LVDS Controller Windows Device Driver Package

1 Introduction

1.1 The WinSystems PPM-C407 full-featured, high-performance single board computer offers a single Low-Voltage Differential Signaling (LVDS) video interface. LVDS panel brightness can be controlled with a backlight enable signal and a PWM brightness control signal. The provided driver and DLL allow user control of the panel backlight under the Windows 7 and 10 operating systems.

1.2 The package utilizes the Intel® Atom™ Processor E3800 Windows 7 I^2C Driver. The driver files are provided in this package. The driver name is *iaioi2c.sys*.

1.3 The LVDS Controller driver package is designed for and has been verified with 32-bit and 64-bit versions of Microsoft Windows 7 and 10.

2 Installation

2.1 Before using the DLL, the Intel I^2C driver (*iaioi2c.sys*) may have to be installed on the SBC. Open the Windows Device Manager to see if the devices are already installed. They will be under the System Devices item.

2.2 The driver, support files, and console applications are supplied in a zip file. The following files are included:

- iaioi2c.sys – Windows Driver file
- iaioi2c.inf - Windows installation file
- iaioi2c.cat - Windows catalog file
- WdfCoinstaller01011.dll – Windows co-installer
- lvdsControllerDLL.dll – Windows DLL file
- lvdsControllerDLL.lib – Windows library file
- lvdsControllerDLL.h – Windows include file
- panel.cpp – Windows application source
- panel.exe – Windows application
- vcredist_x86 or vcredist_x64 - Microsoft Visual C++ Redistributable

2.3 Installation is accomplished via the ‘Add legacy hardware’ selection found in the Action menu of the Windows Device Manager. Navigate to the drive and folder containing the driver files and select *iaioi2c.inf*. The Windows installer will copy the *iaioi2c.sys* driver file to the appropriate directory in the Windows installation.

2.4 In Device Manager, three Intel Atom/Celeron/Pentium Processor I2C Controller devices should appear under the System Devices item. A reboot may be required after installation.

2.5 The included console application, *panel.exe, c*an be used to verify driver installation and functionality. Usage of the programs is described later in this document. These applications require an LVDS panel with the proper cable connected to J5 and J500.

3 Driver Overview and Architecture

3.1 The file *lvdsControllerDLL.dll* is a Windows Dynamic-Link Library (DLL) which facilitates access to the underlying hardware through the Intel I^{2}C driver.

3.2 The driver utilizes the I/O Control (IOCTL) Request framework to control the register set of the LVDS Bridge.

4 Driver Usage

4.1 The *lvdsControllerDLL.h* file included in the driver distribution contains the function definitions to be used by an application to communicate with the *iaioi2c.sys* driver. This file is included in Appendix A: lvdsControllerDLL.h.

4.1 An application calls the function *InitializeSession* to open the driver. This is required before any of the other functions can be called. The following example opens the Intel I^{2}C driver. If a zero is returned, then the driver has been successfully initialized. Any other returned value indicates that an error has occurred and the device is unusable.

```
if (InitializeSession())
      printf(“Error opening device.\n”);
```

4.3 Once the driver is initialized, the other functions can be used to control the LVDS panel brightness. Following is a description and sample code for each function. For all functions, if a zero is returned, the function was successful. Any other returned value indicates that an error or board condition prevented function completion.

4.3.1 int blOnOff(unsigned int blControl)

This function enables or disables the LVDS panel.

```
if (blOnOff(BL_ON))
       printf("\nError turning backlight on.\n\n");
```

4.3.2 int blSetFrequency(unsigned int blFrequency)

This function sets the desired frequency of the backlight PWM signal. The *blFrequency* variable must be one of the preconfigured values found in the enum variable of the same name.

```
if (blSetFrequency(BL_FREQ_225))
    printf("Error setting frequency.\n");
```

Valid options for *blFrequency* are:

- BL_FREQ_500 → 500Hz
- BL_FREQ_400 → 400 Hz
- BL_FREQ_300 → 300 Hz
- BL_FREQ_275 → 275 Hz
- BL_FREQ_250 → 250 Hz
- BL_FREQ_225 → 225 Hz
- BL_FREQ_200 → 200 Hz
- BL_FREQ_100 → 100 Hz

4.3.3 int blSetDutyCycle(unsigned int blDutyCycle)

This function sets the desired duty cycle of the backlight PWM signal. The *blDutyCycle* variable must be one of the preconfigured values found in the enum variable of the same name.

```
if (blSetDutyCycle(BL_DUTY_75))
    printf("Error setting duty cycle.\n");
```

Valid options for *blDutyCycle* are:

- BL_DUTY_100 → 100%
- BL_DUTY_87_5 → 87.5%
- BL_DUTY_75 → 75%
- BL_DUTY_62_5 → 62.5%
- BL_DUTY_50 → 50%
- BL_DUTY_37_5 → 37.5%
- BL_DUTY_25 → 25%
- BL_DUTY_12_5 → 12.5%
- BL_DUTY_0 → 0%

The duty cycle settings are dependent on the frequency selected. As a result, **the duty cycle should always be set after any frequency change**.

4.3.4 int CloseSession(void)

This function is used to close the Intel I^{2}C driver. If a zero is returned, the driver is closed. Otherwise there was an error and the driver is still open.

```
if (CloseSession())
```

```
		printf(“Error closing driver\n”);
```

4.4 Every function returns a zero or a positive integer value indicating success or failure. If a zero is returned, the function has completed successfully. If a failure occurs, the specific value returned provides more clarity as to the failure mechanism.

4.4.1 DRIVER_ERROR (1)

This error indicates that some function within the driver has failed. This error indicates that one of the IOCTL calls within the driver itself has not completed successfully. Using Windows Device Manager, verify that the driver is loaded and has no resource conflicts.

