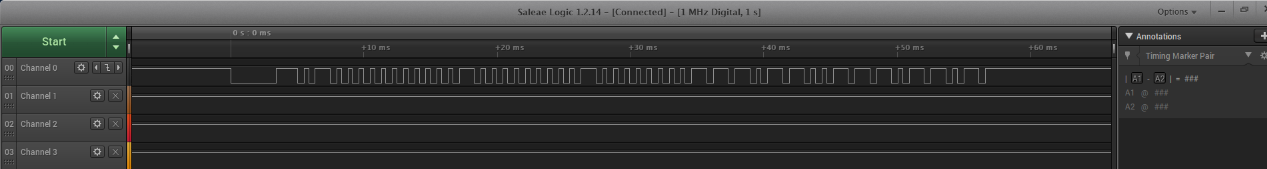
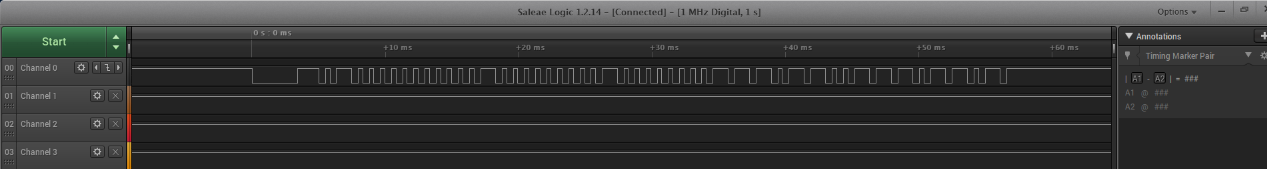
Lab Report 3

Honglei Liu

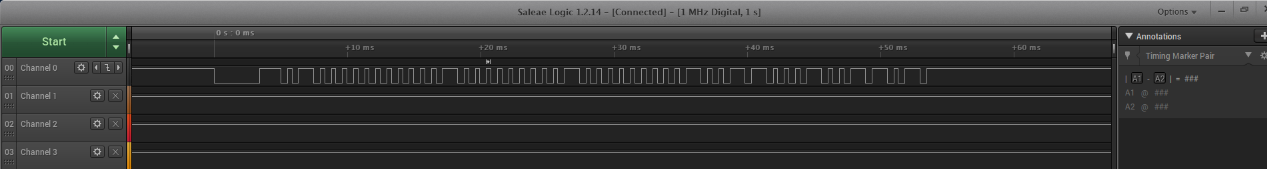
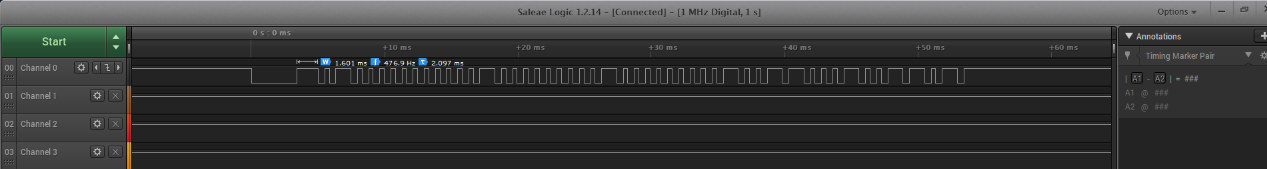
Bingxuan Ying

ScreenShots of the Waveform

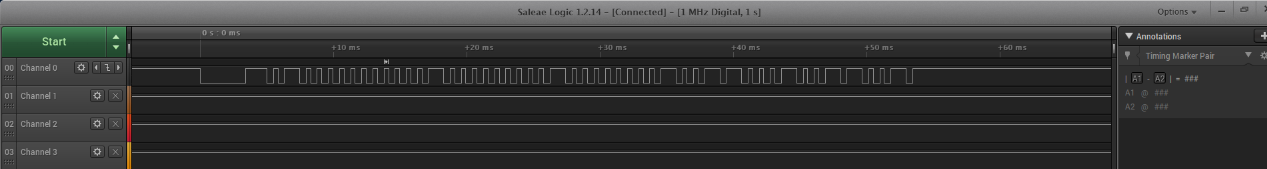
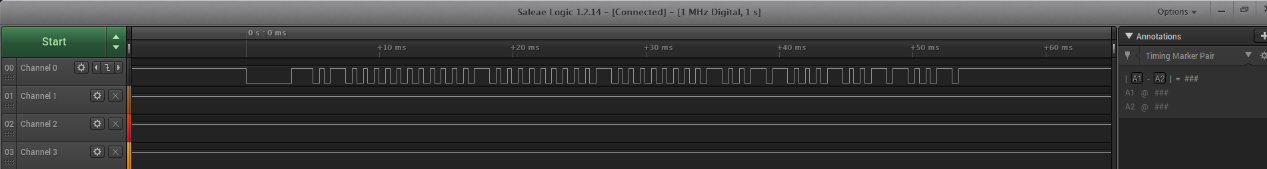
LAST: 0b1010000000000010000000001000000001110110011101101



