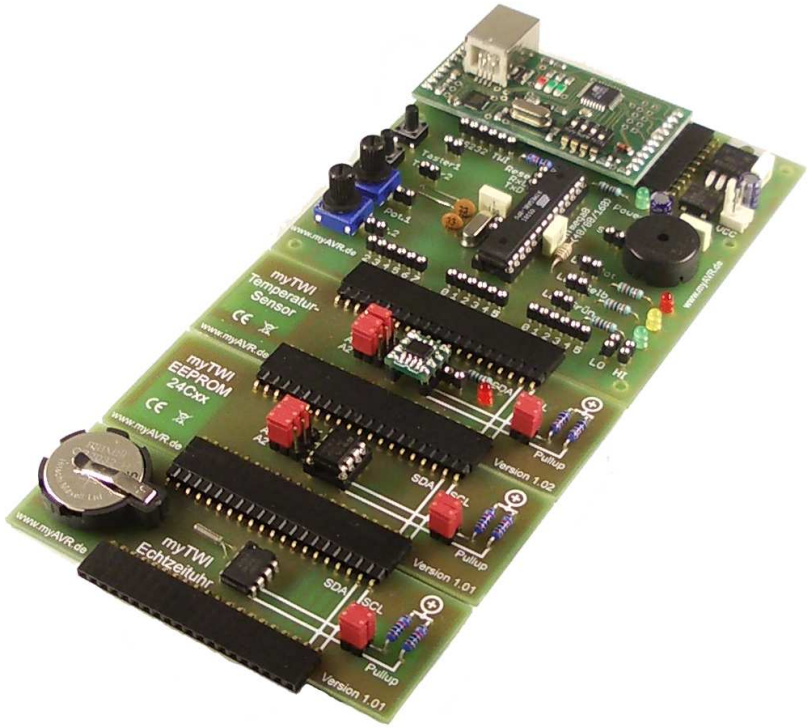
Dipl. Ing. Toralf Riedel
Dipl. Ing. Päd. Alexander Huwaldt

Projekt „myTWI“

Projektpaket für das myAVR Board MK2 USB
zum Aufbau und zur Programmierung
eines mikrocontrollerbasierenden
Temperaturerfassungssystems



Leseprobe

3. Auflage: März 2010

www.laser-co.de
www.myavr.de
service@myavr.de
Tel: ++49 (0) 3585 470 222
Fax: ++49 (0) 3585 470 233

1 Einleitung

Das Projekt *myTWI* soll anhand einer konkreten Aufgabenstellung die Struktur, Programmierung und Anwendung lokaler Bus-Systeme in Mikrocontrollerlösungen (Embedded Systems) demonstrieren. Es kann im Selbststudium oder in der Ausbildung als handlungsorientierte Projektarbeit angewendet werden.
Die Aufgabe besteht darin, ein Datenerfassungsgerät (Datenlogger) auf der Basis von Standard-I²C-Bausteinen für die Erfassung und Speicherung von Temperatur daten über einen längeren Zeitraum zu realisieren.

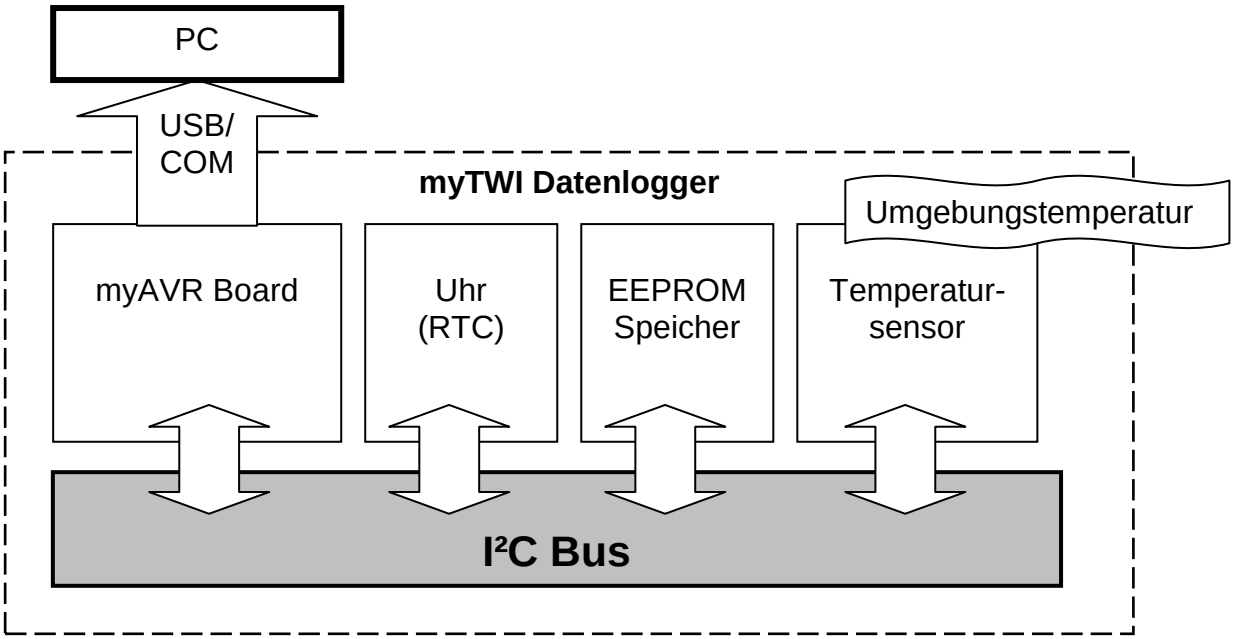


Bild : Blockschaltbild „myTWI“

Für die Realisierung des Datenerfassungsgerätes werden fertige myAVR Komponenten verwendet, die in einem Bussystem zusammenwirken.

myTWI Komponenten

- myAVR Board MK2 USB mit Atmel ATmega8
- I²C RTC Add-On (mit DS1307)
- I²C EEPROM Add-On (mit 24C02)
- I²C Temperatursensor Add-On (mit LM75)

Technische Daten

Versorgungsspannung :	9 – 12V / DC am myAVR Board MK2 USB
Betriebsspannung:	4,8 – 5,3 V
Betriebsstrom :	max. 25 mA
Temperaturbereich:	0 ℃ bis 30 ℃
Busgeschwindigkeit:	100 kHz bis 400 kHz

2 Allgemeine Grundlagen zum I²C Bus

2.1 Entwicklungsgeschichte

I²C ist ein serieller Datenbus für eingebettete Mikrocontrollersysteme und wurde von Philips Semiconductors in den 1980er Jahren entwickelt.

Das Einsatzprofil sind Geräte, die aus intelligenten Komponenten bestehen, welche über geringe Entfernungen kleine bis mittlere Datenmengen austauschen. Die erste offizielle Spezifikation stammt aus den frühen 1990er Jahren und wurde von Philips vorgesehen um die verschiedenen Komponenten in Heimelektronikgeräten zu steuern.

I²C Version 1.0 – 1992,
I²C Version 2.0 – 1998,
I²C Version 2.1 – 2000,

Beim I²C Bus handelt es sich um einen synchronen seriellen Zweidraht-Bus. Er verfügt über eine Leitung für Daten und eine Taktleitung. I²C steht für IIC = Inter IC Bus. Aus Lizenzgründen heißt der I²C Bus bei Atmel Controllern TWI (two wire interface).

Die Übertragungsrate in diesem Bus-System beträgt im Standard-Mode bis zu 100 kbit/s und beim Fast-Mode bis zu 400 kbit/s. Ab der Spezifikation 2.0 von 1998 gibt es zusätzliche einen High-Speed-Mode mit 3,4 MBit/s. Die Bus-Länge ist letztlich von der Buskapazität abhängig (max. 400 pF) und liegt typischerweise im Bereich von einigen Zentimeter bis wenige Meter. Dabei ist zu beachten, dass jedes Gerät im Bus durch seine Eingangskapazität die mögliche Leitungslänge reduziert. Die Adressbildung erfolgt über 7 oder 10 Bit Adressen und ermöglicht damit theoretisch bis zu 128 bzw. 1024 Geräte in einem Bussystem anzusprechen.

2.2 Anwendungsgebiete des I²C Bus

Die Vorteile des I²C Bus liegen im hohen Standardisierungsgrad, geringen Schaltungsaufwand und der Verfügbarkeit von sehr vielen Standardschaltkreisen für verschiedene Anwendungsfälle. Vor allem liegt die Stärke von I²C darin begründet, dass ein Mikrocontroller ein ganzes Netzwerk von intelligenten Chips mit nur zwei I/O-Pins und recht simplen Softwarebausteinen kontrollieren kann.

„The I2C-bus has become a de facto world standard that is now implemented in over 1000 different ICs and licensed to more than 50 companies.“
PHILIPS, THE I 2C-BUS SPECIFICATION, VERSION 2.1, JANUARY 2000

Obwohl der I²C Bus im Verhältnis zu anderen Bussystemen recht langsam ist, besticht er durch niedrige Kosten bei der Realisierung konkreter Systeme. Er wird häufig für Audioregler, Analog-Digital- oder Digital-Analog-Wandler, Echtzeituhren, EEPROM-Speicher, Schalter und Multiplexer, LCDs, verschiedenste Sensoren und vieles mehr eingesetzt. Selbst während des Betriebes können Komponenten zum Bus hinzugefügt oder entfernt werden, was I²C auch für im Betrieb tauschbare Geräte attraktiv macht.

Andere Bussysteme wurden in Anlehnung an den I²C entwickelt und sind zum Teil zu ihm kompatibel wie das VESA Monitordaten-Interface (Display Data Channel, kurz DDC) oder der SMBus von Intel für Mainboardkomponenten.

