```
/*
 * .h file for SHT35D
 */

#ifndef SHT35D
#define SHT35D
#define MAX_READ_COUNT 5
#define MAX_ERROR_COUNT 5
#define ADDR_SHT 0x45

#include <Arduino.h>

//List of Commands for SHT35D Sensor:
typedef enum {
    SHT3XD_CMD_READ_SERIAL_NUMBER = 0x3780,

    SHT3XD_CMD_READ_STATUS = 0xF32D,
    SHT3XD_CMD_CLEAR_STATUS = 0x3041,

    SHT3XD_CMD_HEATER_ENABLE = 0x306D,
    SHT3XD_CMD_HEATER_DISABLE = 0x3066,

    SHT3XD_CMD_SOFT_RESET = 0x30A2,

    SHT3XD_CMD_CLOCK_STRETCH_H = 0x2C06,
    SHT3XD_CMD_CLOCK_STRETCH_M = 0x2C0D,
    SHT3XD_CMD_CLOCK_STRETCH_L = 0x2C10,

    SHT3XD_CMD_POLLING_H = 0x2400,
    SHT3XD_CMD_POLLING_M = 0x240B,
    SHT3XD_CMD_POLLING_L = 0x2416,

    SHT3XD_CMD_ART = 0x2B32,

    SHT3XD_CMD_PERIODIC_HALF_H = 0x2032,
    SHT3XD_CMD_PERIODIC_HALF_M = 0x2024,
    SHT3XD_CMD_PERIODIC_HALF_L = 0x202F,
    SHT3XD_CMD_PERIODIC_1_H = 0x2130,
    SHT3XD_CMD_PERIODIC_1_M = 0x2126,
    SHT3XD_CMD_PERIODIC_1_L = 0x212D,
    SHT3XD_CMD_PERIODIC_2_H = 0x2236,
    SHT3XD_CMD_PERIODIC_2_M = 0x2220,
    SHT3XD_CMD_PERIODIC_2_L = 0x222B,
    SHT3XD_CMD_PERIODIC_4_H = 0x2334,
    SHT3XD_CMD_PERIODIC_4_M = 0x2322,
    SHT3XD_CMD_PERIODIC_4_L = 0x2329,
    SHT3XD_CMD_PERIODIC_10_H = 0x2737,
    SHT3XD_CMD_PERIODIC_10_M = 0x2721,
    SHT3XD_CMD_PERIODIC_10_L = 0x272A,

    SHT3XD_CMD_FETCH_DATA = 0xE000,
    SHT3XD_CMD_STOP_PERIODIC = 0x3093,

    SHT3XD_CMD_READ_ALR_LIMIT_LS = 0xE102,
    SHT3XD_CMD_READ_ALR_LIMIT_LC = 0xE109,
    SHT3XD_CMD_READ_ALR_LIMIT_HS = 0xE11F,
    SHT3XD_CMD_READ_ALR_LIMIT_HC = 0xE114,
    SHT3XD_CMD_WRITE_ALR_LIMIT_HS = 0x611D,
    SHT3XD_CMD_WRITE_ALR_LIMIT_HC = 0x6116,
    SHT3XD_CMD_WRITE_ALR_LIMIT_LC = 0x610B,
    SHT3XD_CMD_WRITE_ALR_LIMIT_LS = 0x6100,

    SHT3XD_CMD_NO_SLEEP = 0x303E,
} SHT31D_Commands;

// List of repeatability options for SHT35D:
typedef enum {
  SHT3XD_REPEATABILITY_HIGH,
  SHT3XD_REPEATABILITY_MEDIUM,
```

```
  SHT3XD_REPEATABILITY_LOW,
} SHT31D_Repeatability;

// List of modes:
typedef enum {
  SHT3XD_MODE_CLOCK_STRETCH,
  SHT3XD_MODE_POLLING,
} SHT31D_Mode;

// List of frequency choices
typedef enum {
  SHT3XD_FREQUENCY_HZ5,
  SHT3XD_FREQUENCY_1HZ,
  SHT3XD_FREQUENCY_2HZ,
  SHT3XD_FREQUENCY_4HZ,
  SHT3XD_FREQUENCY_10HZ
} SHT31D_Frequency;

// List of errors:
typedef enum {
  SHT3XD_NO_ERROR = 0,

  SHT3XD_CRC_ERROR = -101,
  SHT3XD_TIMEOUT_ERROR = -102,

  SHT3XD_PARAM_WRONG_MODE = -501,
  SHT3XD_PARAM_WRONG_REPEATABILITY = -502,
  SHT3XD_PARAM_WRONG_FREQUENCY = -503,
  SHT3XD_PARAM_WRONG_ALERT = -504,

  // Wire I2C translated error codes

  SHT3XD_WIRE_I2C_DATA_TOO_LOG = -10,
  SHT3XD_WIRE_I2C_RECEIVED_NACK_ON_ADDRESS = -20,
  SHT3XD_WIRE_I2C_RECEIVED_NACK_ON_DATA = -30,
  SHT3XD_WIRE_I2C_UNKNOW_ERROR = -40
} SHT31D_ErrorCode;

// List of statuses:
typedef union {
    uint16_t rawData;
    struct {
        uint8_t WriteDataChecksumStatus : 1;
        uint8_t CommandStatus : 1;
        uint8_t Reserved0 : 2;
        uint8_t SystemResetDetected : 1;
        uint8_t Reserved1 : 5;
        uint8_t T_TrackingAlert : 1;
        uint8_t RH_TrackingAlert : 1;
        uint8_t Reserved2 : 1;
        uint8_t HeaterStatus : 1;
        uint8_t Reserved3 : 1;
