

Implementation of a Temperature Monitoring Circuit

Nakasit Niltawach, Joel T. Johnson, and Grant A. Hampson

May 6th, 2003

1 Introduction

As mentioned in [1], monitoring and recording the physical temperature of a microwave terminator is crucial for accurate internal calibration of the IIP radiometer. This document presents an implementation of an electronic circuit for such purposes. The idea of this circuit is based on the initial plan explained in [1].

2 Implementation of the circuit

This circuit is composed of three major components which are a thermistor YSI 44035, an analog-to-digital (A/D) converter AD7711 and a RCM2100 “Rabbit” microcontroller (\$89 each) with its prototype board. The details of how to setup these components are explained in [1]. Here, only additional connections which have not been described in [1] are presented. The circuit diagram is shown in Figure 1. The operation of this circuit can be described briefly as follows. AD7711 converts the ratio of the thermistor resistance (inversely proportional to temperature) to the reference resistor $R1$ (10 k Ω) to a digital signal. The four-wire configuration of the thermistor is employed to reduce error due to voltage drop in wires. A code written in “Dynamic C Version 7.04T2” (an integrated C compiler, editor, loader, and debugger created specifically for embedded control and communication applications) is used to program the microcontroller to setup the AD7711 chip and read the digital temperature data from AD7711 (the source code is *temp_micro.c* shown in Appendix A). The data are then transferred from it to a computer via an ethernet interface. Another program developed in LabWindows (C language based code, source code files: *tempmeas.c*, *tempmeas.h*, and *tempmeas.uir* shown in Appendices B, C, and D, respectively) is used to convert the data into temperature (by using a lookup table which converts the thermistor resistance into temperature, linear interpolation is used for the values that are not in the table. This table can be obtained from the file *ther_1000_ohm.xls* which is shown in Appendix E, but the file *YSI_thermistor.txt* is the one actually used in the program.) including presenting them on the screen and recording them into a text file. Figure 2 shows the complete circuit.

In the following, additional connections of AD7711 which have not been described in [1] are presented.

- PIN1 (SCLK): Connect to PIN PA5 of the microcontroller. PA5 generates clock pulses for read/write operation to/from the data/control registers of AD7711.
- PIN2 (MCLK IN): Connect to 10 MHz clock pulses.
- PIN3 (MCLK OUT): Left unconnected because CMOS compatible clock is used.
- PIN4 (AO): Connect to PIN PA7 of the microcontroller.
- PIN19 ($\overline{\text{TFS}}$): Connect to PIN PD1 of the microcontroller.
- PIN20 ($\overline{\text{RFS}}$): Connect to PIN PD3 of the microcontroller.
- PIN21 ($\overline{\text{DRDY}}$): Connect to PIN PD2 of the microcontroller.
- PIN22 (SDATA): Connect to PIN PD0 of the microcontroller.

3 Implementation Results

To check whether the circuit operates correctly, the temperature obtained from this circuit is compared to one obtained from a digital thermometer. The readings of the room temperature show that the difference in data is approximately 0.1-0.2 degree Celsius. This indicates that the circuit can monitor the temperature with reasonable accuracy. Note that when running the program in LabWindows, it is necessary to set up the “Run Option” as follows. In the Project window, click at Options Menu, choose Run Option, select Debugging Level to be Standard and uncheck the box “break on library errors.” Following this procedure will reduce the chance of unnecessary program termination while collecting data for a long period of time.

4 References

[1] J. T. Johnson and G. A. Hampson, “Initial Plan for a Precision Temperature Sensor,” September 10, 2002. <http://esl.eng.ohio-state.edu/~swe/iip/therm.pdf>.

AD7711 (TOP VIEW)

