## ECEN260 – Final Project

## Car Alarm System

Jeanette Eldredge

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Objectives (5 points)**

Project Objective and requirements from Brother Fisher:

The Final Project is a project each laboratory partnership creates to show what has been learned while in the class. You may adapt an existing project you find online to the MSP432 LaunchPad if you wish. Simply copying an existing MSP432 LaunchPad project, however, is not acceptable. The Final Project is worth 300 points.

You must show an understanding of course concepts by using at least three of the following concepts in your project (30 pts each, up to 90 pts):

ADC

I/O

Interrupt

UART/I2C/SPI

Display

For the Final Project, each laboratory partnership will make a video and turn in a Lab report.

The video, worth 100 points, should have these as a minimum:

An oral explanation of your project and how it works. (50 pts)

A demonstration of your project working. (50 pts)

You will also need to turn in a lab report, worth 110 points, before your presentation so the TA can grade your entire project all together.

The Lab report should include the following

A Schematic of the board and peripherals and their connections. (20 pts)

The test plan for your project. (20 pts)

A copy of the code (30 pts)

A written explanation of your project (20 pts)

The test results and any portions you struggled with. (20 pts)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Program Specifications and Description (10 points)**

Requirements:

1. 2 MSP432 board
2. 1 Magnet Switch
3. 8 connecting wires

This Project is meant to mimic the Locking and Alarm system on a car. This includes locking, unlocking, and Alarm. The LEDs should mimic a car lights when pressing buttons on the fob. One flash for locking, two flashes for unlock, and constant on and off if the alarm is triggered. Board 1 LEDS should represent the front head lights of the car. Board 2 LEDS should represent the back lights of the car. Use UARTS to send signals to both boards.

Switch 1 should be coded to lock the car and LED1 and LED2\_RED should flash once on both boards. Switch 2 should be coded to unlock the car and LED1 and LED2\_RED should flash twice on both boards

If the car is locked and the magnet switch is opened, the alarm is triggered. The trigger of alarm should cause constant flashing of LED1 and LED2\_RED on both boards.  
The Lock Button (SW1) should not do anything in the alarm state. If the Unlock Button (SW2) is pressed, the alarm status should be cleared and the flashing lights should stop.

This project fulfills the project requirements by using 3 of the 5 things we learned this semester. Those things are Input/Outputs, Interrupts, and UARTS.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Schematic or Wiring Diagram (10 points)**

Diagram

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Plan and Results (10 points)**

The test plan includes 4 parts:

Part 1: Lock the car by pressing switch 1. This should cause the two red LEDs to flash once on both boards

Part 2: Unlock the car by pressing switch 2. This should cause the two red LEDs to flash twice on both boards

Part 3: Lock the car by pressing switch 1 again. This should cause the two red LEDS to flash once on both boards. Then separate the magnet switch. This should cause the lights to flash on and off constantly on both boards.

Part 4: After testing part 3, press the Switch 2 again. This should clear the error, turn off the lights, and clear the display board.

Results turned in with this Final Report.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Things I learned:**

It was really good for me to review UARTS. They were much easier to do this time around than they were during the lab. I learned how to use them to send more than just 2 different codes like 0xAA and 0x55.

I was also glad to review an interrupt. Interrupts were needed in this project because it was easier to handle the “door” opening with an interrupt instead of constantly in a while loop.

