Using TSOP2236 to Measure Human Responses

Anahit Sarao
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: indianvip60@gmail.com

Abstract

This report will go through the full project which interfaces a TSOP2236 with an LPC1769 microcontroller. The TSOP2236 is a receive for infrared signals. The usage of the IR receiver is to generate an interrupt every time a signal is sent form the IR transmitter which can be any device such as a remote control. The microcontroller will process the signal as an interrupt to illuminate LEDs. The microcontroller is coded to measure the time of the signal activating the LED vs the microcontroller activating the led. A separate LED is used to signal the microcontroller will blink the led once at a random moment then wait for a response in the form of IR.

1. Introduction

The point of this lab is to measure human response time using several different parts. The response testing will be performed by using a microcontroller, IR sensor, IR transmitter. A simple television remote was used as the transmitter. The receiver was connected to the interrupt pin while the microcontroller drove two different LEDs which represented two different states of the program. The frequency range is hard to configure this is why a low pass filter was used to filter out lower frequencies as noise.

2. Methodology

To provide an elaborate but concise and effective report; steps were taken to achieve high quality work. This document is the representation of all the designs, material, documentation produced by the author stated above.

2.1. Objectives and Technical Challenges

To be able to test CPU interrupts through an external signal requires the external signal and an output. The external signal was sent and received using the ISP_EN port. This sends a direct signal to the interrupt vector. Ports for driving the LED’s included GPIO ports which would alternate on and off to indicate if tests were successful or not. The code was written inside LPCX using the provided header files. The whole test is completed within milliseconds which can be hard for a user tocapture custom code was written to convert time from millisecond to seconds

2.2. Problem Formulation and Design

The interface used here was IR and GPIO. To be able to capture an IR signal code and hardware was constructed. The console was used to display either debugging information or actual output of the test in progress. The receiver was soldered onto the board to allow a steady connection and smooth signal capture. Below is the bill of materials used for this project.

Bill of Material	
Qty	Item Description
1 pc	LPC1769 Module
1 pc	TSOP 2236 Infrared receiver
1 pc	1000 Ohm Resistor
1 pc	0.47uF Capacitor
1 pc	Wire wrapping tool
1 pc	Remote control

Table 1. Bill of Material

3. Implementation

The following sections provide in detail technical and analytical information. This includes hardware and software sections.

3.1. Hardware Design

Figure one and two provide shows the overall connections and schematics of the multiple devices used for this project. The IR module is connected to the microcontroller using selected pins. The project is powered by the microcontroller is self.

