**Group 6: Temperature Monitoring**

**Members: Ngo Yi Run (1718773)**

**Ong Jing Wei (1719266)**

Name of Library Routines (File Name):

1. Temp\_Monitoring.c (Under Source Files) \*Main Program\*
2. Temp\_Monitoring.h (Under Header Files) \*Contains functions for Module\*
3. amt.h (Under Header Files) \*Provided by School\*
4. delay.c (Under Source Files) \*Provided by School\*
5. timer.c (Under Source Files) \*Provided by School\*

Function Prototypes:

* void ADCinit() [Initialise the ADC]
* void ADCread() [Start Conversion of ADC]
* float getTempC() [Temperature in Celsius, in float format]
* char\* getTempCString() [Temperature in Celsius, in pointer format]
* float getTempF() [Temperature in Fahrenheit, in float format]
* char\* getTempFString() [Temperature in Fahrenheit, in pointer format]
* float getTempK() [Temperature in Kelvin, in float format]
* char\* getTempKString() [Temperature in Kelvin, in pointer format]

General Overview:

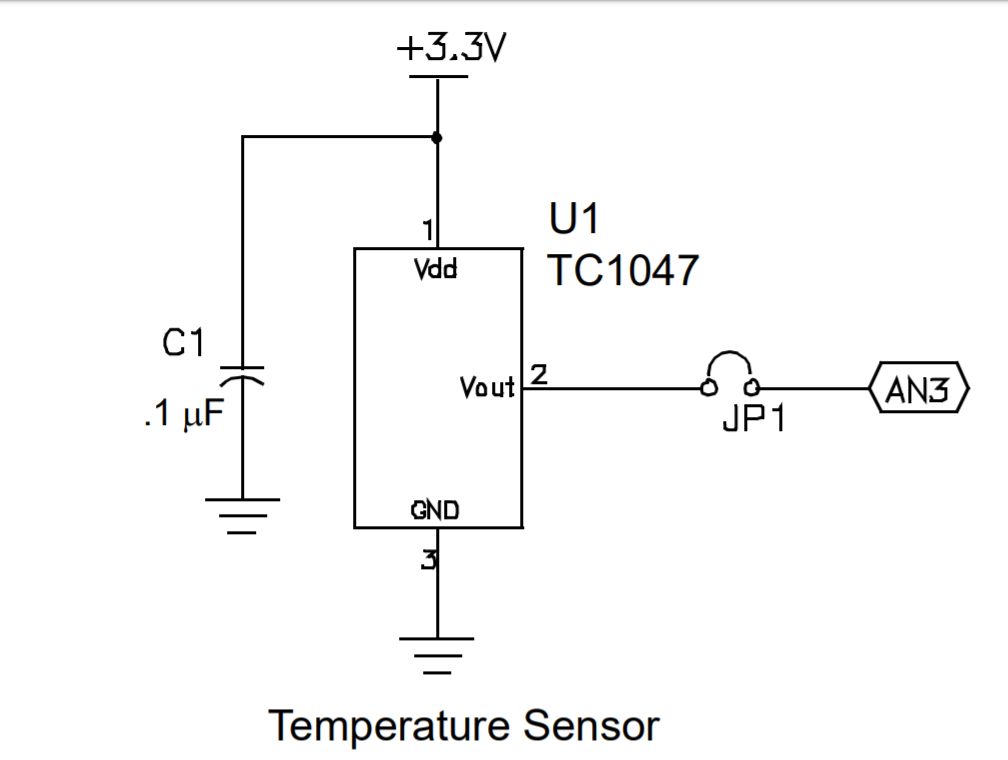
* For the Temperature Monitoring team, our main objective is to obtain the room temperature using the temperature sensor and display it onto the Liquid Crystal Display (LCD).
* Requires a I/O Port to configure the Temperature Sensor as an input.
* Output of temperature sensor connected to A/D Converter.
* The Temperature Sensor that is currently being used is an analog temperature sensor known as the Microchip TC1047 (U1), which is connected to an analog I/O pin of the PIC18F97J60 microcontroller. It can be disconnected by using a jumper.
* The TC1047 can measure temperature from -40C to +125C. For the TC1047, the output voltage range is typically 100mV at -40C, 500mV at 0C, 750mV at +25C, and 1.75V at +125C.
* In order to convert the analog data to digital, we will be using the 10-Bit Analog-To-Digital Converter (A/D) Module. The A/D Control Registers (ADCON0, ADCON1 and ADCON 2) will also be used to assist in the conversion.

