```
/*
This sketch runs a Standalone Datalooger for two general purpose inputs
and two Thermo Couple inputs. Please be aware, you MUST use K-Type Thermocouples.
specification for input can be found at: http://www.maximintegrated.com/datasheet/index.mvp/id/7273
(max31855 Cold-Junction Compensated Thermocouple-to-Digital Converter)
The breakoutboard for max31855 and specs are found at http://www.adafruit.com/products/269
The logfiles are written to an SD Card and and a RealTimeClock is setup.
For schematic, libraries etc please visit http://www.ladyada.net/make/logshield/design.html


PinLayout:
A0
A1
A2 = GPIP    / General Input 0-5V Channel C
A3 = GPIP    / General Input 0-5V Channel D
A4 = I2C bus / RTC
A5 = I2C bus / RTC

DIO 0  =
DIO 1  =
DIO 2  = LED   / Green Indicator LED
DIO 3  = LED   / Red Indicator LED
DIO 4  = CLK    / Thermocouple Channel \
DIO 5  = CS    / Thermocouple Channel   |> A
DIO 6  = DO   / Thermocouple Channel   /
DIO 7  = CLK    / Thermocouple Channel \
DIO 8  = CS    / Thermocouple Channel   |> B
DIO 9  = DO   / Thermocouple Channel   /
DIO 10 = CS    / sd card
DIO 11 = MOSI  / sd card
DIO 12 = MISO  / sd card
DIO 13 = SCK   / sd card
*/

#include <Wire.h>             // I2C communication used for RealTimeClock
#include "RTClib.h"           // Realtime Clock
#include <SD.h>               // ReadWrite to SD Card
#include "Adafruit_MAX31855.h" // readout data from MAX31855

//LED
const int m_led_r = 2;              // status led
const int m_led_g = 3;              // status led
// SD Card and logfile
const int m_sdcard_CS = 10;         // chip selection pin for SD card
int m_active_logfile = 0;           // check if a new logfile is needed
File m_logfile;                     // file handle for logfile writing to sd card
char m_filename[13] = "19700101.csv"; // set a default value in case the rtc is out of battery
DateTime m_t_stamp;                 // object to store snapshot from RTC
String m_time, m_date, yyyy,mm,dd;   // time date snapshot
String m_data;                      // assembly of outputlilne to send to sdcard after reading all sensors
String m_header;                    // set first line for new logfiles
// Channel A
const int m_ch_A_CLK = 4;
const int m_ch_A_CS = 5;
const int m_ch_A_DO = 6;
double m_ch_A, m_ch_A_internal;   // store the temp from max 31855
// Channel B
const int m_ch_B_CLK = 7;
const int m_ch_B_CS = 8;
const int m_ch_B_DO = 9;
double m_ch_B, m_ch_B_internal;   // store the temp from max 31855
//Channel C
const int ch_C = A2;
int m_ch_C;                     // store the Analog Reading
//Channel D
const int ch_D = A3;
int m_ch_D;                     // store the Analog Reading

//General Settings
int m_timer;                    // main delay of main loop for readings
bool m_errorcheck = false;       // start with no error..


//create two objects for thermocouple breakoutboard
Adafruit_MAX31855 ch_A(m_ch_A_CLK,m_ch_A_CS,m_ch_A_DO);
Adafruit_MAX31855 ch_B(m_ch_B_CLK,m_ch_B_CS,m_ch_B_DO);

// setup RealTimeClock
RTC_DS1307 RTC;               // setup RealTimeClock

void setup()
{
```

```
  pinMode(m_led_r, OUTPUT);
  digitalWrite(m_led_r, LOW);
  pinMode(m_led_g, OUTPUT);
  digitalWrite(m_led_g, LOW);
  pinMode(m_ch_A_CS, OUTPUT);
  pinMode(m_ch_B_CS, OUTPUT);
  pinMode(m_sdcard_CS, OUTPUT);

  /*
  set default timer to one second (1000 ms), since the RTC provides seconds as smallest unit
  however the reading of the sensors can be done faster and hence this timer may be
  set lower to capture more readings per seconds. The logfile entries will have the
  same time stamp
  */
  m_timer = 1000;

  // set the header file for new logfiles
  m_header = "Date,Time,CHA int, CHA, CHB int, CHB, CHC, CHD" ;

  Serial.begin(9600);

  // start communication for RTC (using default values of Analog 4 and Analog 5
  Wire.begin();

  // start the lib for RTC so we can get to the timestamp
  RTC.begin();

  if (!SD.begin(m_sdcard_CS))
  {
    blink(m_led_r, 4, 200);
    m_errorcheck = true;
  }else{
    m_errorcheck = false;
  }