2.3 Typische I²C Bausteine

Als I²C-Baustein erhältlich sind RAMs, EEPROMS, Porterweiterungsbausteine, A/D- und D/A- Wandler, Uhren- und Kalenderbausteine, Anzeigetreiber (z.B. LCD), Sensorbausteine wie Temperatursensoren und natürlich Mikrocontroller.

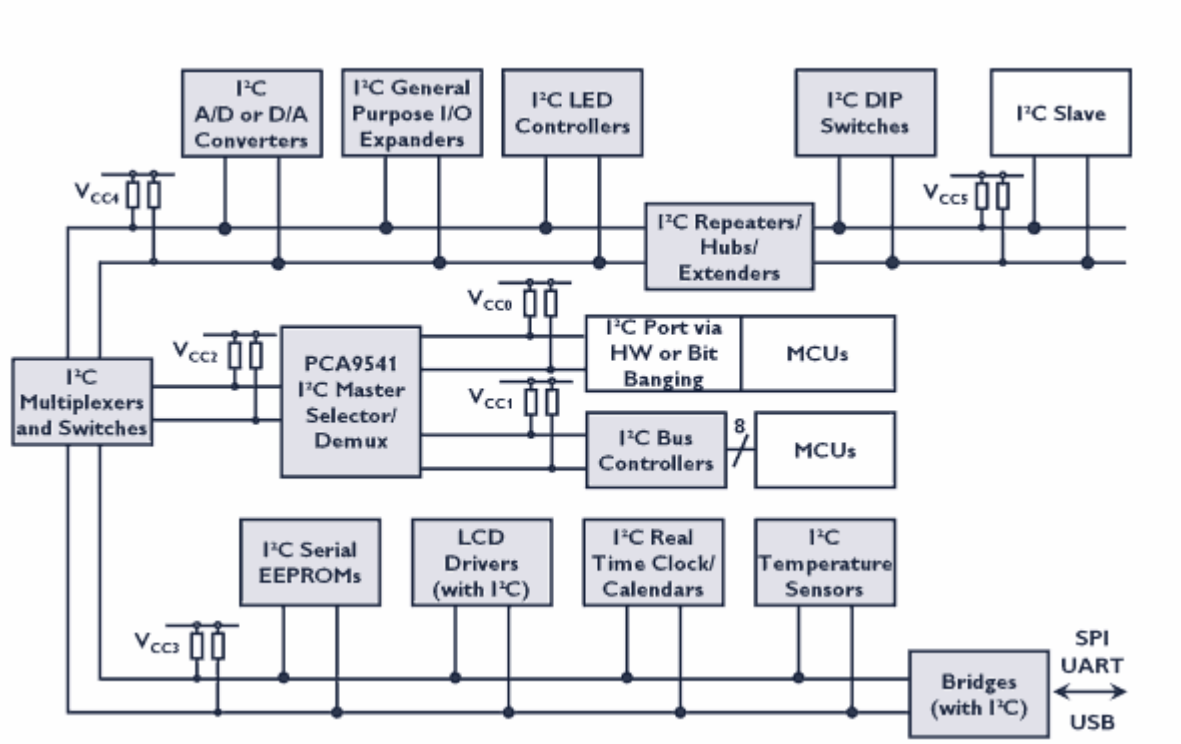


Bild: PHILIPS I²C Portfolio

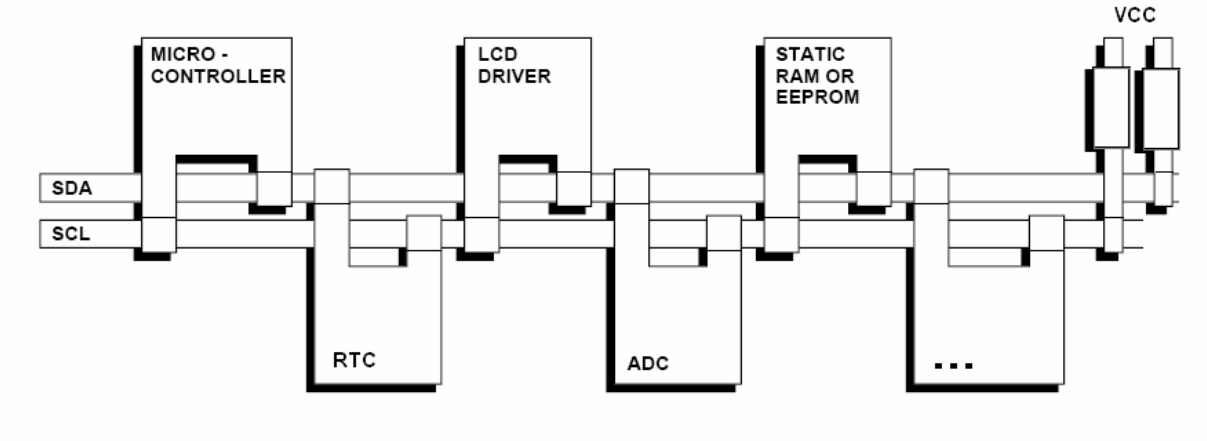
Beispiele für über 1000 verschiedene I²C Bausteine

I2C EEPROMs von Philips	**TWI EEPROMs von STMicroelectronics**
PCA8581 128 x 8-Bit	M24C01 1Kbit,
PCF8570 256 x 8-Bit	M24C02 2Kbit
PCF8594 512 x 8-Bit	M24C04 4Kbit
PCF8598 1024 x 8-Bit	M24C08 8Kbit
PCF85116 2048 x 8-Bit	M24C16 16Kbit
…	…
TWI EEPROMs von Atmel	**I2C Temperatursensoren von Philips**
AT24C01 1Kbit,	LM75 digital temperature sensor
AT24C02 2Kbit	NE1617 temperature monitor
AT24C04 4Kbit	NE1618 temperature monitor
AT24C08 8Kbit	SE95 temperature sensor
AT24C16 16Kbit	SE98 temperature sensor
…	…
I2C multiplexers/switches, Philips	**I2C bus repeaters/hubs/extenders**
PCA9542 2-channel multiplexer	P82B715 extender
PCA9543 2-channel switch	P82B96 buffer
PCA9544 4-channel multiplexer	PCA9509 repeater
PCA9545 4-channel switch	PCA9516 5-channel hub
…	…

2.4 Struktur und Funktion des I²C Bus

In einem I²C-Bus gibt es mindestens einen Baustein der als I²C-Master fungiert und eine „beliebige“ Anzahl I²C-Slaves. Es ist aber auch möglich mehrer Master in einem I²C-Bus zu betreiben (Multi-Master-Bus). Jeder Slave besitzt eine eindeutige Adresse die sich aus einer Geräteidentifikation und evtl. wählbaren Adressbits bildet. Der Master allein übernimmt die Buskontrolle und spricht die Slaves an. Ein I²C-Slave ist nicht in der Lage von sich aus Daten zu senden. Er wird grundsätzlich vom Master dazu aufgefordert (Read/Write/Acknowledge). Die Reihenfolge der Aktionen (Kommandos und Daten), die auf dem Bus gesendet und empfangen werden, ist streng festgelegt (I²C-Protokoll).
I²C nutzt einen Adressraum von 7 Bit, damit sind 128 Adressen möglich. Das Bit 0 der Adresse wird als READ/WRITE-Flag benutzt und kennzeichnet die Datenrichtung der nächsten Datenübertragung (WRITE = Master sendet - Slave empfängt oder READ = Master empfängt - Slave sendet). Zu beachten ist, dass einige Adressen jedoch für Spezialzwecke reserviert sind. Wie bereits beschrieben, wird in neueren Spezifikationen der Adressraum um einen 10 Bit-Modus erweitert. Der I²C-Bus arbeitet standardmäßig mit 100 kbit/s. Zusätzlich sind ein Fast-Mode mit 400 kbit/s und ein High-Speed-Mode mit 3,4 Mbit/s möglich. AVR Controller verfügen über den Standard- und den Fast-Mode mit 100 bzw. 400 kbit/s und unterstützen den 7 Bit-Adress-Mode.

Die zwei Leitungen zum Datenaustausch, werden mit SDA (Serial Data Line) und SCL (Serial Clock Line) bezeichnet. Diese werden von Pull-Up-Widerständen definiert auf High gezogen. Die Geräte ziehen zur Kommunikation die Leitungen auf Low. Adresse, Kommandos, Steuerinformationen und Daten werden gemeinsam auf der Datenleitung (SDA) seriell übertragen. Jeder Gerätetyp ist durch einen Geräte-ID in der Adresse codiert.



