DOC101781-1- rev 01 – 2013-11-27 - Created by: 6644

Programmers Guide

HT 00273 OPT-A1

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

INDEX

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

Revision History

	Created		Approved		
Rev	Date	By	Date	By	Description
1.0	2013-05-21	Lukasz Madej			
1.2	2013-11-27	Wei Jing	2014-03-18	André Kråkenes	HW Rev2: - Invert Output State

Abbreviation

Abbreviation	Description

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

1 Introduction

An USB IO card (UIOC) bases on a FT232H chip produced by FTDI. This chip allows controlling the UIOC via an USB port trough dedicated dynamic D2XX library which is available for Windows and Linux operating systems. Suitable library version can be downloaded from an official FTDI site (http://www.ftdichip.com/Drivers/D2XX.htm). This document is focused on Windows operating systems only nevertheless an example codes can be ported to Linux systems very easily.

A library for Windows should be statically linked with an executable file (a proper ftd2xx.lib file and a ftd2xx.h file should be used in the project and a ftd2xx.dll library should be installed in the system).

Additional information needed to develop software for UIOC can be found in:

- "D2XX Programmer's Guide" published by the FTDI.
- "Application note AN_108 Command processor for MPSSE" published by the FTDI.
- "Application note AN_135 FTDI MPSSE Basics" published by the FTDI.
- "Application note AN_180 FT232 MPSSE Example" published by the FTDI.

1. USB IO card's output and input list

UIOC is equipped with 4 general purpose inputs (table 2.1) and 4 general purpose outputs (table 2.2). For more these outputs are equipped with 4 diagnostic inputs (table 2.3). Thanks to these inputs an outputs' state can be examined by software.

Table 2.1. A list of UIOC general purpose inputs

UIOC input name	FT232 pin name	FT232 access command name	State for input high	State for input low
INPUT A	ADBUS4	"Read Data Bits LowByte"	0	1
INPUT B	ADBUS5			
INPUT C	ADBUS6			
INPUT D	ADBUS7			

Table 2.2. A list of UIOC general purpose outputs

UIOC output name	FT232 pin name	FT232 access command name	State for output high	State for output low
OUTPUT 0	ACBUS0	"Set Data Bits HighByte"	**0**	**1**
OUTPUT 1	ACBUS1			
OUTPUT 2	ACBUS2			
OUTPUT 3	ACBUS3			

Table 2.3. A list of UIOC diagnose inputs

UIOC diagnose input name	UIOC related output name	FT232 pin name	FT232 access command name	State for output high	State for output low
INPUT E	OUTPUT 0	ACBUS4	"Read Data Bits HighByte"	0	1
INPUT F	OUTPUT 1	ACBUS5			
INPUT G	OUTPUT 2	ACBUS6			
INPUT H	OUTPUT 3	ACBUS7			

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

2. Basic procedures

2.1. Configuration of FT232 chip

In order to configure FT232 chip to control the UIOC following steps have to be taken:

1) Confirm existence of FT232 device and open it with **FT_Open** function (only if device is not opened).
2) Reset the peripheral side of FT232 port with **FT_ResetDevice** function.
3) Purge FT232 input and output buffers with **FT_Purge** function.
4) Configure the FT232 USB transfer sizes for minimum values (64 bytes) with **FT_SetUSBParameters** function.
5) Disable FT232 event and error characters with **FT_SetChars** function.
6) Disable FT232 timeouts with **FT_SetTimeouts** function.
7) Set FT232 latency timer for its minimum value (minimum value recommended by FTDI is 2 ms) with **FT_SetLatencyTimer** function in order to maximize performance.
8) Configure FT232 flow control for RTS/CTS with **FT_SetFlowControl** to ensure that the driver will not issue IN requests if the buffer is unable to accept data.
9) Perform a general reset on the FT232 MPSSE controller with **FT_SetBitMode** with proper parameters.
10) Enable the FT232 MPSSE controller with **FT_SetBitMode** function.
11) Verify that FT232 accepts MPSSE commands.
12) Configure direction and states of UIOC output and input ports with **FT_Write** function and proper MPSSE command. A complete list of UIOC's inputs and outputs is shown in point 2.
13) Close FT232 device with **FT_Close** function (only if there is no other data to exchange with UIOC).