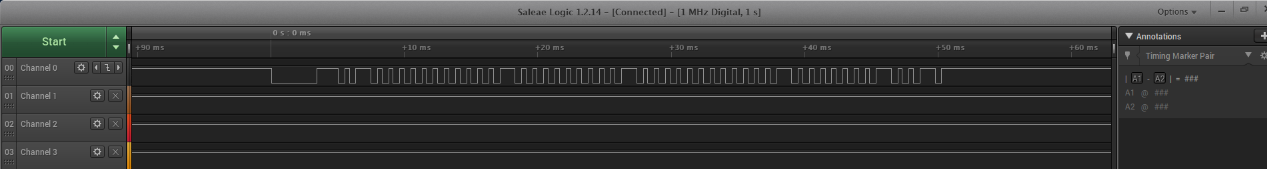
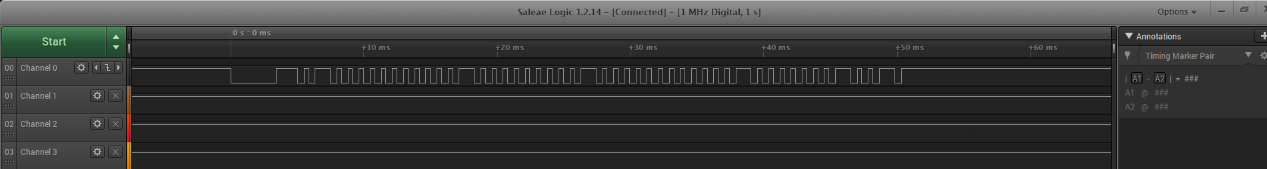
MUTE: 0b1010000000000010000000001000000000100110001001101



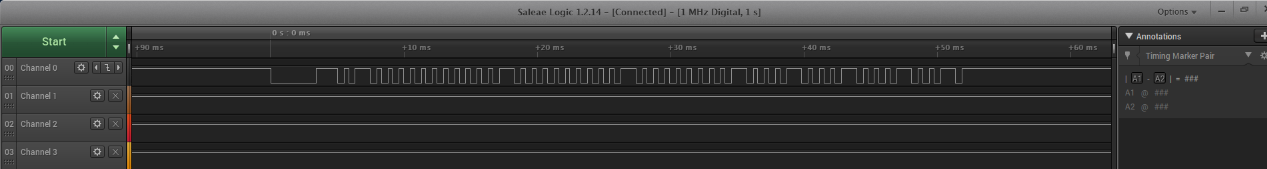
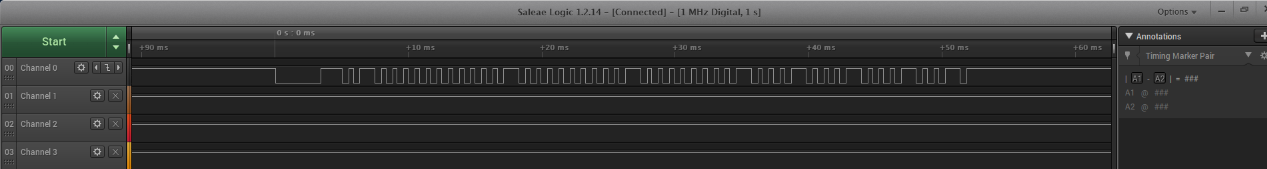
Button 0: 0b1010000000000010000000001000000001001100010011001



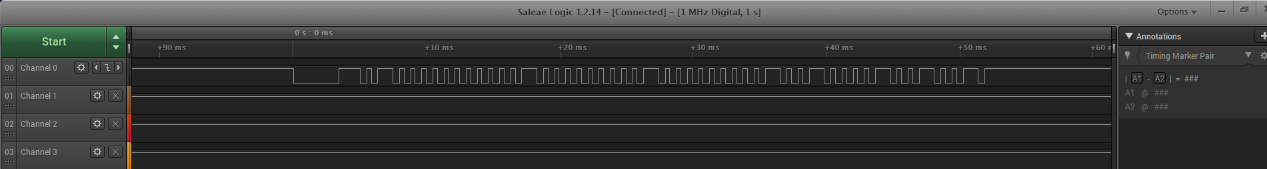
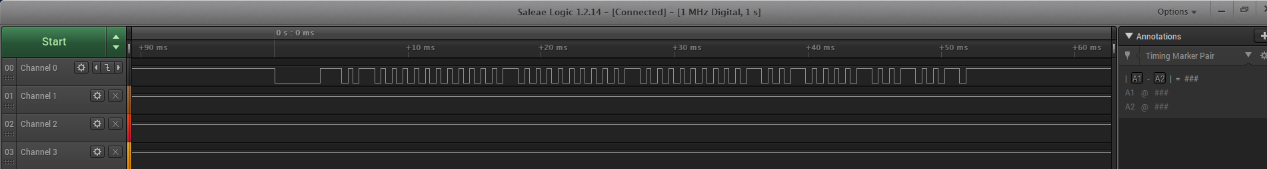
Button 1: 0b1010000000000010000000001000000000000100000001001



