External Interrupt Lab

Nadim Samir Sarras.
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: nadim.sarras@gmail.com

Abstract

The purpose of this lab is to be able to utilize the TSOP1236 to measure human response time. The overall design included using LED's to begin the overall and test and to signify whenever the TSOP1236 received a signal. The time delay is then recorded and the standard deviation and average are calculated. Interrupts were sent to the TSOP device through a t.v. remote.

1. Introduction

According to Dr. Li's lecture notes [1], this assignment revolved around the use of signal interrupts and the TSOP device. This report will breakdown the methodology and steps taken to fulfill the objective of this lab. The objective of this lab was to use the TSOP device and external interrupts as a tool to measure human response time. Specifically, this was achieved by using a television remote to send an external interrupt to the TSOP device on the prototype board, once the TSOP receives the interrupt it sends a signal to the LPC1769 Module. Depending on the signal received the LPC1769 will then turn LED's on and off using general purpose pins.

2. Methodology

The approach to completing the objective for the assignment, as well as technical challenges, and external problems, are outlined in the sections below.

2.1. Objectives and Technical Challenges

Objective 1: Light up a LED when the test is about to begin. The test must begin using a random interval.

Objective 2: A 2nd LED will light up after the random delay and will toggle on and off when an external interrupt is received by the TSOP device.

Objective 3: Detect time intervals of external interrupts received from a television remote. Using all of the recorded times, come up with the average and standard deviations of the external interrupt response times. The time begins to count directly after the 2nd LED has been turned on.

Technical Difficulties: Technical Difficulties in making sure the TSOP device was able to receive external interrupt signal from a remote control.

2.2. Problem Formulation and Design

The following sections will provide the design and implementation methods for this lab. The Hardware Design section will include system block diagrams, schematics, and a bill of material. The schematics will represent the design created on the prototype board while the block diagrams will visualize the functional relationships between the different modules. The Software Design incorporates flow charts, algorithms, and pseudo code. The flow charts visualize the software process of the program. Algorithms and pseudo code act as a description of what the source code is implementing.

3. Implementation

This section will go over the steps taken as well as the thought process behind the hardware and software design of the project

3.1. Hardware Design

The Hardware Designed involved determining which pins would be utilized for certain functions. In this lab, it was chose that pin J6-23 and pin J6-24 were to be used for the LED outputs. Resistors were tied to the outputs to not allow the LED to burnout. The TSOP was chosen to be connected to pin J6-52, which was set as an input. The TSOP required a capacitor between the voltage source and the ground in order to help eliminate random interrupts received. A 1kohm resistor was used to help regulate the voltage entering the TSOP device.

