Temperature work

Temp = Vtemp / 10mv

Vtemp = analog voltage output form temperature sensor

Temp = Vtemp / 0.01V

Temp = Vtemp \* 100

Voltage Chart:

|  |  |  |  |
| --- | --- | --- | --- |
| Tempature name | Value in hex | Value in volts | Value in degrees C |
| High Threshold | 0x0FFF | 0.81V | 81 C |
| Low Threshold | 0x0A00 | 0.54 | 54 C |

Code:

//////////////////////

//Willard Wider

//5/29/18

//ELEC 3800

//Lab 07

//////////////////////

/\* DriverLib Includes \*/

**#include** <ti/devices/msp432p4xx/driverlib/driverlib.h>

/\* Standard Includes \*/

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.h>

**#define** TEMP\_THRESHOLD\_LOW 0x0A00//the low tempature for the temp sensor

**#define** TEMP\_THRESHOLD\_HIGH 0x0FFF//the high tempature for the temp sensor

**#define** MODE\_TOO\_COLD 0

**#define** MODE\_TOO\_HOT 1

**#define** DELAY 1000

/\* Statics \*/

**static** **volatile** uint16\_t curADCResult = 0x0000;

//0 = too cold, keep on untill we get to high threshold

//1 = too warm, keep off untill we get to low threshold

**static** **int** mode = 0;

**int** i = 0;

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

{

/\* Halting the Watchdog \*/

MAP\_WDT\_A\_holdTimer();

/\* Setting Flash wait state \*/

MAP\_FlashCtl\_setWaitState(FLASH\_BANK0, 1);

/\* Setting DCO to 48MHz \*/

MAP\_CS\_setDCOCenteredFrequency(CS\_DCO\_FREQUENCY\_48);

//![Single Sample Mode Configure]

/\* Initializing ADC (MCLK/1/4) \*/

MAP\_ADC14\_enableModule();

//MAP\_ADC14\_initModule(ADC\_CLOCKSOURCE\_MCLK, ADC\_PREDIVIDER\_1, ADC\_DIVIDER\_4, 0);

/\* Configuring GPIOs (5.5 A0) \*/

//now it's 4.0

MAP\_GPIO\_setAsInputPin(GPIO\_PORT\_P4,GPIO\_PIN0);//clears pull up resistors i guess

MAP\_GPIO\_setAsPeripheralModuleFunctionInputPin(GPIO\_PORT\_P4, GPIO\_PIN0, GPIO\_TERTIARY\_MODULE\_FUNCTION);

/\* Configuring ADC Memory \*/

MAP\_ADC14\_configureSingleSampleMode(ADC\_MEM0, true);

MAP\_ADC14\_configureConversionMemory(ADC\_MEM0, ADC\_VREFPOS\_AVCC\_VREFNEG\_VSS, ADC\_INPUT\_A13, false);

/\* Configuring Sample Timer \*/

MAP\_ADC14\_enableSampleTimer(ADC\_MANUAL\_ITERATION);

/\* Enabling/Toggling Conversion \*/

MAP\_ADC14\_enableConversion();

MAP\_ADC14\_toggleConversionTrigger();

//![Single Sample Mode Configure]

/\* Enabling interrupts \*/

MAP\_ADC14\_enableInterrupt(ADC\_INT0);

MAP\_Interrupt\_enableInterrupt(INT\_ADC14);

MAP\_Interrupt\_enableMaster();

//setup the output pin for relay and LED

//high = turn heater on

//low = turn header off

//use pin 4.2 for the output

//GPIO\_setAsOutputPin(GPIO\_PORT\_P4,GPIO\_PIN2);

//tmep use default led as output

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

//get the initial value for logick set before the loop

//if it's below the low threshold, set mode to 0

//else if it's inbetween both threholds, set mode to 1

//else if it's above high threshold, set mode to 1

**printf**("heater init, result is...");

**if**(curADCResult < TEMP\_THRESHOLD\_LOW)

{

**printf**("value below low, turn on heater\n");

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

}

**else** **if** ((curADCResult > TEMP\_THRESHOLD\_LOW) && (curADCResult < TEMP\_THRESHOLD\_HIGH))

{

**printf**("value in middle, turn on heater\n");

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

}

**else** **if** (curADCResult > TEMP\_THRESHOLD\_HIGH)

{

**printf**("value above high, turn off heater");

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

}

**else**

{

**printf**("ERROR: this condition should not be reached on init!\n");

**return** 1;

}

**while** (1)

{

//look at logik above

**if**(curADCResult < TEMP\_THRESHOLD\_LOW && mode == MODE\_TOO\_HOT)

{

**printf**("temp below low threshold from fall, turning heater on\n");

mode = MODE\_TOO\_COLD;

//toggle pin on

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

}

**else** **if** (curADCResult > TEMP\_THRESHOLD\_HIGH && mode == MODE\_TOO\_COLD)

{

**printf**("temp above high threshold from rise, turning heater off\n");

mode = MODE\_TOO\_HOT;

//toggle pin off

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P1, GPIO\_PIN0);

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P4, GPIO\_PIN2);

}

//delay so it's not running 24/7

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

}

}

//![Single Sample Result]

/\* ADC Interrupt Handler. This handler is called whenever there is a conversion

\* that is finished for ADC\_MEM0.

\*/

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

{

uint64\_t status = MAP\_ADC14\_getEnabledInterruptStatus();

MAP\_ADC14\_clearInterruptFlag(status);

**if** (ADC\_INT0 & status)

{

curADCResult = MAP\_ADC14\_getResult(ADC\_MEM0);

//for debugging

**if**(0)

{

**printf**("curADCResult=%x\n",curADCResult);

}

MAP\_ADC14\_toggleConversionTrigger();

}

}

//![Single Sample Result]