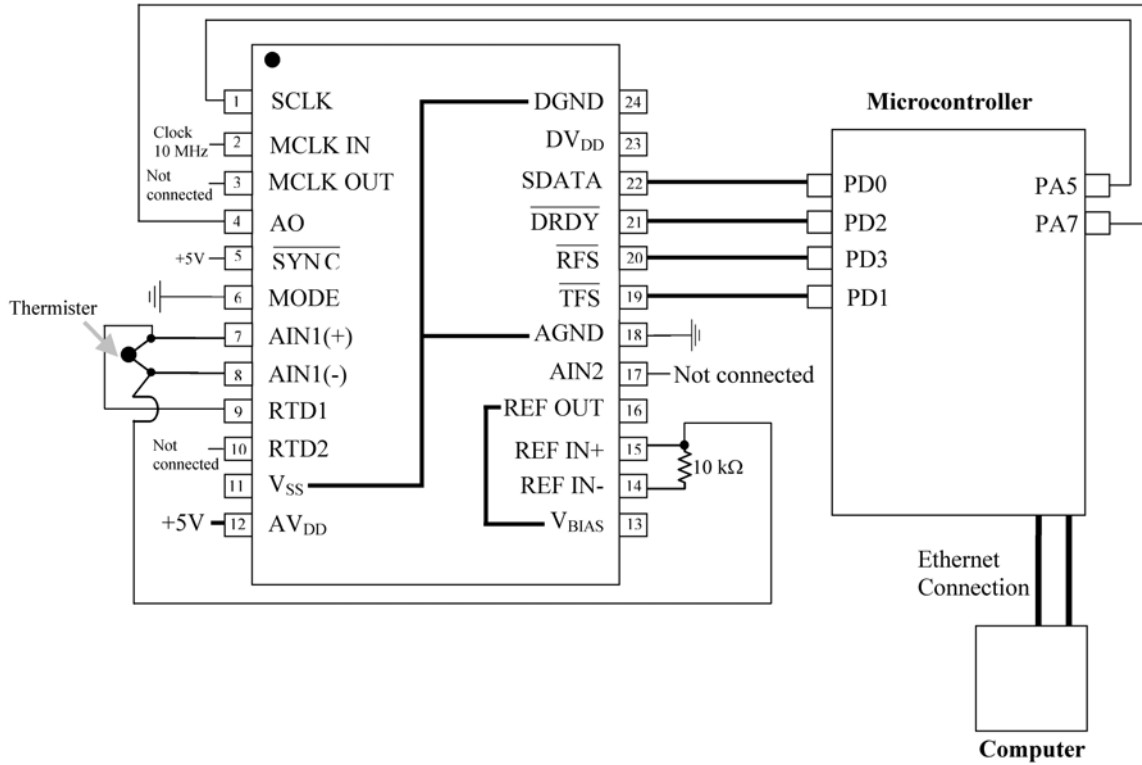


Figure 1 Circuit diagram.

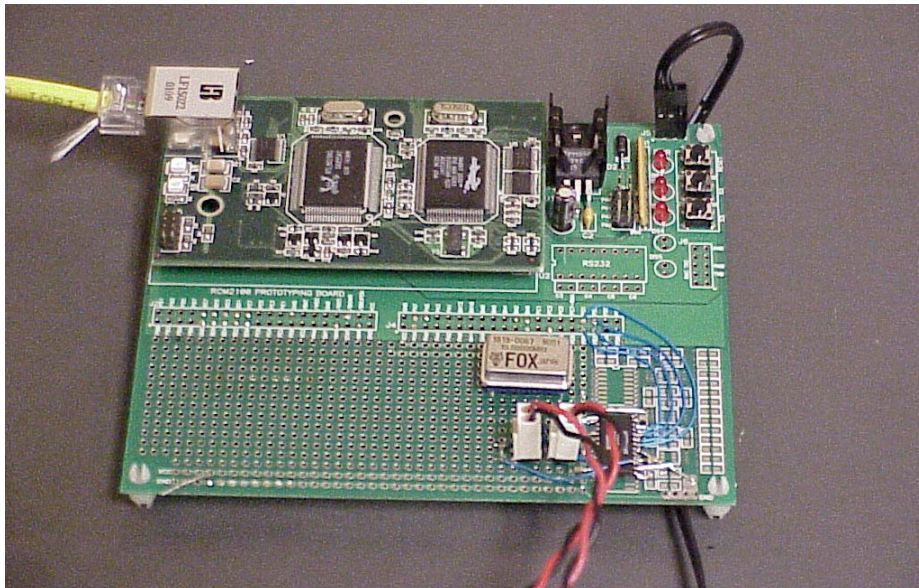


Figure 2: Complete circuit.

