**#include** <msp430.h>

**#include** <stdio.h>

**unsigned** **char** **UART\_RX**(**void**);

**void** **UART\_TX**(**unsigned** **char** ch);

**void** **UART\_TX\_string**(**char** \*str);

**unsigned** **int** **read\_ADC\_temp**(**void**);

**float** **convert\_to\_Fahrenheit**(**unsigned** **int** adc\_value);

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

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

PM5CTL0 &= ~LOCKLPM5; // Unlock GPIO

// UART configuration

UCA0CTLW0 = UCSWRST;

UCA0CTLW0 |= UCSSEL\_2; // SMCLK

UCA0BRW = 6;

UCA0MCTLW = UCOS16 | (8 << 4) | (32 << 8);

P2SEL1 |= BIT0 | BIT1; // P2.0 = UCA0RXD, P2.1 = UCA0TXD

P2SEL0 &= ~(BIT0 | BIT1);

UCA0CTLW0 &= ~UCSWRST;

// ADC configuration

ADC12CTL0 = ADC12SHT0\_2 | ADC12ON; // Sample hold time, ADC ON

ADC12CTL1 = ADC12SHP; // Pulse-mode

ADC12CTL2 |= ADC12RES\_2; // 12-bit resolution

ADC12MCTL0 |= ADC12INCH\_30 | ADC12VRSEL\_1; // Temp sensor, Vref = 1.5V

REFCTL0 |= REFON | REFVSEL\_0; // Internal ref = 1.5V

**while** (1) {

**unsigned** **char** ch = UART\_RX();

**if** (ch == '1') {

UART\_TX\_string("\r\n1. List\r\n2. Name\r\n3. Temp\r\n4. Center\r\n5. Left\r\n6. Right\r\n");

} **else** **if** (ch == '2') {

UART\_TX\_string("ESET 369: Kyle Rex\r\n");

} **else** **if** (ch == '3') {

// If a ‘3’ key is pressed, read the ADC value from the internal temperature sensor and display it on the serial terminal in Fahrenheit (°F) once.

}

}

**return** 0;

}

**unsigned** **char** **UART\_RX**(**void**) {

**while** (!(UCA0IFG & UCRXIFG));

**return** UCA0RXBUF;

}

**void** **UART\_TX**(**unsigned** **char** ch) {

**while** (!(UCA0IFG & UCTXIFG));

UCA0TXBUF = ch;

}

**void** **UART\_TX\_string**(**char** \*str) {

**while** (\*str) {

UART\_TX(\*str++);

}

}

**unsigned** **int** **read\_ADC\_temp**(**void**) {

ADC12CTL0 |= ADC12ENC | ADC12SC; // Enable and start conversion

**while** (ADC12CTL1 & ADC12BUSY);

**return** ADC12MEM0; // Return ADC value

}

**float** **convert\_to\_Fahrenheit**(**unsigned** **int** adc\_value) {

// Based on datasheet for 1.5V ref

**float** temp\_C = ((adc\_value - 630.0f) / 10.0f) + 30.0f;

**return** (temp\_C \* 9.0f / 5.0f) + 32.0f;

}

**#include** <msp430.h>

**unsigned** **int** adc\_raw;

**float** TempDegC;

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

WDTCTL = WDTPW | WDTHOLD; // hold watchdog timer

PM5CTL0 &= ~LOCKLPM5; // clear LOCKLPM5 bit

P1DIR |= 0x01; // output direction (P1.0)

**while**(REFCTL0 & REFGENBUSY); // wait until the busy flag is cleared

REFCTL0 |= REFVSEL\_2 | REFON; // enable internal 2.5V ref.

**int** Ref\_T30 = \*((**unsigned** **int** \*)(TLV\_START + TLV\_ADC12CAL + 0x09)); // temperature calibration (30degC/12bit)

**int** Ref\_T85 = \*((**unsigned** **int** \*)(TLV\_START + TLV\_ADC12CAL + 0x0B)); // temperature calibration (85degC/12bit)

ADC12CTL0 = ADC12SHT0\_6 | ADC12ON; // ADC CTL0 setup

ADC12CTL1 = ADC12SHP; // ADC CTL1 set up

ADC12CTL2 = ADC12RES\_2; // 12-bit conversion

ADC12CTL3 = ADC12TCMAP; // select ADC12TCMAP channel

ADC12MCTL0 = ADC12VRSEL\_1 | ADC12INCH\_30; // VR+, ADC channel

**while**(1) {

ADC12CTL0 |= ADC12ENC | ADC12SC; // ADC, Start conversion

**while** ((ADC12IFGR0 & BIT0) == 0); // flag check

adc\_raw = ADC12MEM0; // read ADC

TempDegC = (((**float**) adc\_raw - Ref\_T30) \* 55) / (Ref\_T85 - Ref\_T30) + 30.0; // temperature conversion

