Michael Tangy

ECE 381 – Spring 2015

Lab 9

Digital Thermometer

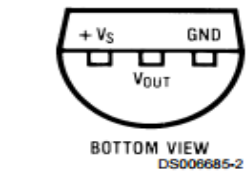
The purpose of this lab was for us to gain an understanding of how to read and analyze Analog signals using our PSoC microcontroller. We learned how to Measure convert and amplify samples taken from our LM34CZ temperature sensor using the PSoC's incremental analog to digital converter and programmable gain amplifier.

Parts Needed:

- PSoC 1 Microcontroller with 2x16 LCD
- LM34CZ temperature sensor

Hardware/Pin-Out Configuration:

After hooking your PSoC up with your computer you need to wire your Temperature sensor to your PSoC. Connect the temperature sensors power and ground to the PSoC's power and ground then connect the Vout pin to port 0 pin 1.



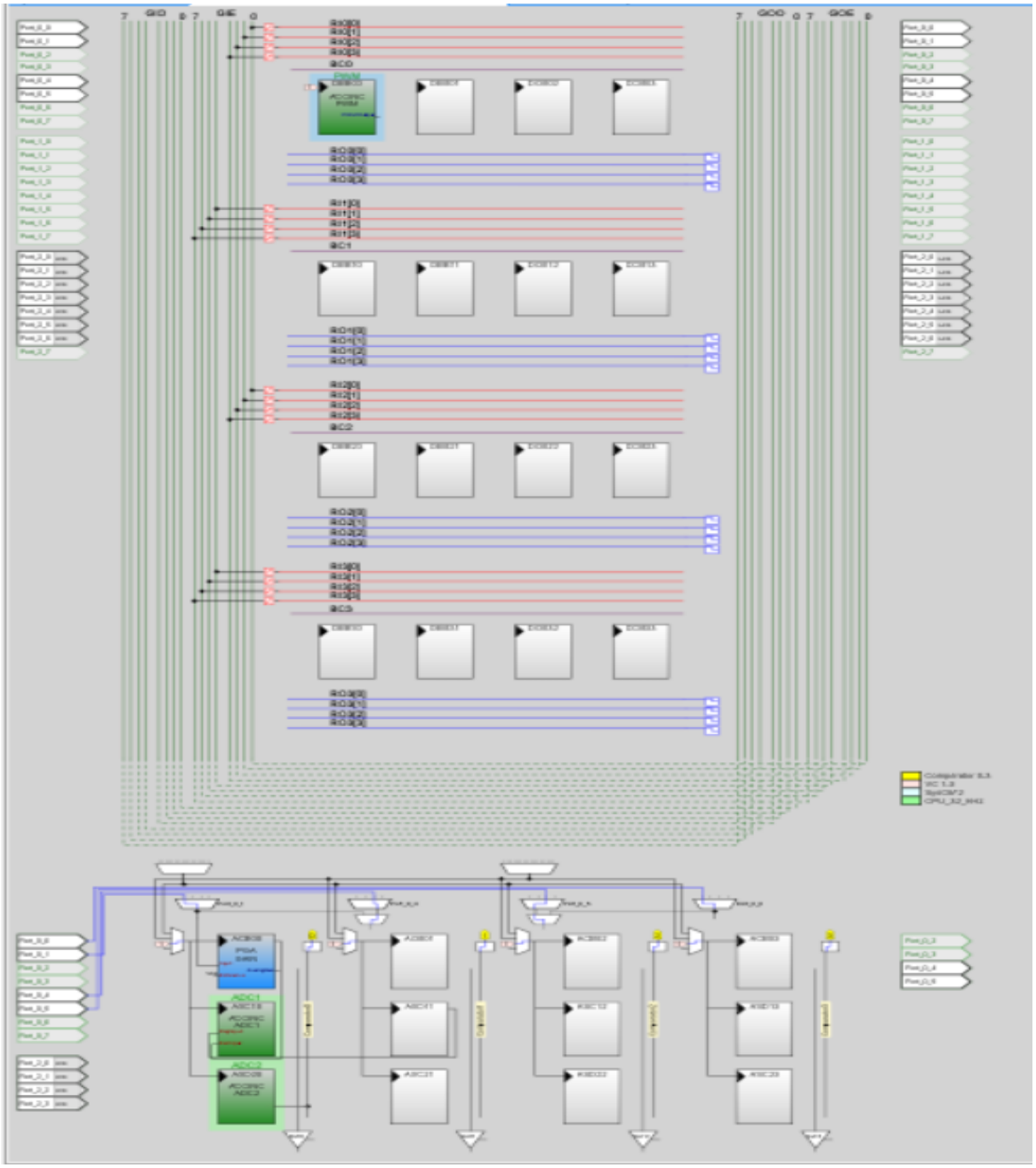
TO-92
Plastic Package

BOTTOM VIEW
DS006685-2