Appendix A: Rabbit Microcontroller Source Code (temp_micro.c)

```
/******  
temp_micro.c  
Grant Hampson June 21, 2002  
  
Downloading this program into RCM2100 "Rabbit" microcontroller  
for setting up and reading digital temperature from AD7711 chip  
  
*****/  
#define NUM_REG 2048  
#define ADCBITS 16  
#define MY_IP_ADDRESS "128.146.90.107" // 107 is INT, 108 is FFT, 109 is Blanker, 111 is DIF  
#define MY_NETMASK "255.255.255.0"  
#define MY_GATEWAY "128.146.90.1"  
#define PORT 7  
#memmap xmem  
#use dcrtcp.lib  
tcp_socket APB_Socket;  
unsigned long adcout;  
  
Init_Rabbit_Ports(void)  
{  
    // Initialisation of Rabbit Processor Ports  
  
    WrPortI(SPCR, &SPCRShadow, 0x84); // Port A is a byte-wide output  
    WrPortI(PDFR, &PDFRShadow, (PDFRShadow | 0x0F)); // Set PD3-0 to be inputs/outputs  
    WrPortI(PDDCR, &PDDCRShadow, (PDDCRShadow & 0xF0)); // Set PD3-0 to be normal I/O  
    WrPortI(PDDDR, &PDDDRShadow, (PDDDRShadow | 0x0A)); // Set PD3 (RFS) and PD1 (TFS) to be outputs  
    WrPortI(PDDDR, &PDDDRShadow, (PDDDRShadow & 0xFB)); // Set PD2 to be an input  
    WrPortI(PDDDR, &PDDDRShadow, (PDDDRShadow | 0x01)); // Set SDATA to be an output  
}  
  
Write_Control_Register(int show)  
{  
    const int control_reg[24] = {1,0,1,0,0,0,0,0,0,1,0,1,0,1,1,1,1,0,1,0,0,0,0,1};  
                                                                    // Background 16-bit  
    int i;  
  
    //  
    // Initialise Control Register of ADC  
    //  
    if (show)  
        printf("Writing to Control Reg :");  
  
    WrPortI(PADR, &PADRShadow, (PADRShadow & 0x7F)); // Set A0 low to access Control Register  
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow | 0x08)); // set RFS high to disable read  
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow & 0xFD)); // set TFS low to start write  
    WrPortI(PADR, &PADRShadow, (PADRShadow & 0xCF)); // set SCLK low to start transmission  
  
    for (i=0; i<24; i++) // write out control register loop  
    {  
        if (show)  
            printf("%d",control_reg[i]);  
  
        if (control_reg[i])  
            WrPortI(PDDR, &PDDRShadow, (PDDRShadow | 0x01)); // write data high  
        else  
            WrPortI(PDDR, &PDDRShadow, (PDDRShadow & 0xFE)); // write data low  
  
        WrPortI(PADR, &PADRShadow, (PADRShadow | 0x20)); // make SCLK high  
        WrPortI(PADR, &PADRShadow, (PADRShadow & 0xCF)); // make sclk low  
    }  
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow | 0x02)); // set TFS high  
    WrPortI(PDDDR, &PDDDRShadow, (PDDDRShadow & 0xFE)); // set SDATA to be an input  
  
    if (show)  
        printf("\n");  
}
```

```

Read_Control_Register(int show)
{
    int i, sdata[24];

    //
    //    Read Data from ADC Data Register
    //
    if (show)
        printf("Reading from control reg:");

    WrPortI(PADR, &PADRShadow, (PADRShadow & 0x7F));    // Set A0 low to access Control Register
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow & 0xF7));    // set RFS low to start read

    for (i=0; i<24; i++)    // read out data loop
    {
        sdata[i] = RdPortI(PDDR) & 0x01;
        WrPortI(PADR, &PADRShadow, (PADRShadow | 0x20));    // make SCLK high
        WrPortI(PADR, &PADRShadow, (PADRShadow & 0xCF));    // make sclk low
    }
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow | 0x08));    // set RFS high

    if (show)
    {
        for (i=0; i<24; i++)    // read out data loop
            printf("%d",sdata[i]);
        printf("\n");
    }
}

```

```

Read_ADC_Data(int show)
{
    int i, sdata[ADCBITS];
    unsigned long pwrtwo;

    //
    //    Read Data from ADC Data Register
    //

    WrPortI(PADR, &PADRShadow, (PADRShadow | 0x80));    // Set A0 high to access Data Register
    while (RdPortI(PDDR) & 0x04) {}    // wait for DRDY to go low

    WrPortI(PDDR, &PDDRShadow, (PDDRShadow & 0xF7));    // set RFS low to start write

    for (i=0; i<ADCBITS; i++)    // read out data loop
    {
        sdata[i] = RdPortI(PDDR) & 0x01;
        if (show) printf("%d",sdata[i]);

        WrPortI(PADR, &PADRShadow, (PADRShadow | 0x20));    // make SCLK high
        WrPortI(PADR, &PADRShadow, (PADRShadow & 0xCF));    // make sclk low
    }
    WrPortI(PDDR, &PDDRShadow, (PDDRShadow | 0x08));    // set RFS high

    if (show) printf("\n");

    pwrtwo = 1;
    adcout = 0;
    for (i=0; i<ADCBITS; i++)    // read out data loop
    {
        adcout = adcout + pwrtwo*sdata[ADCBITS-i-1];
        pwrtwo = pwrtwo * 2;
    }

    if (show) printf("%u\n",adcout);
}