  errorCheck();
}

void loop()
{

  errorCheck();
    if(m_errorcheck)
    {
      delay(m_timer*2);
    } else {
      getTimeStamp();   // set m_date, m_time
      // get temperature reading for channel A
      m_ch_A_internal = ch_A.readInternal();
      checkTemp(m_ch_A_internal);
      m_ch_A = ch_A.readCelsius();
      checkTemp(m_ch_A);

      // get temperature reading for channel B
      m_ch_B_internal = ch_B.readInternal();
      checkTemp(m_ch_B_internal);
      m_ch_B = ch_B.readCelsius();
      checkTemp(m_ch_B);

      // get the GPIO input for AnalogSignal 2&3
      m_ch_C = analogRead(ch_C);
      m_data  = m_data + ',' + m_ch_C;
      m_ch_D = analogRead(ch_D);
      m_data  = m_data + ',' + m_ch_D;

      //write data to sd card
      setLogfile();     // check if the logfilename needs changing
      m_logfile = SD.open(m_filename, FILE_WRITE);
      if (m_logfile)
      {
        m_logfile.println(m_data);
        m_logfile.close();
        blink(m_led_g,1,200);
      } else {
        blink(m_led_r,1,200);
      }
    }
    debug();
    delay(m_timer);
  }


void errorCheck()
```

```
{
  // check if the realtime clock is running
  if (! RTC.isrunning() )
  {
    blink(m_led_r, 2, 200);
    m_errorcheck = false; // set to true.....
  } else
  {
    m_errorcheck = false;
  }
}

void debug()
{
    Serial.println(m_filename);
    Serial.println(m_data);
    Serial.println("-------------------------");
}

void checkTemp(float tVal)
{
  if (isnan(tVal))
  {
    m_data = m_data + ",NaN";
  }else{
    m_data = m_data + "," +floatToString(tVal,2);
  }
}


void getTimeStamp()
{
  m_t_stamp = RTC.now();
  yyyy = String(m_t_stamp.year());
  mm = String(m_t_stamp.month());
  if (mm.length() < 2){mm = '0' + mm;}
  dd = String(m_t_stamp.day());
  if (dd.length() < 2){dd = '0' + dd;}
  m_date = yyyy + '/' + mm + '/'+ dd;

  // start of assembly of data line
  m_data = m_date;

  m_time = String(m_t_stamp.hour()) + ':';
  m_time = m_time + String(m_t_stamp.minute()) + ':';
  m_time = m_time + String(m_t_stamp.second());
  m_data = m_data + "," + m_time;
}

void blink(int led, int freq, int velocity)
{
  for (int i=0; i<freq; i++)
  {
    digitalWrite(led, HIGH);
    delay(velocity);
    digitalWrite(led, LOW);
    delay(velocity);
  }
}

void setLogfile()
{
  String filename;
  m_t_stamp = RTC.now();
  yyyy = String(m_t_stamp.year());
  mm = String(m_t_stamp.month());
  if (mm.length() < 2){mm = '0' + mm;}
  dd = String(m_t_stamp.day());
  if (dd.length() < 2){dd = '0' + dd;}
  filename = yyyy+mm+dd + ".csv";
  // convert to char arrary for sending to sd.open cmd
  filename.toCharArray(m_filename, 13);

 // in case we need a new logfile, write the header first.
 if (! SD.exists(m_filename))
 {
   m_logfile = SD.open(m_filename, FILE_WRITE);
   if (m_logfile)
   {
     m_logfile.println(m_header);
     m_logfile.close();
   } else {
```

```
247       blink(m_led_r,2,200);
248     }
249   }
250  }
251
252  //convert double numbers to a string
253  String floatToString(double number, uint8_t digits)
254  {
255    String resultString = "";
256    // Handle negative numbers
257    if (number < 0.0)
258    {
259      resultString += "-";
260      number = -number;
261    }
262
263    // Round correctly so that print(1.999, 2) prints as "2.00"
264    double rounding = 0.5;
265    for (uint8_t i=0; i<digits; ++i)
266      rounding /= 10.0;
267
268    number += rounding;
269
270    // Extract the integer part of the number and print it
271    unsigned long int_part = (unsigned long)number;
272    double remainder = number - (double)int_part;
273    resultString += int_part;
274
275    // Print the decimal point, but only if there are digits beyond
276    if (digits > 0)
277      resultString += ".";
278
279    // Extract digits from the remainder one at a time
280    while (digits-- > 0)
281    {
282      remainder *= 10.0;
283      int toPrint = int(remainder);
284      resultString += toPrint;
285      remainder -= toPrint;
286    }
287    return resultString;
288  }
289
```