

```

// USBTimer.CPP -- test program for measuring Windows latency to USB interrupts
// Works in cooperation with USB Timer firmware
//
// Copyright (C) 2001, Intel Corporation
// All rights reserved.
// Permission is hereby granted to merge this program code with other program
// material to create a derivative work. This derivative work may be distributed
// in compiled object form only. Any other publication of this program, in any form,
// without the explicit permission of the copyright holder is prohibited.
//
// Send questions and comments to John.Hyde@intel.com

#include "stdafx.h"
#include "winioctl.h"
#include "objbase.h"
#include <iostream.h>

extern "C" {
// Declare the C libraries used
#include "setupapi.h"
#include "hidsdi.h"
}

// Need a global variable
HANDLE HIDHandle;

bool OpenUSBinterface() {
    struct _GUID GUID;
    SP_INTERFACE_DEVICE_DATA DeviceInterfaceData;
    struct {DWORD cbSize; char DevicePath[256];} FunctionClassDeviceData;
    int Device, i;
    HANDLE PnPHandle;
    char buffer[256];
    unsigned long BytesReturned;
    bool Success, Openned;
    const char *DeviceName = "Timer";
    SECURITY_ATTRIBUTES SecurityAttributes;

// Initialize the GUID array and setup the security attributes for Win2000
    HidD_GetHidGuid(&GUID);
    SecurityAttributes.nLength = sizeof(SEcurity_ATTRIBUTES);
    SecurityAttributes.lpSecurityDescriptor = NULL;
    SecurityAttributes.bInheritHandle = false;

// Get a handle for the Plug and Play node and request currently active devices
    PnPHandle = SetupDiGetClassDevs(&GUID, NULL, NULL, DIGCF_PRESENT|DIGCF_INTERFACEDevice);
    if (int(PnPHandle) == -1) { printf("Could not attach to PnP node"); return false; }
// Lets look for a maximum of 20 Devices
    HIDHandle = INVALID_HANDLE_VALUE; Openned = false;
    for (Device = 0; (Device < 20) && !Openned; Device++) {
// Initialize our data
        DeviceInterfaceData.cbSize = sizeof(DeviceInterfaceData);
// Is there a device at this table entry
        Success = SetupDiEnumDeviceInterfaces(PnPHandle, NULL, &GUID, Device, &DeviceInterfaceData);
        if (Success) {
// There is a device here, get it's name
            FunctionClassDeviceData.cbSize = 5;
            Success = SetupDiGetDeviceInterfaceDetail(PnPHandle, &DeviceInterfaceData,
                (PSP_INTERFACE_DEVICE_DETAIL_DATA)&FunctionClassDeviceData, 256, &BytesReturned,
                NULL);
            if (!Success) { printf("Could not find the system name for this Device\n"); return
                false; }
// Can now open this Device
            HIDHandle = CreateFile(FunctionClassDeviceData.DevicePath, GENERIC_READ | GENERIC_WRITE,
                FILE_SHARE_READ | FILE_SHARE_WRITE, &SecurityAttributes, OPEN_EXISTING, 0, NULL);
            if (HIDHandle == INVALID_HANDLE_VALUE) printf("Could not open HID device\n");
            else {
// Is it OUR Device?
                HidD_GetProductString(HIDHandle, buffer, sizeof(buffer));
// Compare incoming string with UNICODE string
                Openned = true; i = 0;
                while (DeviceName[i] != 0) { if (buffer[2*i] != DeviceName[i]) {Openned = false;}
                    i++;}
                printf("HID Interface %sfound\n", Openned ? "" : "not ");
                if (!Openned) CloseHandle(HIDHandle);
            }
        } // if (SetupDiEnumDeviceInterfaces . . .
    } // for (Device = 0; (Device < 20); Device++)
    SetupDiDestroyDeviceInfoList(PnPHandle);
}

```

```

        return Openned;
    }

int main(int argc, char* argv[]) {
    char InputCharacter;
    UCHAR ReplyBuffer[3];
    INT ResultsArray[1024];
    int Choice, i, Index, OverArray;
    unsigned long BytesRead;

    // Say Hello to the user
    printf("USB Design By Example: Measure the latency of USB interrupts\n\n");
    if (!OpenUSBinterface()) {
        printf("\nERROR EXIT\n");
        cin >> InputCharacter;
        return -1;
    }

    do {
        // First clear the results array
        for (i=0; i<1024; i++) ResultsArray[i] = 0;
        OverArray = 0;

        // Collect the data
        for (i=0; i<10000; i++) {
            if (!ReadFile(HIDHandle, ReplyBuffer, sizeof(ReplyBuffer), &BytesRead, NULL))
                { printf("Error reading from HID device\n"); return -2; }
            Index = ReplyBuffer[1] + (256 * ReplyBuffer[2]);
            if (Index > 1024) OverArray++;
            else ResultsArray[Index]++;
            if (i%100 == 0) printf("."); // Give some user feedback
        }

        // Display the results
        printf("OverArray = %d\n", OverArray);
        for (i=0; i<1024; i++) if (ResultsArray[i] != 0) printf("I = %d, Value = %d\n", i,
ResultsArray[i]);
        // Do we collect more data?
        printf("Run the test again?");
        cin >> InputCharacter;
        Choice = InputCharacter;
        if (tolower(Choice) == 'n') { CloseHandle(HIDHandle); return 0; }
        } while(1);

    } // Main

```