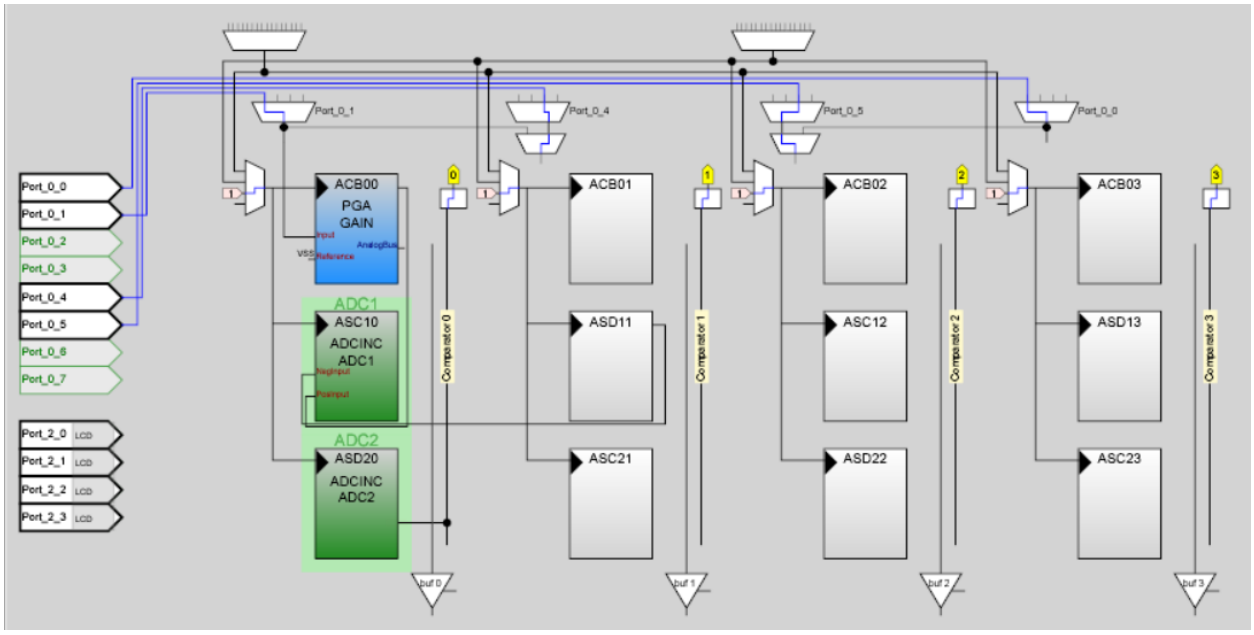
Order Number LM34CZ, LM34CAZ or LM34DZ
See NS Package Number Z03A

After the temperature sensor is hooked up to your PSoC go into the chip level configuration of the project and add an ADCINC (Double modulator), PGA, Sleep timer and LCD modules. Make sure you ADC is set to produce unsigned 14bit samples and make sure your data clock is set to VC1 so you can configure the correct sample rate. Next set your PGA to a

gain of 2 then set your Sleep timers tick counter size to 4 Bytes. Next make sure the global parameters of the project is set to the one below. Once your settings are correct, set the input of the PGA to port 0 pin 1 and set the output of the PGA to the input of the ADCINC. This can be done by shift clicking the inputs/outputs to the appropriate place. Refer to the picture below to confirm your settings are correct.



