**Lab 2 Report Alex Elkman**

**EEC 172 - A02 Hardik Patel**

**Spring 2016 May 4, 2016**

**Objective:**

The purpose of this lab was to use the Saleae Logic Analyzer to characterize the transmissions for the buttons from an AT&T remote. Moreover, we were to write a program to use interrupts to monitor and decode the signals from the IR receiver module, so as to determine which remote was pressed on the remote. Finally, we used UART to send text message from one board to another using the multi-tap text entry system.

**Design and Test Procedure:**

**Part 1:** For this part we created the circuit as shown in the lab manual using the IR receiver, the resistor and the capacitor. The IR receiver was powered by the 3.3 V port on the CC3200. Once the circuit was in place, the Saleae Logic Analyser was connected to the GND port and the output of the receiver. Once the circuit was ready, for the input to the receiver the AT&T remotes were calibrated by following the steps in the manual. For our group the code that was provided was, 1287. Using the remote and the circuit, the waveforms of the “1-9” and ‘mute’ and ‘last’ were captured on the analyzer. These waveforms were then decoded to actual unique bit values that were used in part 2. It is important to mention that the actual waveforms were not converted to 1s and 0s, but we used the waveforms to interpret the data as 1s and 0s.

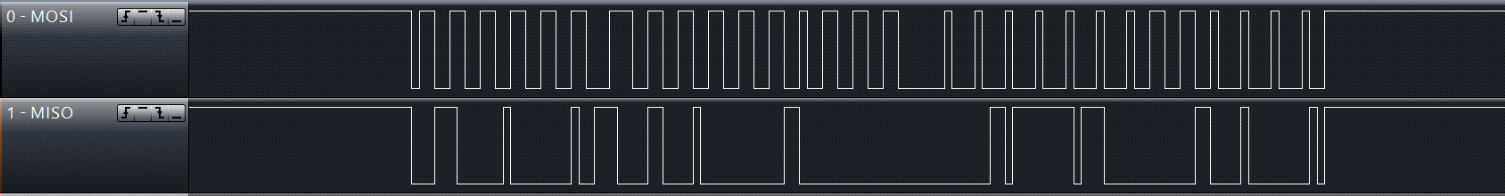
**Part 2:** In this subsection, the waveforms were inputted in a C program, which used a series of logical analyzations and made use of the Timer A, to convert these waveforms to a unique bit pattern for all the inputs. This was accomplished using Interrupts. The timer was set to go off and count the difference between two positive edges. This was used to create unique bit patterns as evident from the code below. Once the bits were obtained, they were compared to the ones already defined and used to report the value on the terminal. The ‘Last’ button acted as out delete key. To delete the last character, we just print the same character as black over it.

Figure 1: Waveform I2C from part 2-2 (Note, “MOSI” is the CLK, “MISO” is the DAC)

**Part 3:** This part follows from the previous part and uses the UART to send and receive text messages. To implement this we started at the pin muxing. Since, Pins 55 and 57 could not be used, we instead used pins 1 and 45. The OLED initialization and pins were imported from Lab 1 files. Once everything was initialized, the UARTCharGet() and UARTCharPut() functions were used to get and put the decoded characters. Since the RX of one board was connected to the TX of the other board, bidirectional texting was possible. Other additions to the board were included to satisfy the lab requirements. The ‘Mute’ button acted as the enter key.

***Use of interrupts by our program:***Interrupts in our program are set to be called by UARTA1\_BASE, which is connected to the output signal of the IR remote. The interrupts also only monitor the negative edges. As soon as an interrupt occurs, the program enters the interrupt handler where it assigns a binary value to a int array. This way we store the entire bit pattern and compare that to the defined bit patterns to decode it.

***Explanation of the waveform:***The waveform generated by the TV remote are all 26 bits long. The bits are characterized by the pulse lengths. Shorter pulses are interpretes as a 0 and longer pulses are 1s. The signal is high before any button is pressed and the start pattern is same for all the buttons. The start bit is also a longer pulse width than the rest of the pulses. When the signal has finished propagating, to signify the stop the signal returns to high and waits for the next button press.

***Problems and difficulties faced:*** The biggest problem that we faced was of the decoding of the input signal. The values we were getting at that time were never a constant. Which meant, definition of a button to work. To fix this we introduced the ‘delta’ variable that basically says, if the pulse is long make it a else make it a 0.

**Conclusion**

Through this lab we learned the usage of UART and how to send signals through UART functions.We also learned the use of interrupts and how to implement timers. We also learned how to decode signals from a source.