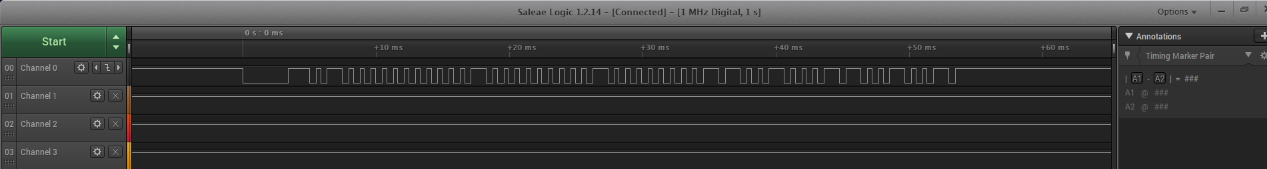
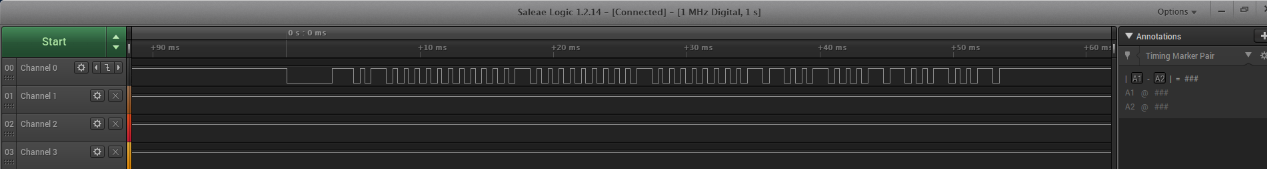
Button 2: 0b1010000000000010000000001000000001000100010001001



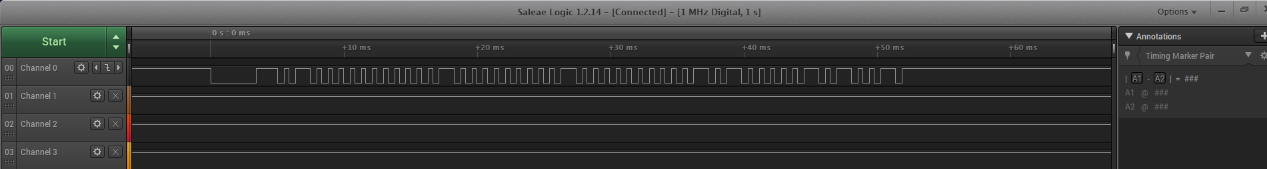
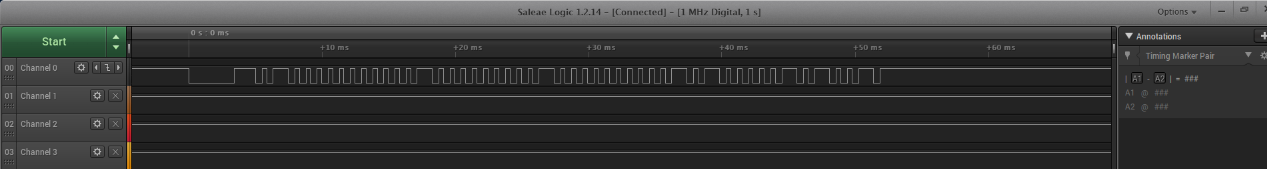
Button 3: 0b1010000000000010000000001000000000100100001001001



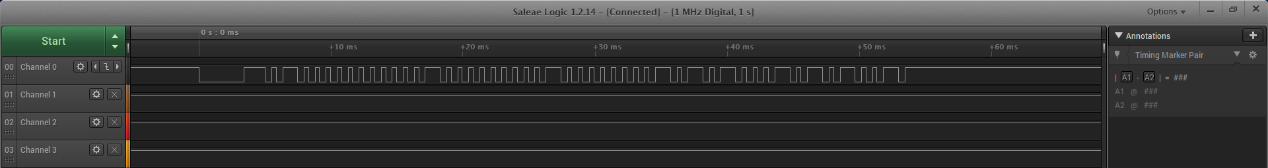
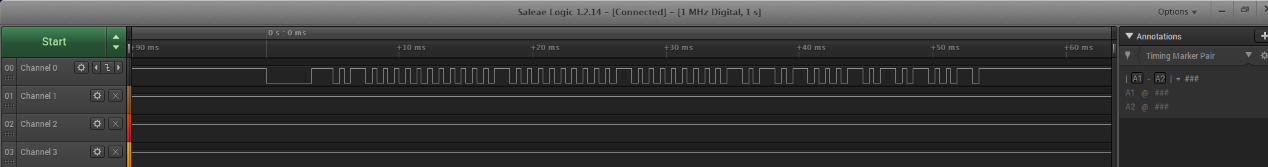
Button 4: 0b1010000000000010000000001000000001100100011001001