All functions listed above are available in the D2XX library. The example of configuration procedure can be found in a `usbIoCard_c::configure()` function from usbIoCard C++ class (a **usbIoCard.cpp** file and **usbIoCard.h** file).

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

2.2. Controlling an output of UIOC

In order to control outputs of UIOC following steps have to be taken:

1) Confirm existence of FT232 device and open it with **FT_Open** function (only if device is not opened).
2) Configure the FT232 chip to control the UIOC (only if device is not configured; please read point 1.1).
3) Send **Read Data Bits HighByte** MPSSE command with **FT_Write** function in order to read current state of UIOC outputs and read this state with **FT_Read** function.
4) Change value of output corresponding bit (0 to reset the output and 1 to set the output).
5) Send **Write Data Bits HighByte** MPSSE command with **FT_Write** function in order to write new state of UIOC outputs.
6) Close FT232 device with **FT_Close** function (only if there is no other data to exchange with UIOC).

All functions listed above are available in the D2XX library. Detailed description of MPSSE commands can be found in the AN_108 document (please read point 1). The example of controlling an output is introduced in a **`usbIoCard_c::setOutputState()`** function from usbIoCard C++ class (a **usbIoCard.cpp** file and **usbIoCard.h** file). A list of available outputs is introduced in point 2.

2.3. Reading an input of UIOC

In order to read inputs' state of UIOC following steps have to be taken:

1) Confirm existence of FT232 device and open it with **FT_Open** function (only if device is not opened).
2) Configure the FT232 chip to control the UIOC (only if device is not configured; please read point 1.1).
3) Send **Read Data Bits LowByte** MPSSE command with **FT_Write** function in order to read current state of UIOC inputs and read this state with **FT_Read** function.
4) Examine value of input corresponding bit (1 for input in low state, 0 for input in high state).
5) Close FT232 device with **FT_Close** function (only if there is no other data to exchange with UIOC).

All functions listed above are available in the D2XX library. Detailed description of MPSSE commands can be found in the AN_108 document (please read point 1). The example of reading an input state is shown in a **`usbIoCard_c::readInputState()`** function from usbIoCard C++ class (a **usbIoCard.cpp** file and **usbIoCard.h** file). A list of available inputs is defined in point 2. Please note that on UIOC inputs' state logic is inverted due to construction of the hardware.

Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com

3. USB IO card usage with an example class

An example C++ class can be used for controlling UIOC. The class consists of two files – a **usbIoCard.cpp** file and **usbIoCard.h** file. A usage of this class is shown on below listing.

```
#include <iostream>
using namespace std;

usbIoCard_c ioCard; // create an usbIoCard_c class object

if (NO_ERR == ioCard.open()) // open an USB IO card
{
	if (NO_ERR == ioCard.configure()) // configure an USB IO card
	{
		if (NO_ERR != ioCard.setOutputState(GPO0, OUT_HI)) // change output's state
		{
			cout << "Output state change ERROR!" << endl;
		}
		outputStateName_e outState;
		if (NO_ERR != diagOutputState(GPO0, outState) // diagnose output's state
		{
			cout << "Output state diagnose ERROR!" << endl;
		}
		inputStateName_e inState;
		if (NO_ERR != readInputState(IN0, inState) // read input's state
		{
			cout << "Input state read ERROR!" << endl;
		}
	}
	else  // USB IO card configuration failed
	{
		cout << "USB IO card configuration ERROR!" << endl;
	}
	ioCard.close();
}
else // USB IO card open failed
{
	cout << "Cannot open USB IO card! Is it connected?" << endl;
}
```


Hatteland Display AS, Åmsosen, N-5578 Nedre Vats, Norway
Tel: (+47) 4814 2200 - mail@hatteland-display.com - www.hatteland-display.com