Application Note

AN_191

Vinculum-II Interrupts

Version 1.0

Issue Date: 2012-06-22

This application note introduces interrupts, and how they are implemented in the FTDI Vinculum-II device. A hardware example is provided.

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
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This document explains what interrupts are, why they are useful for embedded programmers, and how they are implemented in Vinculum-II (VNC2) firmware. A sample hardware application is used to demonstrate how interrupts are used in a real world example. The FTDI Vinco board and a 3rd party LCD shield will be used in the example.

The reader should be familiar with the VNC2 IDE and Vinco development board. It is also assumed that the reader has an understanding of the Vinculum Operating System (VOS).

Interrupts are used in embedded systems to signal the processor that an external event has taken place. Examples could include the arrival of data on a bus, flipping a switch or if a specific amount of time has passed.

Interrupts allow designers to isolate time critical operations from the main program to guarantee they are processed in a timely manner.

Since interrupts are asynchronous, they can occur at any time when the main program is running. An interrupt mask is used to select the desired type of interrupt.

After reading this app note, the reader will be able to understand and write interrupt based code for the Vinculum II.

1.1 Hardware used to demonstrate GPIO interrupt

- Vinco development board
- LCD Shield for Arduino/Vinco
- VNC2 Debugger module
- External pushbutton

Appendix A has the datasheet links for the above hardware.

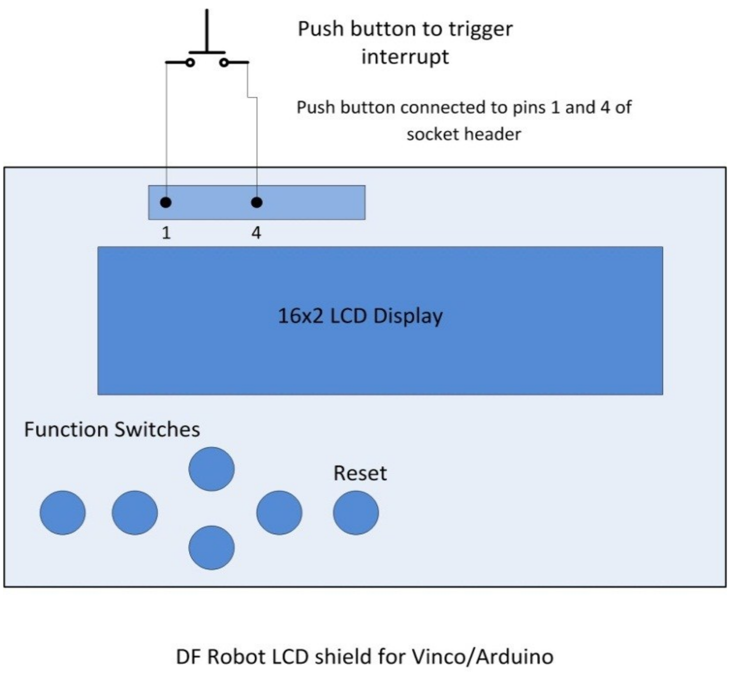


Figure 1.1 LCD Shield for Vinco

1.2 Available VNC2 Interrupts

The following VNC2 ports and interfaces have interrupt support:

USB Port	GPIO Port	SPI Port	PWM Interface	FIFO Port
USB Host 0	GPIO Port A	SPI Master	PWM	FIFO
USB Host 1	GPIO Port B	SPI Slave 1		
USB Slave 1		SPI Slave 2		
USB Slave 2				

Table 1.1 Available VNC2 Interrupts

An *interrupt mask* is used to select one of the above interrupts. In this application note, GPIO Port pin B_0 (Pin 51) will be used as an interrupt. Multiple interrupts such as PWM and GPIO are allowed since they are different functions.

For GPIO interrupts, Port B supports interrupts from a specified pin. Port A will trigger an interrupt if any of the Port A pins toggle.

1.3 Levels and Edges

Level sensitive interrupts force the VNC2 to respond as long as the interrupt signal is at a specified level. Edge sensitive interrupts force the VNC2 to respond when the interrupt signal makes a transition. Levels can be logic low or logic high, edges can be rising or falling. Figure 1.2 shows a positive level sensitive interrupt and a falling edge sensitive interrupt.