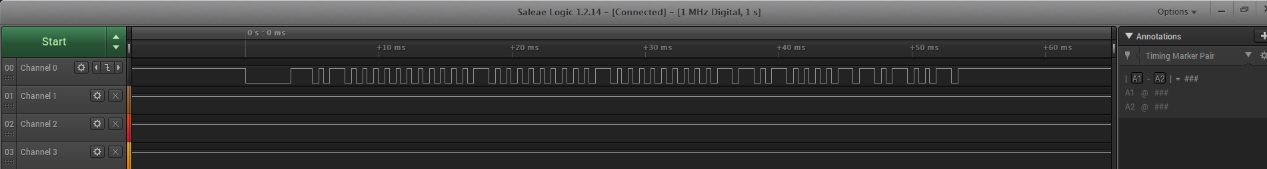
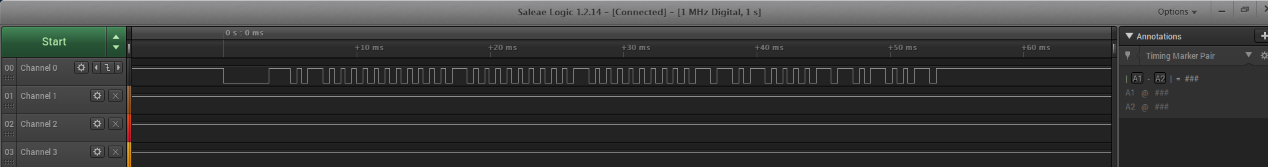
Button 5: 0b1010000000000010000000001000000000010100000101001



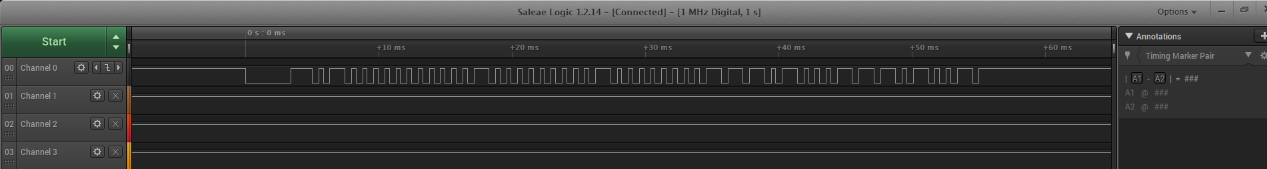
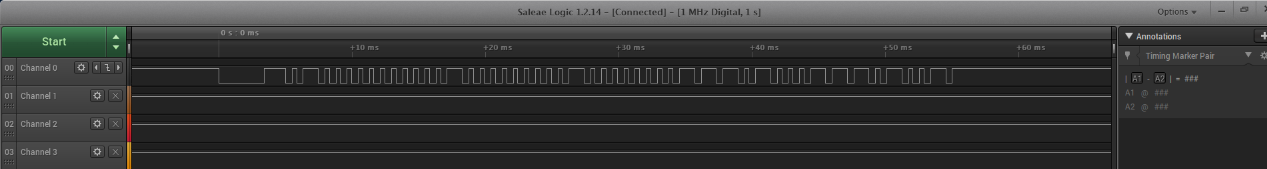
Button 6: 0b1010000000000010000000001000000001010100010101001



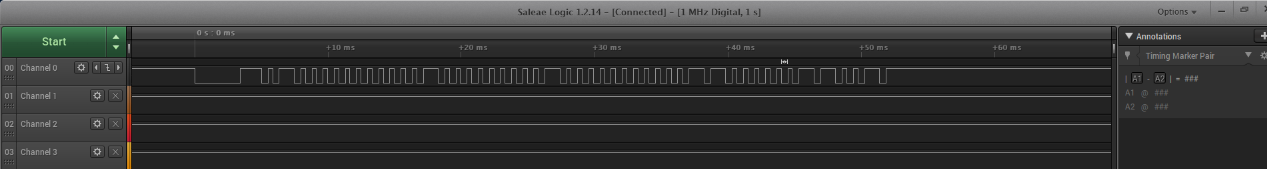
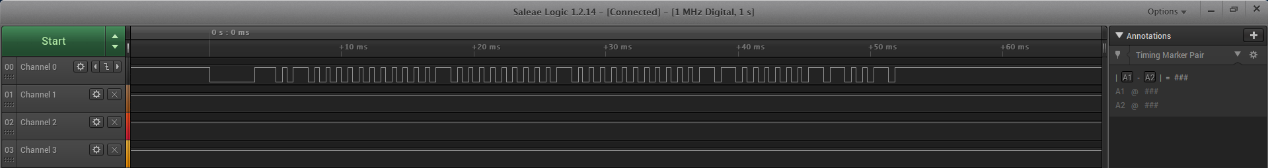
Button 7: 0b1010000000000010000000001000000000110100001101001



Button 8: 0b1010000000000010000000001000000001110100011101001



Button 9: 0b1010000000000010000000001000000000001100000011001



PINMUX.C

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

// pinmux.c

//

// configure the device pins for different peripheral signals

//

// Copyright (C) 2014 Texas Instruments Incorporated - http://www.ti.com/

//

//

// Redistribution and use in source and binary forms, with or without

// modification, are permitted provided that the following conditions

// are met:

//

// Redistributions of source code must retain the above copyright

// notice, this list of conditions and the following disclaimer.

