```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL  // 8 MHz.h>
#include <util/delay.h>

        unsigned char buttonPressed;
        int tempOut;
        int tempIn;
        int realTempOut;
        int realTempIn;
        int maxAlarm;
        int maxAlarmArmed = 0;
        int minAlarm;
        int minAlarmArmed = 0;
        int maxAlarmNeg = 0;
        int minAlarmNeg = 0;
        int alarmActive = 0;
        char printDigits[3];
        int counter;

        int maxTempOut;
        int minTempOut;
        int maxTempIn;
        int minTempIn;

        char hundredsOut;
        char tensOut;
        char singlesOut;

        char hundredsIn;
        char tensIn;
        char singlesIn;



        /*
        modeSet sets which mode the weather station is in
        0 = shows the current temperature (in/out).
        1 = enters the max-alarm mode (lets the user enter the upper limit)
        2 = enters the min-alarm mode (lets the user enter the lower limit)
        3 = shows the max/min/avg temperature for the out sensor
        */
        int modeSet = 0;


//DISPLAY
void disp_wait(void){
        _delay_ms(5);
        return;

}

void write_cmd(char val) {
        disp_wait();
        PORTB=val;
        PORTD=0x00;
        PORTD=0x04;
        PORTD=0x00;
        return;
}
```

```
void disp_clear() {
        write_cmd(0x01); //clear disp
        return;
}

void disp_init() {
        disp_clear();
        write_cmd(0x38); //func set
        write_cmd(0x0E); //disp on
        write_cmd(0x06); //entry mode set
        //write_cmd(0x1C); //cursor set
        return;
}

void start_tempRead(){
        ADCSRA = 0xDC;
        ADMUX = 0x20;
        return;
}

void resetDigits(){
        for(int i = 0; i < 3; i++){
                printDigits[i] = ' ';
        }
        maxAlarm = 100;
        minAlarm = -100;
        counter = 0;
        maxAlarmNeg = 0;
        minAlarmNeg = 0;
        maxAlarmArmed = 0;
        minAlarmArmed = 0;
        alarmActive = 0;
        return;
}

void startUp(){

        //AVR PIN-settings
        DDRA = 0x08; //sets PA3 to output and all other PA0-2 & PA4-7 to
input.
        DDRB = 0xff; //sets all B-ports to output (to the display)
        DDRC = 0x01; //sets the backlight-transistor-controller to 1.
        DDRD = 0x07; //sets the display controller pins to output.

        //interrupt-settings
        MCUCR = 0x0C; //sets the MCU control register to generate
interuption requests on any logical change on INT1.
        GICR = 0x80; //sets the General Interrupt Control register to
enable INT1
        GIFR = 0x80; //sets the General Interrupt Flag register fir INT1

        //keyboard-settings
        PORTA = 0x08 & 0x08; //controller is "high" as default.

        //Display-settings
        disp_init(); //Initializes the display.
        PORTC = 0x01 & 0x01; //sets the backlight to "Off"

        //Start temperature readings.
        start_tempRead();
```

```
        //Resets the alarm limits.
        resetDigits();

        //Prints the 'main' screen to the display.
        disp_tempOut();

        //Initializes the max/min temperature variables.