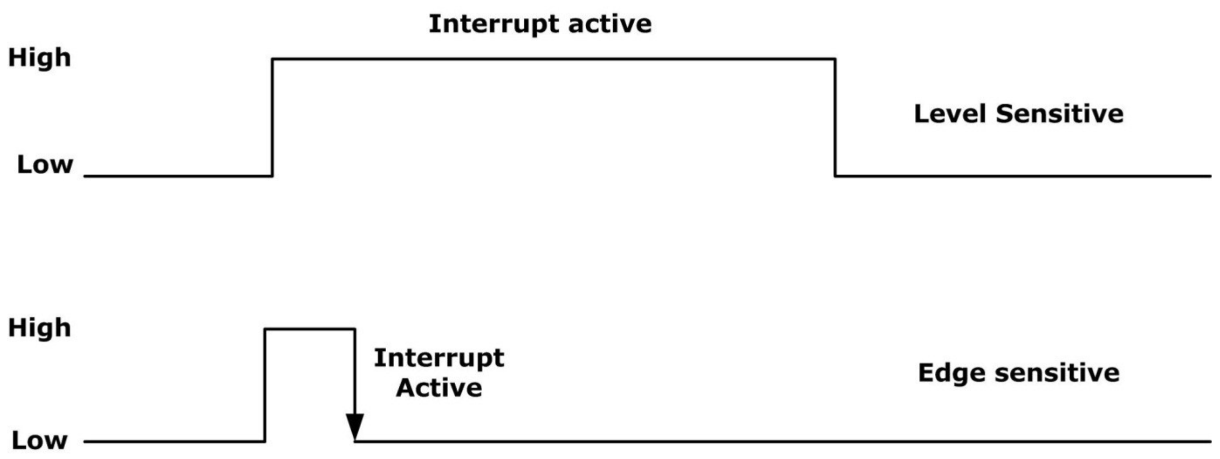


Figure 1.2 Interrupt Types

2 Firmware Details

In this section, the function calls used to configure an interrupt and send the result to a LCD display will be discussed. The source code can be downloaded from the following link

http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/AN_191_Source_Code.zip

Figure 2.1 shows the basic operation of the interrupt application code.