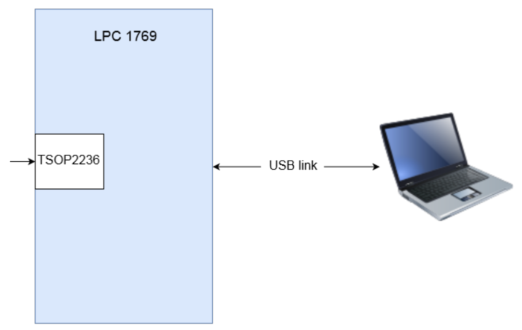


Figure 1. Overall Block Diagram for Hardware

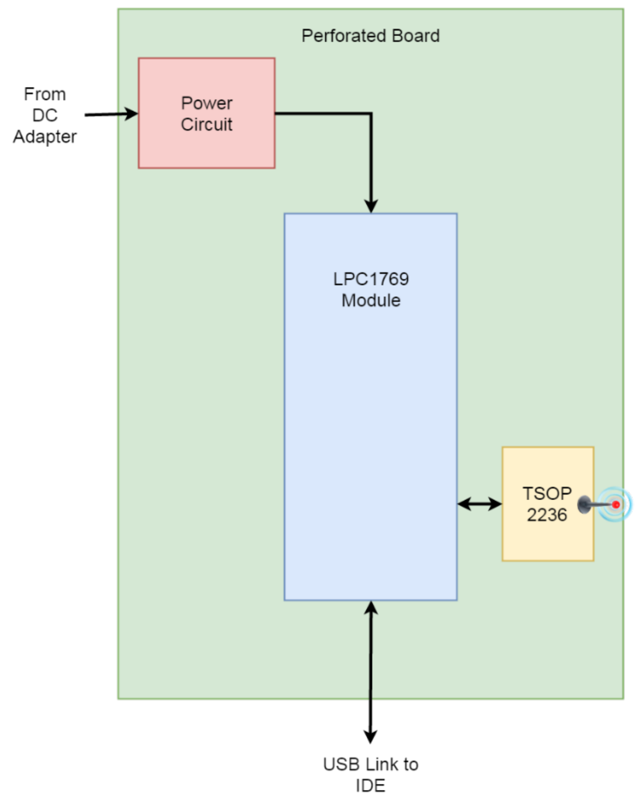


Figure 2. Circuit Schematic Block Diagram

The pins used in this lab are stated below in table two. The pins from the TSOP output goes directly into the Interrupt port 1. The GPIO ports are used to control the LEDs.

TSOP 2236			LPC 1769		
Pin No.	Pin Label	Description	Pin No.	Pin Label	Description
1	Out	Signal Output	J6-42	P2[0]	GPIO Output
2	Vcc	Power Supply	J6-46	P2[4]	GPIO Output
3	Gnd	Ground	J6-51	P2[10]	Interrupt Port 1

Table 1. Pin Connections.

$$f = \frac{1}{2\pi RC}$$

Eq. 1 Frequency Equation

Equation one helps determine the capacitor and resistor values used for the TSOP low pass filter circuit. The filter circuit helps reduce noise of frequencies which are lower than the threshold provided from the data sheet. The capacitor had a 0.47uF capacity and resistance of 1000Ohms. Figure three provides the schematic for the TSOP connections to the low pass filter. These components filter out frequencies below 338Hz

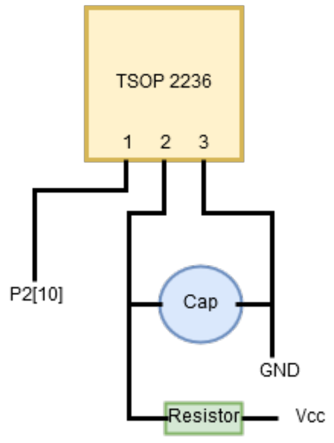


Figure 3. Schematic for TSOP and Low Pass Filter

3.2. Software Design

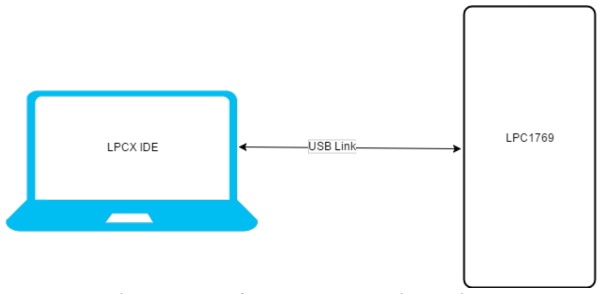


Figure 4. Software Connection Diagram

Figure four shows the overall connection between the computer and the LPC microcontroller. The software design was very complex and it involved heavy use of port triggering and loops. Below is some pseudo code for some of the main functions created.

```
main(void)
        Clear all pins
        Set needed pins
        Begin input reading for IR
        printf(IR signal to BEGIN TEST)
        prepareData()
        while (testing())
        printf(Test Completed)
        exit
        return 0}
```

main(void) {

//initializing the GPIO port direction
//clearing all the GPIO port so that the connected LED's are off initially
// Start GPIO ports is done in the function which is defined in the same file.
//All the functions and timers are also set in this function together with the wait and cycling through of the states.