//

// Redistributions in binary form must reproduce the above copyright

// notice, this list of conditions and the following disclaimer in the

// documentation and/or other materials provided with the

// distribution.

//

// Neither the name of Texas Instruments Incorporated nor the names of

// its contributors may be used to endorse or promote products derived

// from this software without specific prior written permission.

//

// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//

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

// This file was automatically generated on 7/21/2014 at 3:06:20 PM

// by TI PinMux version 3.0.334

//

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

#include "pinmux.h"

#include "hw\_types.h"

#include "hw\_memmap.h"

#include "hw\_gpio.h"

#include "pin.h"

#include "rom.h"

#include "rom\_map.h"

#include "gpio.h"

#include "prcm.h"

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

void

PinMuxConfig(void)

{

//

// Set unused pins to PIN\_MODE\_0 with the exception of JTAG pins 16,17,19,20

//

PinModeSet(PIN\_01, PIN\_MODE\_0);

PinModeSet(PIN\_02, PIN\_MODE\_0);

PinModeSet(PIN\_04, PIN\_MODE\_0);

PinModeSet(PIN\_05, PIN\_MODE\_0);

PinModeSet(PIN\_06, PIN\_MODE\_0);

PinModeSet(PIN\_15, PIN\_MODE\_0);

PinModeSet(PIN\_21, PIN\_MODE\_0);

PinModeSet(PIN\_52, PIN\_MODE\_0);

PinModeSet(PIN\_53, PIN\_MODE\_0);

PinModeSet(PIN\_58, PIN\_MODE\_0);

PinModeSet(PIN\_59, PIN\_MODE\_0);

PinModeSet(PIN\_60, PIN\_MODE\_0);

PinModeSet(PIN\_61, PIN\_MODE\_0);

PinModeSet(PIN\_62, PIN\_MODE\_0);

PinModeSet(PIN\_63, PIN\_MODE\_0);

PinModeSet(PIN\_64, PIN\_MODE\_0);

//

// Enable Peripheral Clocks

//

MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA1, PRCM\_RUN\_MODE\_CLK);

MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA2, PRCM\_RUN\_MODE\_CLK);

MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA3, PRCM\_RUN\_MODE\_CLK);

MAP\_PRCMPeripheralClkEnable(PRCM\_GSPI, PRCM\_RUN\_MODE\_CLK);

MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA0, PRCM\_RUN\_MODE\_CLK);

MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA1, PRCM\_RUN\_MODE\_CLK);

//

// Configure PIN\_03 for GPIO Output (DC)

//

MAP\_PinTypeGPIO(PIN\_03, PIN\_MODE\_0, false);

MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x10, GPIO\_DIR\_MODE\_OUT);

//

// Configure PIN\_08 for GPIO Output (R)

//

MAP\_PinTypeGPIO(PIN\_08, PIN\_MODE\_0, false);

MAP\_GPIODirModeSet(GPIOA2\_BASE, 0x2, GPIO\_DIR\_MODE\_OUT);

//

// Configure PIN\_18 for GPIO Output (OC)

//

MAP\_PinTypeGPIO(PIN\_18, PIN\_MODE\_0, false);

MAP\_GPIODirModeSet(GPIOA3\_BASE, 0x10, GPIO\_DIR\_MODE\_OUT);

//

// Configure PIN\_05 for GPIOOutput (IR)

//

MAP\_PinTypeGPIO(PIN\_05, PIN\_MODE\_0, false);

MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x40, GPIO\_DIR\_MODE\_IN);

//

// Configure PIN\_50 for SPI0 GSPI\_CS

//

MAP\_PinTypeSPI(PIN\_50, PIN\_MODE\_9);

//

// Configure PIN\_45 for SPI0 GSPI\_CLK (CL)

//

MAP\_PinTypeSPI(PIN\_45, PIN\_MODE\_7);

//

// Configure PIN\_07 for SPI0 GSPI\_MOSI (SL)

//

MAP\_PinTypeSPI(PIN\_07, PIN\_MODE\_7);

//

// Configure PIN\_55 for UART0 UART0\_TX

//

MAP\_PinTypeUART(PIN\_55, PIN\_MODE\_3);

//

// Configure PIN\_57 for UART0 UART0\_RX

//

MAP\_PinTypeUART(PIN\_57, PIN\_MODE\_3);

//

// Configure PIN\_01 for UART1 UART1\_TX

//

MAP\_PinTypeUART(PIN\_01, PIN\_MODE\_7);

//

// Configure PIN\_02 for UART1 UART1\_RX

//

MAP\_PinTypeUART(PIN\_02, PIN\_MODE\_7);

}

MAIN.C

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

//

// Application Name - int\_sw

// Application Overview - The objective of this application is to demonstrate

// GPIO interrupts using SW2 and SW3.