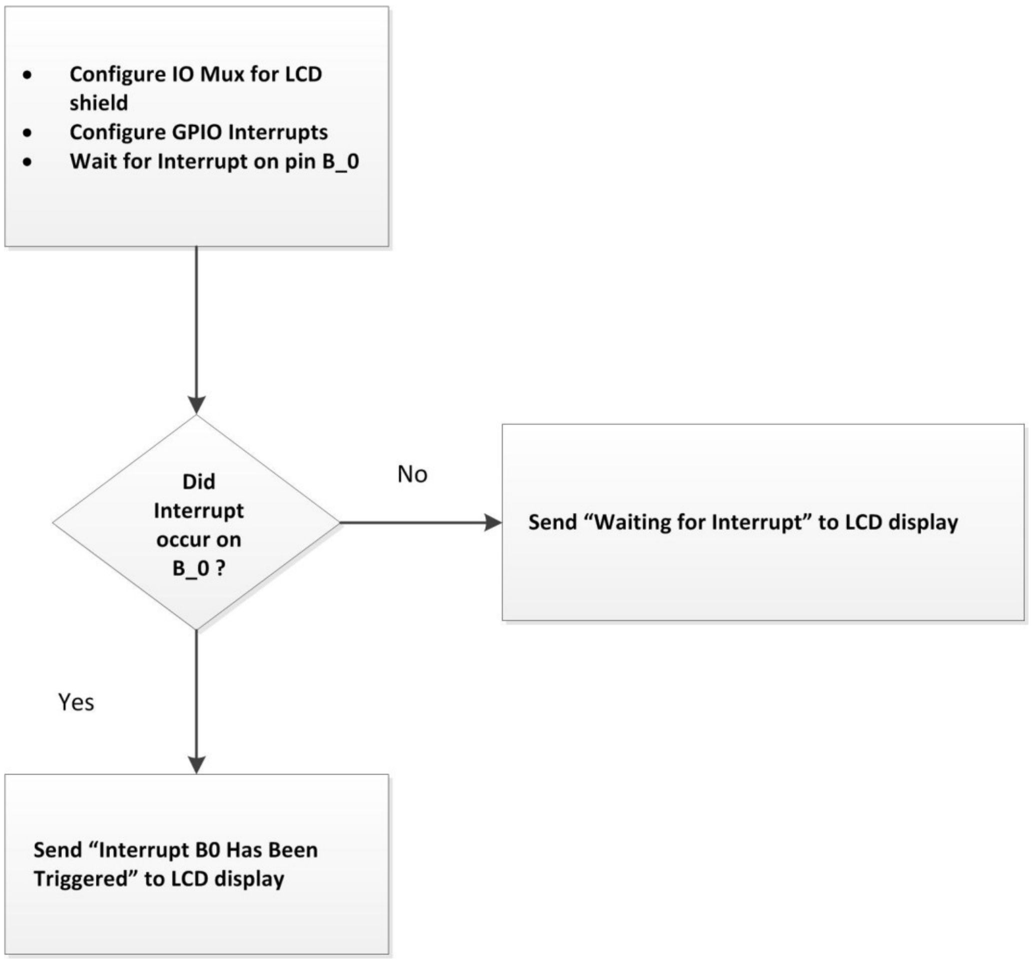


Figure 2.1 Interrupt Application Flowchart

2.1 Configure IOMux

The Vinco module needs to have its GPIO signals configured to interface with the LCD shield. A 16x2 monochrome LCD is used. The LCD control code is explained in more detail in Application Note AN_153, Vinco LCD Interface Example.

Pin 51 is the interrupt input. To sensitize this input for a falling edge interrupt, it has been configured as:

- Schmitt Trigger
- Fast slew rate
- 75K Pull Up

```
if (vos_get_package_type() == VINCULUM_II_64_PIN)
  {
        // GPIO_Port_A_0 to pin 57 as Output.
        vos_iomux_define_output(57, IOMUX_OUT_GPIO_PORT_A_0); //LCD DB4 input
        // GPIO_Port_A_1 to pin 58 as Output.
        vos_iomux_define_output(58, IOMUX_OUT_GPIO_PORT_A_1); //LCD DB5 input
        // GPIO_Port_A_2 to pin 59 as Output.
        vos_iomux_define_output(59, IOMUX_OUT_GPIO_PORT_A_2); //LCD DB6 input
        // GPIO_Port_A_3 to pin 60 as Output.
        vos_iomux_define_output(60, IOMUX_OUT_GPIO_PORT_A_3); //LCD DB7 input
        // GPIO_Port_A_6 to pin 17 as Output.
        vos_iomux_define_output(17, IOMUX_OUT_GPIO_PORT_A_6); //LCD RS input
        // GPIO_Port_A_7 to pin 18 as Output.
        vos_iomux_define_output(18, IOMUX_OUT_GPIO_PORT_A_7); //LCD EN input
        // GPIO_Port_C_3 to pin 23 as Output.  //Port C is used to isolate this signal
        vos_iomux_define_output(23, IOMUX_OUT_GPIO_PORT_C_3); //Backlight for LCD
        // GPIO_Port_B_0 to pin 51 as Input.  // Configure for Port B_0 Interrupt
        vos_iomux_define_input(51, IOMUX_IN_GPIO_PORT_B_0);
        vos_iocell_set_config(51,
                              VOS_IOCELL_DRIVE_CURRENT_4MA,
                              VOS _IOCELL_TRIGGER_SCHMITT,
                              VOS_IOCELL_SLEW_RATE_FAST,
                              VOS_IOCELL_PULL_UP_75K);
  }
```

2.2 Configure GPIO Interrupts

To set up a GPIO interrupt, the following steps must take place:

- Set interrupt mask for interrupts on GPIO B
- Specify which pin on port B to monitor
- Specify if interrupt is level sensitive or edge sensitive
- Specify the polarity of the interrupt (high/low level, rising/falling edge)
- Enable/Disable the interrupt

```
//Open GPIO Port B
hGpioB = vos_dev_open(GPIOB);

// Setup Interrupt Mask for GPIO B
gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_MASK;
gpio_iocb.value = 0x00 ; //set all as input
vos_dev_ioctl(hGpioB, &gpio_iocb);

// Setup Interrupt for INT_0 on port B_0
gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_PROG_INT0_PIN;
gpio_iocb.value = GPIO_PIN_0;  // Port B_0
vos_dev_ioctl(hGpioB, &gpio_iocb);

// Setup for Interrupt  on falling edge transition
gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_SET_PROG_INT0_MODE;
gpio_iocb.value = GPIO_INT_ON_NEG_EDGE;
vos_dev_ioctl(hGpioB, &gpio_iocb);

// Enable interrupt
vos_enable_interrupts(VOS_GPIO_INT_IEN);
```

2.3 Wait for Interrupt

The GPIO_B0 interrupt works by monitoring the state of this input. A "Waiting on Interrupt" message is sent to the LCD before configuring the interrupt.

Before the interrupt triggers, the VOS_IOCTL_WAIT_ON_INT0 call blocks a 2nd write to the LCD display. A pushbutton is connected between port B0 and a ground pin, as illustrated in Figure 1.1

When B0 makes a high to low transition (pushbutton is pressed), the rest of the thread is unblocked and is allowed to execute.

To simplify the design of the interrupt, the state of port B is read by VOS_DEV_READ when the interrupt triggers. The default state of port has been set to 0xFF. When a B0 goes low, port B is no longer 0xFF, and an "Interrupt Triggered" message is sent to the LCD display.

Pre-interrupt string to display:

```
unsigned char *lcd_str ;
unsigned char *lcd_str1, *lcd_str2;
lcd_str1 = "Waiting.....";
lcd_str2 = "For Interrupt";
```

Wait for interrupt code:

```
        gpio_iocb.ioctl_code = VOS_IOCTL_GPIO_WAIT_ON_INT0;
        vos_dev_ioctl(hGpioB, &gpio_iocb);
        //Code is blocked from executing until interrupt occurs
        //Read the state of Port B (preset to 0xFF)
        vos_dev_read(hGpioB, &B0,1,&num_read);

        // If Port B is not 0xFF, we have confirmed the interrupt
        // Write message to LCD indicating interrupt was triggered
        if (B0 != 0xFF)

                {
                lcd_str1 = "Interrupt B0 Has";
                lcd_str2 = "Been Triggered";
                }
    }
} while (1);
```

3 Using IDE to Monitor Thread States

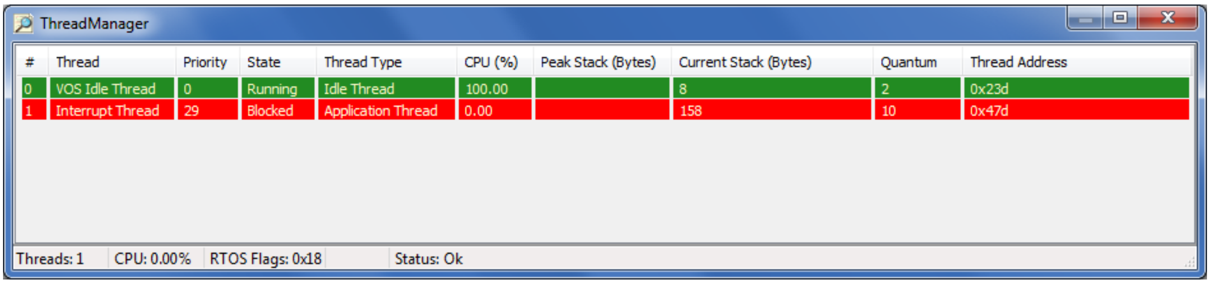
The **firmware** thread in the code has been labeled as "Interrupt Thread" by the **vos_create_thread_ex** call.

```
// create thread for firmware application (no parameters)

tcbFirmware = vos_create_thread_ex(29, SIZEOF_THREAD_MEMORY, firmware, "Interrupt Thread", 0);
```

This allows the IDE's Thread Manager to display the state of the interrupt thread before and after execution. The IDE must be used in debug mode for the Thread Manager to work.