return 0}

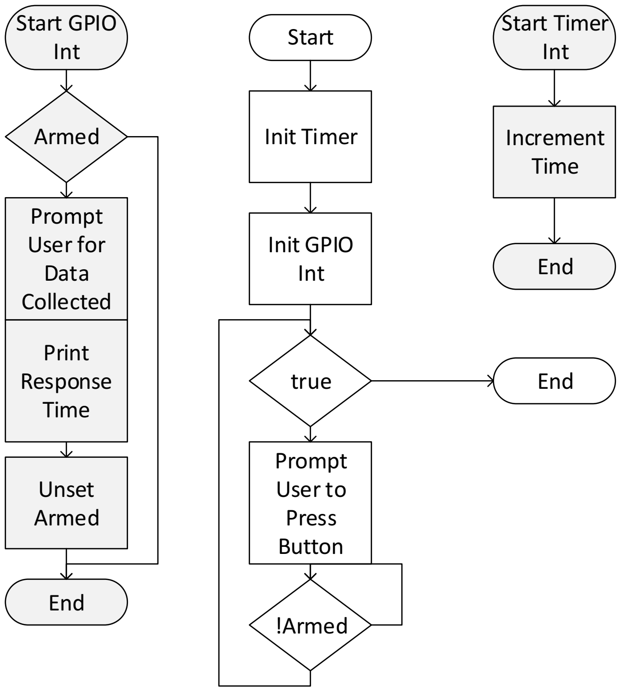


Figure 5. Flowchart for code.

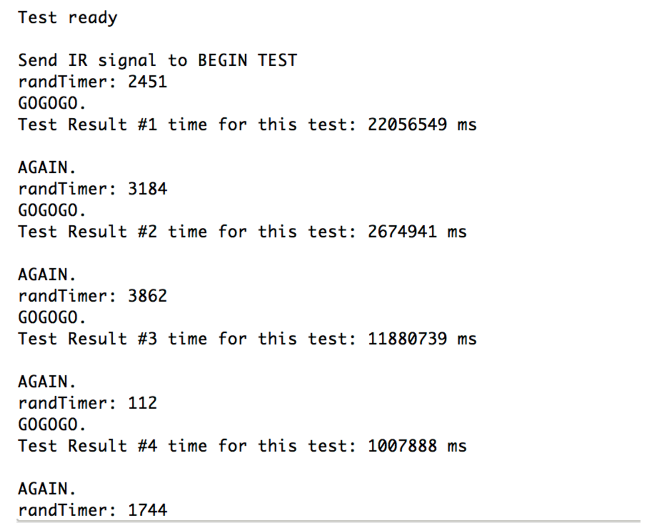
The figure above shoes the flow chart of the code. The code is given a command to go and beings the test. The timers are started and turned off for every test. In the end the result is stored and calculations are done to provide the standard deviation and average response time.

4. Testing and Verification

The steps taken for verifying the hardware and software cohesion are list as:

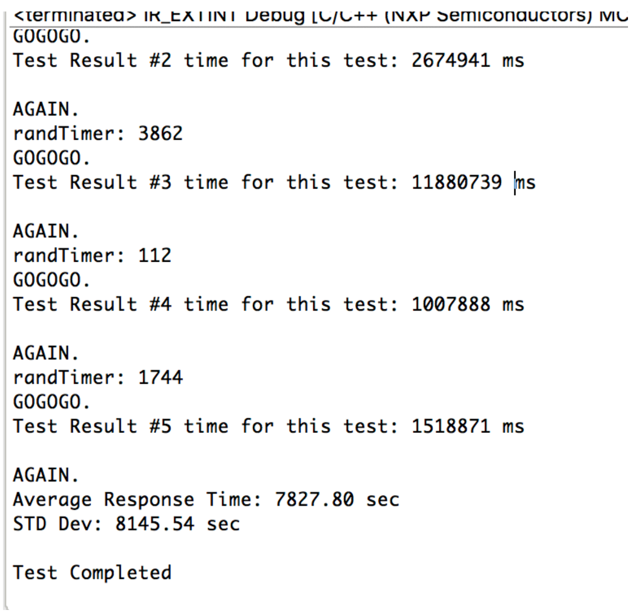
1. Load code onto LPC board
2. Start the Program
3. Read the IR signal
4. Repeat 5 times
5. Calculate standard deviation and average
6. Exit