        uint8_t AlertPending : 1;
    };
} SHT31D_RegisterStatus;

struct SHT31D {
    /*
     * Structure for SHT31D
     * t - temperature
     * rh - relative humidity
     * error - error of type SHT31D_ErrorCode
     */
    float t;
    float rh;
    SHT31D_ErrorCode error;
};

class ClosedCube_SHT31D {
```

```
139  /*
140   * Class definition for ClosedCube_SHT31D
141   */
142  public:
143      ClosedCube_SHT31D();
144
145      bool start_sht(void);
146      bool run_sht(void);
147      float get_t_ave(void);
148      float get_rh_ave(void);
149
150
151      SHT31D_ErrorCode begin(uint8_t address);
152      SHT31D_ErrorCode clearAll();
153      SHT31D_RegisterStatus readStatusRegister();
154
155      SHT31D_ErrorCode heaterEnable();
156      SHT31D_ErrorCode heaterDisable();
157
158      SHT31D_ErrorCode softReset();
159      SHT31D_ErrorCode generalCallReset();
160
161      SHT31D_ErrorCode artEnable();
162
163      uint32_t readSerialNumber();
164
165      SHT31D printResult(String text, SHT31D result);
166      SHT31D readTempAndHumidity(SHT31D_Repeatability repeatability, SHT31D_Mode mode,
         uint8_t timeout);
167      SHT31D readTempAndHumidityClockStretch(SHT31D_Repeatability repeatability);
168      SHT31D readTempAndHumidityPolling(SHT31D_Repeatability repeatability, uint8_t
         timeout);
169
170      SHT31D_ErrorCode periodicStart(SHT31D_Repeatability repeatability, SHT31D_Frequency
         frequency);
171      SHT31D periodicFetchData();
172      SHT31D_ErrorCode periodicStop();
173
174      SHT31D_ErrorCode writeAlertHigh(float temperatureSet, float temperatureClear, float
         humiditySet, float humidityClear);
175      SHT31D readAlertHighSet();
176      SHT31D readAlertHighClear();
177
178      SHT31D_ErrorCode writeAlertLow(float temperatureClear, float temperatureSet, float
         humidityClear, float humiditySet);
179      SHT31D readAlertLowSet();
180      SHT31D readAlertLowClear();
181
182  private:
183      float t_buf[MAX_READ_COUNT];
184      float rh_buf[MAX_READ_COUNT];
185      bool is_average_taken;
186      int read_count;
187      int error_count;
188      float t_average;
189      float rh_average;
190
191      SHT31D save_to_buffer(SHT31D result);
192      SHT31D read_sht(void);
193      void calculate_average(void);
194
195      uint8_t _address;
196      SHT31D_RegisterStatus _status;
197
198      SHT31D_ErrorCode writeCommand(SHT31D_Commands command);
199      SHT31D_ErrorCode writeAlertData(SHT31D_Commands command, float temperature, float
         humidity);
200
201      uint8_t checkCrc(uint8_t data[], uint8_t checksum);
```

```
202       uint8_t calculateCrc(uint8_t data[]);
203
204       float calculateHumidity(uint16_t rawValue);
205       float calculateTemperature(uint16_t rawValue);
206
207       uint16_t calculateRawHumidity(float value);
208       uint16_t calculateRaWTemperature(float value);
209
210       SHT31D readTemperatureAndHumidity();
211       SHT31D readAlertData(SHT31D_Commands command);
212       SHT31D_ErrorCode read(uint16_t* data, uint8_t numOfPair);
213
214       SHT31D returnError(SHT31D_ErrorCode command);
215   };
216
217   #endif
```