 Friedrich-Ebert-Schule Esslingen FES	<b>Mikrocontroller-Labor</b>	Name: Rahm Datum: 13.11.2017 6_3_2_Datenlogger_Bibliotheken.docx
	Datenlogger-Bibliotheken	6.3.2.1


Datei: datenlogger.h (Auszug)

```
// Globale Variablen als Übergabe-Parameter für das Datenlogger-Programm (main)
volatile uint8_t jahr,monat,tag,stunde,minute,sekunde;
volatile int16_t temperatur;
volatile uint16_t aktueller_Datensatz;

#define LM75_ADDR_R 0x91 // i2c-Adressen für Temperatursensor
#define LM75_ADDR_W 0x90

#define _REC_SIZE_ 8 // Bytes je Datensatz
#define RECORD_MSB 0x08 // Speicheradressen für aktuellen Datensatz
#define RECORD_LSB 0x09 // wird im RTC-Ram batteriegepuffert
#define SAMPLETIME_MSB 0x0A // Speicheradressen für Sample-Time
#define SAMPLETIME_LSB 0x0B // wird im RTC-Ram batteriegepuffert
#define MAX_RECORD 4095 // Maximale Anzahl an Datensätzen je EEPROM
// 24LC512: 32kByte / _REC_SIZE_ = 4095

//Führt einen Speichertest von EEPROM#1 durch
void eeprom_speichertest (void);
// Eingabe der aktuellen rtc-Zeit über ein Eingabeprompt
// Wartet nach Aufruf 5s auf Eingabe.
void rs232_set_time (void);
// Speichert oder holt den aktuellen Datensatz aus dem EEPROM
// Parameter: i2c_address = EEPROM_1, EEPROM_2, EEPROM_3
// record = Datensatznummer (= Speicheradresse im EEPROM)
void eeprom_get_record (uint8_t i2c_address, uint16_t record);
void eeprom_set_record (uint8_t i2c_address, uint16_t record);
// Gibt den angegebenen Datensatz über die serielle Schnittstelle aus
// Format: DD.MM.YY hh:mm:ss;(-)xx,y
// Tag.Monat.Jahr Stunde:Minute:Sekunde;Temperatur,Dezimale(0,5)
// Bsp: 02.11.17 20:47:34;21,5
void rs232_print_record (uint8_t i2c_address, uint16_t record);
// Speichert oder holt die aktuelle Uhrzeit im Uhrenbaustein
void rtc_set (void);
void rtc_get (void);
// Schreibt oder liest die aktuelle Datensatznummer spannungsausfallsicher
// im Uhren-RAM (Adresse: RECORD_MSB, RECORD_LSB)
// record_number: 0x0000 ... EEPROM_END_ADDRESS (0x7fff)
uint16_t read_current_recordnumber_from_rtc (void);
void write_current_recordnumber_to_rtc (uint16_t record_number);
// Schreibt oder liest die aktuelle Samplezeit spannungsausfallsicher
// im Uhren-RAM (Adresse: SAMPLETIME_MSB, SAMPLETIME_LSB)
// sample_time: 0 ... 65525 Sekunden
uint16_t read_current_sampletime_from_rtc (void);
void write_current_sampletime_to_rtc (uint16_t sample_time);
// Gibt alle gespeicherten Datensätze über RS232 aus
void serial_print_all_records(void);
// Liest den aktuellen Temperaturwert vom LM75-Temperatursensor
// Rückgabe als 16-Bit Wert (Linksbündig) mit Vorzeichen.
int16_t lm75_read (void);
// Zeigt den Temperaturwert auf dem LC-Display an. (z.B.: 24,5°C)
void lcd_print_temperatur (int16_t degree);
// Gibt den Temperaturwert über RS232 aus.
// mode = 0: Mit Kurvenname T1, mit \n (z.B.: T1=23,5)
// mode = 1: ohne Name, mit \r (z.B.: 23,5)
// mode = 2: ohne Name, mit °C mit \r (z.B.: 23,5°C)
void rs232_print_temperatur(int16_t degree, uint8_t mode);
//Eingabe einer 5-stelligen Ganzzahl (0...99999) am Terminal.
//Wird als 32-Bit Wert zurückgegeben.
uint32_t rs232_get_sampletime(void);
```

 Friedrich-Ebert-Schule Esslingen FES	<b>Mikrocontroller-Labor</b>		Name: Rahm Datum: 13.11.2017 6_3_2_Datenlogger_Bibliotheken.docx
	Datenlogger-Bibliotheken		6.3.2.2

## Bibliothek: eeprom.h (Auszug)

```
// i2c-Adressen (Definiert ist jeweils nur die Schreibadresse (R/W = 0)
#define EEPROM_1          0xa0
#define EEPROM_2          0xa2
#define EEPROM_3          0xa4
#define EEPROM_END_ADDRESS 0x7fff // 32kByte 24LC512

//**** EEPROM-Funktionsprototypen
// Initialisierung
void eeprom_init (void);
// Schreibt das Byte "value" an die angegebene Speicheradresse
// address: 0x0000...EEPROM_END_ADDRESS
// i2c_address: EEPROM_1, EEPROM_2, EEPROM_3
void eeprom_write (uint8_t i2c_address, uint16_t address, uint8_t value);
// dito. ein Byte lesen
uint8_t eeprom_read (uint8_t i2c_address, uint16_t address);
// führt einen Speichertest des angegebenen EEPROM durch.
// getestet wird Adresse 0x0000 bis EEPROM_END_ADDRESS
// Rückgabewert: -1 = Fehler, 0 = OK
int8_t eeprom_memtest (uint8_t i2c_address);
```

## Bibliothek: rtc.h (Auszug)

```
// i2c-Adressen
#define RTC_ADDR_R 0xd1
#define RTC_ADDR_W 0xd0

// Adressen der Timekeeper-Register DS1307
#define _SEC_      0x00
#define _MIN_      0x01
#define _HR_       0x02
#define _DAY_      0x03
#define _DATE_     0x04
#define _MONTH_    0x05
#define _YEAR_     0x06

// Funktionsprototypen für Echtzeituhr
// Initialisierung
void rtc_init (void);
// Schreiben der Timekeeperregister (BCD-Format)
// reg: _SEC_, _MIN_, _HR_, _DAY_, _DATE_, _MONTH_, _YEAR_
// value: 00...59, 59, 23, 31, 7, 12, 99
void rtc_write (uint8_t reg, uint8_t value);
// Lesen der Timekeeper-Register (BCD-Format)
uint8_t rtc_read (uint8_t reg);
// Byteweises Schreiben ins interne RAM (Binär-Format).
// Z.B. Speichern von batteriegepufferten Werten.
// reg: 0x08....0x3F, 0x00..0x07 (=Timekeeper-Register!!!)
// value: 0x00...0xff
void rtc_lowlevel_write (uint8_t reg, uint8_t value);
// Byteweises Lesen aus dem internen RAM
uint8_t rtc_lowlevel_read (uint8_t reg);
```

Table 2. Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00-59
01h	0	10 Minutes			Minutes				Minutes	00-59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1-12 +AM/PM 00-23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01-07
04h	0	0	10 Date		Date				Date	01-31
05h	0	0	0	10 Month	Month				Month	01-12
06h	10 Year				Year				Year	00-99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h-3Fh									RAM 56 × 8	00h-FFh