Analog configuration:

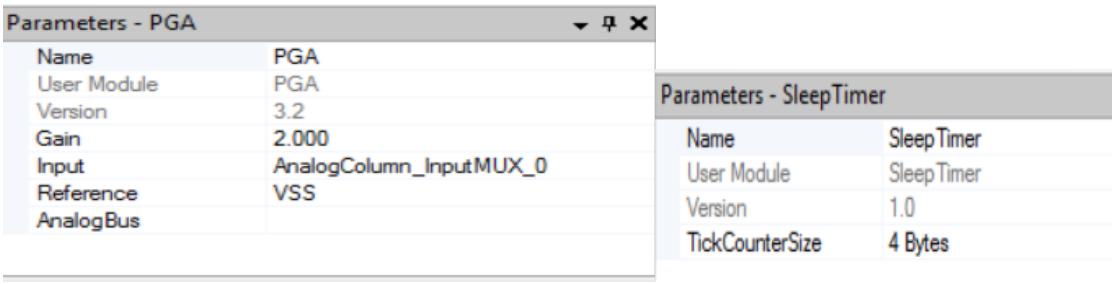


Parameters - ADCINC	
Name	ADCINC
User Module	ADCINC
Version	1.20
DataFormat	Unsigned
Resolution	14 Bit
Data Clock	VC1
PosInput	ACB00
NegInput	ASD11
NegInputGain	Disconnected
ClockPhase	Normal
PulseWidth	1
PWM Output	None

Global Resources - lab9_ece381	
Power Setting [Vcc / S	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	5
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	256
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	BandGap+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMF	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

Equation used to find sample rate:

$$SampleRate = \frac{DataClock}{256 \cdot (2^{Bits-6} + 1)}$$



Software Description:

Your code should have all the variable's declared globally above main. The main method should include the start functions to all your modules as well as your scale factor which should be your maximum voltage divide by two to the number bits in each sample and both those numbers should be type casted to floats. Your infinite while loop gets a single sample from the ADC multiplies it by the scale factor then multiplies it by 100 and adds one half to move the value up two decimal places and round it up. This value then goes through an if-else statement to determine where to put the decimal place depending on its value. The value is then printed on the LCD.

CODE:

```
//--------------------------------------------------------------------------
// C main line
//--------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include <stdbool.h>
#include <stdlib.h>
#include "stdlib.h"
#include "math.h"
```