```

```

main ()
{
    int i, status;
    int j;
    char buffer[NUM_REG];
    char sendbuffer[20]; // = "Go away\0";

    const int control_reg_1[24] = {1,0,1,0,0,0,0,0,0,1,0,1,0,1,1,1,0,1,0,0,0,0,1};
                                                // Background 16-bit

    Init_Rabbit_Ports(); // Initialise Rabbit ports

    Write_Control_Register(1); // Initialis ADC
    Read_Control_Register(1); // Verification

    sock_init(); // Initialise the TCP/IP chip

    while(1)
    {
        tcp_listen(&APB_Socket, PORT, 0, 0, NULL, 0); // listen to TCP/IP
        sock_wait_established(&APB_Socket, 0, NULL, &status); // wait for connection
        sock_mode(&APB_Socket, TCP_MODE_ASCII); // make an ascii connection
        Read_ADC_Data(1);
        // read ADC data
        sprintf(sendbuffer, "%d\n\0", adcout);

        // check the number we get from A/D
        printf("Number from A/D = %d\n", adcout);

        printf("Receiving connection\n"); //*****
        while(tcp_tick(&APB_Socket))
        {
            sock_wait_input(&APB_Socket, 0, NULL, &status); // Wait for recieved data
            if(sock_gets(&APB_Socket, buffer, 2048))
            {
                sock_puts(&APB_Socket, sendbuffer); // transmit temperature
                printf("received %s\n sending %s\n", buffer, sendbuffer); //*****
            }
        }

        sock_err:
        switch(status)
        {
            case 1: printf("Client Closed Connection\n"); break;
            case -1: printf("Connection Timed Out\n"); break;
        }
    }
}

```

Appendix B: tempmeas.c

```
#include <userint.h>
/* Grant Hampson 26 October 2002 */

/* The program is modified by Nakasit Niltawach */

#include <ansi_c.h>
#include <utility.h>
#include "tempmeas.h"
#include <tcpsupp.h>
#include <stdio.h>
#include <time.h>

#define FALSE 0
#define TRUE 1
#define NO_WAIT 0
#define NUM_PT 720 // Number of points we want to display on the screen (1 point ~ 1 sec)
// we may set it to be 720 which corresponds to 12 minutes

#define T_min 10 // unit in degree Celsius
#define T_max 100 // Temperature max and min

int temp; // Global variables
float tempfl; // Similar to "temp" but in float type
float t_delay_n; // time delay for each sample
float t_n_between; // time delay between each consecutive samples

float const2; // Use 2^15 (follow Prof Johnson's note)
int hpMain; /* handle to panel "MAIN" */

float YSI_data_fl[181]; // YSI thermister data in float format , 181 data points
float YSI_data[181]; // Reduce size of YSI_data_fl

int *TCPcallback (unsigned handle, int xType, int errCode, void *callbackData)
{
    if (xType == TCP_DISCONNECT)
        printf("Error occured %d",errCode);
    return(0);
}

int Get_ADC_Value(void)
{
    char ServerName[256] = "128.146.90.107\0";
    int i, PortNumber = 7, mode, status;
    unsigned char buffer[2];
    unsigned handle;
    char sendstring[20] = "Hello Grant\n\0", recstring[20];

    // printf("Connecting to %s\n", ServerName); // ***
    status = ConnectToTCPServer (&handle, PortNumber, ServerName, (tcpFuncPtr)&TCPcallback, NULL,
    50000); //***
    // printf("status %d\n",status);
    // exit(0);

    if (status<0) // status <0 means error which we can read what it means from "Help"
    {
        printf("Connection failed : error: %d\n",status); // ***
        status= DisconnectFromTCPServer (handle); // ***
        printf("Disconnecting : Status = %d\n",status); // ***
        //printf("TCP Client, Connection to server failed !\n");

        // try to make the program not stop if error occurs
        for (i=0;i<20;i++) recstring[i]='\0'; // initialise received string

        return(0);
    }
}
```

```

}
else
{
    status = ClientTCPWrite (handle, sendstring, 20, 1000);
    //printf("Sending %d, %s\n",status,sendstring); //*****

    for (i=0;i<20;i++) recstring[i]='\0'; // initialise received string
    //recstring[0]='\0';

    status = ClientTCPRead (handle, recstring, 20, 1000); // "status" = number of bytes read
    //printf("Reading %d, %s\n",status,recstring); //*****

    while (status <= 0)
    {
        //Delay(0.2);
        status = ClientTCPRead (handle, recstring, 20, 1000);
        //printf("Retry:Reading %d, %s\n",status,recstring);
    }
}

DisconnectFromTCPserver(handle);
//printf("status %d\n",status);
//printf("exit"); exit(0);

if (status > 0)
    return(atoi(recstring));
else
    return(-1);
}