While waiting for interrupt on GPIO B0, thread is blocked:



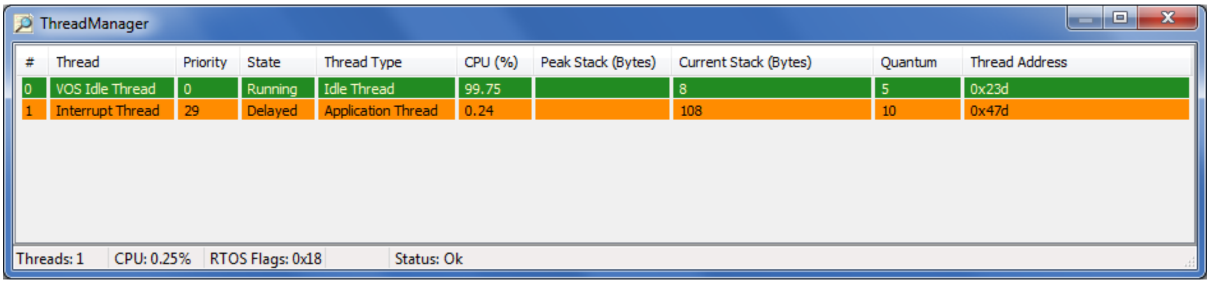
ThreadManager

#	Thread	Priority	State	Thread Type	CPU (%)	Peak Stack (Bytes)	Current Stack (Bytes)	Quantum	Thread Address
0	VOS Idle Thread	0	Running	Idle Thread	100.00		8	2	0x23d
1	Interrupt Thread	29	Blocked	Application Thread	0.00		158	10	0x47d

Threads: 1 CPU: 0.00% RTOS Flags: 0x18 Status: Ok

Figure 3.1 Thread Manager Pre-Interrupt

After Interrupt has triggered on GPIO B0, thread is unblocked:



ThreadManager

#	Thread	Priority	State	Thread Type	CPU (%)	Peak Stack (Bytes)	Current Stack (Bytes)	Quantum	Thread Address
0	VOS Idle Thread	0	Running	Idle Thread	99.75		8	5	0x23d
1	Interrupt Thread	29	Delayed	Application Thread	0.24		108	10	0x47d

Threads: 1 CPU: 0.25% RTOS Flags: 0x18 Status: Ok

Figure 3.2 Thread Manager Post-Interrupt

4 Conclusion

This document has introduced basic GPIO interrupts with the Vinculum-II. Other Vinculum-II interfaces can initiate interrupts, such as pulse width modulation (PWM) and Serial Peripheral Interface (SPI).

5 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

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

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)	asia.sales1@ftdichip.com
E-mail (Support)	asia.support1@ftdichip.com
E-mail (General Enquiries)	asia.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)

Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
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

http://ftdichip.com

Appendix A – References

Document References

AN_142 Vinculum II Tool Chain Getting Started Guide

AN_153 Vinco LCD Interface Example

AN_151 Vinculum User Guide

Arduino LCD Keypad Shield Web Page

Arduino LCD Keypad Shield Technical Details

Vinco Datasheet

http://www.ftdichip.com/Support/SoftwareExamples/VinculumIIProjects/AN_191_Source_Code.zip

Acronyms and Abbreviations

Terms	Description
LCD Shield	An Arduino compatible plug in LCD display that is easily adapted for use with the FTDI Vinco board.
Interrupt	A signal that indicates an external event has taken place.
Interrupt Mask	A command that selects which type of hardware interrupt to process.
Edge Sensitive Interrupt	An interrupt event triggered by a falling or rising edge
Level Sensitive Interrupt	An interrupt triggered by a high or low logic level.

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_191 Vinculum 2 Interrupts

Document Reference No.: FT_000531

Clearance No.: FTDI# 299

Product Page: http://www.ftdichip.com/FTProducts.htm

Document Feedback: Send Feedback

Revision	Changes	Date
draft	Initial draft subject to change	2012-05-16
1.0	First Release	2012-02-22