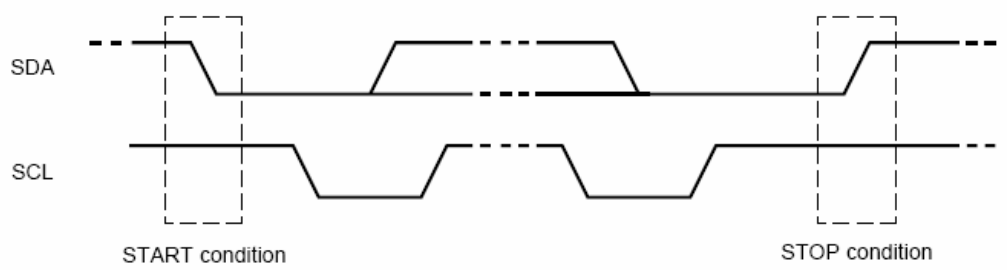
Typische Konfiguration

Geräte ID				wählbare Adressbits			R/W
Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Gesamte 7 Bit Adresse							

Adressbildung im I²C Bus

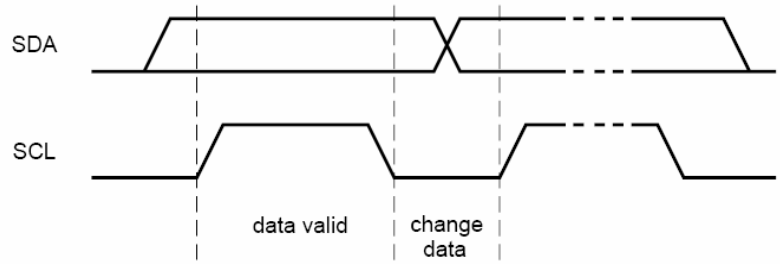
2.5 Prinzipielle Arbeitsweise des I²C Bus

Der Datentransfer auf dem I²C Bus wird durch zwei Ereignisse vom Master kontrolliert: **START** und **STOP**. Das START-Ereignis einer Datenübertragung wird dadurch definiert, dass die Leitung SDA von High auf Low wechselt während die SCL Leitung High bleibt. Das STOP-Ereignis liegt vor, wenn der Level der SDA Leitung von Low nach High wechselt, währen die SCL Leitung High bleibt. Diese beiden vom Master ausgelösten Ereignisse rahmen jede Kommunikation auf dem I²C-Bus ein.



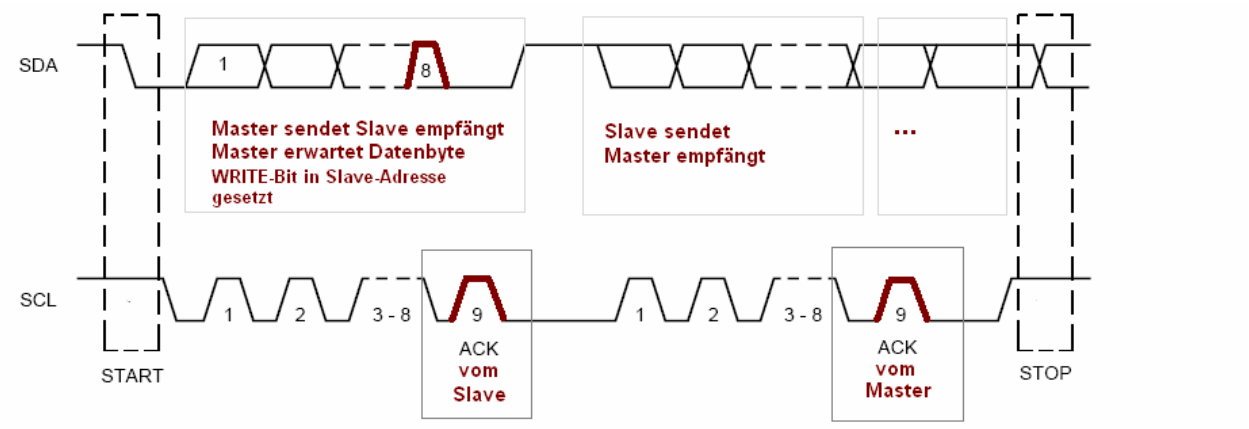
START, STOP Ereignisse entsprechend der original Philips Spezifikation

Bei der eigentlichen Datenübertragung zwischen den Ereignissen START und STOP werden auf der Leitung SCL (Clock) das Taktsignal zur Synchronisation und auf der Leitung SDA (Data) die Datenbits übertragen.



Bitübertragung entsprechend der original Philips Spezifikation

Damit ist erst einmal klar geregelt wie über die eine einzige Datenleitung Datenbytes vom Master an einen Slave übertragen werden können. Da ein Slave von sich aus keine Datenübertragung initiieren kann dazu jedoch aktiv die Datenleitung SDA benutzen muss, wird er vom Master durch das Setzen des READ/WRITE Bits aufgefordert Daten zu senden. Im Übertragungsprotokoll wird ein zusätzliches Bestätigungsbit für den Datenempfang (Acknowledge) gesendet. Unter diesen Bedingungen darf der Slave für die Übertragung des folgenden Bytes die Datenleitung aktiv als Sender benutzen und der Master empfängt die Datenbytes. Die konkrete Folge von Signalen ist dem jeweiligen I²C Protokoll des entsprechenden Bausteins zu entnehmen (Datenblätter)



3 Projektaufgabe und spezielle Grundlagen

3.1 Aufgabenstellung

Es ist ein Datenerfassungsgerät (Datenlogger) auf der Basis von Standard-I²C-Bausteinen für die Erfassung und Speicherung von Temperaturdaten über einen längeren Zeitraum zu realisieren. Die Steuerung soll durch einen 8 Bit RISC Mikrocontroller der AVR-Serie von Atmel übernommen werden. Das Gerät soll über einen wählbaren Zeitraum in einem einstellbaren Intervall Temparaturdaten und den Erfassungszeitpunkt autonom (ohne PC) erfassen und speichern. Die Temperatur wird von einem I²C Temperatursensor und die Erfassungszeit von einer I²C Echtzeituhr geliefert. Die Messdaten sollen in einem I²C EEPROM gespeichert werden bis der Datenlogger an einen PC angeschlossen wird um die Messdaten zu übernehmen. Mit diesem Gerät soll zum Beispiel der Temperaturverlauf an einem Ort oder in einem Raum während der Nacht, während eines Tages oder über eine Woche hinweg erfasst werden. Die Auswertung kann dann durch Übertragen der Daten an einen PC zum Beispiel in EXCEL erfolgen.

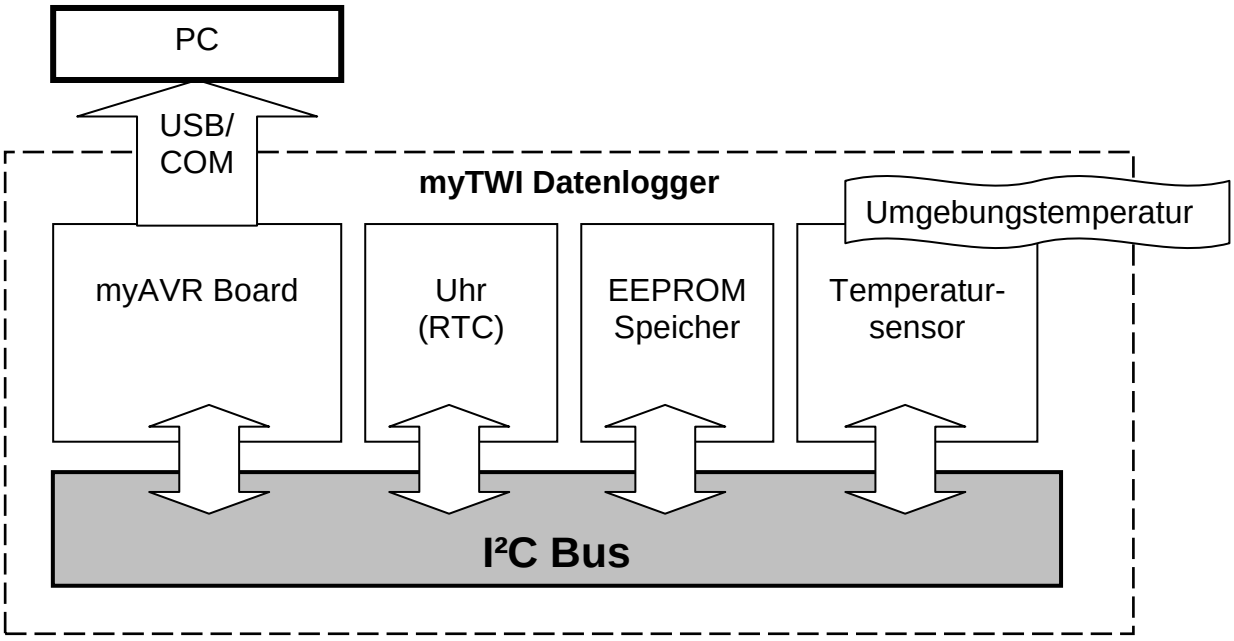
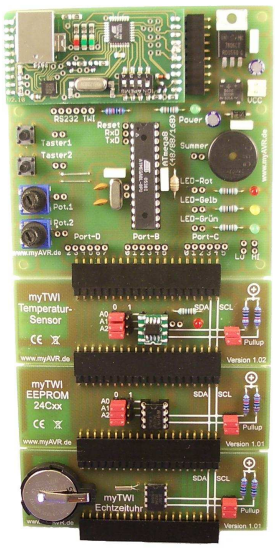


Bild : Blockschaltbild „myTWI“

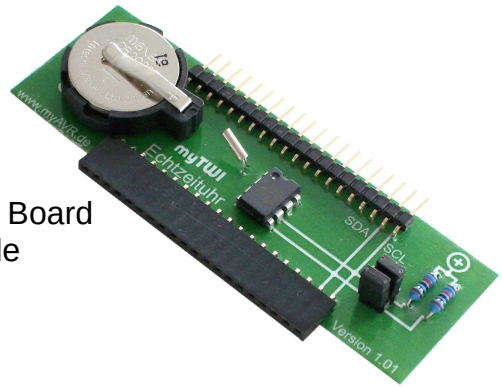
Für die Realisierung des Datenerfassungsgerätes werden fertige myAVR Komponenten verwendet die in einem Bussystem zusammenwirken.



myTWI Komponenten

- myAVR Board MK2 USB
- I²C Temperatursensor Add-On (mit LM75)
- I²C EEPROM Add-On (mit 24C02)
- I²C RTC Add-On (mit DS1307)