The main thing I learned was how to set an interrupt to react on the falling edge rather than the rising edge. You must use the IES command and set it equal to 0x00. This will tell the interrupt to activate when the change goes from a 1 to a 0 which is what I needed for my “door” to work.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code (15 points)**

***Board 1 code:*  
#include** "msp.h"

**#define** LED1 BIT0 //P1.0

**#define** LED2\_RED BIT0 //P2.0

**#define** S1 BIT1 //P1.1

**#define** S2 BIT4 //P1.4

**#define** DOOR BIT1 // P4.1

**#define** DELAY 300 // used for SW switch debouncer using 300 clock cycles

**volatile** **int** Open;

/\*\*

\* main.c

\*/

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

//Initialize Variables

**int** Alarm = 0;

Open = 0;

**int** Lock = 0;

**int** i = 0;

//Initialize LED1 and Switch 1

P1->DIR |= LED1; // set P1.0 as output

P1->DIR &= ~(S1); // set P1.1 as input

P1->REN |= S1; // turn on P1.1 pull resistor

P1->OUT |= S1; // configure P1.1 resistor as pull-up

//Initialize LED2\_RED and Switch 2

P2->DIR |= LED2\_RED; // set P2.0 as output

P1->DIR &= ~(S2); // set P1.4 as input

P1->REN |= S2; // turn on P2.1 pull resistor

P1->OUT |= S2; // configure P2.1 resistor as pull-up ?????

// Turn all alarm lights off

P1->OUT &= ~LED1; // turn off Car right light

P2->OUT &= ~LED2\_RED; // turn off Car Left light

// Configure port 3 as UART2.

P3->SEL0 |= BIT2 | BIT3; // Set bit 2 and bit 3 of P3SEL0 to 1

P3->SEL1 &= ~(BIT2 | BIT3); // Reset bit 2 and bit 3 of P3SEL1 to 0

EUSCI\_A2->CTLW0 |= BIT0; // Set WRST to put UART0 in reset

// Leave all other bits = 0

EUSCI\_A2->CTLW0 |= BIT6 | BIT7; // While keeping bit 0 = 1, set the

//remaining control bits in

//UCA2CTLW0 to the

// values shown in section

// UART2 (UCA2) Registers.

EUSCI\_A2->BRW= 0x138; // Baud rate = 9600 clk = 3000000 BRW = clk/BR

//convert to Hexadecimal

EUSCI\_A2->MCTLW &= ~BIT0; // UCOS16 bit = 0

EUSCI\_A2->CTLW0 &= ~ BIT0; // Clear WRST to resume UART operation.

**char** data; //Defining the variable data to be used in the while loop

//Set up interrupt on Port 3 for door watch

P3->DIR &= ~BIT0; /\* set up pin P3.0 (DA (interrupt pin)) as input \*/

P3->REN |= BIT0; /\* connect pull resistor to pin P3.0 \*/

P3->OUT |= BIT0; /\* configure pull resistor as pull up \*/

P3->IFG &= ~BIT0; /\* clear interrupt flag for pin P3.0 (DA (int pin)) \*/

P3->IE |= BIT0; /\* enable the interrupt for pin P3.0 (DA (interrupt pin)) \*/

P3->IES = 0x00; /\*Enables activation on falling edge\*/

NVIC->ISER[1] |= 0x20; /\* enable port 3 interrupts (see p. 89 in text)\*/

\_enable\_interrupts();

// While loop handles Lock, Unlock, and Alarm clearing.

**while** (1)

{

// delay for switch debouncing

**for** (i = 0; i < DELAY; i++){;}

// check the door

**if**((P1->IN & S1) == 0x00)

{

data = 0xAA; //If switch is 0, that means the switch is pressed

//and data is set equal to 0xAA

UART\_OutChar(data); //OxAA is passed through the UART in this

//function call.

P1->OUT |= LED1; // Flash Car lights once

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

Lock = 1;

**if**(Alarm ==0)

{ Open = 0;}

}

**else** **if**((P1->IN & S2) == 0x00)

{

data = 0xBB; //If switch is 0, that means the switch is pressed

//and data is set equal to 0xBB

UART\_OutChar(data); //OxAA is passed through the UART in this

//function call.