4.4.2 INVALID_HANDLE (2)

This error indicates that the driver has not initialized or closed. The driver attempts to obtain a handle to the I^2C device, and this error indicates that the handle was not obtained. Verify that the driver has loaded successfully.

4.4.3 INVALID_PARAMETER (3)

This error indicates that one of the parameters in a DLL function is out of bounds. Check your parameter definitions.

5 Sample Applications

There is one sample Windows console application provided in the driver package, *panel.exe*. The source code for the application is provided in the Appendix section.

5.1 Panel Application

The *panel* sample application is a simple program that cycles through every possible frequency and duty cycle combination with a two second pause between each step. It then sets the PWM at 225Hz with a 75% duty cycle. The panel is then disabled for two seconds and then re-enabled. Any error conditions are reported and the program is terminated immediately.

Appendix A: lvdsControllerDLL.h

```
//*****************************************************************************
//
//     Copyright 2017 by WinSystems Inc.
//
//*****************************************************************************
//
//     Name    : lvdsControllerDLL.h
//
//     Project : LVDS Controller Windows DLL
//
//     Author  : Paul DeMetrotion
//
//*****************************************************************************
//
//       Date        Rev                    Description
//     --------    -------    ---------------------------------------------
//     08/07/17      1.0           Original Release of DLL
//
//*****************************************************************************

#ifndef _LVDS_CONTROLLER_DLL_H_
  #define _LVDS_CONTROLLER_DLL_H_

#if defined DLL_EXPORT
#define DECLDIR __declspec(dllexport)
#else
#define DECLDIR __declspec(dllimport)
#endif

extern "C"
{
    DECLDIR int InitializeSession();
    DECLDIR int blOnOff(unsigned int blControl);
    DECLDIR int blSetFrequency(unsigned int blFrequency);
    DECLDIR int blSetDutyCycle(unsigned int blDutyCycle);
    DECLDIR int CloseSession();
}

typedef enum {
    SUCCESS = 0,
    DRIVER_ERROR,
    INVALID_HANDLE,
    INVALID_PARAMETER
} ErrorCodes;

typedef enum {
    BL_ON = 0,
    BL_OFF
} blControl;

typedef enum {
    BL_FREQ_500 = 0,  // 500Hz
    BL_FREQ_400,      // 400 Hz
    BL_FREQ_300,      // 300 Hz
    BL_FREQ_275,      // 275 Hz
    BL_FREQ_250,      // 250 Hz
    BL_FREQ_225,      // 225 Hz
    BL_FREQ_200,      // 200 Hz
    BL_FREQ_100,      // 100 Hz
    MAX_BL_FREQ_OPTIONS
} blFrequency;

typedef enum {
    BL_DUTY_100 = 0,  // 100 %
    BL_DUTY_87_5,     // 87.5%
    BL_DUTY_75,       // 75%
    BL_DUTY_62_5,     // 62.5%
    BL_DUTY_50,       // 50%
    BL_DUTY_37_5,     // 37.5%
    BL_DUTY_25,       // 25%
    BL_DUTY_12_5,     // 12.5%
    BL_DUTY_0,        // 0%
    MAX_BL_DUTY_OPTIONS
} blDutyCycle;
#endif
```

Appendix B: panel.cpp

```
//****************************************************************************
//
//     Copyright 2017 by WinSystems Inc.
//
//****************************************************************************
//
//     Name    : panel.cpp
//
//     Project : LVDS Controller Console Application
//
//     Author  : Paul DeMetrotion
//
//****************************************************************************
//
//       Date        Rev                   Description
//     --------    -------     ---------------------------------------------
//     08/07/17      1.0          Original Release
//
//****************************************************************************

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "lvdsControllerDLL.h"

#define MAJOR_VER   1
#define MINOR_VER   0

int _tmain(int argc, _TCHAR* argv[])
{
    int dllReturn = 0;

    printf("LVDS Controller Application : panel\n");
    printf("Version %d.%d\n\n", MAJOR_VER, MINOR_VER);

    dllReturn = InitializeSession();

    if (dllReturn)
    {
        printf("Error initializing session with code %d.\n", dllReturn);
        exit(dllReturn);
    }
    else
    {
        printf("Device opened.\n");
    }

    for (int i = 0; i < MAX_BL_FREQ_OPTIONS; i++)
    {
        if (blSetFrequency(i))
        {
            printf("\nError setting frequency with code %d.\n", dllReturn);
            exit(dllReturn);
        }
        for (int j = 0; j < MAX_BL_DUTY_OPTIONS; j++)
        {
            if (blSetDutyCycle(j))
            {
                printf("\nError setting duty cycle with code %d.\n", dllReturn);
                exit(dllReturn);
            }

            Sleep(2000);
        }
    }

    if (blSetFrequency(BL_FREQ_225))
    {
        printf("\nError setting frequency with code %d.\n", dllReturn);
        exit(dllReturn);
    }

    if (blSetDutyCycle(BL_DUTY_75))
    {
        printf("\nError setting duty cycle with code %d\n", dllReturn);
        exit(dllReturn);
    }

    printf("Panel will be disabled in 2 seconds.\n");

    Sleep(2000);

    if (blOnOff(BL_OFF))
        printf("\nError turning backlight off.\n\n");

    Sleep(2000);

    printf("Panel enabled again.\n");

    if (blOnOff(BL_ON))
        printf("\nError turning backlight on.\n\n");

    dllReturn = CloseSession();

    if (dllReturn)
    {
        printf("Error closing session with code %d.\n", dllReturn);
        exit(dllReturn);
    }

    printf("Device closed.\n\n");

    return dllReturn;
}
```