3.2 Grundlagen zum RTC Baustein



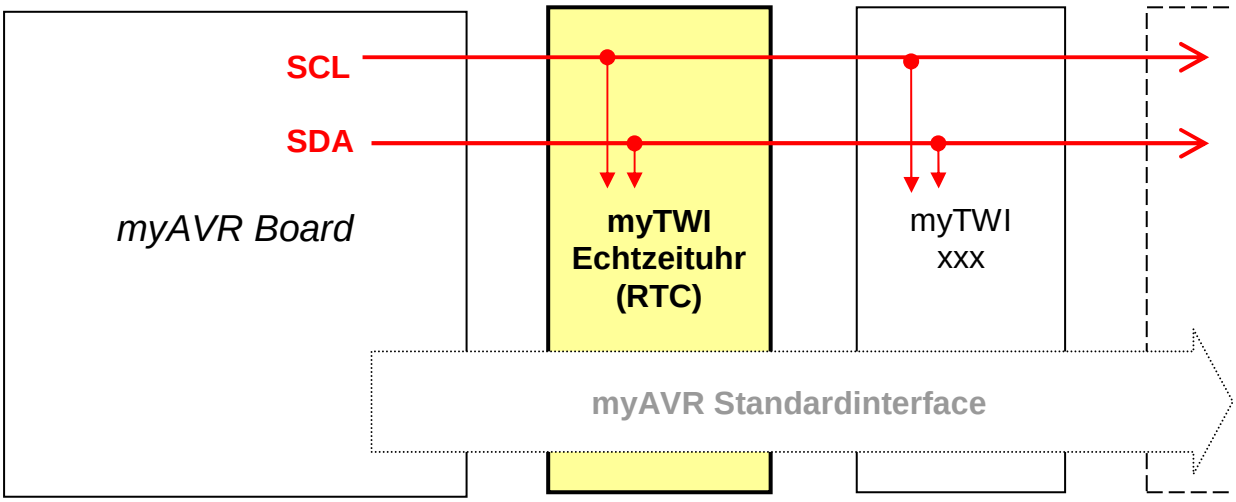
Eigenschaften

- TWI (I²C) Real Time Clock Modul
- 1 x RTC DS1307 Echtzeituhr
- Steckerleiste für den Anschluss an das myAVR Board
- Buchsenleise für den Anschluss weiterer Module
- Robust, mit Dokumentationsdruck
- Industriefertigung
- Material: FR4; 1,5 mm; 0,35 µm Cu
- Gebohrt, verzinnt, Lötstoppmaske

Allgemeine Beschreibung

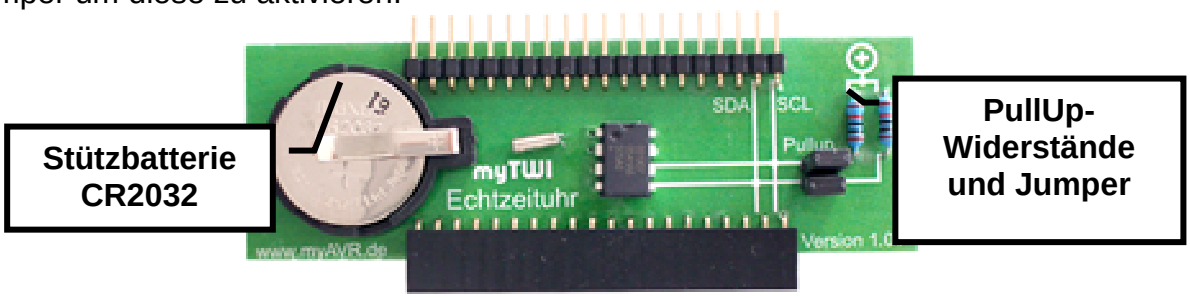
Die Zusatzplatine (Add-On) „myTWI Echtzeituhr“ ist ein Teil der TWI-Serie für das myAVR Board. Damit wird es möglich das myAVR-System um eine externe Echtzeituhr (RTC, Real Time Clock) vom Typ DS1307 zu erweitern. Es kann mit weiteren TWI (I²C) Add-Ons am myAVR Erweiterungsport angeschlossen werden.

Blockbild

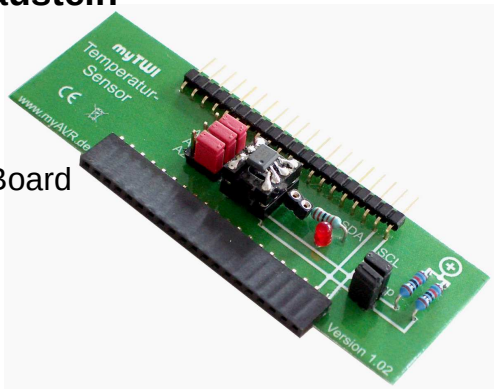


Handhabung

Das myAVR TWI Echtzeituhr Add-On kann mit weiteren TWI Add-Ons in einem I²C-Bus betrieben werden. Ein I²C Gerät bildet aus seinem Geräte-ID und den möglichen Adress-Pins (A0-A2) seine Geräteadresse im Bus. Bei der Echtzeituhr ist nur ein Gerät im Bus möglich. Diese verfügt über keine zusätzlichen Adress-Pins. Für den Fall, dass das System abgeschaltet ist, kann die Echtzeituhr mit einer Stützbatterie (CR2032) kontinuierlich weiterlaufen. Des Weiteren muss der I²C Bus mit PullUp-Widerständen auf High gezogen werden. Dies sollte jeweils nur von einem Add-On erfolgen. Dazu verfügt jedes Add-On über entsprechende PullUp-Widerstände und Jumper um diese zu aktivieren.



3.3 Grundlagen zum Temperatursensor Baustein



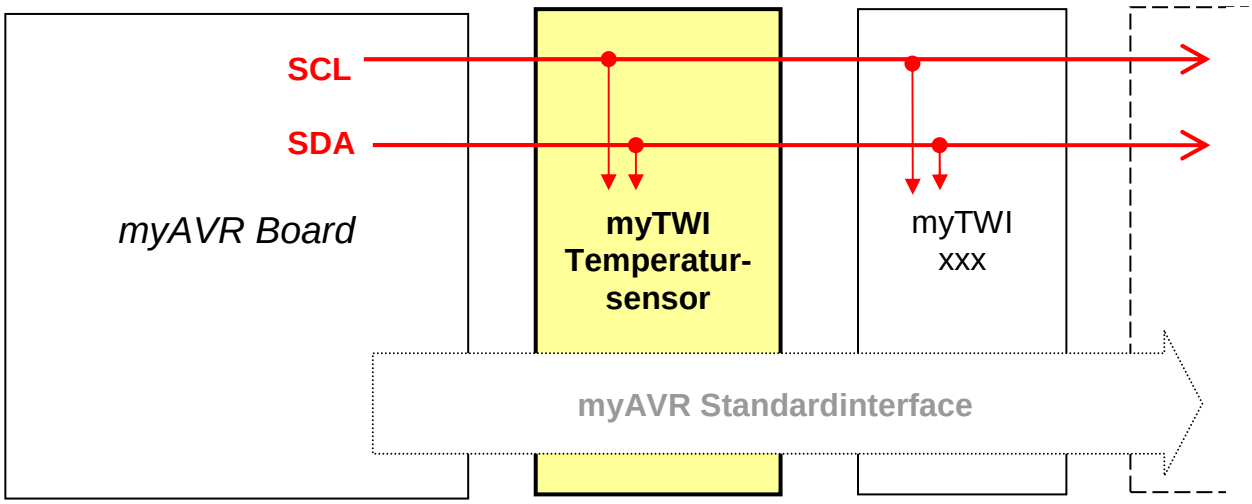
Eigenschaften

- TWI (I²C) Temperatursensor Modul
- 1 x Temperatursensor LM75
- Steckerleiste für den Anschluss an das myAVR Board
- Buchsenleise für den Anschluss weiterer Module
- Robust, mit Dokumentationsdruck
- Industriefertigung
- Material: FR4; 1,5 mm; 0,35 µm Cu
- Gebohrt, verzinnt, Lötstoppmaske

Allgemeine Beschreibung

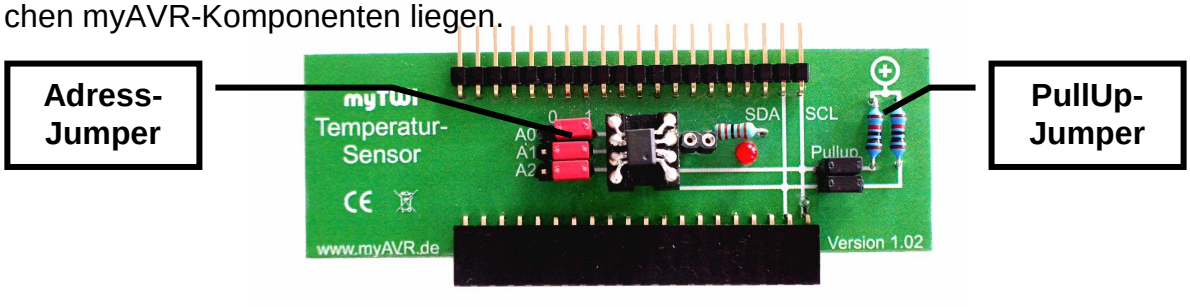
Die Zusatzplatine (Add-On) „myTWI Temperatursensor“ ist ein Teil der TWI-Serie für das myAVR Board. Damit wird es möglich, das myAVR-System um einen externen Temperatursensor mit dem Standardbaustein LM75 zu erweitern. Es kann mit weiteren TWI (I²C) Add-Ons am myAVR Erweiterungsport angeschlossen werden.