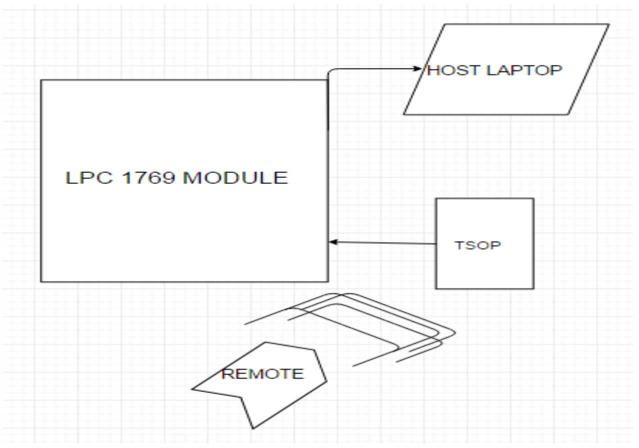


Figure 1: Interconnection Block Diagram

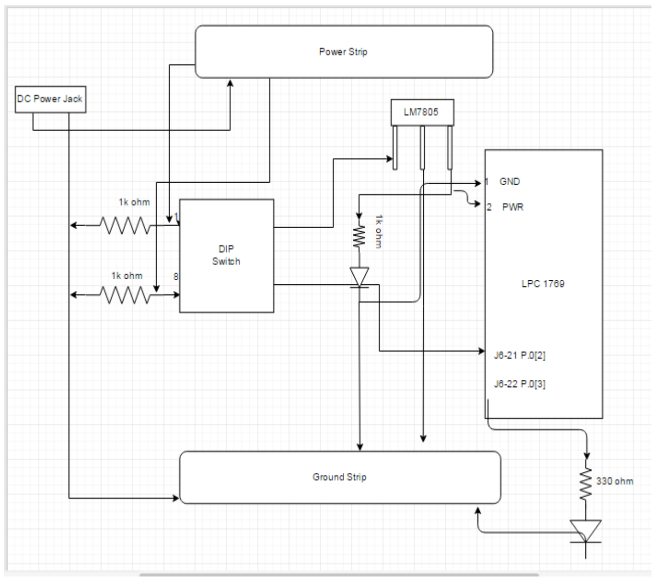


Figure 2: Power Circuit Schematic

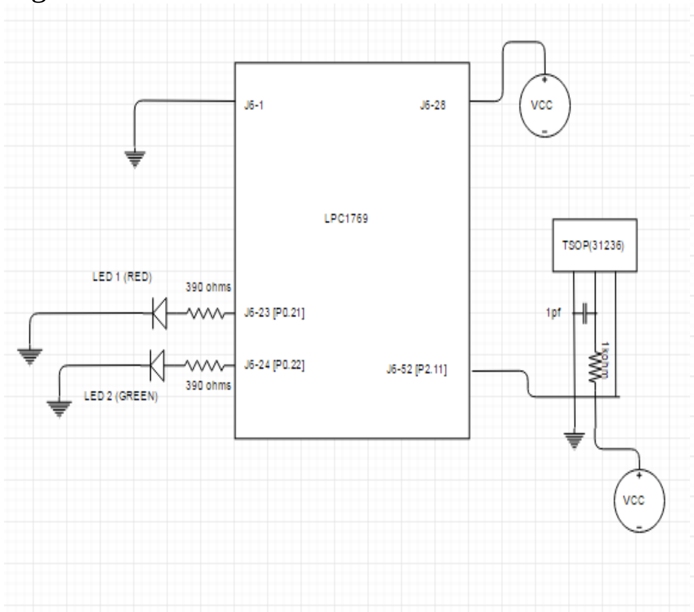


Figure 3: LPC Board to TSOP Connection Schematic

Table 1. Pin Usage

Functional Name	Pin Number	Purpose	CPU Model LPC 1769 Port Num / Pin
GND	J6-1	Ground	GNDX
GPP	J6-23	TOGGLE LED1	Port 0/ Pin 21
GPP	J6-24	TOGGLE LED 2	Port 0/ Pin 22
Interrupt Pin	J6-52	Interrupt reciever	Port 2/Pin 11
VOUT	J6-28	Output Power	VIO_3V3X

Table 2. Bill of Materials

Part	Function
LPC1769CD	CPU Module
TSOP 31236	IR Reciever
Power Supply, 9v/2A	Power Circuit
USB Cable	Loading Program/ Debugging