// NOTE: the switches are not debounced!

//

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

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

//

//! \addtogroup int\_sw

//! @{

//

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

// Standard includes

#include <stdio.h>

#include "string.h"

// Driverlib includes

#include "hw\_types.h"

#include "hw\_ints.h"

#include "hw\_memmap.h"

#include "hw\_common\_reg.h"

#include "interrupt.h"

#include "hw\_apps\_rcm.h"

#include "prcm.h"

#include "rom.h"

#include "rom\_map.h"

#include "prcm.h"

#include "gpio.h"

#include "utils.h"

#include "timer.h"

#include "systick.h"

#include "spi.h"

#include "uart.h"

// Common interface includes

#include "uart\_if.h"

#include "pinmux.h"

#include "timer\_if.h"

#include "gpio\_if.h"

#define SPI\_IF\_BIT\_RATE 100000

#define BLACK 0x0000

#define WHITE 0xFFFF

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

// GLOBAL VARIABLES -- Start

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

extern void (\* const g\_pfnVectors[])(void);

volatile unsigned long count;

volatile unsigned long value;

volatile unsigned long lastValue;

volatile unsigned long diff;

volatile unsigned char nextFlag;

volatile unsigned char confirmFlag;

volatile unsigned char isNew;

static volatile unsigned long g\_lastKey;

static volatile unsigned long g\_ulBase;

char cmp[60];

char cmpCheck[60];

char ifRepeat[60];

char buf[60];

int i;

int j;

int k;

int x;

int y;

int Rx;

int Ry;

int m;

int maxLength;

char \*Button0 = "1010000000000010000000001000000001001100010011001";

char \*Button1 = "1010000000000010000000001000000000000100000001001";

char \*Button2 = "1010000000000010000000001000000001000100010001001";

char \*Button3 = "1010000000000010000000001000000000100100001001001";

char \*Button4 = "1010000000000010000000001000000001100100011001001";

char \*Button5 = "1010000000000010000000001000000000010100000101001";

char \*Button6 = "1010000000000010000000001000000001010100010101001";

char \*Button7 = "1010000000000010000000001000000000110100001101001";

char \*Button8 = "1010000000000010000000001000000001110100011101001";

char \*Button9 = "1010000000000010000000001000000000001100000011001";

char \*LAST = "1010000000000010000000001000000001110110011101101";

char \*MUTE = "1010000000000010000000001000000000100110001001101";

char checkList[12][4] = {

{' '}, //Button 0

{',', '.', '!'}, //Button 1

{'a', 'b', 'c'}, //Button 2

{'d', 'e', 'f'}, //Button 3

{'g', 'h', 'i'}, //Button 4

{'j', 'k', 'l'}, //Button 5

{'m', 'n', 'o'}, //Button 6

{'p', 'q', 'r', 's'}, //Button 7

{'t', 'u', 'v'}, //Button 8

{'w', 'x', 'y', 'z'}, //Button 9

{'+'}, //Button LAST

{'-'}, //Button MUTE

};

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

// GLOBAL VARIABLES -- End

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

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

// LOCAL FUNCTION PROTOTYPES

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

static void BoardInit(void);

int time\_diff(unsigned long value, unsigned long lastValue) {

unsigned long interval;

if (lastValue > value) interval = (lastValue - value)/80000; //80000

else interval = (2^24 + (lastValue - value))/80000;

return interval;

}

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

//

//check input character

//

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

static char

checkButton(char \*cmp) {

char message;

if(strcmp(cmp, Button0) == 0) { message = '0';}

else if(strcmp(cmp, Button1) == 0) { message = '1';}

else if(strcmp(cmp, Button2) == 0) { message = '2';}

else if(strcmp(cmp, Button3) == 0) { message = '3';}

else if(strcmp(cmp, Button4) == 0) { message = '4';}

else if(strcmp(cmp, Button5) == 0) { message = '5';}

else if(strcmp(cmp, Button6) == 0) { message = '6';}

else if(strcmp(cmp, Button7) == 0) { message = '7';}

else if(strcmp(cmp, Button8) == 0) { message = '8';}

else if(strcmp(cmp, Button9) == 0) { message = '9';}

else if(strcmp(cmp, LAST) == 0) { message = '+';}

else if(strcmp(cmp, MUTE) == 0) { message = '-';}

else{message = 'x';}

return message;

//return checkList[i][j];

}

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

//

//check pressed button

//

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

static int

checkRow(char text) {

int message;

if(text == '0') { message = 0; maxLength = 0;}

else if(text == '1') { message = 1; maxLength = 2;}

else if(text == '2') { message = 2; maxLength = 2;}

else if(text == '3') { message = 3; maxLength = 2;}

else if(text == '4') { message = 4; maxLength = 2;}

else if(text == '5') { message = 5; maxLength = 2;}

else if(text == '6') { message = 6; maxLength = 2;}

else if(text == '7') { message = 7; maxLength = 3;}

else if(text == '8') { message = 8; maxLength = 2;}

else if(text == '9') { message = 9; maxLength = 3;}

else if(text == '+') { message = 10; maxLength = 0;}

else if(text == '-') { message = 11; maxLength = 0;}

else{message = 12;}

return message;