        maxTempOut = -100;
        minTempOut = 100;
        maxTempIn = -100;
        minTempIn = 100;
        maxAlarmArmed = 0;
        minAlarmArmed = 0;

        sei();
        return;

}

void disp_home(){
        write_cmd(0x03);
        return;
}

void disp_writeCh(char val) {
        disp_wait();
        PORTB=val;
        PORTD=0x01;
        PORTD=0x05;
        PORTD=0x01;
        return;
}

char writeDigit(char digitToConvert){
        if(digitToConvert == 0){
                return '0';
        }
        if(digitToConvert == 1){
                return '1';
        }
        if(digitToConvert == 2){
                return '2';
        }
        if(digitToConvert == 3){
                return '3';
        }
        if(digitToConvert == 4){
                return '4';
        }
        if(digitToConvert == 5){
                return '5';
        }
        if(digitToConvert == 6){
                return '6';
        }
        if(digitToConvert == 7){
                return '7';
        }
        if(digitToConvert == 8){
                return '8';
        }
```

```
        if(digitToConvert == 9){
                return '9';
        }
        else
                return ' ';
}

void disp_tempOut(){
        disp_home();

        disp_writeCh('T');
        disp_writeCh('e');
        disp_writeCh('m');
        disp_writeCh('p');
        disp_writeCh(' ');
        disp_writeCh('i');
        disp_writeCh('n');
        disp_writeCh('s');
        disp_writeCh('i');
        disp_writeCh('d');
        disp_writeCh('e');
        disp_writeCh(':');
        disp_writeCh(' ');
        disp_writeCh(' ');

        if(alarmActive == 1){
                write_cmd(0x94);
                disp_writeCh('A');
                disp_writeCh('l');
                disp_writeCh('a');
                disp_writeCh('r');
                disp_writeCh('m');
                disp_writeCh(' ');
                disp_writeCh('a');
                disp_writeCh('c');
                disp_writeCh('t');
                disp_writeCh('i');
                disp_writeCh('v');
                disp_writeCh('e');
                disp_writeCh('!');
        } else
        write_cmd(0x94);
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');

        write_cmd(0xC0);

        disp_writeCh('T');
        disp_writeCh('e');
        disp_writeCh('m');
```

```
        disp_writeCh('p');
        disp_writeCh(' ');
        disp_writeCh('o');
        disp_writeCh('u');
        disp_writeCh('t');
        disp_writeCh('s');
        disp_writeCh('i');
        disp_writeCh('d');
        disp_writeCh('e');
        disp_writeCh(':');
        disp_writeCh(' ');
        if(tempOut >= 0){
                disp_writeCh(' ');
        }
        if(tempOut < 0){
                disp_writeCh('-');
        }

        write_cmd(0xD4);

        disp_writeCh('B');
        disp_writeCh('=');
        disp_writeCh('S');
        disp_writeCh('t');
        disp_writeCh('a');
        disp_writeCh('t');
        disp_writeCh('s');
        disp_writeCh(' ');
        disp_writeCh('C');
        disp_writeCh('=');
        disp_writeCh('M');
        disp_writeCh('a');
        disp_writeCh('x');
        disp_writeCh(' ');
        disp_writeCh('D');
        disp_writeCh('=');
        disp_writeCh('M');
        disp_writeCh('i');
        disp_writeCh('n');
        return;
}

void enter_maxAlarm(){
        disp_home();
        disp_writeCh('E');
        disp_writeCh('n');
        disp_writeCh('t');
        disp_writeCh('e');
        disp_writeCh('r');
        disp_writeCh(' ');
        disp_writeCh('u');
        disp_writeCh('p');
        disp_writeCh('p');
        disp_writeCh('e');
        disp_writeCh('r');
        disp_writeCh(' ');
        disp_writeCh('l');
        disp_writeCh('i');
        disp_writeCh('m');
        disp_writeCh('i');
        disp_writeCh('t');
```

```
        disp_writeCh(':');

        write_cmd(0xC0);

        if(maxAlarmNeg == 1){
                disp_writeCh('-');
        }
        disp_writeCh(printDigits[0]);
        disp_writeCh(printDigits[1]);
        disp_writeCh(printDigits[2]);
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');

        write_cmd(0xD4);
        disp_writeCh('A');
        disp_writeCh('=');
        disp_writeCh('M');
        disp_writeCh('a');
        disp_writeCh('i');
        disp_writeCh('n');
        disp_writeCh(' ');
        disp_writeCh('E');
        disp_writeCh('=');
        disp_writeCh('N');
        disp_writeCh('E');
        disp_writeCh('G');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh('F');
        disp_writeCh('=');
        disp_writeCh('O');
        disp_writeCh('K');
        disp_writeCh(' ');
        return;
}

void enter_minAlarm(){
        disp_home();
        disp_writeCh('E');
        disp_writeCh('n');
        disp_writeCh('t');
        disp_writeCh('e');
        disp_writeCh('r');
        disp_writeCh(' ');
        disp_writeCh('l');
        disp_writeCh('o');
        disp_writeCh('w');
        disp_writeCh('e');
```

```
        disp_writeCh('r');
        disp_writeCh(' ');
        disp_writeCh('l');
        disp_writeCh('i');
        disp_writeCh('m');
        disp_writeCh('i');
        disp_writeCh('t');
        disp_writeCh(':');

        write_cmd(0xC0);

        if(minAlarmNeg == 1){
