Future Technology Devices International Ltd.

Application Note

AN_140

Vinculum-II PWM Example

Document Reference No.: FT_000241

Version 1.1

Issue Date: 2010-03-25

This document introduces the PWM (Pulse Width Modulation) capability of Vinculum II (VNC2). It then explains how to use it and provide a worked 'C' code example to configure the PWM interface.

Future Technology Devices International Limited (FTDI)

Unit 1,2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
E-Mail (Support): **support1@ftdichip.com** Web: http://www.ftdichip.com

Table of Contents

1 Introduction

1.1 Overview

Vinculum-II (VNC2) is FTDI's 2nd generation USB host solution device and expands on the capabilities of the VNC1L. One of the capabilities of the device is that it provides a PWM interface capability. This document introduces PWM and gives an example of programming the PWM interface functions of Vinculum II (VNC2).

2 What is PWM?

Q. What exactly is Pulse Width Modulation (PWM) and how does it work?

A. The simplest form of PWM is duty cycle control. Consider the simple square wave in Figure 2.1.

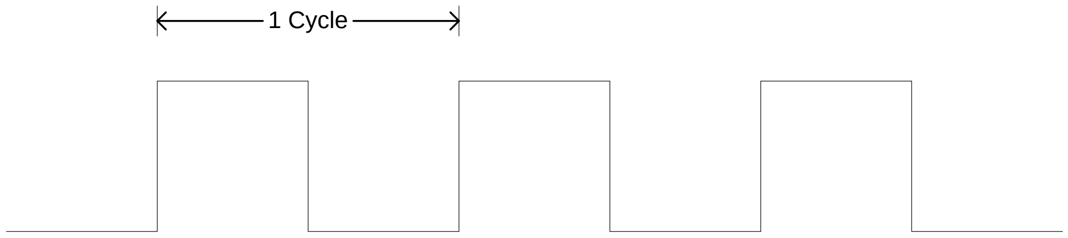

Figure 2.1 Square wave with 50% Duty Cycle

In this example, the width of the high pulse is equal to the width of the low pulse. This waveform has a 50 % duty cycle. If the amplitude of this square wave is 5V, the RMS voltage can be calculated as:

$$V_{RMS} = V_{PEAK} * SQRT\ (Duty\ Cycle)$$

$$V_{RMS} = 5V * SQRT\ (.50)$$

$$V_{RMS} = 3.54V$$

If the duty cycle is reduced, then the RMS voltage is reduces. This is illustrated in the following example which shows a waveform with a 20% duty cycle:

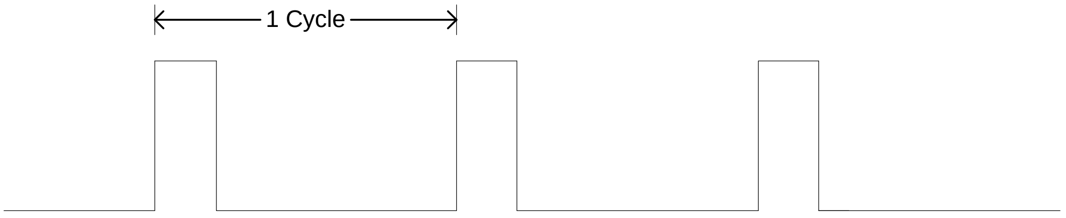

Figure 2.2 Square wave with 20% Duty Cycle

With the same 5V peak amplitude as the 50% duty cycle waveform, the 20% duty cycle waveform has the following RMS voltage:

$$V_{RMS} = 5V * SQRT\ (.20)$$

$$V_{RMS} = 2.24V$$

By changing the duty cycle, the effective RMS is modified without changing the signal amplitude.

Why is this important?

By changing the amplitude and duty cycle of the signal, it is essentially generating an **analogue** signal from a **digital** source. PWM is a method which can be used to interface to analogue hardware using a digital source such as a microcontroller.

Real world applications of PWM include lamp brightness, electric motor control and servo control.

With VNC2, the duty cycle is controlled by the PWM controller block. VNC2 can be programmed to generate a variety of PWM signals, as described in Section 2.