Blockbild

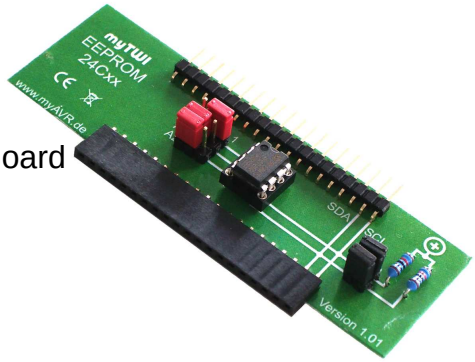


Handhabung

Das myAVR TWI Temperatursensor Add-On kann mit weiteren TWI Add-Ons in einem Bus betrieben werden. Ein I²C Gerät bildet aus seinem Geräte-ID und den möglichen Adress-Pins (A0-A2) seine Geräteadresse im Bus. Somit lassen sich auch mehrere gleiche Geräte in einem Bus betreiben. Auf diesem myTWI Add-On sind die Adresspins per Jumper konfigurierbar. Des Weiteren muss der I²C Bus mit PullUp-Widerständen auf High gezogen werden. Dies sollte jeweils nur von einem Add-On erfolgen. Dazu verfügt jedes Add-On über entsprechende PullUp-Widerstände und Jumper, um diese zu aktivieren. Beachten Sie, dass die technischen Möglichkeiten (Temperaturbereiche) des LM75 weit außerhalb der Einsatzbedingungen der restlichen myAVR-Komponenten liegen.



3.4 Grundlagen zum EEPROM Baustein



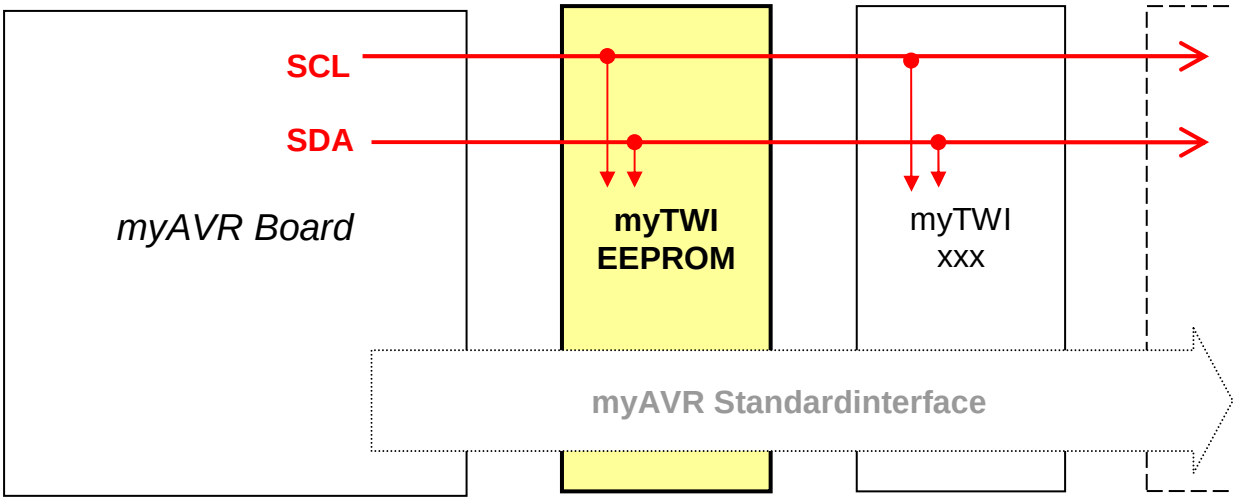
Eigenschaften

- TWI (I²C) EEPROM Modul
- 1 x EEPROM 24Cx (im Lieferumfang 24C02)
- Steckerleiste für den Anschluss an das myAVR Board
- Buchsenleise für den Anschluss weiterer Module
- Robust, mit Dokumentationsdruck
- Industriefertigung
- Material: FR4; 1,5 mm; 0,35 µm Cu
- Gebohrt, verzinnt, Lötstoppmaske

Allgemeine Beschreibung

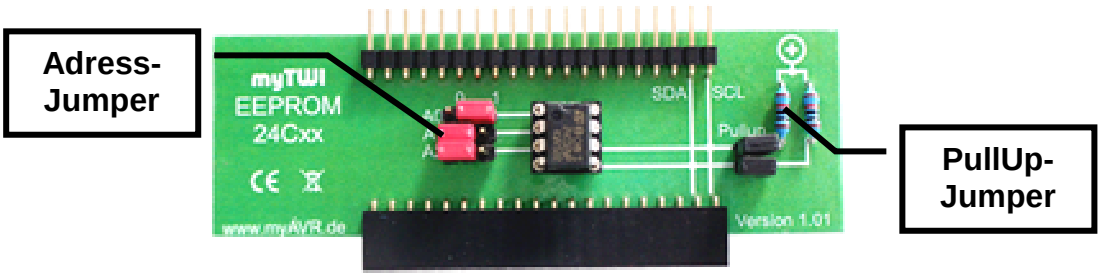
Die Zusatzplatine (Add-On) „myTWI EEPROM“ ist ein Teil der TWI-Serie für das myAVR Board. Damit wird es möglich das myAVR-System um einen externen EEPROM der 24Cx Typenreihe zu erweitern. Es kann mit weiteren TWI (I²C) Add-Ons am myAVR Erweiterungsport angeschlossen werden.

Blockbild



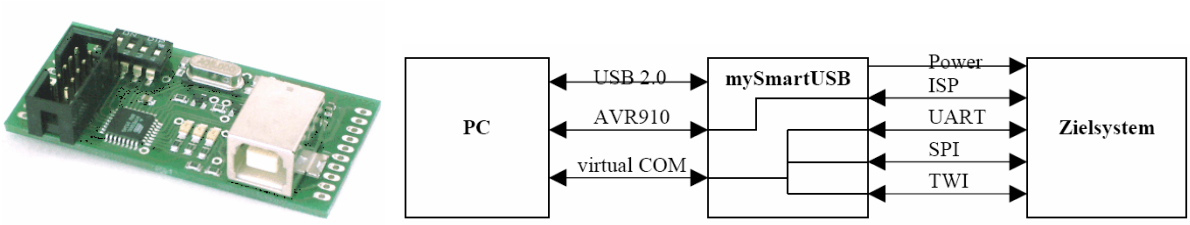
Handhabung

Das myAVR TWI EEPROM Add-On kann mit weiteren TWI Add-Ons in einem I²C-Bus betrieben werden. Ein I²C Gerät bildet aus seinem Geräte-ID und den möglichen Adresspins (A0-A2) seine Geräteadresse im Bus. Somit lassen sich auch mehrere gleiche Geräte in einem BUS betreiben. Auf diesem myTWI Add-On sind die Adress-pins per Jumper konfigurierbar. Des Weiteren muss der I²C Bus mit PullUp-Widerständen auf High gezogen werden. Dies sollte jeweils nur von einem Add-On erfolgen. Dazu verfügt jedes Add-On über entsprechende PullUp-Widerstände und Jumper um diese zu aktivieren.

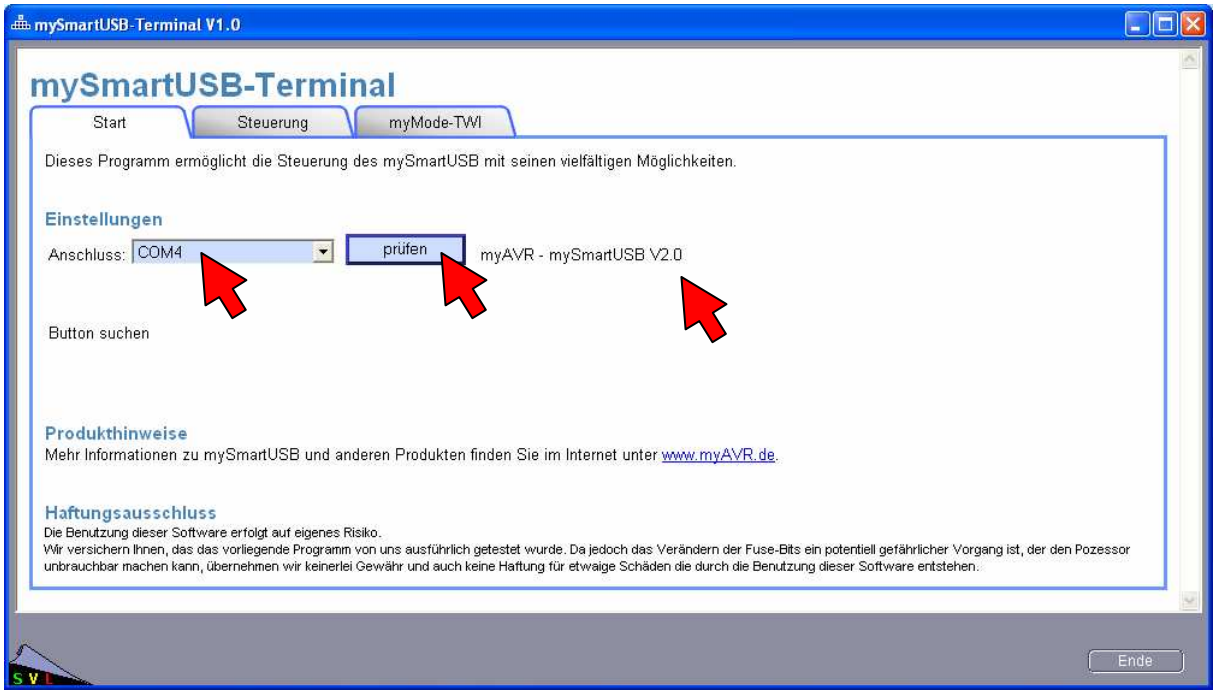


3.5 mySmartUSB MK2 und das mySmartUSB-Terminal

Der mySmartUSB MK2 ist in der Lage AVR Controller im Zielsystem per USB zu programmieren (ISP) oder mit dem Controller zu kommunizieren (UART, TWI, SPI). Dabei kann mySmartUSB MK2 wahlweise die Spannungsversorgung des Zielsystems 5V bis zu 80mA übernehmen. Somit sind für viele Anwendungsfälle keine externen Spannungsquellen wie Netzteile oder Batterien nötig. Als USB Programmer ist er zum Atmelstandard AVR910/AVR911 kompatibel. Die zusätzlichen Eigenschaften des mySmartUSB MK2 ermöglichen weitere Anwendungsvarianten, so zum Beispiel für einen UART-USB, SPI-USB, TWI-USB Konverter.