                disp_writeCh('-');
        }
        disp_writeCh(printDigits[0]);
        disp_writeCh(printDigits[1]);
        disp_writeCh(printDigits[2]);
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');

        write_cmd(0xD4);
        disp_writeCh('A');
        disp_writeCh('=');
        disp_writeCh('M');
        disp_writeCh('a');
        disp_writeCh('i');
        disp_writeCh('n');
        disp_writeCh(' ');
        disp_writeCh('E');
        disp_writeCh('=');
        disp_writeCh('N');
        disp_writeCh('E');
        disp_writeCh('G');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh('F');
        disp_writeCh('=');
        disp_writeCh('O');
        disp_writeCh('K');
        disp_writeCh(' ');
        return;
}

void show_stats(){
        disp_home();
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
```

```
        disp_writeCh(' ');
        disp_writeCh('M');
        disp_writeCh('a');
        disp_writeCh('x');
        disp_writeCh(' ');
        disp_writeCh('/');
        disp_writeCh(' ');
        disp_writeCh('M');
        disp_writeCh('i');
        disp_writeCh('n');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');

        write_cmd(0x94);

        disp_writeCh('O');
        disp_writeCh('u');
        disp_writeCh('t');
        disp_writeCh(':');

        int maxHundredsOut = maxTempOut/100;
        int maxTensOut = (maxTempOut-maxHundredsOut*100)/10;
        int maxSinglesOut = maxTempOut-maxHundredsOut*100-maxTensOut*10;

        if(maxTempOut < 0){
                disp_writeCh('-');
        } else
                disp_writeCh(' ');
        if(maxTempOut >= 10){
                disp_writeCh(writeDigit(maxTensOut));
        } else
                disp_writeCh(' ');
        if(maxTempOut >= 0){
                disp_writeCh(writeDigit(maxSinglesOut));
        }
        if(maxTempOut < 0){
                maxSinglesOut = maxSinglesOut-2*maxSinglesOut;
                disp_writeCh(writeDigit(maxSinglesOut));
        }
        if(maxTempOut < -10){
                maxTensOut=maxTensOut-2*maxTensOut;
                disp_writeCh(writeDigit(maxTensOut));
        }

        disp_writeCh(' ');
        disp_writeCh('/');

        int minHundredsOut = minTempOut/100;
        int minTensOut = (minTempOut-minHundredsOut*100)/10;
        int minSinglesOut = minTempOut-minHundredsOut*100-minTensOut*10;

        if(minTempOut < 0){
                disp_writeCh('-');
        } else
                disp_writeCh(' ');
        if(minTempOut >= 10){
                disp_writeCh(writeDigit(minTensOut));
        } else
                disp_writeCh(' ');
```

```
        if(minTempOut >= 0){
                disp_writeCh(writeDigit(minSinglesOut));
        }
        if(minTempOut < 0){
                minSinglesOut = minSinglesOut-2*minSinglesOut;
                disp_writeCh(writeDigit(minSinglesOut));
        }
        if(minTempOut < -10){
                minTensOut=minTensOut-2*minTensOut;
                disp_writeCh(writeDigit(minTensOut));
        }

        write_cmd(0xC0);

        disp_writeCh('I');
        disp_writeCh('n');
        disp_writeCh(':');
        disp_writeCh(' ');

        int maxHundredsIn = maxTempIn/100;
        int maxTensIn = (maxTempIn-maxHundredsIn*100)/10;
        int maxSinglesIn = maxTempIn-maxHundredsIn*100-maxTensIn*10;

        if(maxTempIn < 0){
                disp_writeCh('-');
        } else
                disp_writeCh(' ');
        if(maxTempIn >= 10){
                disp_writeCh(writeDigit(maxTensIn));
        } else
                disp_writeCh(' ');
        if(maxTempIn >= 0){
                disp_writeCh(writeDigit(maxSinglesIn));
        }
        if(maxTempIn < 0){
                maxSinglesIn = maxSinglesIn-2*maxSinglesIn;
                disp_writeCh(writeDigit(maxSinglesIn));
        }
        if(maxTempIn < -10){
                maxTensIn = maxTensIn-2*maxTensIn;
                disp_writeCh(writeDigit(maxTensIn));
        }

        disp_writeCh(' ');
        disp_writeCh('/');

        int minHundredsIn = minTempIn/100;
        int minTensIn = (minTempIn-minHundredsIn*100)/10;
        int minSinglesIn = minTempIn-minHundredsIn*100-minTensIn*10;

        if(minTempIn < 0){
                disp_writeCh('-');
        } else
                disp_writeCh(' ');
        if(minTempIn >= 10){
                disp_writeCh(writeDigit(minTensIn));
        } else
                disp_writeCh(' ');
        if(minTempIn >= 0){
                disp_writeCh(writeDigit(minSinglesIn));
        }
```

```
        if(minTempIn < 0){
                minSinglesIn = minSinglesIn-2*minSinglesIn;
                disp_writeCh(writeDigit(minSinglesIn));
        }
        if(minTempIn < -10){
                minTensIn = minTensIn-2*minTensIn;
                disp_writeCh(writeDigit(minTensIn));
        }

        write_cmd(0xD4);
        disp_writeCh('A');
        disp_writeCh('=');
        disp_writeCh('M');
        disp_writeCh('a');
        disp_writeCh('i');
        disp_writeCh('n');
        disp_writeCh(' ');
        disp_writeCh('F');
        disp_writeCh('=');
        disp_writeCh('R');
        disp_writeCh('e');
        disp_writeCh('s');
        disp_writeCh('e');
        disp_writeCh('t');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        disp_writeCh(' ');
