EE4930 Advanced Embedded Systems

Section 011, Winter 2019/20

Draven Schilling

1-30-20

Lab 5 – Low Power

# Objectives

The objective for this lab was to experiment with the MSP432 low power modes. We were tasked with periodically reading a temperature sensor (~10min) using an ADC, then storing its result and signaling that the conversion was completed. Between conversions, the MSP432 would reside in low power mode such that the program would have a lifetime of ~70days using an CR2032 battery.

# Description

In this lab, I started by configuring the ADC to read from the temperature sensor. I used the TMP36 datasheet with the graph to create a formula to convert the ADC readings to temperature in Fahrenheit. Once I verified the temperature sensor was working, I then setup the watchdog timer in interval, interrupt mode such that it would fire off approximately every 4 minutes. If I only triggered an ADC conversion every other interrupt, we would have a new reading every 8minutes which was just under the required 10minute maximum. Next, once I verified the watchdog was running correctly, I moved on to configuring the clock subsystem and SCB/PCM to run in low power mode 3. This part was the trickiest because I had hardware problems combined with clock problems and it took quite awhile to get the board to actually draw less current while in LPM3 mode.

# Conclusions

This lab was overall quite challenging, I’m not sure if it was meant to be as hard as it was for me but after first running into hardware problems, then problems with current draw in LPM3 still being too high it took a long time and overcoming many challenges to complete this lab. First, I spent a large amount of time struggling with EnergyTrace and measuring the current draw from my program. Eventually with Dr. Widder’s help I was able to find that the problem resided within my board, so I ended up having to borrow a friends MSP432 for the remaining tests. Next, once I was able to obtain some testing time, I quickly got my board to enter LPM3 but even during LPM3 the current draw was still significantly higher than it should be.

# Attachments

**Source Code:**

/\*

\* fsm.h

\*

\* Created on: Jan 13, 2020

\* Author: schillingdl

\*/

**#ifndef** FSM\_H\_

**#define** FSM\_H\_

**typedef** **enum** {

*NORMAL\_OFF*,

*NORMAL\_ON*,

*DEFROST\_MODE*

}state;

**typedef** **enum** {

*IDLE*,

*TURN\_ON*,

*TURN\_OFF*,

*THERE\_IS\_ICE*

}event;

**typedef** **void** (\*fp)(**void**);

**typedef** **struct**{

state nextstate;

fp action;

} stateElement;

// turns the fan and compressor off

**void** **outputsOff**();

//turns the fan and compressor on

**void** **outputsOn**();

//turns the fan on and compressor off (defrost mode)

**void** **defrost**();

// function to handle the state machine update

// accepts current state and current event, returns next state

state **stateUpdate** (state current, event input);

**#endif** /\* FSM\_H\_ \*/

/\*\*

\* fsm.c

\*/

**#ifndef** FSM\_C\_

**#define** FSM\_C\_

**#include** "fsm.h"

**#include** "msp.h"

// create state table array

stateElement stateTable [3][4] = {

{{*NORMAL\_OFF*, outputsOff},{*NORMAL\_ON*, outputsOn},{*NORMAL\_OFF*, outputsOff},{*DEFROST\_MODE*, defrost}},

{{*NORMAL\_ON*, outputsOn},{*NORMAL\_ON*, outputsOn},{*NORMAL\_OFF*, outputsOff},{*DEFROST\_MODE*, defrost}},

{{*NORMAL\_OFF*, outputsOff},{*NORMAL\_ON*, outputsOn},{*NORMAL\_OFF*, outputsOff},{*DEFROST\_MODE*, defrost}},

};

// function to handle the state machine update

// accepts current state and current event, returns next state

state **stateUpdate** (state current, event input){

stateElement currentstate = stateTable[current][input];

// run the proper action function

(\*currentstate.action)();

//return next state info

**return** currentstate.nextstate;

}

// turns the fan and compressor off

**void** **outputsOff**(){

P2->OUT &= ~((1<<4)|(1<<6));

