## horizontal line



Lab 7 (Creative Lab)

11.23.2015

**─**

Yazan Halawa

Adam Hastings

# Description

In this lab we interfaced a phone into our project to use as a controller instead of the push buttons. We managed to do this using the Adafruit Bluefruit LE UART Friend chip with a breadboard. We built a small circuit using the chip and connected it to the UART port in the ATLYS board. We had to add the IP to our hardware and connect the interrupt line, and a transfer and transmit signals from the bluetooth chip to the ATLYS board. Once the hardware was generated, we moved on to the software. We used the “xuartlite” library provided by Xilinx and used some code from github to manage interrupts on the bluetooth chip. We had to make some modifications to the code but we finally managed to connect the interrupts. Finally, we used an iphone app from adafruit that connects to the chip and provides a controller pad UI that was easy to interface and parse data from. The final code is attached. We spent approximately 10 hours on this lab.

# Source Code

**/\***

**\* bluefruit.h**

**\***

**\* Created on: Nov 19, 2015**

**\* Author: superman**

**\*/**

**#ifndef BLUEFRUIT\_H\_**

**#define BLUEFRUIT\_H\_**

**#include "xparameters.h"**

**#include "xuartlite.h"**

**#include "xintc.h"**

**#include "xil\_exception.h"**

**#include <stdio.h>**

**#include "platform.h"**

**#include "time.h"**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Constant Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\***

**\* The following constants map to the XPAR parameters created in the**

**\* xparameters.h file. They are defined here such that a user can easily**

**\* change all the needed parameters in one place.**

**\*/**

**#define UARTLITE\_DEVICE\_ID XPAR\_UARTLITE\_0\_DEVICE\_ID**

**#define INTC\_DEVICE\_ID XPAR\_INTC\_0\_DEVICE\_ID**

**#define UARTLITE\_INT\_IRQ\_ID XPAR\_INTC\_0\_UARTLITE\_0\_VEC\_ID**

**/\***

**\* The following constant controls the length of the buffers to be sent**

**\* and received with the UartLite device.**

**\*/**

**#define TEST\_BUFFER\_SIZE 15**

**static u8 pressed = 0;**

**int UartLiteIntrInit(u16 DeviceId);**

**u8 intr\_test();**

**u8 isPressed();**

**void setPressed(u8 newVal);**

**int SetupInterruptSystem(XUartLite \*UartLitePtr);**

**void SendHandler(void \*CallBackRef, unsigned int EventData);**

**void RecvHandler(void \*CallBackRef, unsigned int EventData);**

**#endif /\* BLUEFRUIT\_H\_ \*/**

**/\***

**\* bluefruit.c**

**\***

**\* Created on: Nov 19, 2015**

**\* Author: superman**

**\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include Files \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#include "bluefruit.h"**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**XUartLite UartLite; /\* The instance of the UartLite Device \*/**

**XIntc InterruptController; /\* The instance of the Interrupt Controller \*/**

**/\***

**\* The following buffers are used in this example to send and receive data**

**\* with the UartLite.**

**\*/**

**u8 SendBuffer[TEST\_BUFFER\_SIZE];**

**u8 ReceiveBuffer[TEST\_BUFFER\_SIZE];**

**/\***

**\* The following counters are used to determine when the entire buffer has**

**\* been sent and received.**

**\*/**

**static volatile int TotalReceivedCount = 0;**

**static volatile int TotalSentCount = 0;**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**int UartLiteIntrInit(u16 DeviceId){**

**int Status;**

**/\***

**\* Initialize the UartLite driver so that it's ready to use.**

**\*/**

**Status = XUartLite\_Initialize(&UartLite, DeviceId);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Perform a self-test to ensure that the hardware was built correctly.**

**\*/**

**Status = XUartLite\_SelfTest(&UartLite);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Connect the UartLite to the interrupt subsystem such that interrupts can**

**\* occur. This function is application specific.**

**\*/**

**Status = SetupInterruptSystem(&UartLite);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Setup the handlers for the UartLite that will be called from the**