//return checkList[i][j];

}

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

// LOCAL FUNCTION DEFINITIONS

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

static void GPIOA1IntHandler(void) { // ############################

unsigned long ulStatus;

unsigned long interval;

char text;

int l;

lastValue = value;

value = SysTickValueGet();

diff = time\_diff(value, lastValue);

if(diff >= 10 ){

k = 0;

count = 0;

}else{

if(diff >= 1){

cmp[k] = '1';

k++;

}

else{

cmp[k] = '0';

k++;

}

count++;

if(count >= 49){

cmp[k] = '\0';

text = checkButton(cmp);

if (g\_lastKey == -1) MAP\_TimerEnable(TIMERA0\_BASE, TIMER\_A);

unsigned long tik = Timer\_IF\_GetCount(TIMERA0\_BASE, TIMER\_A);

unsigned long tok = g\_lastKey;

g\_lastKey = tik;

int wait = 535 - (tik - tok)/8000000;

if(wait>2 && wait<20) {nextFlag = 0;}

else if (wait >= 20) {

nextFlag = 1;

j = 0;

}

else {nextFlag = 2;}

if(strcmp(cmp, cmpCheck) != 0 || text == 'x' || text == '+' || text == '-' || nextFlag == 1){

//Report("------------ %d\r\n", interval);

if(text != 'x' && text != '+' && text != '-' && confirmFlag == 1){

m++;

x+=8;

if(x > 127){

x = 0;

y+=8;

if(y > 127){

y = 0;

}

}

}

if(text == '-'){

buf[m] = '\0';

m--;

drawChar(x, y, ' ', WHITE, BLACK, 1);

x -= 8;

if (x == 0 && y >= 8) {

x = 119;

y -= 8;

}

if (x <= 3 && y <= 3) {

x = 0;

y = 0;

confirmFlag = 0;

}

}

else if(text == '+'){

fillRect(0, 0, 127, 63, BLACK);

buf[m+1] = '\0';

int n;

for(n = 0; n <= strlen(buf); n++) {MAP\_UARTCharPut(UARTA1\_BASE, buf[n]);}

//MAP\_UARTCharPut(UARTA1\_BASE, '\0');

x = 0;

y = 0;

m = 0;

confirmFlag = 0;

}

i = checkRow(text);

j = 0;

if(i != 12 && i != 11 && i != 10) {

drawChar(x, y, checkList[i][j], WHITE, BLACK, 1);

buf[m] = checkList[i][j];

}

if(confirmFlag == 0 && i != 11 && i != 10) {confirmFlag = 1;}

if(text == 'x') {Report("This key is not in the target button set.\r\n", text);}

else if(text == '+') {Report("Button ENTER/LAST is pressed.\r\n");}

else if(text == '-') {Report("Button DELETE/MUTE is pressed.\r\n");}

else {Report("Button #%c is pressed.\r\n", text);}

strcpy(cmpCheck, cmp);

}

else if(strcmp(cmp, cmpCheck) == 0 && nextFlag == 0 && j < maxLength){

j++;

if(i != 12 && i != 11 && i != 10) {

drawChar(x, y, checkList[i][j], WHITE, BLACK, 1);

buf[m] = checkList[i][j];

}

}

}

}

ulStatus = MAP\_GPIOIntStatus(GPIOA1\_BASE, true);

MAP\_GPIOIntClear(GPIOA1\_BASE, ulStatus); // clear interrupts on GPIOA1

}

static void UARTIntHandler(void) {

unsigned char transChar;

MAP\_UARTIntDisable(UARTA1\_BASE,UART\_INT\_RX);

while(UARTCharsAvail(UARTA1\_BASE))

{

transChar = MAP\_UARTCharGet(UARTA1\_BASE);

if(isNew)

{

fillRect(0, 65, 127, 127, BLACK);

isNew = 0;

}

if(Rx > 127) // About to go off right edge.

{

Rx = 0;

Ry += 8; // go to next line.

}

if(transChar == '\0')

{

isNew = 1;

Rx = 0;

Ry = 65;

}

else

{

drawChar(Rx, Ry, transChar, WHITE, BLACK, 1);

Rx += 8;

}

}

MAP\_UARTIntClear(UARTA1\_BASE,UART\_INT\_RX);

MAP\_UARTIntEnable(UARTA1\_BASE,UART\_INT\_RX);

}

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

//

//! Board Initialization & Configuration

//!

//! \param None

//!

//! \return None

//

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

static void

BoardInit(void) {

MAP\_IntVTableBaseSet((unsigned long)&g\_pfnVectors[0]);

// Enable Processor

//

MAP\_IntMasterEnable();

MAP\_IntEnable(FAULT\_SYSTICK);

PRCMCC3200MCUInit();

}

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

//

//! Main function

//!

//! \param none

//!

//!