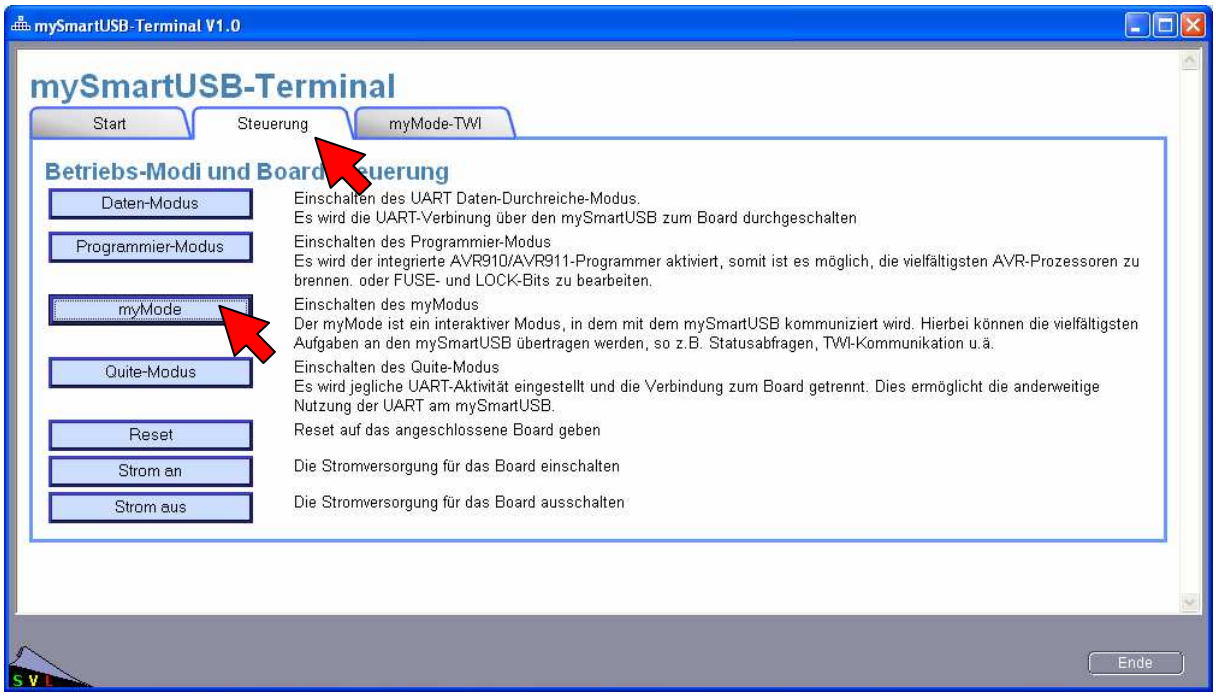
Für die erweiterten Fähigkeiten der Firmware gibt es spezielle Werkzeuge von myAVR, wie das myAVR Controlcenter und das mySmartUSB-Terminal. Das mySmartUSB-Terminal ist sehr gut geeignet die einzelnen Schritte eines I²C Protokolls auszuführen und die Abfolge der Aktionen und Rückmeldungen zu beobachten. Wir nutzen es um die I²C Protokolle der verwendeten Bausteine (LM57, DS1307, 24C02) zu analysieren und zu verstehen. Das mySmartUSB-Terminal finden Sie im Downloadbereich von www.myAVR.de. Installieren und starten Sie das Programm.



Beim ersten Schritt ist es wichtig, dass die USB Treiber für den mySmartUSB MK2 installiert sind und dass der Programmer an einem USB Anschluss angesteckt ist. Sie können den virtuellen COM Port des mySmartUSB MK2 manuell einstellen oder automatisch suchen lassen. Prüfen Sie die Verbindung zum mySmartUSB MK2. Die Firmwareversion muss für die I²C Kommunikation die Nummer 2.0 oder höher besitzen.
Nach dem Anstecken des mySmartUSB MK2 ist dieser automatisch im Programmiermodus. Um ihn in einen anderen Betriebsmodus zu versetzen, nutzt man die Mög-

lichkeit der Softwaresteuerung. Es ist auch möglich, bestimmte Betriebsmodi über die DIP Schalter auf dem Programmer zu erzwingen. Öffnen Sie die Dialogseite „Steuerung“ und aktivieren den myMode. Dieser stellt uns einen Terminalmode mit unterschiedlichen Kommandos zur Verfügung. Der myMode wird durch die Status-LEDs auf dem mySmartUSB MK2 angezeigt.

- **ROT - - = Programmier-Modus**
- **- GRÜN GRÜN = Daten-Modus**
- **- GRÜN - = myMode**
- **ROT GRÜN GRÜN = Bootloader-Modus**



Beachten Sie, dass die DIP Schalter für diese Betriebsart nicht verstellt werden dürfen.

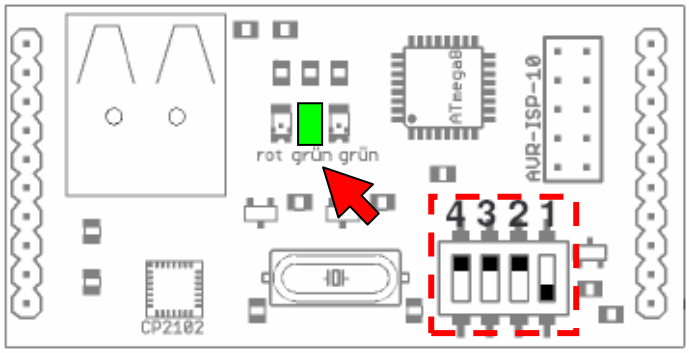
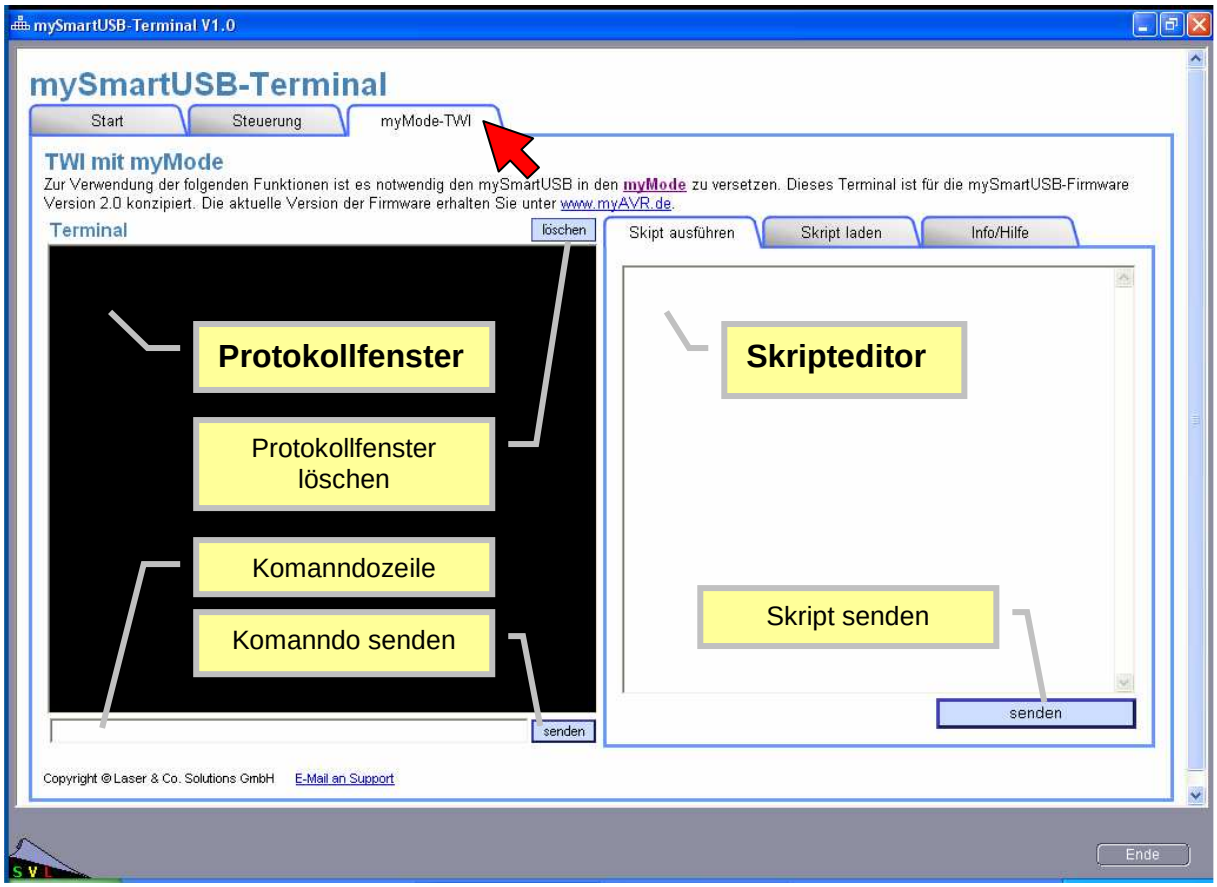
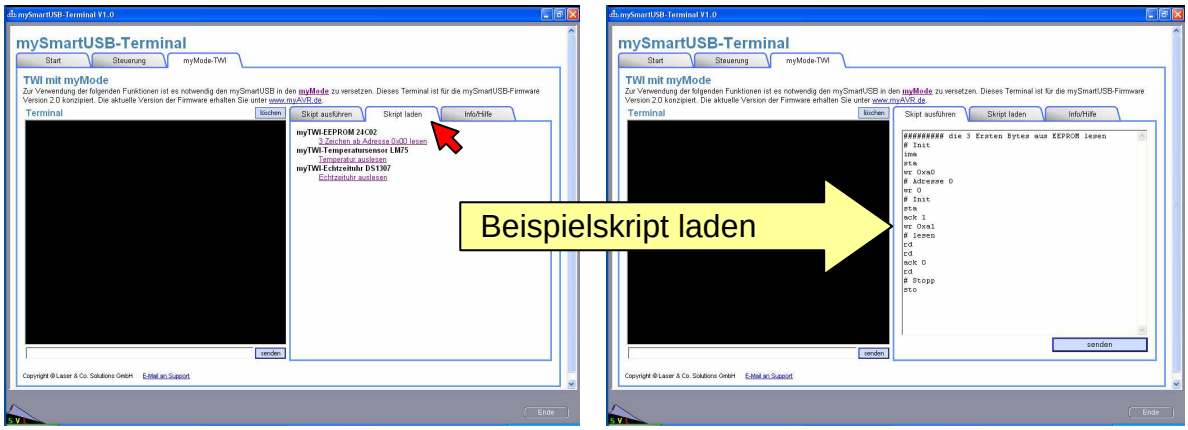