When everything was checked out then the program is looped back to the beginning signifying that it is ready for another test. Overall this lab was very successful in the testing and presenting parts. By following these steps, it was verified that the board was programmed and the hardware was working correctly. Figure six and seven show all the tests being completed.



```
Test ready

Send IR signal to BEGIN TEST
randTimer: 2451
GOGOGO.
Test Result #1 time for this test: 22056549 ms

AGAIN.
randTimer: 3184
GOGOGO.
Test Result #2 time for this test: 2674941 ms

AGAIN.
randTimer: 3862
GOGOGO.
Test Result #3 time for this test: 11880739 ms

AGAIN.
randTimer: 112
GOGOGO.
Test Result #4 time for this test: 1007888 ms

AGAIN.
randTimer: 1744
```

Figure 6. Console Output



```
<terminated> IR_EXTINT Debug [C/C++ (NXP Semiconductors) MC
GOGOGO.
Test Result #2 time for this test: 2674941 ms

AGAIN.
randTimer: 3862
GOGOGO.
Test Result #3 time for this test: 11880739 ms

AGAIN.
randTimer: 112
GOGOGO.
Test Result #4 time for this test: 1007888 ms

AGAIN.
randTimer: 1744
GOGOGO.
Test Result #5 time for this test: 1518871 ms

AGAIN.
Average Response Time: 7827.80 sec
STD Dev: 8145.54 sec

Test Completed
```

Figure 7. Console Output

The verification for this lab is successful as the console displays the standard deviation and average.

5. Conclusion

In conclusion, this project was successful in helping understand and get familiar with LPCX and the interrupts inside of microcontrollers. The board can be used to control USB devices and such as they all use interrupts. This is a key function which is built into almost all devices in modern day embedded electronics. A more advance design would be to add parallel process and interrupts so that multiple devices can be used. The TSOP IR sensor opens up so much more options and shows how useful microprocessors are in the modern day. This experience was very positive and educating even though there were many problems such as wires detaching, connections touching other junctions and the board not being fully utilized by the IDE. Additional figures are given within the appendix to show the functionality and verification of this lab.

6. Acknowledgement

Provide acknowledgement if needed, such as support, help, or assistance from someone. These support, help, assistance are crucial.

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.