3.2. Software Design

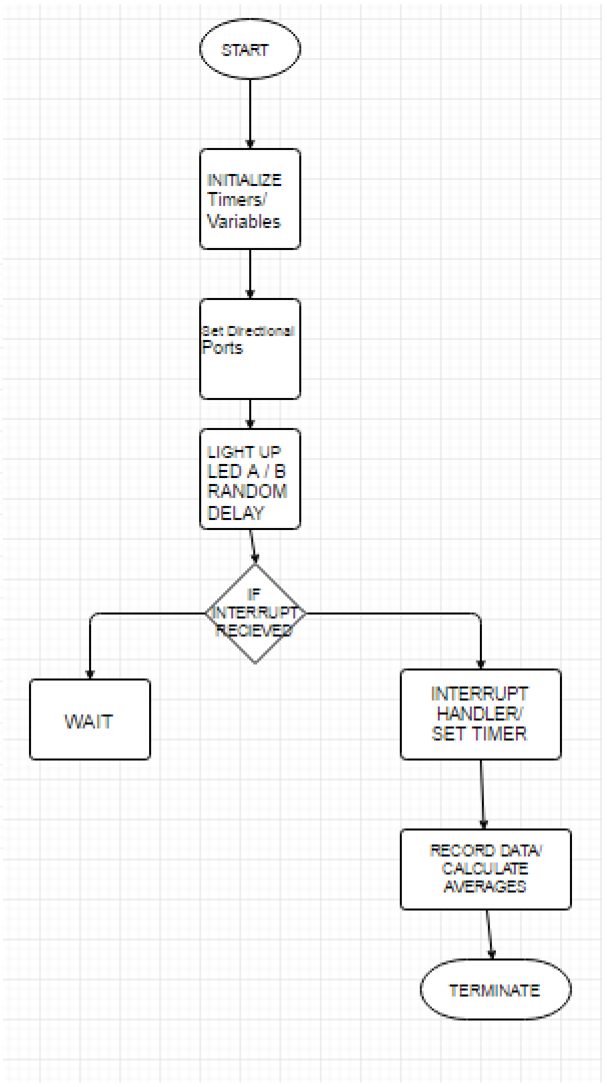

START
INITIALIZE Timers/ Variables
Set Directional Ports
LIGHT UP LED A / B RANDOM DELAY
IF INTERRUPT RECIEVED
WAIT
INTERRUPT HANDLER/ SET TIMER
RECORD DATA/ CALCULATE AVERAGES
TERMINATE

Figure 4: Software Process Flowchart

Pseudo Code

```
//Set Directions of Ports
//Initialize used Pins
Generate rand ( ) function
Delay(rand_number)
//FIOCLR to Turn on Test LED // Active Low
```

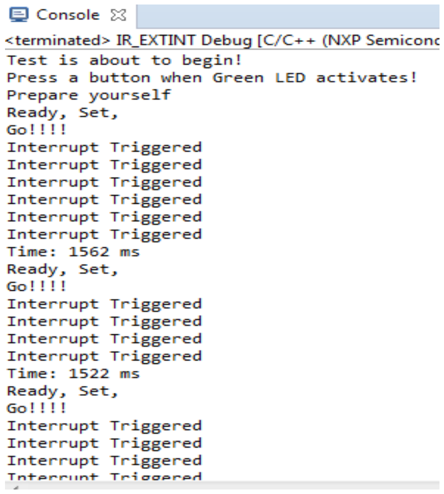
Start Test
EINTInit( );
TOGGLE LED
LPC_GPIO0->FIOCLR |= (1<<22);//turn on LED2
Restart Test, Once Finished. Continue.
Store = time_taken
Calculate average
Calculate STD. Deviation
Turn OFF LED

Algorithm

In order to successfully capture interrupts and to calculate the time delays, some functions had to be extracted from other header files, such as Eint,h and timer.h. The code begins by initializes and determining the inputs and outputs of the respective ports. Then using the srand function a random number is generated to begin the test. The RED LED is then set to clear since it is active low to turn it on. This signifies that the test has begun. The GREEN LED is then toggled with FIOSET and FIOCLR to show when the TSOP receives an external interrupt. The time recorded is extracted from the counter implemented within the interrupt handler function, EINTInit(); After the test loops multiple times, all of the recorded times are stored into an array. Using these recorded responses the average time delay is calculated, as well as the standard deviation between the data points.