**Waveforms**



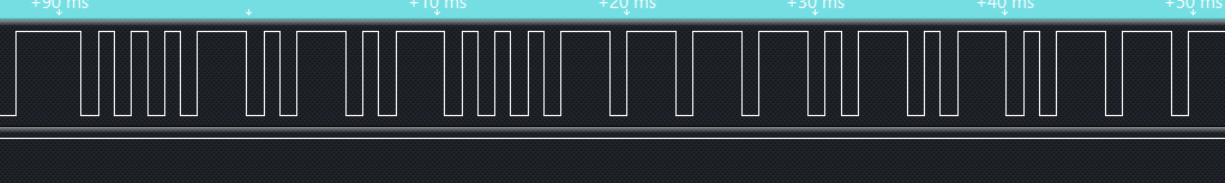
Waveform ‘0’. Binary: 1001101000010110010111000



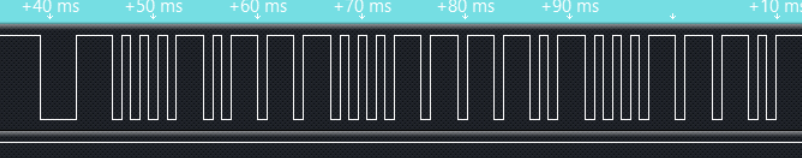
Waveform ‘1’. Binary: 11111101000100000010111000



Waveform ‘2’. Binary: 1101000011110010111000



Waveform ‘3’. Binary: 10101000011101010111000



Waveform ‘4’. Binary: 11101000011100010111000



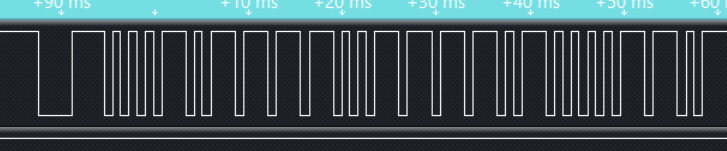
Waveform ‘5’. 100101000011011010111000



Waveform ‘6’. Binary: 101101000011010010111000



Waveform ‘7’. Binary: 110101000011001010111000

Waveform ‘8’. Binary: 111101000011000010111000

Waveform ‘9’. Binary: 1000101000010111010111000

**Code for this lab -** Only the essential code that was added is presented here.

**//Alex Elkman**

**//Hardik Patel**

#define MASTER 0

//These are the decimal values that corrospont to the binary data from the buttons as decoded bt our inturrupt handler

#define BUTTON\_ONE 66339000

#define BUTTON\_TWO 3423416

#define BUTTON\_THREE 5520056

#define BUTTON\_FOUR 7616696

#define BUTTON\_FIVE 9713336

#define BUTTON\_SIX 11809976

#define BUTTON\_SEVEN 13906616

#define BUTTON\_EIGHT 16003256

#define BUTTON\_NINE 18099896

#define BUTTON\_ZERO 20196536

#define BUTTON\_LAST 30728880

#define BUTTON\_MUTE 41162936

#define SPI\_IF\_BIT\_RATE 100000

static unsigned long g\_ulSamples[2];

static char messageBuffer[100]; //Keeps track of the message in a global variable

static char number[100]; //Keeps track the binary strings of the buttons

static char lastPressed; //Keeps track of the last digit passes.

static char tempString[1]; // Is used for the UARTCharGet functions to store the character in a temporary variable befor we display it on the OLED

static int keyBuffer[13] = {0}; // An array that helps with the implementation of the multi-tap aspect of the program.

static int value = 1; // Generates the binary value

static int i = 0;

//static int index = 0;

static int delta = 0;

#if defined(ccs) || defined(gcc)

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

#endif

#if defined(ewarm)

extern uVectorEntry \_\_vector\_table;

#endif

static void TimerIntHandler()

{

g\_ulSamples[1] = MAP\_TimerValueGet(TIMERA1\_BASE,TIMER\_A);

TimerLoadSet(TIMERA1\_BASE, TIMER\_A,0xffff);

g\_ulSamples[0] = g\_ulSamples[1];

delta = g\_ulSamples[1];

//Delta is used to differentiate betwwn a 0 pulse width and a 1 pulse width.

if(delta < 54000)

value = 0;

else

value = 1;

if(i==0)

value = 0;

number[i] = (char)value; //Storing a binary value in an array.

i++;

MAP\_TimerIntClear(TIMERA1\_BASE,TIMER\_CAPA\_EVENT); // Clearing interrupts

}

//Generates a small delay

void delay(unsigned long ulCount){

int i;

do{

ulCount--;

for (i=0; i< 63000; i++) ;