```

```

void main(void)
{
    int    numsamples, ns, i;
    int    iControlID, hHandle, bQuit = FALSE; // LabWindows/CVI-specific GUI stuff

    int    ind_read, ind_Tmin, ind_Tmax, ind_write, ind_extra , ind_lock;
    float  temporary, temp_low, temp_high, R_low, R_high, delta_temp ;

    double t1, t2, tn ;
    char   *time_full_start, *time_full_stop ;
    char   *date_start, *date_stop ;

    float  *x_extra, *y_extra; // Pointer, limit the number of points shown on the screen

    int ind_save, ind_wr_output, num_x ;

    int loop_while;
    int flag_1; // flag_1==0, use tempfl_1
               // flag_1==1, use tempfl_2

    float t_delay_pre, tempfl_1[2*NUM_PT], tempfl_2[2*NUM_PT];
    float t_delay_n_1[2*NUM_PT], t_delay_n_2[2*NUM_PT];

    /* Read Resistance-Temperature data of YSI Thermistor *****/

    FILE *InFile;
    FILE *OutFile;

    if ((InFile=fopen("YSI_thermistor.txt","r")) == NULL)
        // check whether the file is in the correct directory
    {
        fprintf(stderr, "Can't open input file 'YSI_thermistor'. Maybe the file is in the wrong
        directory.\n");
        exit(0);
    }
}

```



```

if (i==0) printf("\n\n\n");

temp = Get_ADC_Value(); // read data from the microcontroller

// check that we can read the input OK
if (temp<= 0.00001)
{
    temp=0;
    printf("\n\ntemp = %f\n", temp);
    printf("Something wrong with input from the thermistor\n");
    printf("or we may try to close the LabWindow program and run it again\n");
    printf("We may have to download the program in the microcontroler again\n");
    //exit(0);
}

// -----

// Check starting and stoping time + time delay of each sample
// 1st          nth (last sample obtained)
// | ..... |
// t1 ..... tn
// The variable "t" is "tn-t1"

// starting time
if (i==0)
{
    t1=Timer();
    time_full_start=TimeStr();
    printf("The STARTING time in HOUR:MINUTE:SECOND is ");
    fputs(time_full_start, stdout); printf("\n");

    date_start=DateStr();
    printf("The STARTING date in MONTH:DATE:YEAR is ");
    fputs(date_start, stdout); printf("\n");

    // Write "Date and Time" to OUTPUT.txt (create a new file)
    if ( ( OutFile = fopen("OUTPUT.txt","w")==0 )
        {
            fprintf(stderr, "Can't open output file.\n");
            printf("Can't open output file.\n");
        }

    fprintf(OutFile,"Time Start -- %s\n", time_full_start);
    fprintf(OutFile,"Date Start -- %s\n", date_start);
    fclose(OutFile);
}

// time delay for each sample, we use "i" because later
// we have to use "PlotWaveForm" which requires index i=0

if (i==0)
{
    t_delay_n=t1-t1;
    t_n_between=0.0;
    t_delay_pre=t1;
}

if (i>=0.5)
{
    tn=Timer();
    t_delay_n=tn-t1;
    t_n_between=tn-t_delay_pre;
    t_delay_pre=tn;

    //printf("Time delay = %f", (tn-t1));
}

// The equation of the themistor resistance (RT) is (R1=10k ohm, 2^15 = 32,768)