        return;
}

char checkButton(char buttonPressed){
        char result;
        switch(buttonPressed){
                case 0x30:
                        result = 0;
                        break;
                case 0x20:
                        result = 1;
                        break;
                case 0x10:
                        result = 2;
                        break;
                case 0x00:
                        result = 3;
                        break;
                case 0x70:
                        result = 4;
                        break;
                case 0x60:
                        result = 5;
                        break;
                case 0x50:
                        result = 6;
                        break;
                case 0x40:
                        result = 7;
                        break;
                case 0xB0:
                        result = 8;
```

```
                        break;
                case 0xA0:
                        result = 9;
                        break;
                case 0x90:
                        result = 'A';
                        break;
                case 0x80:
                        result = 'B';
                        break;
                case 0xF0:
                        result = 'C';
                        break;
                case 0xE0:
                        result = 'D';
                        break;
                case 0xD0:
                        result = 'E';
                        break;
                case 0xC0:
                        result = 'F';
                        break;
                default:
                        result = -1;
                        break;
        }
        return result;
}


void catchMaxAlarmDigit(char digit){
        if(digit != 'F'){
                int tempInt = (int)digit;
                maxAlarm = maxAlarm * 10 + tempInt;
                if(digit == 1){
                        printDigits[counter] = '1';
                        counter++;
                }
                if(digit == 2){
                        printDigits[counter] = '2';
                        counter++;
                }
                if(digit == 3){
                        printDigits[counter] = '3';
                        counter++;
                }
                if(digit == 4){
                        printDigits[counter] = '4';
                        counter++;
                }
                if(digit == 5){
                        printDigits[counter] = '5';
                        counter++;
                }
                if(digit == 6){
                        printDigits[counter] = '6';
                        counter++;
                }
                if(digit == 7){
                        printDigits[counter] = '7';
                        counter++;
```

```
                }
                if(digit == 8){
                        printDigits[counter] = '9';
                        counter++;
                }
                if(digit == 9){
                        printDigits[counter] = '9';
                        counter++;
                }
                if(digit == 0){
                        printDigits[counter] = '0';
                        counter++;
                }
                disp_clear();
                disp_wait();
                disp_clear();
                enter_maxAlarm();
        }

}

void catchMinAlarmDigit(char digit){
        if(digit != 'F'){
                int tempInt = (int)digit;
                minAlarm = minAlarm * 10 + tempInt;
                if(digit == 1){
                        printDigits[counter] = '1';
                        counter++;
                }
                if(digit == 2){
                        printDigits[counter] = '2';
                        counter++;
                }
                if(digit == 3){
                        printDigits[counter] = '3';
                        counter++;
                }
                if(digit == 4){
                        printDigits[counter] = '4';
                        counter++;
                }
                if(digit == 5){
                        printDigits[counter] = '5';
                        counter++;
                }
                if(digit == 6){
                        printDigits[counter] = '6';
                        counter++;
                }
                if(digit == 7){
                        printDigits[counter] = '7';
                        counter++;
                }
                if(digit == 8){
                        printDigits[counter] = '9';
                        counter++;
                }
                if(digit == 9){
                        printDigits[counter] = '9';
                        counter++;
                }
```

```
                if(digit == 0){
                        printDigits[counter] = '0';
                        counter++;
                }
                disp_clear();
                disp_wait();
                disp_clear();
                enter_minAlarm();
        }

}

void main(void)
{
        startUp();

    while(1)
        {
                if(modeSet == 0){
                        //Main menu. Do nothing.