P1->OUT |= LED1; // Flash Car lights twice

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

**for** (i = 0; i < 60000; i++){;}

P1->OUT |= LED1;

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

Lock = 0;

Open = 0;

}

**else** **if**((Open ==1) && (Lock ==1))

{

Alarm = 1;

**while**(Alarm == 1)

{

**if**((P1->IN & S2) == 0x00)

{

Alarm =0;

Open = 0;

}

**else**{

data = 0xCC; //If switch is 0, that means the switch is pressed

//and data is set equal to 0xCC

UART\_OutChar(data); //OxAA is passed through the UART in this

//function call.

P1->OUT |= LED1; // Flash Car lights As alert

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

**for** (i = 0; i < 60000; i++){}

}

}

}

**else**

{

data = 0x55; //If switch is 0, that means the switch is pressed

//and data is set equal to 0xCC

UART\_OutChar(data); //OxAA is passed through the UART in this

//function call.

}

}

}

**void** **PORT3\_IRQHandler**(**void**){

uint32\_t status;

status = P3->IFG; /\* get the interrupt status for port 3 \*/

**if**(status & BIT0){ /\* constant for the pin 0 mask \*/

Open = 1;

}

P3->IFG &= ~BIT0; /\* clear the interrupt for port 3, pin 0 \*/

}

**void** **UART\_OutChar** (**char** data) {

**while** ( (EUSCI\_A2->IFG & BIT1) == 0); // Busy. Wait for previous output.

EUSCI\_A2->TXBUF = data; // Start transmission when IFG = 1.

}

***Board 2 code:***

**#include** "msp.h"

**#define** LED1 BIT0 //P1.0

**#define** S1 BIT1 //P1.1

**#define** LED2\_RED BIT0 //P2.0

/\*\*

\* main.c

\*/

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

**char** data; //Defines the variable data as a char to be used in a later while statement

P1->DIR |= LED1; // set P1.0 as output

P2->DIR |= LED2\_RED;

// Configure port 3 as UART2.

P3->SEL0 |= BIT2 | BIT3; // Set bit 2 and bit 3 of P3SEL0 to 1

P3->SEL1 &= ~(BIT2 | BIT3); // Reset bit 2 and bit 3 of P3SEL1 to 0

EUSCI\_A2->CTLW0 |= BIT0; // Set WRST to put UART0 in reset

// Leave all other bits = 0

EUSCI\_A2->CTLW0 |= BIT6 | BIT7; // While keeping bit 0 = 1, set the

//remaining control bits in

//UCA2CTLW0 to the

// values shown in section

// UART2 (UCA2) Registers.

EUSCI\_A2->BRW= 0x138; // Baud rate = 9600 clk = 3000000 BRW = clk/BR

//convert to Hexadecimal

EUSCI\_A2->MCTLW &= ~BIT0; // UCOS16 bit = 0

EUSCI\_A2->CTLW0 &= ~BIT0; // Clear WRST to resume UART operation.

**int** i = 0;

**while**(1){

data = UART\_InChar(); //Set data equal to the bits read off of the UART

**if**(data == 0xAA)

{

P1->OUT |= LED1; // Flash Car lights once

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

}

**if**(data == 0xBB)

{

P1->OUT |= LED1; // Flash Car lights twice

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

**for** (i = 0; i < 60000; i++){;}

P1->OUT |= LED1;

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){;}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

}

**if**(data == 0xCC)

{

P1->OUT |= LED1; // Flash Car lights As alert

P2->OUT |= LED2\_RED;

**for** (i = 0; i < 40000; i++){}

P1->OUT &= ~LED1;

P2->OUT &= ~LED2\_RED;

**for** (i = 0; i < 60000; i++){}

}

**if**(data == 0x55)

{

P1->OUT &= ~LED1; /\*turn off LED1 at pin P1.0 if data is=to 0x55\*/

}

}

}

**char** **UART\_InChar** (**void**)

{

**while** ( (EUSCI\_A2->IFG & BIT0) == 0); // Busy. Wait for received data.

**return** ( (**char**) (EUSCI\_A2->RXBUF) ); // Get new input when IFG = 1.

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_