```

```

// Obtained number from microcontroller * 0.5 * R1 / 2^15
const2=32768; // 2^15

// We use index "i+1" because "i" starts at 0 index
// At this point, we get resistance
tempfl= ((float)temp)*0.5*1.0e4/(float)const2;

// We now convert "Resistance" to "Temperature" -----
ind_look=0;
temporary=YSI_data[ind_look];

while (temporary >= tempfl)
{
    ind_look=ind_look+1;
    temporary=YSI_data[ind_look];
    //printf("temporary %f , tempfl[i]= %f\n",temporary,tempfl[i]); exit(0);
}
ind_look=ind_look-1;

// Don't forget negative temperature coefficient!!
// define the lower limit of the temperature range
// define the higher limit of the temperature range

temp_low =(float)(T_min+(ind_look));
temp_high=(float)(temp_low+1);

// higher value of R corresponding to lower temperature
R_high=YSI_data[ind_look];
R_low =YSI_data[ind_look+1];

// convert Resistance to temperature with linear interpolation

tempfl=temp_low + (tempfl-R_high)/(R_low-R_high) ;
printf("Td_between_sam = %f, Time delay = %f, Temperature = %f. \n", t_n_between,
t_delay_n, tempfl);

// -----

if ((OutFile = fopen("OUTPUT.txt","a"))==0 )
    // "a" continue to write the same file as above
{
    fprintf(stderr, "Can't open output file 'OUTPUT.txt'.\n");
    printf("Can't open output file.\n");
}

fprintf(OutFile,"%f %f\n", t_delay_n, tempfl);
fclose(OutFile);

// %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

// Dealing with plot data
// No memory limitation, collect data as long as we need

// choose the right number of data to plot, num_x = number of samples
if (ind_save==0) num_x=i+1;

// plot with index on the x-axis
// PlotWaveform (hpMain, PANEL_TEMPCHART, tempfl, num_x, VAL_FLOAT, 1, 0, 0,
//              1, VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_RED);

// Plot time delay on x-axis *****
// Note that we have to use "DeleteGraphPlot" first so that "PlotXY"
// will not create excessive time delay later
// (NUM_PT = number of points we want to show on the screen)