**\* interrupt context when data has been sent and received, specify a**

**\* pointer to the UartLite driver instance as the callback reference so**

**\* that the handlers are able to access the instance data.**

**\*/**

**XUartLite\_SetSendHandler(&UartLite, SendHandler, &UartLite);**

**XUartLite\_SetRecvHandler(&UartLite, RecvHandler, &UartLite);**

**/\***

**\* Enable the interrupt of the UartLite so that interrupts will occur.**

**\*/**

**XUartLite\_EnableInterrupt(&UartLite);**

**}**

**u8 isPressed(){**

**return pressed;**

**}**

**void setPressed(u8 newVal){**

**pressed = newVal;**

**}**

**u8 intr\_test(){**

**/\***

**\* Start receiving data before sending it since there is a loopback.**

**\*/**

**XUartLite\_Recv(&UartLite, ReceiveBuffer, TEST\_BUFFER\_SIZE);**

**int i = 0;**

**for (i = 0; i < 5; i++){**

**xil\_printf("byte %d is %c\n\r", i, ReceiveBuffer[i]);**

**}**

**u8 temp;**

**if ((char) ReceiveBuffer[0] == 'B') {**

**temp = ReceiveBuffer[1];**

**if ((char) ReceiveBuffer[2] == '0')**

**temp = 0;**

**} else if ((char) ReceiveBuffer[1] == 'B') {**

**temp = ReceiveBuffer[2];**

**if ((char) ReceiveBuffer[3] == '0')**

**temp = 0;**

**} else {**

**temp = 0;**

**}**

**int Index;**

**TotalReceivedCount = 0;**

**TotalSentCount = 0;**

**for (Index = 0; Index < TEST\_BUFFER\_SIZE; Index++) {**

**ReceiveBuffer[Index] = 0;**

**}**

**return temp;**

**}**

**void SendHandler(void \*CallBackRef, unsigned int EventData)**

**{**

**TotalSentCount = EventData;**

**}**

**void RecvHandler(void \*CallBackRef, unsigned int EventData)**

**{**

**TotalReceivedCount = EventData;**

**}**

**int SetupInterruptSystem(XUartLite \*UartLitePtr)**

**{**

**int Status;**

**/\***

**\* Initialize the interrupt controller driver so that it is ready to**

**\* use.**

**\*/**

**Status = XIntc\_Initialize(&InterruptController, INTC\_DEVICE\_ID);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Connect a device driver handler that will be called when an interrupt**

**\* for the device occurs, the device driver handler performs the**

**\* specific interrupt processing for the device.**

**\*/**

**Status = XIntc\_Connect(&InterruptController, UARTLITE\_INT\_IRQ\_ID,**

**(XInterruptHandler)XUartLite\_InterruptHandler,**

**(void \*)UartLitePtr);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Start the interrupt controller such that interrupts are enabled for**

**\* all devices that cause interrupts, specific real mode so that**

**\* the UartLite can cause interrupts through the interrupt controller.**

**\*/**

**Status = XIntc\_Start(&InterruptController, XIN\_REAL\_MODE);**

**if (Status != XST\_SUCCESS) {**

**return XST\_FAILURE;**

**}**

**/\***

**\* Enable the interrupt for the UartLite device.**

**\*/**

**XIntc\_Enable(&InterruptController, UARTLITE\_INT\_IRQ\_ID);**

**/\***

**\* Initialize the exception table.**

**\*/**

**Xil\_ExceptionInit();**

**/\***

**\* Register the interrupt controller handler with the exception table.**

**\*/**

**Xil\_ExceptionRegisterHandler(XIL\_EXCEPTION\_ID\_INT,**

**(Xil\_ExceptionHandler)XIntc\_InterruptHandler,**

**&InterruptController);**

**/\***

**\* Enable exceptions.**

**\*/**

**Xil\_ExceptionEnable();**

**return XST\_SUCCESS;**

**}**