In section 3, a simple motor controller using the VNC2 PWM function application is discussed.

3 VNC2 PWM Description

Pulse width modulation (PWM) is a common interface on microcontrollers.

VNC2 has 8 separate independent PWM channels. The following section describes the building blocks used to control these PWM channels.

Pre-scaler - This is a programmable counter that reduces the frequency of the system clock (48 MHz, 24 MHz, or 12 MHz) to the desired frequency. The pre-scaler is shared by all 8 PWM channels.

16 Bit Counter Block - This is a programmable counter that determines the period of the PWM signal. The input clock is from the pre-scaler block. The 16 bit counter is shared by all 8 PWM channels.

16 Bit Comparator – Up to 8 comparators can be used per PWM channel. The number of comparators assigned to a PWM channel determines the toggle events (up to 8), which give up to 4 data pulses. Simple duty cycle based pulse width modulation can be programmed by using only 2 comparators. There are a total of 8 comparators in the PWM module.

Control Block – This controls the number of times to repeat the PWM waveform. The control block is shared by all 8 PWM channels.

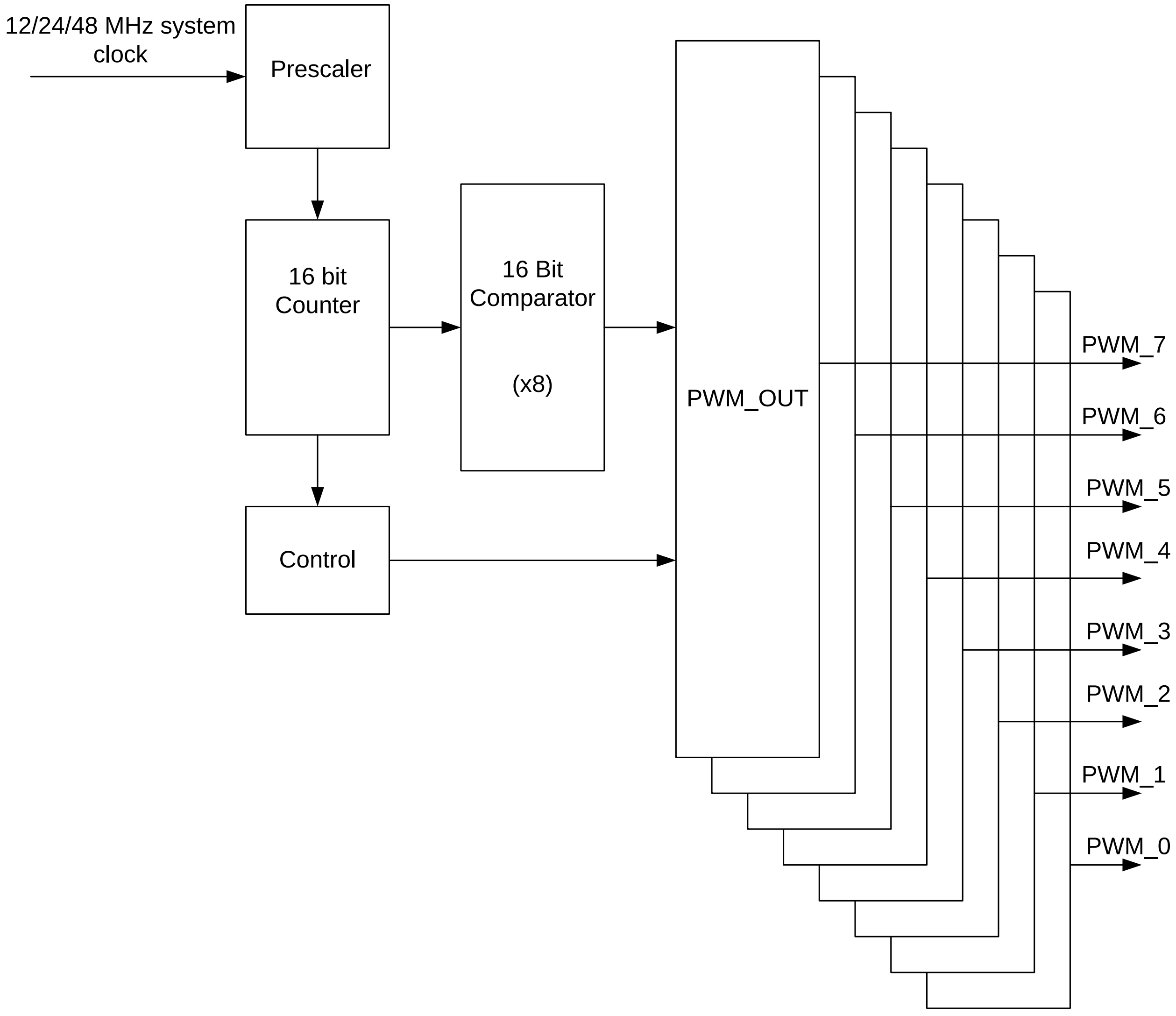

Figure 3.1 VNC2 PWM Block Diagram

3.1 VNC2 Duty Cycle Description

The following section describes an example of how to generate a 4 pulse PWM waveform using VNC2, which toggles at the following counter states:

7, 8 12,14 15,16 19,22