TC1047 Main Features:

1. Supply Voltage Range: 2.7V to 4.4V
2. Wide Temperature Measurement Range: -40°C to +125°C
3. High Temperature Converter Accuracy: 2°C, Maximum at 25C
4. Linear Temperature Slope: 10mV/°C
5. Very Low Supply Current: 35uA Typical

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Board ID(s) | Type | Description |
| 1 | JP1 | Bridge | U1 (temperature sensor) to RA3 |

Jumper Description



Connection for the Temperature Sensor TC1047

**Code Explanations**

Under Temp\_Monitoring.h, we are initialising the function prototypes. We are using float as the temperature we are extracting will show a decimal point, the decimal point will help in making the temperature in giving a accurate reading.

Under Temp\_Monitoring.c, we are calling the function prototypes.

* The ADCinit() function is to initialise the Analog-To-Digital Converter.
* The ADCread function is to start the conversion of the Analog-To-Digital, this function is for internal use, users do not need to use it.
* The getTempC is to obtain the temperature in degree celsius and will return a floating value.
* The getTempCString is to obtain the temperature in degree celsius and will return a pointer value.
* The getTempF is to obtain the temperature in Fahrenheit and will return a floating value.
* The getTempFString is to obtain the temperature in Fahrenheit and will return a pointer value.
* The getTempK is to obtain the temperature in Kelvin and will return a floating value.
* The getTempKString is to obtain the temperature in Kelvin and will return a pointer value.

**Full Code Below**

**Under Temp\_Monitoring.h**

#ifndef \_TEMP\_MONITORING\_H

#define \_TEMP\_MONITORING\_H

void ADCinit(void);

float getTempC();

float getTempF();

float getTempK();

char\* getTempCString();

char\* getTempFString();

char\* getTempKString();

#endif

**Under Temp\_Monitoring.c**

#include <xc.h>

#include <math.h>

#include <stdio.h>

#include “amt.h”

#pragma config XINST = OFF // X Instructions

#pragma config FOSC = HS //Frequency Oscillator

#pragma config WDT = OFF //Watchdog Timer

void ADCinit(void);

float getTempC();

float getTempF();

float getTempK();

void ADCinit(void)//unsigned char ch)

{

TRISAbits.RA3 = 1; // Set RA3 as input

ADCON0 = 0b00001101; //Channel 3, A/D is off

ADCON1 = 0b00001011; // Vref(+)=AVdd, Vref(-)=AVss, AN3=analog input

ADCON2 = 0b00111000; // Left Justified, 20TAD, Fosc/2

}

unsigned int ADCread(int ch)///unsigned char ch)

{

ADCON0bits.CHS = ch;

ADCON0bits.GO\_DONE=1; // Start conversion

while(ADCON0bits.GO\_DONE); // Wait for the conversion to finish

return ADRESH;

}

float getTempC() // Temperature in Degree Celsius

{

unsigned int val=ADCread(3); //voltage in binary

float temperatureC;

temperatureC = round((val\*(11/850.0)-0.5)/0.01\*10)/10;

return temperatureC;

}

char\* getTempCString()

{

char message[5];

sprintf(message, "%f", getTempC());

return message;

}

float getTempF() // Temperature in Fahrenheit

{

unsigned int val=ADCread(3); //voltage in binary

float temperatureC, temperatureF;

temperatureC = (val\*(11/850.0)-0.5)/0.01\*10/10;

temperatureF = temperatureC\*1.8+32;

return temperatureF;

}

char\* getTempFString()

{

char message[5];

sprintf(message, "%f", getTempF());

return message;

}

float getTempK() // Temperature in Kelvin

{

unsigned int val=ADCread(3); //voltage in binary

float temperatureC, temperatureK;

temperatureC = (val\*(11/850.0)-0.5)/0.01\*10/10;

temperatureK = temperatureC + 273.15;

return temperatureK;

}

char\* getTempKString()

{

char message[5];

sprintf(message, "%f", getTempK());

return message;

}

**In the Main Program (For Testing)**

#include <xc.h>

#include "amt.h"

#include "Temp\_Monitoring.h"

#include <stdio.h>

#pragma config XINST = OFF //pragma assembler compiler directive

#pragma config FOSC = HS

#pragma config WDT = OFF

void main(void) {

char\* message;

LCD8init();

delay\_us(100);

ADCinit();

while(1)

{

LCD8send(0x01,0); //clear

LCD8send(0x80,0); //move cursor start of 1st line

message=getTempCString();

for (int i = 0; i < 5 ;i++) {

LCD8send(message[i],1);

}

delay\_ms(1000);

}

}