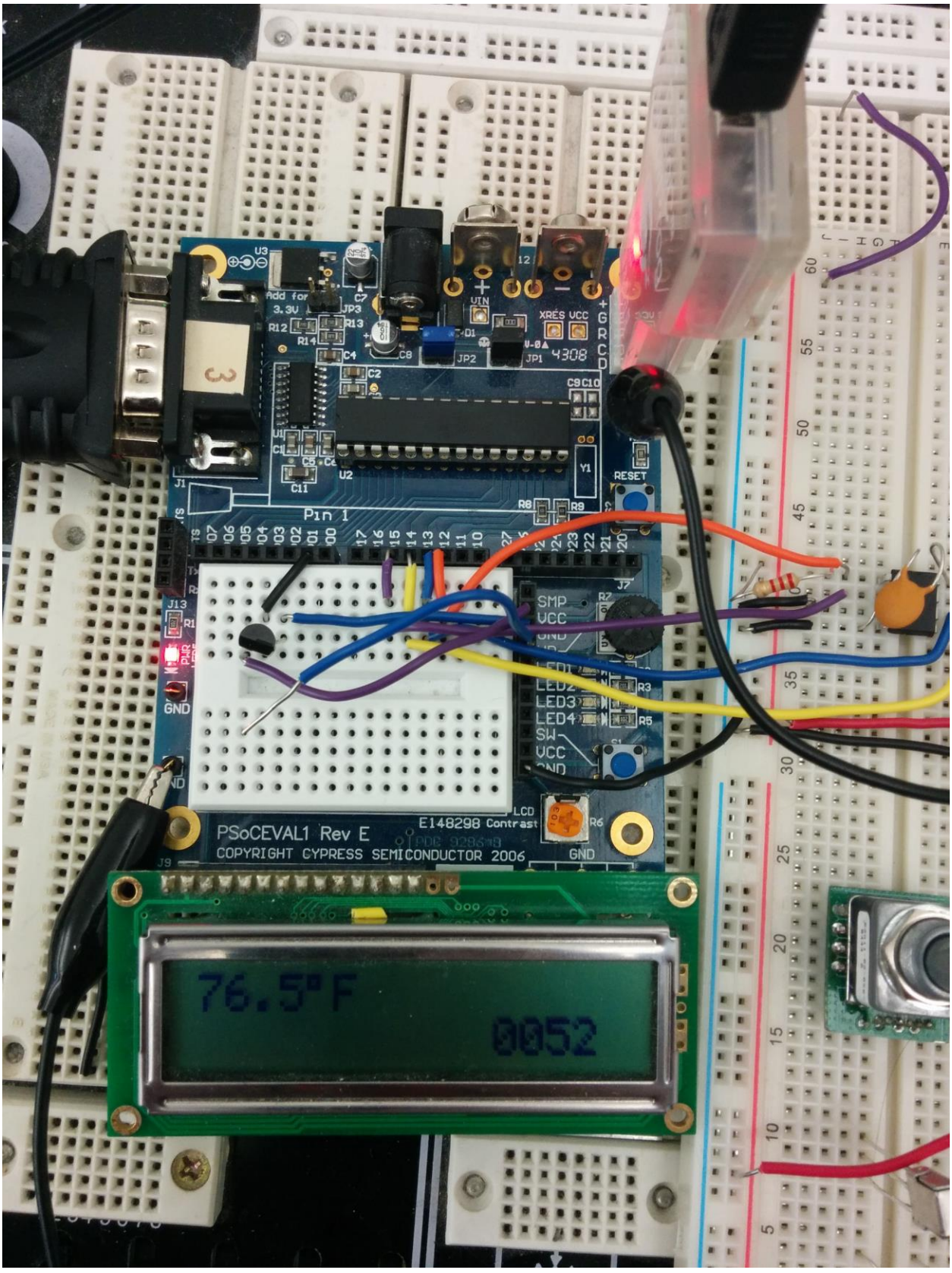
```
void clearScreen(void);
int sampleIn;
char *Degree;
float voltageIn;
int*  tempValue = 0x00;
char *voltageValuep = 0x00;
BYTE *tempValuep = 0x00;
int *status;
int i;
int j;
int count=0;
int iStatus;
float scaleFactor;

void main(void)
{
M8C_EnableGInt;
PGA_Start(PGA_HIGHPOWER);
ADCINC_Start (ADCINC_HIGHPOWER);
LCD_Start();
SleepTimer_Start();
SleepTimer_EnableInt();
SleepTimer_SetInterval(SleepTimer_1_HZ);
scaleFactor = ((float)1.3)/((float)16384);
```