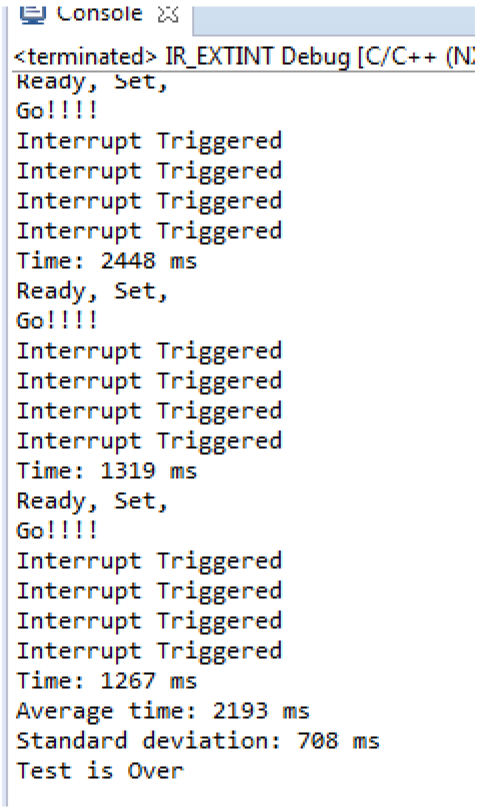
4. Testing and Verification

The test was successful as shown by the console outputs. The test waits for the GO statement to be printed and then the GREEN LED lights up to signify the user to send an external interrupt with a remote control. These signals are received and used to calculate the average and standard deviation.



```
<terminated> IR_EXTINT Debug [C/C++ (NXP Semicond
Test is about to begin!
Press a button when Green LED activates!
Prepare yourself
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Time: 1562 ms
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Time: 1522 ms
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
```

Figure 5: Console Output



```
<terminated> IR_EXTINT Debug [C/C++ (N
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Time: 2448 ms
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Time: 1319 ms
Ready, Set,
Go!!!!
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Interrupt Triggered
Time: 1267 ms
Average time: 2193 ms
Standard deviation: 708 ms
Test is Over
```

Figure 6: Console Output

5. Conclusion

In conclusion, this lab was very effective in understanding how to handle interrupts, as well as being able to create a working device to measure human response time. Understanding the active low and active high status of different pins on the LPC Module was crucial to this lab. Some debugging was done to make sure the TSOP device was functioning properly. In order to test for this the voltage was measured across the device while an external interrupt was sent to the device. If the voltage dropped, then the TSOP device successfully received a interrupt. This lab has provided the basis to using external interrupts and understanding how to capture them.

6. Acknowledgement

N/A

7. References

[1] H. Li, "Author Guidelines for CMPE 146/242 Project Report", *Lecture Notes of CMPE 146/242*, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.

8. Appendix

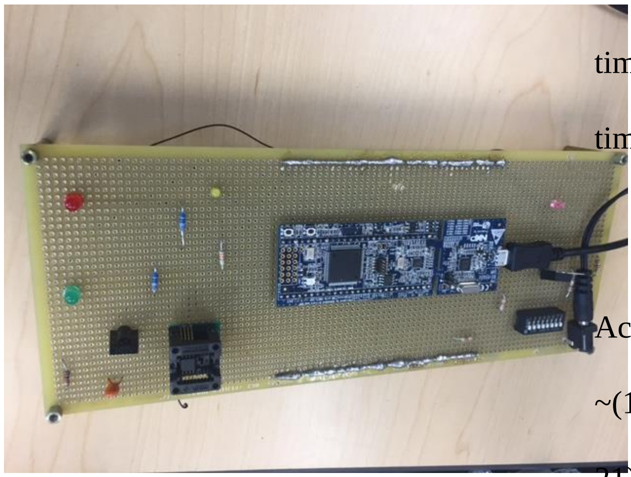


Figure 9: Prototype Board Config.