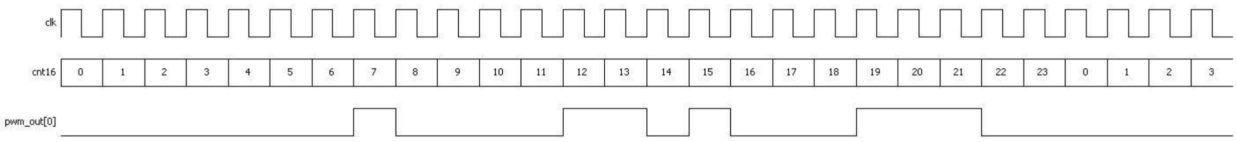

Figure 3.2 4 Pulse Waveform generated by 8 Comparators

In this example there are 8 toggle events and all 8 comparators are used. In this example, the 16 bit counter is programmed to count 24 states and then re-start, and 4 pulses are generated.

Comparator #	Programmed Toggle Value	Pulse Width (clock cycles)
0	7	
1	8	1
2	12	
3	14	2
4	15	
5	16	1
6	19	
7	22	3

Table 3.1 Programming 8 VNC2 Comparators to generate above waveform

For a simple duty cycle PWM, where only a single pulse is required only 2 comparators would be necessary and only a single pulse is generated. For example, to generate a 50% duty cycle waveform for the clock/counter combination, comparators 0 & 1 could be programmed as follows:

Comparator #	Programmed Toggle Value	Pulse Width (clock cycles)
0	2	
1	14	12

Table 3.2 Programming 2 VNC2 Comparators for 50 % Duty Cycle

In this example, there are 24 clocks per cycle and the PWM output changes state (toggles) every 12 clocks (12/24). This produces a 50 % duty cycle. By programming different toggling values into the VNC2 comparators, a wide range of duty cycles can be generated

3.2 VNC2 Example PWM Application

This example shows a common method of interfacing a DC electric motor to VNC2. Since the TTL voltage and current levels of VNC2's output pins cannot be used to directly drive a motor, a specialized interface circuit is required : an H-bridge. The following diagram shows a simplified H –bridge circuit:

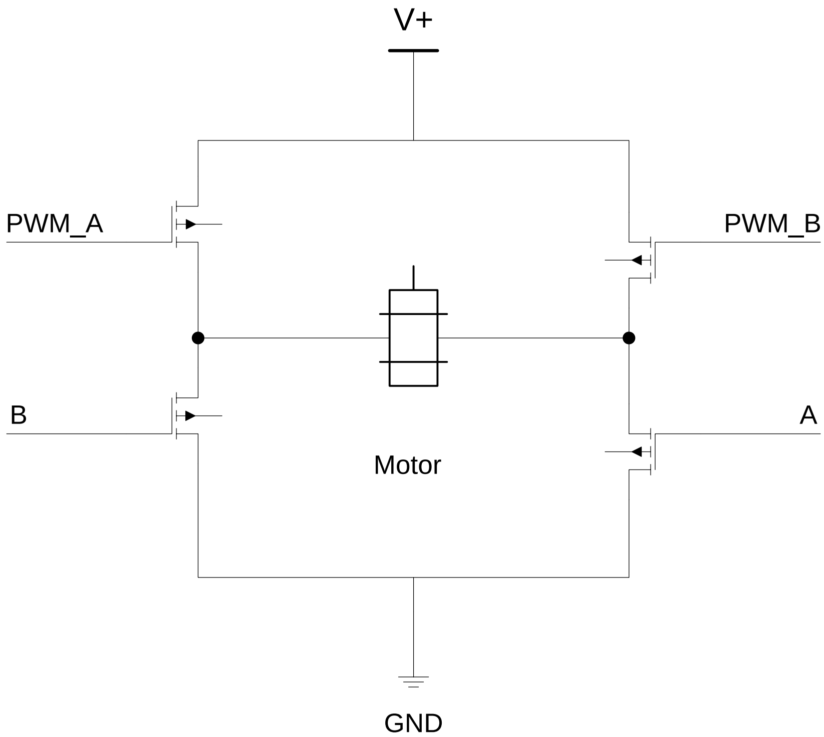

Figure 3.3 Simple H-Bridge circuit

This circuit can control a motor's speed and direction. The N channel MOSFETs function as switches. Initial conditions are all inputs (A, B, PWM_A and PWM_B) off. VNC2 would be configured to drive the signals PWM_A and PWM_B from its PWM outputs and pins A & B from its GPIO pins.

If input A and PWM_A are turned on, current flows in one direction and the motor will start spinning. If a PWM waveform is applied to PWM_A from VNC2, then this can be used to control the speed of the motor in one direction. The motor speed will be determined by the duty cycle of the applied PWM waveform.

If inputs A and PWM_A are switched off, and inputs B and PWM_B switched on, the motor will spin in the opposite direction. By applying a PWM waveform to PWM_B, then this will control the speed of the motor in the reverse direction.

Caution must be followed when supplying signals to the H-bridge. If both MOSFETs on the same side of the bridge are switched on, a low impedance path (short) to ground is created, which will destroy the circuit and the motor. This is referred to as "shoot-through" and this should be avoided. Commercial H-bridge ICs contain protective interfaces to prevent this.

3.3 Using a Commercial H-Bridge Example

The National Semiconductor LMD18200 is a 3 Amp H-bridge designed to control DC motors. It consists of a control block and an H-bridge. The control block is driven by TTL level inputs for the PWM signal and mode control inputs, making it easy to interface directly with VNC2. The H-bridge section can be powered to 55V @ 3A, and is easily interfaced to a DC motor. It also has protection circuitry to prevent shoot through.

Figure 3.4 shows a simplified functional block diagram of the LMD18200.

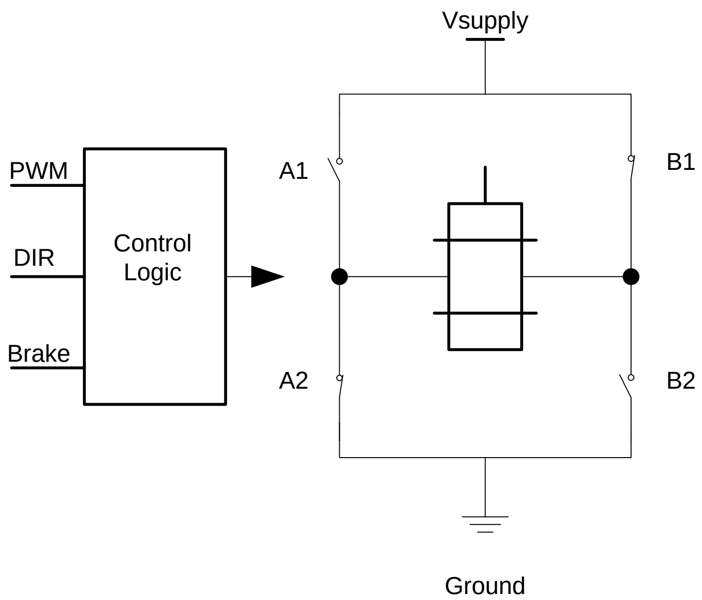

Figure 3.4 LMD 18200 Block Diagram

In Figure 3.4, the N channel MOSFETS are represented by switches. Depending on the state of the input pins, the motor can be driven at different speeds and directions, as shown by the following truth table:

MODE	PWM	DIR	BRAKE	Active Switches (ON)
1	H	H	L	A1,B2
2	H	L	L	A2,B1
3	L	X	L	A1,B1
4	H	H	H	A1,B1
5	H	L	H	A2,B2
6	L	X	H	All Off

Table 3.3 LMD18200 Truth Table

The following describes what happens in modes 1 and 2.

In mode 1, DIR is driven high and BRAKE is driven low. PWM is driven by VNC2.

With a VNC2 PWM signal modulated at 50% duty cycle, the motor will spin forward at normal speed. As the duty cycle is reduced, the motor will slow down. Switch A1 is used for PWM, and switch B2 is closed. Switches A2 and B1 are open

In mode 2, DIR is driven low and BRAKE driven low. PWM is driven by VNC2.

With the PWM signal modulated at 50% duty cycle, the motor will spin in reverse at normal speed. As the duty cycle is reduced, the motor will slow down. Switch B1 is used for PWM and switch A2 is closed. Switches A1 and B2 are open.

The following table summarizes how the controller works in Modes 1 & 2:

Mode	Duty Cycle (controlled by PWM)	Motor Rotation	Motor Speed
1	50%	forward	Normal
2	50%	reverse	Normal
1	20%	forward	Slow
2	20%	reverse	Slow

Table 3.4 LMD 18200 H-Bridge operating modes.