//! \return None.

//

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

int main() {

unsigned long ulStatus;

g\_ulBase = TIMERA0\_BASE;

BoardInit();

PinMuxConfig();

InitTerm();

ClearTerm();

//

// Initialize the device on UART1

//

MAP\_UARTIntRegister(UARTA1\_BASE,UARTIntHandler);

MAP\_UARTIntEnable(UARTA1\_BASE,UART\_INT\_RX);

MAP\_UARTFIFOLevelSet(UARTA1\_BASE,UART\_FIFO\_TX1\_8,UART\_FIFO\_RX1\_8);

MAP\_UARTConfigSetExpClk(UARTA1\_BASE,MAP\_PRCMPeripheralClockGet(PRCM\_UARTA1),

UART\_BAUD\_RATE, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |

UART\_CONFIG\_PAR\_NONE));

//

// Register the interrupt handlers

//

MAP\_GPIOIntRegister(GPIOA1\_BASE, GPIOA1IntHandler);

//

// Configure rising edge interrupts on GPIO

//

MAP\_GPIOIntTypeSet(GPIOA1\_BASE, 0x40, GPIO\_RISING\_EDGE); // GPIO - output, GPIO\_RISING\_EDGE or GPIO\_BOTH\_EDGES

ulStatus = MAP\_GPIOIntStatus (GPIOA1\_BASE, false);

MAP\_GPIOIntClear(GPIOA1\_BASE, ulStatus); // clear interrupts on GPIOA1

SysTickPeriodSet(16777216);

//Timer\_IF\_Init(PRCM\_TIMERA0, g\_ulBase, TIMER\_CFG\_ONE\_SHOT\_UP, TIMER\_A, 0);//ulConfig check

SysTickEnable();

Timer\_IF\_Init(PRCM\_TIMERA0, TIMERA0\_BASE, TIMER\_CFG\_ONE\_SHOT\_UP, TIMER\_A, 0); //ulConfig check

// Enable GPIO interrupts

// clear global variables

i = 0;

j = 0;

k = 0;

x = 0;

y = 0;

Rx = 0;

Ry = 65;

m = 0;

maxLength = 0;

count = 0;

value = 0;

lastValue = 0;

nextFlag = 0;

confirmFlag = 0;

isNew = 1;

memset(cmpCheck, '\0', sizeof(cmpCheck));

strcpy(cmpCheck, "00000000000000000000000000000000");

memset(ifRepeat, '\0', sizeof(cmpCheck));

strcpy(ifRepeat, "00000000000000000000000000000000");

// Enable GPIO interrupts

MAP\_GPIOIntEnable(GPIOA1\_BASE, 0x40);

MAP\_TimerEnable(TIMERA0\_BASE, TIMER\_A);

Message("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

Message("\t\t\t IR Remote TX/RX Application Start \n\r");

Message("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

Message("\n\n\n\r");

Report("count = %d\r\n",count);

//

// Reset SPI

//

MAP\_SPIReset(GSPI\_BASE);

//

// Configure SPI interface

//

MAP\_SPIConfigSetExpClk(GSPI\_BASE,MAP\_PRCMPeripheralClockGet(PRCM\_GSPI),

SPI\_IF\_BIT\_RATE,SPI\_MODE\_MASTER,SPI\_SUB\_MODE\_0,

(SPI\_SW\_CTRL\_CS |

SPI\_4PIN\_MODE |

SPI\_TURBO\_OFF |

SPI\_CS\_ACTIVEHIGH |

SPI\_WL\_8));

//

// Enable SPI for communication

//

MAP\_SPIEnable(GSPI\_BASE);

//

// Initialising the OLED.

//

Adafruit\_Init();

fillScreen(BLACK);

drawLine(0, 64, 127, 64, WHITE);

while (1) {

if(x > 127 || y > 127 || Rx > 127 || Ry > 127){

break;

}

}

while (1) {

//MAP\_UtilsDelay(80000000/5);

}

}

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

//

// Close the Doxygen group.

//! @}

//

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

How we decoded the signals from the receiver module

I decode the signal based on pulse length. Every time there is a rising edge in the waveform, the program will be interrupted, and the signal handler will calculate the time difference between current time and last rising edge using systick. The time difference will be converted to millisecond and then be read as 0, 1, or reset. Later, comparing the number series with the pre-read signals from the IR receiver by Logic, the program will tell which button is pressed.

The difficulty we met in Lab 3

In this lab we do meet some difficulties. At the very beginning of the lab, we spent a lot of time on capturing the waveform generated from the remote control-theoretically, we are supposed to read a 48-bit long data, but we were just able to detect half of the data. After resetting the remote control and Launchpad several times, we figured out the problem located on the receiver. Then we changed our receiver and successfully capturing the full data waveform.

Besides, after we modified the program to be ready for check off, we found that when we kept push buttons to entering command for more than one line, we would have trouble deleting or sending the message. Look through the program, we found the problem is located at the buffer we built which is used for storing the input. And after we fixed this bug, we are now able to input as many characters as we want.