}

//turns the fan and compressor on

**void** **outputsOn**(){

P2->OUT |= ((1<<4)|(1<<6));

}

//turns the fan on and compressor off (defrost mode)

**void** **defrost**(){

P2->OUT &= ~(1<<4);

P2->OUT |= (1<<6);

}

**#endif**

/\*\*

\* main.c

\*/

**#include** "fsm.h"

**#include** "msp.h"

**#include** <stdint.h>

**#include** <string.h>

**#include** <stdio.h>

**#include** "msoe\_lib\_clk.h"

**#include** "msoe\_lib\_lcd.h"

**#include** "msoe\_lib\_delay.h"

**void** **initGPIOInputs**();

**void** **initGPIOOutputs**();

**void** **initADC**();

**#define** ADC\_RANGE 4096 // for 12 bit conversions

//Global Variables

uint8\_t temperature = 0; //temperature reading

uint8\_t humidity = 0; // humidity reading

uint8\_t humidity\_setpoint = 50; // set point (default 50)

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

{

event input;

state current = *NORMAL\_OFF*; // start in off mode

outputsOff(); // start with everything off

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

// setup lcd

LCD\_Config();

LCD\_clear();

LCD\_home();

LCD\_contrast(10);

// init ports

initADC();

initGPIOInputs();

initGPIOOutputs();

**while**(1){

ADC14->CTL0 |= 1; // start ADC conversions

LCD\_goto\_xy(0,0);

LCD\_print\_str("Temp: ");

LCD\_print\_udec5(temperature);

LCD\_goto\_xy(0,1);

LCD\_print\_str("Set: ");

LCD\_print\_udec5(humidity\_setpoint);

LCD\_goto\_xy(0,2);

LCD\_print\_str("Humid: ");

LCD\_print\_udec5(humidity);

LCD\_goto\_xy(0,3);

**if**((P1->IN & 1<<5) >> 5 == 1){ // check for high signal from the ice sensor

LCD\_print\_str("DEFROST: Y ");

} **else** {

LCD\_print\_str("DEFROST: N ");

}

// determine input event

input = *IDLE*;

**if**(humidity >= (humidity\_setpoint + 5)){

input = *TURN\_ON*;

} **else** **if**(humidity <= (humidity\_setpoint - 5)){

input = *TURN\_OFF*;

}

**if**((P1->IN & 1<<5) >> 5 == 1){ // check for high signal from the ice sensor

input = *THERE\_IS\_ICE*;

}

current = stateUpdate(current, input); // call state update

}

}

// Setup both MSP onboard pushbuttons & Ice Sensor

// initializes pushbutton P1.1 as GPIO input

// initializes pushbutton P1.4 as GPIO input

// initializes P1.5 as GPIO input (Ice Sensor)

**void** **initGPIOInputs**()

{

// P1.1

P1->SEL0 &= ~0b10; // use GPIO function

P1->SEL1 &= ~0b10;

P1->DIR &= ~0b10; // make input

P1->REN |= 0b10; // allow pull up/down

P1->OUT |= 0b10; // setup as pull up

P1->IES &= ~0b10; // trigger interrupt on rising edge

P1->IE |= 0b10; // enable interrupt

// P1.4

P1->SEL0 &= ~(0b1<<4); // use GPIO function

P1->SEL1 &= ~(0b1<<4);

P1->DIR &= ~(0b1<<4); // make input

P1->REN |= (0b1<<4); // allow pull up/down

P1->OUT |= (0b1<<4); // setup as pull up

P1->IES &= (0b1<<4); // trigger interrupt on rising edge

P1->IE |= (0b1<<4); // enable interrupt

// P1.5

P1->SEL0 &= ~(0b1<<5); // use GPIO function

P1->SEL1 &= ~(0b1<<5);