                }
                if(modeSet == 1){
                        //Enter upper alarm. Do nothing.
                }
                if(modeSet == 2){
                        //Enter lower alarm. Do nothing.
                }
                if(modeSet == 3){
                        //Show min/max temperature. Do nothing.
                }

        }
        return;
}



ISR(INT1_vect) //interrupt when a key is pressed on the keyboard.
{

        PORTA = 0x00 & 0x08; //sends Output enable from the AVR to the
controller
        buttonPressed = PINA & 0xF0; //masks the four least significant
digits of the A-port

        char pressedButton = checkButton(buttonPressed);

        if(pressedButton == 'A'){
                disp_clear();
                disp_wait();
                disp_clear();
                disp_tempOut();
                modeSet = 0;
                return;
        }
        if(pressedButton == 'B'){
                modeSet = 3;
                disp_clear();
                disp_wait();
                disp_clear();
                show_stats();
```

```
                return;
        }
        if(pressedButton == 'C'){
                resetDigits();
                maxAlarm = 0;
                modeSet = 1;
                disp_clear();
                disp_wait();
                disp_clear();
                enter_maxAlarm();
                return;
        }
        if(pressedButton == 'D'){
                resetDigits();
                minAlarm = 0;
                modeSet = 2;
                disp_clear();
                disp_wait();
                disp_clear();
                enter_minAlarm();
                return;
        }
        if(pressedButton >= 0 && pressedButton <= 9){
                if(modeSet == 1){
                        catchMaxAlarmDigit(pressedButton);
                }
                if(modeSet == 2){
                        catchMinAlarmDigit(pressedButton);
                }
                if(alarmActive == 1){
                        resetDigits();
                        disp_tempOut();
                }
                return;
        }
        if(pressedButton == 'E'){
                if(modeSet == 1){
                        if(maxAlarmNeg == 0){
                                maxAlarmNeg = 1;
                        } else if(maxAlarmNeg == 1){
                                maxAlarmNeg = 0;
                        }
                disp_clear();
                disp_wait();
                disp_clear();
                enter_maxAlarm();
                }
                if(modeSet == 2){
                        if(minAlarmNeg == 0){
                                minAlarmNeg = 1;
                        } else if(minAlarmNeg == 1){
                                minAlarmNeg = 0;
                        }
                disp_clear();
                disp_wait();
                disp_clear();
                enter_minAlarm();
                }
        }
        if(pressedButton == 'F'){
                if(modeSet == 1){
```

```
                        maxAlarmArmed = 1;
                        alarmActive = 1;
                        if(maxAlarmNeg == 1){
                                int temp = maxAlarm;
                                maxAlarm = maxAlarm-2*temp;
                        }
                        modeSet = 0;
                        disp_clear();
                        disp_wait();
                        disp_clear();
                        disp_tempOut();
                        return;
                }
                if(modeSet == 2){
                        minAlarmArmed = 1;
                        alarmActive = 1;
                        if(minAlarmNeg == 1){
                                int temp = minAlarm;
                        minAlarm = minAlarm-2*temp;
                        }
                        modeSet = 0;
                        disp_clear();
                        disp_wait();
                        disp_clear();
                        disp_tempOut();
                        return;
                }
                if(modeSet == 3){
                        maxTempOut = -100;
                        minTempOut = 100;
                        maxTempIn = -100;
                        minTempIn = 100;
                        modeSet = 3;
                        disp_clear();
                        disp_wait();
                        disp_clear();
                        show_stats();
                        return;
                }
        return;
        }
}

ISR(ADC_vect)
