//--------------------------------------------------------------------

// TUCN, Computer Science Department

// Input/Output Systems and Peripheral Devices

//--------------------------------------------------------------------

// http://users.utcluj.ro/~baruch/en/pages/teaching/inputoutput-systems/laboratory.php

//--------------------------------------------------------------------

// File: AppScroll-e.cpp

// Date: 10.10.2015

// Modified: 21.11.2015

//--------------------------------------------------------------------

// IOS application example with vertical scroll bar

//--------------------------------------------------------------------

#include <iostream>

#include <cstdio>

#include <windows.h>

#include "Hw.h"

#include "PCI-e.h"

#include "PCI-vendor-dev.h"

using namespace std;

#define NLIN 500 // number of lines in the display window

#define NCOL 240 // number of columns in the display window

// Global variables

char szBuffer[NLIN][NCOL]; // buffer for the window contents

int cLine; // number of current line in the display buffer

// Declarations of external functions

void DisplWindow(HWND hWnd);

DWORD64 qvFuncAddr;

PPCI\_CONFIG0 returnsPointer(WORD busNumber, WORD deviceNumber, WORD PCIeFunctionNumber){

DWORD64 qwAdrDisp = PCI\_CONFIG\_START;

qwAdrDisp = qwAdrDisp | (busNumber << 20) | (deviceNumber << 15) | (PCIeFunctionNumber << 12);

return (PPCI\_CONFIG0)qwAdrDisp;

}

DWORD readDWORD(byte bus\_number, byte device\_number, byte function\_number, byte dw\_number) {

DWORD data = ((1 << 31) | (bus\_number << 16) | (device\_number << 11) | (function\_number << 8) | (dw\_number << 2));

\_\_outpd(PCI\_CONFIG\_ADR, data);

DWORD readData = \_\_inpd(PCI\_CONFIG\_DATA);

return readData;

}

void returnsInfo(BYTE baseClass, BYTE subClass, BYTE progInterface){

for (int i = 0; i < PCI\_CLASS\_TABLE\_LEN; i++)

{

if (PciClassTable[i].Class == baseClass && PciClassTable[i].SubClass == subClass

&& PciClassTable[i].ProgIf == progInterface)

{

wsprintf(szBuffer[cLine++], "class subclass descriptor = %s progInterface descriptor = %X \n\n",

PciClassTable[i].ClassDesc, PciClassTable[i].ProgIfDesc);

freopen("output.txt", "w", stdout);

cout << PciClassTable[i].ProgIfDesc;

}

}

}

void returnsIds(DWORD vendorId, DWORD deviceId){

for (int i = 0; i < PCI\_VENTABLE\_LEN; i++)

if (PciVenTable[i].VenId == vendorId)

wsprintf(szBuffer[cLine++], "vendor id = %d vendor descriptor = %s ",

PciVenTable[i].VenId, PciVenTable[i].VenFull);

for (int i = 0; i < PCI\_DEVTABLE\_LEN; i++)

if (PciDevTable[i].VenId == vendorId && PciDevTable[i].DevId == deviceId)

wsprintf(szBuffer[cLine++], "device id = %d device descriptor = %s ",

PciDevTable[i].DevId, PciDevTable[i].ChipDesc);

}

//--------------------------------------------------------------------

// Function AppScroll

//--------------------------------------------------------------------

//

// Function: IOS application example with vertical scroll bar

//

// Parameters: - Handle to the application window

//

// Returns: 0 - Operation completed successfully

// 1 - Error at initializing the Hw driver

//

//--------------------------------------------------------------------

int AppScroll(HWND hWnd)

{

int i;

char szMes0[] = "Error initializing the Hw driver";

char szMes1[] = "IOS Application";

// Initialize the Hw library

if (!HwOpen()) {

wsprintf(szBuffer[0], szMes0);

MessageBox(NULL, szBuffer[0], "HwOpen", MB\_ICONSTOP);

return 1;

}

// Erase the display buffer and the window contents

for (i = 0; i < NLIN; i++) {

memset(szBuffer[i], ' ', NCOL);

}

cLine = 1;

// Copy the start message into the display buffer and display the message

wsprintf(szBuffer[cLine], szMes1);

cLine += 2;

DisplWindow(hWnd);

//--------------------------------------------------------------------

// To be completed with the application's code

//--------------------------------------------------------------------

for (int i = 0; i < 15; i++)

for (int j = 0; j < 31; j++)

for (int k = 0; k < 7; k++)

{

///2.7.2, 2.7.3

PPCI\_CONFIG0 pRegPci = returnsPointer(i, j, k);

WORD wVendorID = \_inmw((DWORD\_PTR)&pRegPci->VendorID);

if (wVendorID == 0xFFFF)

continue;

wsprintf(szBuffer[cLine++], "2.7.3\n", i, j, k);

wsprintf(szBuffer[cLine++], "bus = %d device = %d function = %d", i, j, k);

DWORD baseClass = \_inmw((DWORD\_PTR)&pRegPci->BaseClass);

DWORD subClass = \_inmw((DWORD\_PTR)&pRegPci->SubClass);

DWORD progInterface = \_inmw((DWORD\_PTR)&pRegPci->ProgInterface);

DWORD subSystVendorID = \_inmw((DWORD\_PTR)&pRegPci->SubSystVendorID);

DWORD deviceId = \_inmw((DWORD\_PTR)&pRegPci->BaseClass);

returnsInfo(baseClass, subClass, progInterface);

returnsIds(wVendorID, deviceId);

wsprintf(szBuffer[cLine++], "baseclass = %d subclass = %d progInterface = %d, subsystVendorID = %d",

baseClass, subClass, progInterface, subSystVendorID);

///2.7.4

wsprintf(szBuffer[cLine++], "\n2.7.4\n");

WORD vendor\_id = LOWORD(readDWORD(i, j, k, 0));

WORD device\_id = HIWORD(readDWORD(i, j, k, 0));

if (vendor\_id != 0xFFFF){

wsprintf(szBuffer[cLine++], "bus = %d device = %d function = %d", i, j, k);

wsprintf(szBuffer[cLine++], "class: %x\n", (HIWORD(readDWORD(i, j, k, 0x02)) << 7) | HIBYTE(LOWORD(i, j, k, 0x02)));

wsprintf(szBuffer[cLine++], "vendorId = %d\n", vendor\_id);

for (int index = 0; index < PCI\_VENTABLE\_LEN; index++) {

if (PciVenTable[index].VenId == vendor\_id) {

wsprintf(szBuffer[cLine++], "vendor information: %s\n", PciVenTable[index].VenFull);

}

}

for (int index = 0; index < PCI\_DEVTABLE\_LEN; index++) {

if (PciDevTable[index].DevId == device\_id) {

wsprintf(szBuffer[cLine++], "chip descriptor: %s\n", PciDevTable[index].ChipDesc);

}

}

}

wsprintf(szBuffer[cLine++], "--------------------------------------------------");

wsprintf(szBuffer[cLine++], "\n\n\n\n\n");

}

// Display the messages

DisplWindow(hWnd);

HwClose();

return 0;

}