}while(ulCount);

}

int main()

{

//Local variables used for different purposes.

int z = 0;

int y = 0;

int k = 0;

int sum = 0;

int temp = 0;

BoardInit();

PinMuxConfig();

InitTerm();

DisplayBanner(APP\_NAME);

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

MAP\_PRCMPeripheralReset(PRCM\_GSPI);

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

SPI\_IF\_BIT\_RATE,SPI\_MODE\_MASTER,SPI\_SUB\_MODE\_3,

(SPI\_SW\_CTRL\_CS |

SPI\_4PIN\_MODE |

SPI\_TURBO\_OFF |

SPI\_CS\_ACTIVELOW |

SPI\_WL\_8));

MAP\_SPIEnable(GSPI\_BASE);

MAP\_SPICSEnable(GSPI\_BASE);

Adafruit\_Init();

fillScreen(BLACK);

MAP\_PinConfigSet(PIN\_02,PIN\_TYPE\_STD\_PU,PIN\_STRENGTH\_6MA);

MAP\_TimerIntRegister(TIMERA1\_BASE,TIMER\_A,TimerIntHandler);

MAP\_TimerConfigure(TIMERA1\_BASE, (TIMER\_CFG\_SPLIT\_PAIR | TIMER\_CFG\_A\_CAP\_TIME));

// We are calling interrupts at the positice edges of the input signal

MAP\_TimerControlEvent(TIMERA1\_BASE,TIMER\_A,TIMER\_EVENT\_NEG\_EDGE);

//Starting the timer at 0xffff

MAP\_TimerLoadSet(TIMERA1\_BASE,TIMER\_A,0xffff);

MAP\_TimerIntEnable(TIMERA1\_BASE,TIMER\_CAPA\_EVENT);

MAP\_TimerEnable(TIMERA1\_BASE,TIMER\_A);

//Enable and set up the UARTA1

MAP\_UARTEnable(UARTA1\_BASE);

MAP\_UARTFIFOEnable(UARTA1\_BASE);

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

UART\_BAUD\_RATE,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |

UART\_CONFIG\_PAR\_NONE));

while(1){

// i is the nummerof bits generated by our remote. Hence, we run this loop in 26 times

while(i<26){

}

Report("\n\n\r");

i=0;

Report("You pressed: ");

for(k = 0; k<25; k++){

sum += (int)number[k]\*pow(2, (k+1));

Report("%d", (int)number[k]);

}

Report("\n\rUnique ID: %d", sum);

index++; //increment index

Report("\n\rYou pressed: ");

//This switch statement is where all the multitap aspect is implemented. Also it includes our enter button (MUTE) and delete buttons (LAST).

switch(sum){

case(BUTTON\_ONE):

keyBuffer[1]+=1;

Report("1");

lastPressed = '1';

index-=1;

break;

case(BUTTON\_TWO):

Report("2");

if(keyBuffer[2] == 3){

keyBuffer[2] = 0;

delKey("C");

}

if(lastPressed != '2')

keyBuffer[2] = 0;

if(keyBuffer[2]==0){

Outstr("A");

}

else if(keyBuffer[2]==1){

delKey("A");

Outstr("B");

}

else if(keyBuffer[2]==2){

delKey("B");

Outstr("C");

}

lastPressed = '2';

keyBuffer[2]+=1;

break;

case(BUTTON\_THREE):

Report("3");

if(keyBuffer[3] == 3){

keyBuffer[3] = 0;

delKey("F");

}

if(lastPressed != '3')

keyBuffer[3] = 0;

if(keyBuffer[3]==0){

Outstr("D");

}

else if(keyBuffer[3]==1){

delKey("D");

Outstr("E");

}

else if(keyBuffer[3]==2){

delKey("E");

Outstr("F");

}

lastPressed = '3';

keyBuffer[3]+=1;

break;

case(BUTTON\_FOUR):

Report("4");

if(keyBuffer[4] == 3){

keyBuffer[4] = 0;

delKey("I");

}

if(keyBuffer[4] == 3)

keyBuffer[4] = 0;

if(lastPressed != '4')

keyBuffer[4] = 0;

if(keyBuffer[4]==0){

Outstr("G");

}

else if(keyBuffer[4]==1){

delKey("G");

Outstr("H");

}

else if(keyBuffer[4]==2){

delKey("H");

Outstr("I");

}

lastPressed = '4';

keyBuffer[4]+=1;

break;

case(BUTTON\_FIVE):