```

```

// plot as usual
if ( (flag_1==0) & i<=(NUM_PT-1) )
{
    tempfl_1[i]=tempfl;
    t_delay_n_1[i]=t_delay_n;

    DeleteGraphPlot(hpMain, PANEL_TEMPCHART,-1, VAL_IMMEDIATE_DRAW);
    PlotXY (hpMain, PANEL_TEMPCHART, t_delay_n_1, tempfl_1, num_x,
            VAL_FLOAT, VAL_FLOAT, VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_RED);
}

// switch between "tempfl_1" and "tempfl_2"
// so that we have array to plot the newest 720 data points

if ( (flag_1==0) & (i>(NUM_PT-1)) & (i<=(2*NUM_PT-1)) )
{
    tempfl_1[i]=tempfl;
    t_delay_n_1[i]=t_delay_n;

    // use "Pointer" to show the latest "NUM_PT" data
    x_extra=t_delay_n_1+( i-(NUM_PT-1) );
    y_extra=tempfl_1+( i-(NUM_PT-1) );

    DeleteGraphPlot(hpMain, PANEL_TEMPCHART,-1, VAL_IMMEDIATE_DRAW);
    PlotXY (hpMain, PANEL_TEMPCHART, x_extra, y_extra, NUM_PT,
            VAL_FLOAT, VAL_FLOAT, VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_RED);

    // Prepare another set of array
    tempfl_2[i-NUM_PT]=tempfl;
    t_delay_n_2[i-NUM_PT]=t_delay_n;

    if ( i==(2*NUM_PT-1) )
    {
        i=(2*NUM_PT-1)-NUM_PT;
        flag_1=1;
    }
}

if ( (flag_1==1) & (i>(NUM_PT-1)) & (i<=(2*NUM_PT-1)) )
{
    tempfl_2[i]=tempfl;
    t_delay_n_2[i]=t_delay_n;

    // use "Pointer" to show the latest "NUM_PT" data
    x_extra=t_delay_n_2+( i-(NUM_PT-1) );
    y_extra=tempfl_2+( i-(NUM_PT-1) );

    DeleteGraphPlot(hpMain, PANEL_TEMPCHART,-1, VAL_IMMEDIATE_DRAW);
    PlotXY (hpMain, PANEL_TEMPCHART, x_extra, y_extra, NUM_PT,
            VAL_FLOAT, VAL_FLOAT, VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_RED);

    // Prepare another set of array
    tempfl_1[i-NUM_PT]=tempfl;
    t_delay_n_1[i-NUM_PT]=t_delay_n;

    if ( i==(2*NUM_PT-1) )
    {
        i=(2*NUM_PT-1)-NUM_PT ;
        flag_1=0;
    }
}

// %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

// Botton "Stop collecting data"

if (GetUserEvent(NO_WAIT, &hHandle, &iControlID))
{
    if (iControlID==PANEL_SAVE)
    {
        t2=Timer();
        time_full_stop=TimeStr();
        date_stop=DateStr();
        ind_save=1; // we clicked at "SAVE" botton

        printf("The STOPPING time in HOUR:MINUTE:SECOND is ");
        fputs(time_full_stop, stdout);
        printf("\nThe STOPPING date in MONTH:DATE:YEAR is ");
        fputs(date_stop, stdout); printf("\n");
        printf("\n\nTime between the first and the last samples = %f seconds\n",
            (t2-t1));

        loop_while=FALSE;

        if ((OutFile = fopen("OUTPUT.txt","a")==0 )
            // "a" continue to write the same file as above
        {
            fprintf(stderr, "Can't open output file.\n");
            printf("Can't open output file.\n");
        }

        fprintf(OutFile,"Time Stop -- %s\n", time_full_start);
        fprintf(OutFile,"Date Stop -- %s\n", date_start);
        fprintf(OutFile,"Total monitoring time = %f seconds\n", (t2-t1));
        fclose(OutFile);

    }
}

// adjust time delay such that the time between each sample is 1 second
Delay(0.92);
i=i+1;

} // loop "while" counting index "i"

//break; // exit "while" loop

} // loop "if PANEL_ACQUIRE"

} // loop "if PANEL==hHandle"

Delay(0.1); // stop PC polling so hard

}

// *****

/* ----- */

// ** We have to click "QUIT" on the screen
// to save the stop time and

printf("\n\nThe file is saved, please change its name\n");
printf("before starting to collect another set of data. ***** \n");

/* ----- */

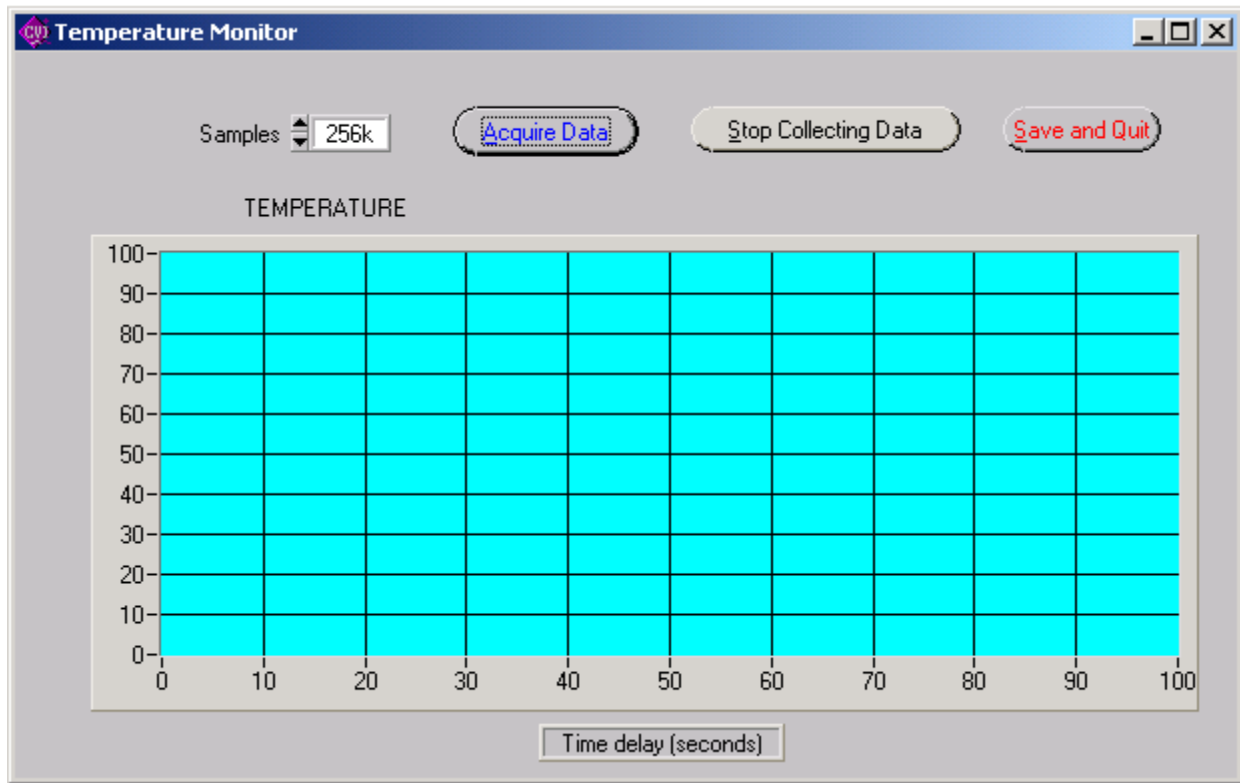
} // main program