P1OUT ^= 0x01; // toggle (P1.0)

**\_\_delay\_cycles**(25000); // delay

}

**return** 0;

}

FINALLLL

**#include** <msp430.h>

**unsigned** **char** **UART\_RX**(**void**);

**void** **UART\_TX**(**unsigned** **char** ch);

**void** **UART\_TX\_string**(**char** \*str);

**void** **UART\_TX\_number**(**unsigned** **int** num);

**unsigned** **int** **read\_ADC\_temp**(**void**);

**float** **convert\_to\_Fahrenheit**(**unsigned** **int** adc\_value);

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

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

PM5CTL0 &= ~LOCKLPM5; // Unlock GPIO

// UART configuration

UCA0CTLW0 = UCSWRST;

UCA0CTLW0 |= UCSSEL\_2; // SMCLK

UCA0BRW = 6;

UCA0MCTLW = UCOS16 | (8 << 4) | (32 << 8);

P2SEL1 |= BIT0 | BIT1; // P2.0 = UCA0RXD, P2.1 = UCA0TXD

P2SEL0 &= ~(BIT0 | BIT1);

UCA0CTLW0 &= ~UCSWRST;

// ADC configuration

ADC12CTL0 = ADC12SHT0\_2 | ADC12ON; // Sample hold time, ADC ON

ADC12CTL1 = ADC12SHP; // Pulse-mode

ADC12CTL2 |= ADC12RES\_2; // 12-bit resolution

ADC12MCTL0 |= ADC12INCH\_30 | ADC12VRSEL\_1; // Temp sensor, Vref = 1.5V

REFCTL0 |= REFON | REFVSEL\_0; // Internal ref = 1.5V

**while** (1) {

**unsigned** **char** ch = UART\_RX();

**if** (ch == '1') {

UART\_TX\_string("\r\n1. List\r\n2. Name\r\n3. Temp\r\n4. Center\r\n5. Left\r\n6. Right\r\n");

} **else** **if** (ch == '2') {

UART\_TX\_string("ESET 369: Kyle Rex\r\n");

} **else** **if** (ch == '3') {

**unsigned** **int** adc\_value = read\_ADC\_temp();

**float** tempF = convert\_to\_Fahrenheit(adc\_value);

**unsigned** **int** int\_part = (**unsigned** **int**) tempF;

**unsigned** **int** frac\_part = (**unsigned** **int**) ((tempF - int\_part) \* 100); // Extract two decimal places

UART\_TX\_string("Temperature: ");

UART\_TX\_number(int\_part);

UART\_TX('.');

UART\_TX\_number(frac\_part);

UART\_TX\_string(" F\r\n");

}

}

**return** 0;

}

**unsigned** **char** **UART\_RX**(**void**) {

**while** (!(UCA0IFG & UCRXIFG));

**return** UCA0RXBUF;

}

**void** **UART\_TX**(**unsigned** **char** ch) {

**while** (!(UCA0IFG & UCTXIFG));

UCA0TXBUF = ch;

}

**void** **UART\_TX\_string**(**char** \*str) {

**while** (\*str) {

UART\_TX(\*str++);

}

}

**void** **UART\_TX\_number**(**unsigned** **int** num) {

**char** buffer[5];

**int** i = 0;

**if** (num == 0) {

UART\_TX('0');

**return**;

}

**while** (num > 0) {

buffer[i++] = (num % 10) + '0';

num /= 10;

}

**while** (i > 0) {

UART\_TX(buffer[--i]);

}

}

**unsigned** **int** **read\_ADC\_temp**(**void**) {

ADC12CTL0 |= ADC12ENC | ADC12SC; // Enable and start conversion

**while** (ADC12CTL1 & ADC12BUSY);

**return** ADC12MEM0; // Return ADC value

}

**float** **convert\_to\_Fahrenheit**(**unsigned** **int** adc\_value) {

// Based on datasheet for 1.5V ref

**float** temp\_C = ((adc\_value - 630.0f) / 10.0f) + 30.0f;

**return** (temp\_C \* 9.0f / 5.0f) + 32.0f;

}