0xAB;   // For brevity we will not calculate a checksum

        // Write the device's product information
        ok = QuickUsbWriteStorage(hDevice, 0, (PQBYTE)&info, sizeof(ProductInfo));
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
```

```
        }

        // Close the device
        ok = QuickUsbClose(hDevice);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Move to the next device in the list
        name += strlen(name) + 1;
    } while (*name != 0);

    // Return successfully
    return 0;

}
```

Reading Product Information to the EEPROM

```
#include "QuickUSB.h"
#include <iostream>

using namespace std;

// The Product Information Memory Structure
struct ProductInfo {
    QWORD productID;
    QBYTE hardwareRev;
    QULONG capabilites;
    // ...additional information
    QBYTE checksum;
};

// Main Program
int main(int argc, char **argv) {
    QCHAR nameList[1024], *name;
    QCHAR serial[32];
    QHANDLE hDevice;
    QRESULT ok;
    QULONG lastError;
    ProductInfo info;

    // Get a list of the connected QuickUSB devices
    ok = QuickUsbFindModules(nameList, 1024);
    if (!ok) {
        ok = QuickUsbGetLastError(&lastError);
        cout << "Error: " << lastError << endl;
        return 1;
    }

    // Loop though all devices
    name = nameList;
    do {
        // Open the device
        ok = QuickUsbOpen(&hDevice, name);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Read the device's product information
        ok = QuickUsbReadStorage(hDevice, 0, (PQBYTE)&info, sizeof(ProductInfo));
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }
```

```
        // Here we could verify the checksum to ensure that the memory contains
        // valid product information, but for clarity we will skip that step

        // Print out the product ID, revision, and capabilities
        cout << name << ": " << info.productID << ", " << (int)info.hardwareRev << ", " <<
info.capabilites << endl;

        // Close the device
        ok = QuickUsbClose(hDevice);
        if (!ok) {
            ok = QuickUsbGetLastError(&lastError);
            cout << "Error: " << lastError << endl;
            return 1;
        }

        // Move to the next device in the list
        name += strlen(name) + 1;
    } while (*name != 0);

    // Return successfully
    return 0;

}
```

Bitwise Systems
6489 Calle Real, Suite E
Goleta, CA 93117
V (805) 683-6469
F (805) 683-4833
info@bitwisesys.com