Bild: DIP-Schalterstellung (default) für den myMode

Für die Arbeit mit den I²C (TWI) Kommandos im Dialogbetrieb schalten Sie bitte auf myMode-Terminal um. Das Terminal bietet folgende Möglichkeiten:

- Dialogbetrieb mit einzelnen Kommandozeilen
- Protokoll- und Ergebnisausgaben
- Skriptarbeit für Komandosequenzen
- Hilfen





Skriptbefehle (Auszug für I²C):

```
- rst 0|1    Resetleitung OFF|ON     - ack 0|1     Acknowledge 0/1
- pwr 0|1    Power OFF|ON            - wr [byte]   TWI Write
- m:main     Hauptmenü               - rd [anz]    TWI Read
- m:twi      Untermenü TWI           - rda [anz]   TWI Read + Ack
- m:?        Menühilfe               - state       TWI Status
- ima        init TWI Master         - # ...       Kommentarzeile
- sta        TWI Start               - !cls        Clear Screen
- sto        TWI Stop                - end         TWI deinit
- sla [adr] w|r Slave-Adress
```

5 Teilprojekt Zeiterfassung

5.1 Analyse der Zeiterfassung

In der Problemanalyse geht es darum den Baustein und die Kommunikation mit ihm zu verstehen und daraus die nötige Systemstruktur abzuleiten.
Aus dem Datenblatt des DS1307 sind folgende Leistungsmerkmale ersichtlich:

- I²C (TWI) Echtzeituhr (Real-time clock, RTC)
- liefert Sekunde, Minute, Stunde, Tag, Monat, Wochentag und Jahr
- Daten liegen im BCD Format vor
- Register 0 = Sekunde, BCD, Bit 7 Stop/Run RTC
- Register 1 = Minute, BCD
- Register 2 = Stunde, BCD
- berücksichtigt Schaltjahre bis 2100
- batteriegestützte Datenhaltung
- verbraucht weniger als 500nA im Batteriebetrieb
- GeräteID 0b1101xxxx = 0xD0
- keine Adresserweiterung, also nur eine RTC in einem Bus

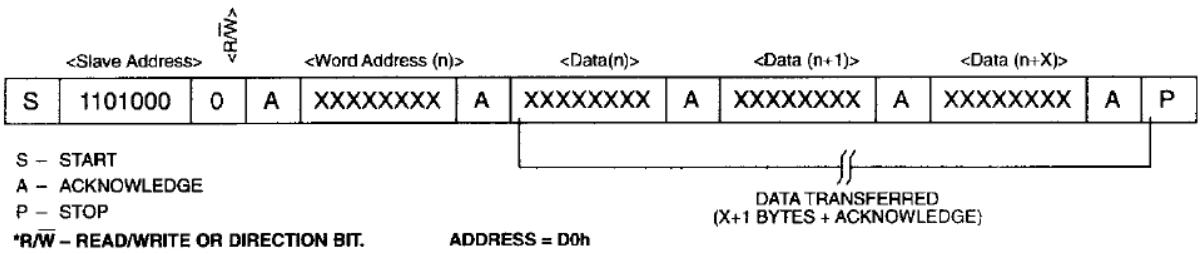
Die Kommunikationsprotokolle des DS1307 sind ebenfalls dem Datenblatt zu entnehmen. Wir konzentrieren uns auf folgende I²C Protokolle für die Echtzeituhr:

- Initialisierung, aktivieren
- Uhr stellen (Sekunde, Minute, Stunde)
- Zeit auslesen (Sekunde, Minute, Stunde)

Initialisierung (stellen) und aktivieren der Echtzeituhr
Die Initialisierung des DS1307 ist verhältnismäßig einfach. Die aktuellen Zeitdaten sind fortlaufend ab Register 0 in der Echtzeituhr abgelegt. Vergleichen Sie dazu die Registerübersicht aus dem Datenblatt. Das höchstwertige Bit des Registers 0 wird mit CH bezeichnet. CH steht für CLOCK HALT. Damit lässt sich die Uhr anhalten (CH=1) und auch starten (CH=0). Das Einfachste ist, beim Stellen der Uhr den Wert 0x00 in das Sekundenregister zu schreiben; damit startet die Uhr.

00H	CH	SECONDS
	MINUTES	
	HOURS	
	DAY	
	DATE	
	MONTH	
	YEAR	
07H	OUT-CONTROL	
08H 3FH	RAM 56 x 8	

Dem Datenblatt entnehmen wir die Protokollsequenz zum Schreiben der Register des DS1307.



Zu beachten ist, dass nur die erste Adresse (Startadresse) der Schreiboperation angegeben werden muss und dann fortlaufend Datenbytes gesendet werden können. Der DS1307 stellt den Adresspointer automatisch weiter (autoincrement). Es sind also folgende I²C Befehle nötig, um die Echtzeituhr für die Messung der Uhrzeit zu konfigurieren:

- den mySmartUSB MK2 als TWI-Master initialisieren
- die START-Sequenz auslösen

- Register Nummer 0, 1 und 2 (Sekunde, Minute und Stunde) laden
  - o DS1307 adressieren
  - o Register 0 adressieren
  - o Sekunde schreiben (Uhr starten)
  - o Minute schreiben
  - o Stunde schreiben
- die STOP-Sequenz senden

Das ergibt folgende Skriptsequenz im myMode Terminal (10 Uhr 45 Minuten):

```
rst 1
m:twi
ima
sta
wr 0xD0
wr 0x00
wr 0x00
wr 0x45
wr 0x10
sto
```

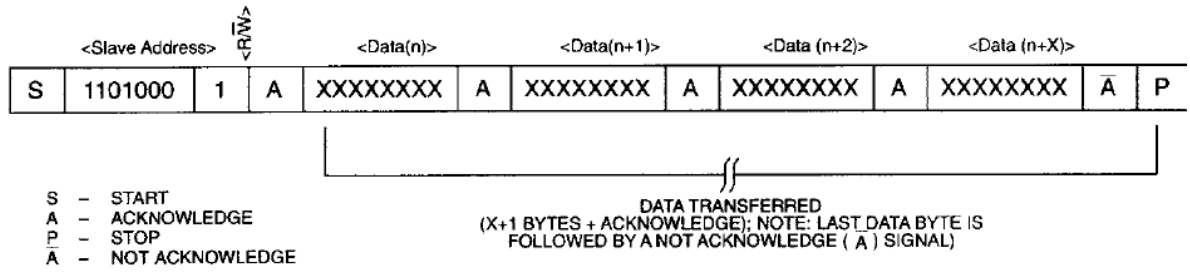
Den Controller auf dem myAVR Board MK2 USB deaktivieren.
In das TWI (I²C) Menü wechseln.
mySmartUSB MK2 als Master initialisieren
TWI START
TWI WRITE Adresse des DS1307 (GeräteID)
TWI WRITE Registernummer 0
TWI WRITE Sekunde, Bit7=0 Uhr starten
TWI WRITE Minute
TWI WRITE Stunde
TWI STOP

Schließen Sie zuerst das Add-On „myTWI Echtzeituhr“ an das myAVR Board MK2 USB an und verbinden Sie danach das myAVR Board mit einem USB Anschluss an Ihrem Rechner. Starten Sie das mySmartUSB Terminal und aktivieren den myMode. Geben Sie die oben gezeigte Kommandosequenz ein und führen sie diese aus.