Report("5");

if(keyBuffer[5] == 3){

keyBuffer[5] = 0;

delKey("L");

}

if(lastPressed != '5')

keyBuffer[5] = 0;

if(keyBuffer[5]==0){

Outstr("J");

}

else if(keyBuffer[5]==1){

delKey("J");

Outstr("K");

}

else if(keyBuffer[5]==2){

delKey("K");

Outstr("L");

}

lastPressed = '5';

keyBuffer[5]+=1;

break;

case(BUTTON\_SIX):

Report("6");

if(keyBuffer[6] == 3){

keyBuffer[6] = 0;

delKey("O");

}

if(lastPressed != '6')

keyBuffer[6] = 0;

if(keyBuffer[6]==0){

Outstr("M");

}

else if(keyBuffer[6]==1){

delKey("M");

Outstr("N");

}

else if(keyBuffer[6]==2){

delKey("N");

Outstr("O");

}

lastPressed = '6';

keyBuffer[6]+=1;

break;

case(BUTTON\_SEVEN):

Report("7");

if(keyBuffer[7] == 4){

keyBuffer[7] = 0;

delKey("S");

}

if(lastPressed != '7')

keyBuffer[7] = 0;

if(keyBuffer[7]==0){

Outstr("P");

}

else if(keyBuffer[7]==1){

delKey("P");

Outstr("Q");

}

else if(keyBuffer[7]==2){

delKey("Q");

Outstr("R");

}

else if(keyBuffer[7]==3){

delKey("R");

Outstr("S");

}

lastPressed = '7';

keyBuffer[7]+=1;

break;

case(BUTTON\_EIGHT):

Report("8");

if(keyBuffer[8] == 3){

keyBuffer[8] = 0;

delKey("V");

}

if(lastPressed != '8')

keyBuffer[8] = 0;

if(keyBuffer[8]==0){

Outstr("T");

}

else if(keyBuffer[8]==1){

delKey("T");

Outstr("U");

}

else if(keyBuffer[8]==2){

delKey("U");

Outstr("V");

}

lastPressed = '8';

keyBuffer[8]+=1;

break;

case(BUTTON\_NINE):

Report("9");

if(keyBuffer[9] == 4){

keyBuffer[9] = 0;

delKey("Z");

}

if(lastPressed != '9')

keyBuffer[9] = 0;

if(keyBuffer[9]==0){

Outstr("W");

}

else if(keyBuffer[9]==1){

delKey("W");

Outstr("X");

}

else if(keyBuffer[9]==2){

delKey("X");

Outstr("Y");

}

else if(keyBuffer[9]==3){

delKey("Y");

Outstr("Z");

}

lastPressed = '9';

keyBuffer[9]+=1;

break;

case(BUTTON\_ZERO):

Report("0");

Outstr(" ");

lastPressed = '0';

break;

case(BUTTON\_LAST):

//We print black over the last pressed as a delete implementation

if(index>1) //since delKey will decrement index, and we want it to stay > 1

index = index - 1;

tempString[0] = messageBuffer[index-1];

messageBuffer[index-1] = NULL;

Report("\n\rLAST");

delKey(tempString);

lastPressed = 'l';

break;

case(BUTTON\_MUTE):

Report("MUTE");

//Running the ENTER command for the MASTER device

Report("\n\rMaster Entered!\n\r");

//Put each character from messageBuffer into the TX FIFO (for UART1)

for(z=0; z<(index-1); z++){

if(MASTER)

UARTCharPut(UARTA1\_BASE, messageBuffer[z]);

else{

tempString[0] = ((char)UARTCharGet(UARTA1\_BASE));

Outstr2(tempString);

Report("\n\rtempString: %s\n\r", tempString);

}

}

lastPressed = 'm';

index-=1;

break;

default:

Report("Unknown code %d", sum);

index-=1;

break;

}

if(lastPressed == 'l' || lastPressed == 'm' || lastPressed == '1') {}

else if (lastPressed == '0')

messageBuffer[index-1] = ' ';

else{

temp = 65 + (lastPressed-48-2)\*3 + keyBuffer[(int)lastPressed-48] - 1; //returns ascii value for the char you selected aelkman

if(lastPressed == '8' || lastPressed == '9')

temp += 1;

messageBuffer[index-1] = (char)temp;

}

Report("\n\rtemp: %d, index: %d", temp, index);

sum=0;

Report("\n\rMessege Buffer: ");

for(y=0;y<100;y++){

Report("%c", messageBuffer[y]);

}

}

}