7. Appendix

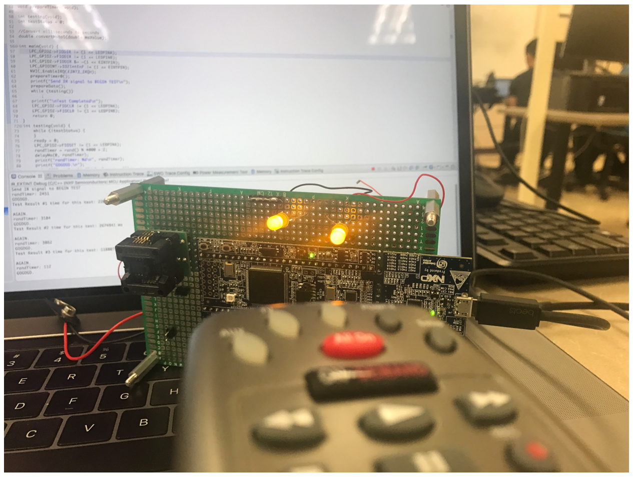


Figure 8. Picture of Board

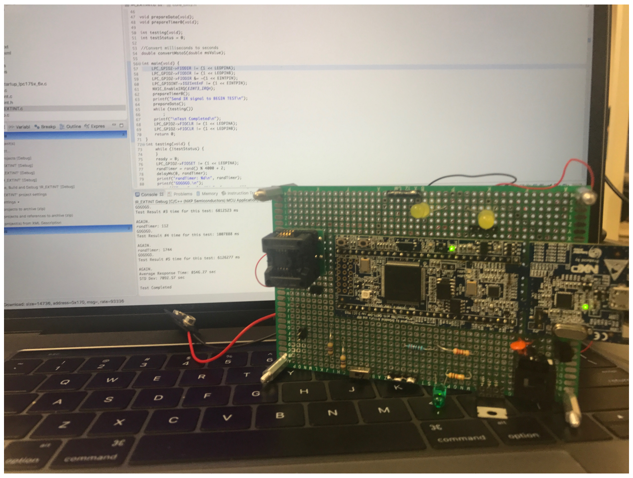


Figure 9. Picture of Board

```
/*
 ============================================
===================================
 Name        : IR_EXTINT.c
 Author      : $(author)
 Version     :
 Copyright   : $(copyright)
 Description : main definition
 ============================================
===================================
 */

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

#include <cr_section_macros.h>

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
// TODO: insert other include files here

// TODO: insert other definitions and
declarations here

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

#define LEDPINA             0
#define LEDPINB             4
#define EINTPIN             10

//Iterations before human response
#define COUNTDOWN    5

//Response Time Attributes
int count = 0;
double resultBuffer[4];
double averageTime = 0;
double variance = 0;
int randTimer = 0;
int ready = 0;

void prepareData(void);
void prepareTimer0(void);

int testing(void);
int testStatus = 0;

//Convert milliseconds to seconds
double convertMstoS(double msValue);

int main(void) {
	LPC_GPIO2->FIODIR |= (1 << LEDPINA);
	LPC_GPIO2->FIODIR |= (1 << LEDPINB);
	LPC_GPIO2->FIODIR &= ~(1 << EINTPIN);
	LPC_GPIOINT->IO2IntEnF |= (1 <<
EINTPIN);
	NVIC_EnableIRQ(EINT3_IRQn);
	prepareTimer0();
	printf("Send IR signal to BEGIN
TEST\n");
	prepareData();
	while (testing())
		;
	printf("\nTest Completed\n");
	LPC_GPIO2->FIOCLR |= (1 << LEDPINA);
	LPC_GPIO2->FIOCLR |= (1 << LEDPINB);
	return 0;
}
int testing(void) {
	while (!testStatus) {
	}
	ready = 0;
	LPC_GPIO2->FIOSET |= (1 << LEDPINA);
	randTimer = rand() % 4000 + 2;
	delayMs(0, randTimer);
	printf("randTimer: %d\n", randTimer);
	printf("GOGOGO.\n");
	printf("Test Result #%d ", (count +
1));
	LPC_GPIO2->FIOSET |= (1 << LEDPINB);
	LPC_TIM0->TCR = 0x02;
	LPC_TIM0->TCR = 0x01;
	while (!ready) {
	}
	if (count >= COUNTDOWN) {
		int n = 0;
		while (n < COUNTDOWN) {
			averageTime +=
resultBuffer[n];
			n++;
		}
		averageTime = averageTime /
COUNTDOWN;
		double averageTimeSec =
convertMstoS(averageTime);
		printf("Average Response Time:
%.2f sec\n", averageTimeSec);
		n = 0;
		while (n < COUNTDOWN) {
			variance +=
(resultBuffer[n] - averageTime)

	* (resultBuffer[n] - averageTime);
			n++;
		}
		variance = variance /
COUNTDOWN;
		double stdrdDev =
sqrt(variance);
		double stdrdDevSec =
convertMstoS(stdrdDev);
		printf("STD Dev: %.2f sec\n",
stdrdDevSec);
		return 0;
	} else {
		return 1;
	}
}
double convertMstoS(double msValue) {
	double sValue = msValue / 1000;
	return sValue;
}
void prepareTimer0(void) {
	LPC_SC->PCONP |= (1 << 1);
	LPC_TIM0->PR = (SystemCoreClock /
4000) - 1;
	LPC_TIM0->TCR = 0x00;
}
void prepareData(void) {
	count = 0;
	averageTime = 0;
	variance = 0;
	resultBuffer[4] = 0;
}
void EINT3_IRQHandler(void) {
	if (testStatus == 0) {
		printf("Test ready\n\n");
		testStatus = 1;
	} else {
		int state = (LPC_GPIO2->FIOPIN
& (1 << LEDPINB) ? 1 : 0);
		if (state) {
			LPC_GPIO2->FIOCLR |= (1
<< LEDPINB);
			LPC_TIM0->TCR = 0x00;
			int timerCount =
LPC_TIM0->TC;
			printf("time for this
test: %d ms\n", timerCount);
			resultBuffer[count] =
(double) timerCount;
			count++;
			ready = 1;
			LPC_GPIO2->FIOSET |= (1
<< LEDPINA);
			printf("\nAGAIN.\n");
		}
		LPC_GPIOINT->IO2IntClr |= (1 <<
EINTPIN);
	}
}
```