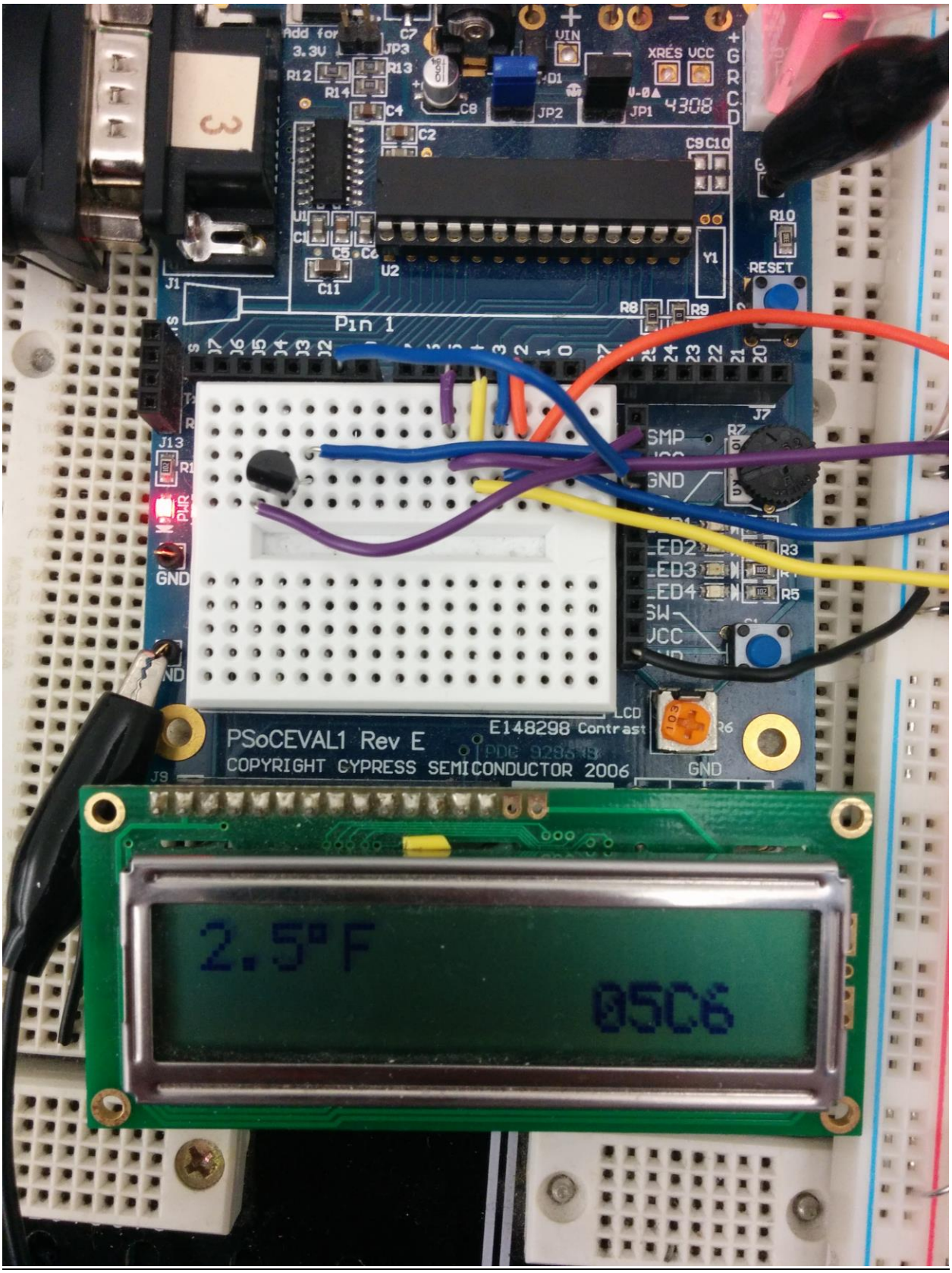
```
while (1){
count++;
ADCINC_GetSamples(1);
SleepTimer_SyncWait(0x01,SleepTimer_WAIT_RELOAD);

while(ADCINC_fIsDataAvailable() == 0);

sampleIn = ADCINC_iClearFlagGetData();
voltageIn = ((scaleFactor*(float)sampleIn)*100)+.05;

voltageValuep = ftoa(voltageIn, &iStatus);
clearScreen();
LCD_Position(0,0);                                  // Set LCD position to row 1 column 0

LCD_PrString(voltageValuep);                              // Print voltage value on LCD

if ((voltageIn<100)&&(voltageIn>10))
{
LCD_Position(0,4);
LCD_WriteData(0xDF);
LCD_PrCString("F          ");
}
else if(voltageIn>100)
{
LCD_Position(0,5);
LCD_WriteData(0xDF);
LCD_PrCString("F          ");
```