```

Appendix C: tempmeas.h

```
/* *****  
/* LabWindows/CVI User Interface Resource (UIR) Include File */  
/* Copyright (c) National Instruments 2003. All Rights Reserved. */  
/*  
/* WARNING: Do not add to, delete from, or otherwise modify the contents */  
/* of this include file. */  
/* *****  
  
#include <userint.h>  
  
#ifdef __cplusplus  
    extern "C" {  
#endif  
  
    /* Panels and Controls: */  
  
#define PANEL 1  
#define PANEL_ACQUIRE 2  
#define PANEL_QUIT 3  
#define PANEL_NUMBERSAMPLES 4  
#define PANEL_SAVE 5  
#define PANEL_TEXTBOX 6  
#define PANEL_TEMPCHART 7  
  
    /* Menu Bars, Menus, and Menu Items: */  
  
    /* (no menu bars in the resource file) */  
  
    /* (no callbacks specified in the resource file) */  
  
#ifdef __cplusplus  
    }  
#endif
```

Appendix D: tempmeas.uir



Appendix E: YSI_thermistor.txt

-80	278,800	-38	17,540	4	2286	46	483.2	88	143.4
-79	258,100	-37	16,590	5	2192	47	467.8	89	139.8
-78	239,100	-36	15,700	6	2102	48	452.9	90	136.2
-77	221,700	-35	14,860	7	2017	49	438.6	91	132.8
-76	205,600	-34	14,070	8	1936	50	424.8	92	129.5
-75	190,800	-33	13,330	9	1859	51	411.6	93	126.3
-74	177,200	-32	12,630	10	1785	52	398.8	94	123.2
-73	164,700	-31	11,970	11	1714	53	386.5	95	120.2
-72	153,100	-30	11,350	12	1647	54	374.7	96	117.3
-71	142,500	-29	10,770	13	1582	55	363.2	97	114.4
-70	132,600	-28	10,220	14	1521	56	352.2	98	111.7
-69	123,500	-27	9705	15	1462	57	341.6	99	109
-68	115,100	-26	9218	16	1406	58	331.3	100	106.4
-67	107,300	-25	8758	17	1353	59	321.5		
-66	100,100	-24	8323	18	1302	60	311.9		
-65	93,480	-23	7914	19	1253	61	302.7		
-64	87,300	-22	7527	20	1206	62	293.9		
-63	81,580	-21	7161	21	1161	63	285.3		
-62	76,280	-20	6815	22	1118	64	277		
-61	71,350	-19	6489	23	1077	65	269		
-60	66,780	-18	6180	24	1038	66	261.3		
-59	62,530	-17	5887	25	1000	67	253.9		
-58	58,590	-16	5611	26	963.9	68	246.7		
-57	54,920	-15	5349	27	929.4	69	239.7		
-56	51,500	-14	5101	28	896.3	70	233		
-55	48,320	-13	4866	29	864.5	71	226.5		
-54	45,360	-12	4643	30	834	72	220.2		
-53	42,600	-11	4432	31	804.8	73	214.1		
-52	40,030	-10	4232	32	776.8	74	208.3		
-51	37,630	-9	4042	33	749.9	75	202.6		
-50	35,390	-8	3862	34	724.1	76	197.1		
-49	33,300	-7	3691	35	699.4	77	191.8		
-48	31,350	-6	3529	36	675.6	78	186.7		
-47	29,520	-5	3374	37	652.7	79	181.7		
-46	27,810	-4	3228	38	630.8	80	176.9		
-45	26,220	-3	3088	39	609.7	81	172.2		
-44	24,720	-2	2956	40	589.5	82	167.7		
-43	23,320	-1	2830	41	570	83	163.3		
-42	22,010	0	2710	42	551.2	84	159.1		
-41	20,790	1	2596	43	533.2	85	154.9		
-40	19,640	2	2487	44	515.9	86	151		
-39	18,560	3	2384	45	499.2	87	147.1		