P1->DIR &= ~(0b1<<5); // make input

P1->REN |= (0b1<<5); // allow pull up/down

P1->OUT |= (0b1<<5); // setup as pull up

P1->IES &= (0b1<<5); // trigger interrupt on rising edge

P1->IE |= (0b1<<5); // enable interrupt

NVIC->ISER[1] |= (1<<3); // enable I/O P1 interrupt in NVIC

}

// Initializes Outputs for both the fan and compressor

// initializes P2.4 as GPIO output (compressor)

// initializes P2.6 as GPIO output (fan)

**void** **initGPIOOutputs**(){

// P2.4

P2->SEL0 &= ~(1<<4); // use GPIO function

P2->SEL1 &= ~(1<<4);

P2->DIR |= (1<<4); // make output

P2->OUT &= ~(1<<4); // setup output low to start

// P2.6

P2->SEL0 &= ~(1<<6); // use GPIO function

P2->SEL1 &= ~(1<<6);

P2->DIR |= (1<<6); // make output

P2->OUT &= ~(1<<6); // setup output low to start

}

//Initialize ADC6 on pin 4.7

//Initialize ADC7 on pin 4.6

//Interrupt upon completed conversion

**void** **initADC**(){

//setup pins 4.6 and 4.7 in analog mode

P4->SEL0 |= 0b11<<6;

P4->SEL1 |= 0b11<<6;

// start sampling on SC bit,

// source a timer,

// use SMCLK,

// repeat sequence of channels conversion mode

// 96 sample/hold time

// turn core on

ADC14->CTL0 &= 0x0;

ADC14->CTL0 |= (1<<26) | (1<<21) | (0b11<<17) | (0b101<<12) | (0b101<<8) | (1<<4);

// 12 bit resolution and use memory location 4 to start

ADC14->CTL1 &= 0xF0000000;

ADC14->CTL1 |= (0b10<<4) | (4<<16);

ADC14->MCTL[4] |= 0x6; // input on A6 for pot1

ADC14->MCTL[5] |= 0x7; // input on A7 for pot2

ADC14->MCTL[5] |= (1<<7); // set EOS for location5

ADC14->IER0 |= (1<<4); // enable interrupt for mem[4]

ADC14->IER0 |= (1<<5); // enable interrupt for mem[5]

ADC14->CTL0 |= 0b10; // enable

NVIC->ISER[0] |= (1<<24); // enable ADC interrupt in NVIC

ADC14->CTL0 |= 1; // start ADC conversions

}

// Interrupt Handler for ADC

// Read ADC and set to temperature/humidity

**void** **ADC14\_IRQHandler**(**void**)

{

uint32\_t pendingInterrupt = ADC14->IFGR0;

**if**((pendingInterrupt & 1<<4) > 0){

//pot1

temperature = 100-(uint8\_t)((((**float**)ADC14->MEM[4])/ADC\_RANGE)\*100);

ADC14->CLRIFGR0 |= 1<<4; // Clear Interrupt

} **else** **if**((pendingInterrupt & 1<<5) > 0){

//pot2

humidity = 100-(uint8\_t)((((**float**)ADC14->MEM[5])/ADC\_RANGE)\*100);

ADC14->CLRIFGR0 |= 1<<5; // Clear Interrupt

}

ADC14->CLRIFGR0 |= 0xFFFFFFFF; // Extra precaution

}

// Interrupt Handler for the GPIO inputs

// increment / decrement the set point

// interrupt for the ice sensor is unused

**void** **PORT1\_IRQHandler**(**void**){

uint16\_t pending\_interrupt = P1->IV;

**if**(pending\_interrupt == 4){

humidity\_setpoint = (humidity\_setpoint < 5) ? 0 : humidity\_setpoint-5;

} **else** **if**(pending\_interrupt == 0xA){

humidity\_setpoint = (humidity\_setpoint > 95) ? 100 : humidity\_setpoint+5;

} **else** **if**(pending\_interrupt == 0xC){

// Unused interrupt

}

}