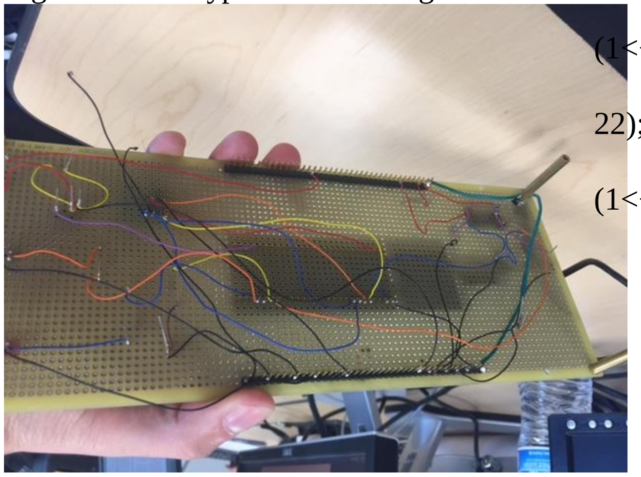


Figure 10: Prototype Board Wiring

Source Code: IR_EXTINT.C

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#include "timer.h"
#include "uart.h"
#include "extint.h"
#include "type.h"
#include <math.h>
#include <time.h>

#endif

#include <cr_section_macros.h>

#include <stdio.h>
```

```
	extern uint32_t timer0_m0_counter,
timer1_m0_counter;
	extern uint32_t timer0_m1_counter,
timer1_m1_counter;

	int main(void)
	{
		// Both PO.21 & PO.22 Are
Active Low outputs
		LPC_GPIO0->FIODIR &=
~(1<<11); //set pin 2.11 as input
		LPC_GPIO0->FIODIR |= (1 <<
21);
		LPC_GPIO0->FIOSET |=
(1<<21);
		LPC_GPIO0->FIODIR |= (1 <<
22);
		LPC_GPIO0->FIOCLR |=
(1<<22);



		srand(time(NULL));
		int random;
		int i;
		int i2;
		int storage[4];//Time Waited for
Interrupt
		int average;//average
		int var[4];//variance
		int std_dev;//standard deviation


		LPC_GPIO0->FIOSET |=
(1<<22);
		printf("Test is about to
begin!\n");
		LPC_GPIO0->FIOCLR |=
(1<<22);
		printf("Press a button when
Green LED activates!\n");

		i = 0;
		printf("Prepare yourself\n");
		delayMs(0,10000);
		LPC_GPIO0->FIOSET |=
(1<<21);//Turn off LED1

		while(i < 6)
		{
			printf("Ready, Set,\n");
			random = rand() %
10000;//random number
			delayMs(0,random); //
Incorporate delay of random number in ms
			LPC_TIM1->TCR =
0x02;// Obtained from Extint.h, initializes timer
and interrupts
			LPC_TIM1->PR  = 0x00;
			LPC_TIM1->IR  = 0xff;
			LPC_TIM1->TCR =
0x01;
			LPC_GPIO0->FIOSET |=
(1<<22);//turn off LED2  CLR
			printf("Go!!!!\n");
			while(LPC_TIM1->TCR
== 0x01)
			{
				EINTInit();
//wait for external interrupt
				storage[i] =
LPC_TIM1->TC;  // Store the elapsed time into
storage array
			}
			LPC_GPIO0->FIOCLR
|= (1<<22);//turn on LED2
			storage[i] =
storage[i]/(9000000 / 1000-1);//Extracted from
Timer.h, returns time elapsed in ms
			printf("Time: %i
ms\n",storage[i]);
			i++;
		}

		LPC_GPIO0->FIOSET |=
(1<<21);//turn off LED1
		average =
(storage[0]+storage[1]+storage[2]+storage[3])/4
;//calculate average
		for(i2 = 0;i2<4;i2++)//calculate
variance for each value
		{
			var[i2] = storage[i2] -
average;
			var[i2] = var[i2]*var[i2];
		}

		std_dev =
sqrt((var[0]+var[1]+var[2]+var[3])/4);//calculate
standard deviation
		printf("Average time: %i
ms\n",average);
		printf("Standard deviation: %i
ms\n",std_dev);
		printf("Test is Over\n");
		LPC_GPIO0->FIOCLR |=
(1<<21);

		return 0;
	}
```