4 VNC2 Duty Cycle Programming Example

The following example code will generate a waveform with a 50% duty cycle when used with VNC2 The code is not guaranteed or supported by FTDI and is provided for illustration purposes only.

```
// *****************************************************************
// PWM example
// This application uses a single PWM output linked to 2 PWM comparators
// to generate a 50% duty cycle pulse 255 times and repeat.
// This code was tested on a V2DIP-64 development board using Version 0.9.7
// the Vinculum 2 IDE.
// *****************************************************************

#include "vos.h"
#include "PWM.h"

#define SIZEOF_tcb                          0x400

#define NUMBER_OF_DEVICES                   1

/* Device definitions*/

#define VOS_DEV_PWM                         0

// *****************************************************************

// Device initialistation

// *****************************************************************

void init_devices(void) {

unsigned char packageType;
```

```
if (NUMBER_OF_DEVICES != 0)

// INITIALISE IOMUX PARAMETERS

// route PWM signals as required
packageType = vos_get_package_type();

if (packageType == VINCULUM_II_48_PIN){

// PWM 1 to pin 12

vos_iomux_define_output(12,IOMUX_OUT_PWM_1);

}

else if (packageType == VINCULUM_II_64_PIN) {

// PWM 1 to pin 62 (IOBUS41 on V2DIP-64 module)

vos_iomux_define_output(62,IOMUX_OUT_PWM_1);

}

//*******************************************************

// INITIALISE PWM PARAMETERS

//*******************************************************

 pwm_init(VOS_DEV_PWM);
 }

}

// ****************************************************************
// Pulse thread
// ****************************************************************

void pulse() {

VOS_HANDLE hPwm;

pwm_ioctl_cb_t pwm_iocb ;

// open pwm and get a handle
```



```
hPwm = vos_dev_open(VOS_DEV_PWM);

// set counter prescaler value to ff (256), reduce system clock
// from 48 Mhz to 187.5 KHz

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_PRESCALER_VALUE;

pwm_iocb.count.prescaler = 0xFF;

vos_dev_ioctl(hPwm, &pwm_iocb);

// *******************************************************************
// Setting a count value of 0x00A0 with toggles at 0x0010 and 0x0060
// will give a 50% duty cycle  (80/160)
// *******************************************************************

// set counter value - cycle complete when internal counter reaches this value

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COUNTER_VALUE;

pwm_iocb.count.value = 0x00A0;

vos_dev_ioctl(hPwm, &pwm_iocb);

// set comparator 0 value - toggle output at a value of 0x0010

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COMPARATOR_VALUE;

pwm_iocb.identifier.comparator_number = COMPARATOR_0;

pwm_iocb.comparator.value = 0x0010;

vos_dev_ioctl(hPwm, &pwm_iocb);


// set comparator 1 value - toggle output at a value of 0x0020

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_COMPARATOR_VALUE;

pwm_iocb.identifier.comparator_number = COMPARATOR_1;

pwm_iocb.comparator.value = 0x0060; // try changing from 0x0060 to change duty cycle

vos_dev_ioctl(hPwm, &pwm_iocb);
```

```
// enable comparators 0 and 1 for PWM 0

// this will cause PWM output 1 to toggle on comparators 0 and 1

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_OUTPUT_TOGGLE_ENABLES;

pwm_iocb.identifier.pwm_number = PWM_1;

pwm_iocb.output.enable_mask = (MASK_COMPARATOR_0 | MASK_COMPARATOR_1);

vos_dev_ioctl(hPwm, &pwm_iocb);

// set initial state - all PWM outputs will be low (0) initially

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_INITIAL_STATE;

pwm_iocb.output.init_state_mask = 0x00;

vos_dev_ioctl(hPwm, &pwm_iocb);


// set restore state - PWM output 0 will return to low state (0)
// at end of cycle

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_RESTORE_INITIAL_STATE;

pwm_iocb.output.restore_state_mask = (MASK_PWM_1);

vos_dev_ioctl(hPwm, &pwm_iocb);

// set mode to 256 cycles

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_SET_NUMBER_OF_CYCLES;

pwm_iocb.output.mode = 0xFF;

vos_dev_ioctl(hPwm, &pwm_iocb);

while(1) {

// enable interrupt - this will fire when the specified number of

// cycles is complete

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_ENABLE_INTERRUPT;
```

```
vos_dev_ioctl(hPwm, &pwm_iocb);

// enable output

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_ENABLE_OUTPUT;

vos_dev_ioctl(hPwm, &pwm_iocb);

// wait on interrupt

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_WAIT_ON_COMPLETE;

vos_dev_ioctl(hPwm, &pwm_iocb);

// When we get to here, we've completed our 256 cycles of 50% duty cycle
// disable output

pwm_iocb.ioctl_code = VOS_IOCTL_PWM_DISABLE_OUTPUT;

vos_dev_ioctl(hPwm, &pwm_iocb);

// no sleep

vos_delay_msecs(0);

  }

}

//******************************************************************

// Main application

// ******************************************************************

void main(void) {

// initialise rtos

vos_init(VOS_QUANTUM, VOS_TICK_INTERVAL, NUMBER_OF_DEVICES);

vos_set_clock_frequency(VOS_48MHZ_CLOCK_FREQUENCY);

// initialise devices (APPLICATION SPECIFIC)

init_devices();
```

```
// initialise threads

// pulse thread

vos_create_thread( 31,SIZEOF_tcb,&pulse,0);


// enable PWM interrupts

vos_enable_interrupts(VOS_PWM_TOP_INT_IEN);

vos_start_scheduler();

main_loop:

goto main_loop;

}
```


Acronyms and Abbreviations

Terms	Description
PWM	Pulse Width Modulation
H-Bridge	H shaped array of transistors designed to control current flow to a electric motor

Table A.0.1 Acronyms and Abbreviations

Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1,2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com
Web Site URL http://www.ftdichip.com
Web Shop URL http://www.ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-mail (General Enquiries) us.admin@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
Shanghai, 200051
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com
Web Site URL http://www.ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – List of Tables and Figures

List of Tables

List of Figures

Appendix B - Revision History

Revision History

Version 1.0		19th March, 2010
Version 1.1	Changed pwm_ioctl_cb to pwm_ioctl_cb_t	25th March, 2010
	Added Void main(void)	