{
        if(ADMUX == 0b00100000){ //reads the temperature outside
                int tempOutLow = ADCL >> 6;
                int tempOutHigh = ADCH << 2;

                tempOut = tempOutHigh + tempOutLow;
                realTempOut = tempOut*500/1024;
                hundredsOut = realTempOut/100;
                tensOut = (realTempOut-hundredsOut*100)/10;
                singlesOut = realTempOut-hundredsOut*100-tensOut*10;

                if(modeSet == 0){
                        disp_wait();
                        write_cmd(0xB6);

                        if(realTempOut < 0){
                                disp_writeCh('-');
```

```
                        } else
                                disp_writeCh(' ');

                        disp_writeCh(' ');

                        if(realTempOut >= 10){
                                disp_writeCh(writeDigit(tensOut));
                        } else
                                disp_writeCh(' ');
                        if(realTempOut >= 0){
                                disp_writeCh(writeDigit(singlesOut));
                        }
                        if(realTempOut < 0){
                                singlesOut = singlesOut-2*singlesOut;
                                disp_writeCh(writeDigit(singlesOut));
                        }
                        if(realTempOut < -10){
                                tensOut=tensOut-2*tensOut;
                                disp_writeCh(writeDigit(tensOut));
                        }
                }
                if(realTempOut > maxTempOut){
                        maxTempOut = realTempOut;
                }

                if(realTempOut < minTempOut){
                        minTempOut = realTempOut;
                }

                if(minAlarmArmed == 1){
                        if(realTempOut <= minAlarm){
                                PORTC = 0x00 & 0x01; //turns ON the
backlight
                        } else if(realTempOut > minAlarm){
                                PORTC = 0x01 & 0x01; //turns OFF the
backlight
                        }
                } else
                        PORTC = 0x01 & 0x01; //turns OFF the backlight

                if(maxAlarmArmed == 1){
                        if(realTempOut >= maxAlarm){
                                PORTC = 0x00 & 0x01; //turns ON the
backlight
                        } else if(realTempOut < maxAlarm){
                                PORTC = 0x01 & 0x01; //turns OFF the
backlight
                        }
                } else
                        PORTC = 0x01 & 0x01; //turns OFF the backlight

                ADMUX = 0b00100001;
                ADCSRA = 0xDC;
                return;
        }
        else if(ADMUX == 0b00100001){ //reads the temperature inside
                int tempInLow = ADCL >> 6;
                int tempInHigh = ADCH << 2;

                tempIn = tempInHigh + tempInLow;
                realTempIn = tempIn*500/1024;
```

```
                hundredsIn = realTempIn/100;
                tensIn = (realTempIn-hundredsIn*100)/10;
                singlesIn = realTempIn-hundredsIn*100-tensIn*10;

                if(modeSet == 0){
                        disp_wait();
                        write_cmd(0x8E);

                        if(realTempIn < 0){
                                disp_writeCh('-');
                        } else
                                disp_writeCh(' ');

                        disp_writeCh(' ');

                        if(realTempIn >= 10){
                                disp_writeCh(writeDigit(tensIn));
                        } else
                                disp_writeCh(' ');
                        if(realTempIn >= 0){
                                disp_writeCh(writeDigit(singlesIn));
                        }
                        if(realTempIn < 0){
                                singlesIn = singlesIn-2*singlesIn;
                                disp_writeCh(writeDigit(singlesIn));
                        }
                        if(realTempIn < -10){
                                tensIn=tensIn-2*tensIn;
                                disp_writeCh(writeDigit(tensIn));
                        }

                }

                if(realTempIn > maxTempIn){
                        maxTempIn = realTempIn;
                }

                if(realTempIn < minTempIn){
                        minTempIn = realTempIn;
                }

                ADMUX = 0b00100000;
                ADCSRA = 0xDC;
                return;
        }
}
```