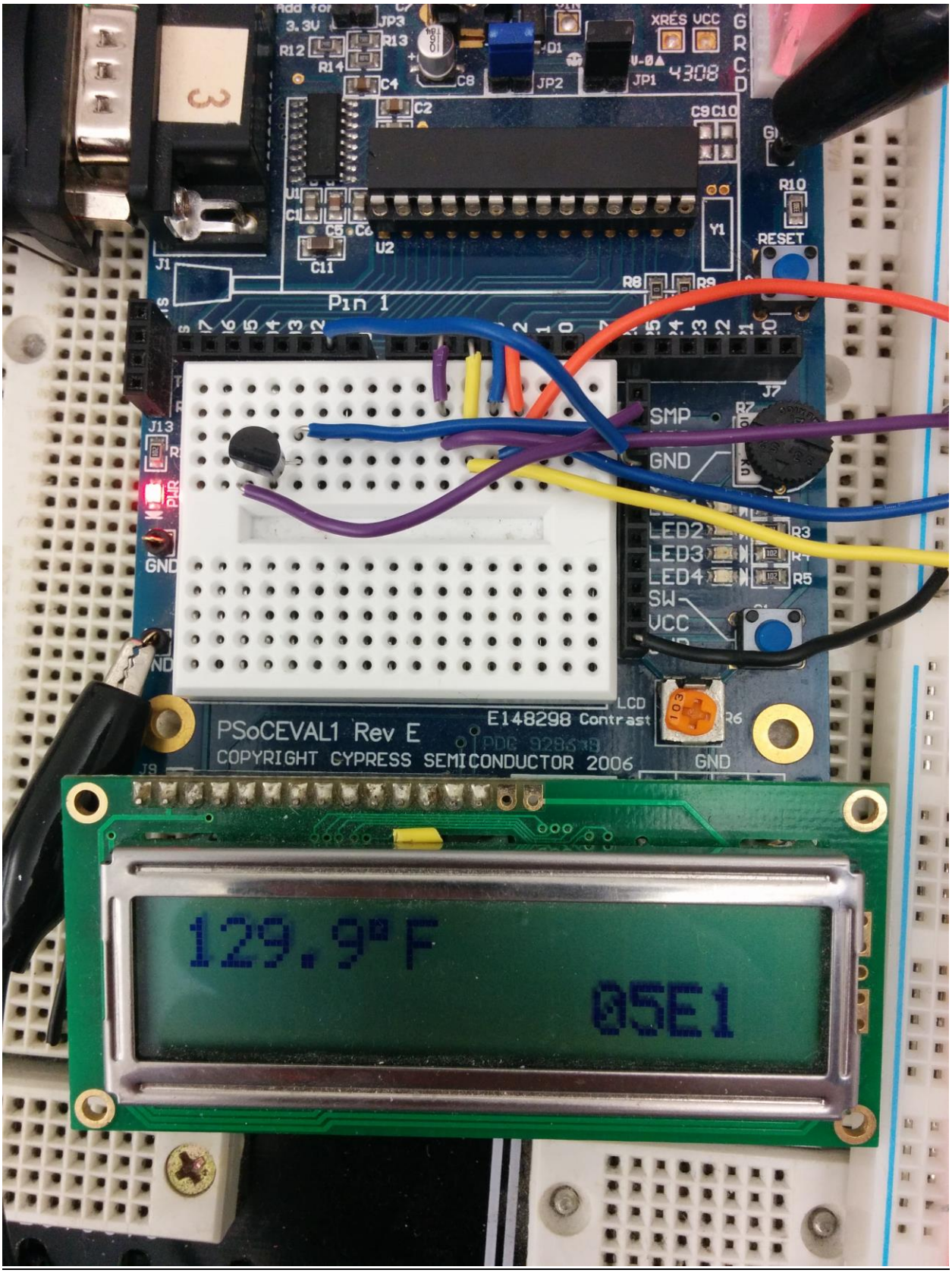
```
}else {
LCD_Position(0,3);
LCD_WriteData(0xDF);
LCD_PrCString("F          ");
}
LCD_Position(1,11);
LCD_PrHexInt(count);
}
}
void clearScreen(void){
LCD_Position(0,0);
LCD_PrCString("              ");
LCD_Position(1,0);
LCD_PrCString("              ");
}
```

Testing:

After the program is written is should be built and uploaded to the PSoC. Turn the PSoC on and Look at the LCD it should display something close to room temperature, around 76˚F. Next plug port 0 pin 1 into the potentiometer on your PSoC evaluation board. Spin the wheel in both directions to confirm your program can hit both the maximum and minimum temperatures. Refer to the pictures below.


76.5°F
0052
PSoCEVAL1 Rev E
COPYRIGHT CYPRESS SEMICONDUCTOR 2006
E148298
RESET
LED1
LED2
LED3
LED4
SW
VCC
GND


2.5°F
05C6
PSoCEVAL1 Rev E
COPYRIGHT CYPRESS SEMICONDUCTOR 2006
E148298
Contrast
RESET
Pin 1
GND
PWR


129.9°F
05E1
PSoCEVAL1 Rev E
COPYRIGHT CYPRESS SEMICONDUCTOR 2006
E148298
LCD Contrast
RESET
Pin 1
SMP
GND
LED2
LED3
LED4
SW
VCC
XRES
VCC
3.3V
4308