Sie können das Skript auf Wochentag, Tag, Monat und Jahr erweitern. Diese I²C Sequenz ist als Initialisierungsfunktion der Echtzeituhr in der zu realisierenden Mikrocontrollerlösung zu implementieren.

Auslesen der Uhrzeit

Nach dem die Uhr gestellt und gestartet wurde, läuft diese mit hoher Genauigkeit und bei eingelegter Batterie auch ohne Spannungsversorgung über das Board. Für das Auslesen der Uhrzeit gibt das Datenblatt der Echtzeituhr folgendes Protokoll vor:



Es sind also folgende I²C Befehle nötig, um die Uhrzeit zu ermitteln:

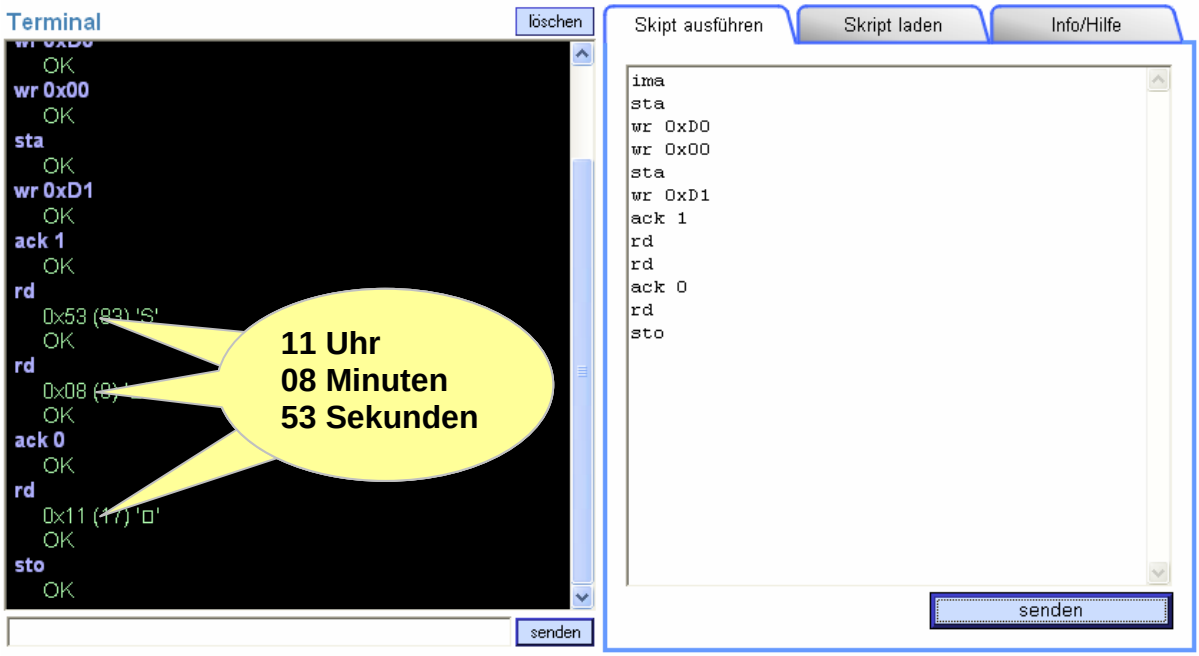
- den mySmartUSB MK2 als TWI-Master initialisieren
- die START-Sequenz auslösen
- Register Nummer 0-2 (Stunde, Minute, Sekunde) auslesen
    - DS1307 adressieren
    - Register 0 adressieren, Registerpointer setzen ab Adresse 0 lesen
    - Wiederholt TWI START senden
    - DS1307 adressieren + Write-Bit = Sendeaufforderung an Slave
    - Slave sendet Sekunde, Master empfängt und bestätigt Bereitschaft das nächste Byte zu empfangen (Acknowledge)
    - Slave sendet Minute, Master empfängt und bestätigt Bereitschaft das nächste Byte zu empfangen (Acknowledge)
    - Slave sendet Stunde, Master empfängt und negiert Bereitschaft das nächste Byte zu empfangen (No), keine weiteren Daten gewünscht
- die STOP-Sequenz senden

Das ergibt folgende Skriptsequenz im myMode Terminal (10 Uhr 45 Minuten):

```
rst 1
m:twi
ima
sta
wr 0xD0
wr 0x00
sta
wr 0xD1
ack 1
rd
rd
ack 0
rd
sto
```

Den Controller auf dem myAVR Board MK2 USB deaktivieren.
In das TWI (I²C) Menü wechseln.
mySmartUSB MK2 als Master initialisieren
TWI START
TWI WRITE Adresse des DS1307 (GeräteID)
TWI WRITE Registernummer 0
Wiederholt TWI START
TWI WRITE Adresse GeräteID+Sendeaufforderung
Acknowledge = 1 (mehr/weitere Daten erwartet)
TWI READ Sekunde
TWI READ Minute
Acknowledge = 0 (letztes Byte keine weiteren Daten erwartet)
TWI READ Stunde
TWI STOP

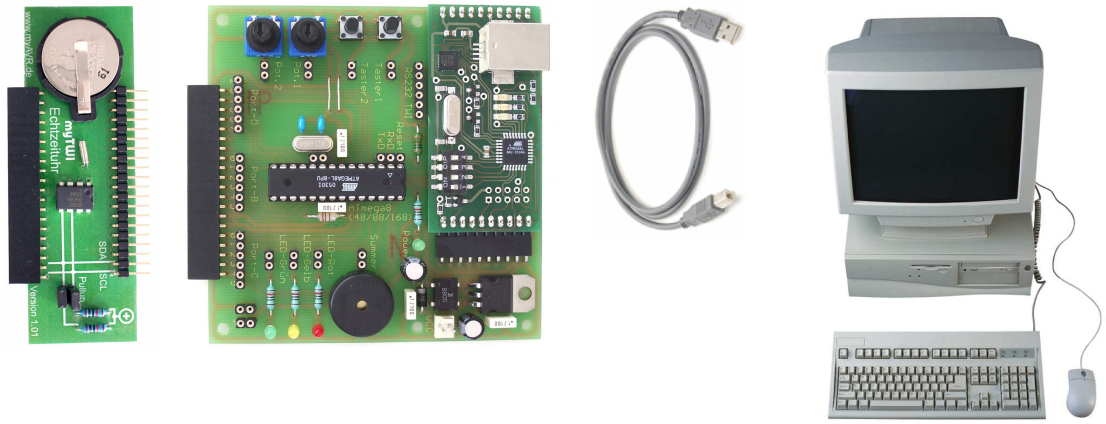
Starten Sie das mySmartUSB-Terminal und aktivieren den myMode. Geben Sie die oben gezeigte Kommandosequenz ein und führen Sie diese aus.



Sie können das Skript auf Wochentag, Tag, Monat und Jahr erweitern. Diese I²C Sequenz ist als Lesefunktion für die Echtzeituhr in der zu realisierenden Mikrocontrollerlösung zu implementieren.

Systemstruktur festlegen

Die Teillösung besteht zunächst nur aus der I²C Echtzeituhr, dem myAVR Controllerboard und einem PC mit Terminalprogramm.



Das Controllerboard soll fortlaufend die aktuelle Zeit von der Echtzeituhr auslesen und in einem lesbaren Format an den PC senden. Aus den Erkenntnissen zum Aufbau und zur Funktion der Echtzeituhr DS1307 lässt sich die nötige Systemstruktur ableiten. Im Folgenden ist die Modulstruktur mit den Mitteln der Strukturierten Analyse visualisiert.

Bild: Kontextdiagramm Teilsystem Temperaturerfassung

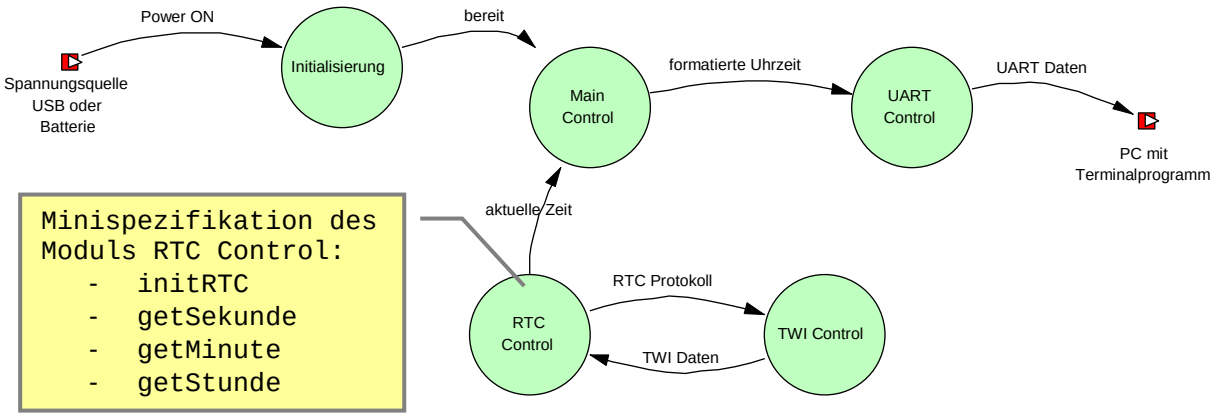


Bild: Level 0 Diagramm Teilsystem Temperaturerfassung

5.2 Entwurf der Zeiterfassung

Der folgende Modulentwurf zeigt den groben Ablauf (von links nach rechts) und die grobe Unterprogrammstruktur (Modulbaum, Structured Chart). Besonders hervorzuheben sind die Gemeinsamkeiten mit der Modulstruktur der vorangegangenen Teillösung.

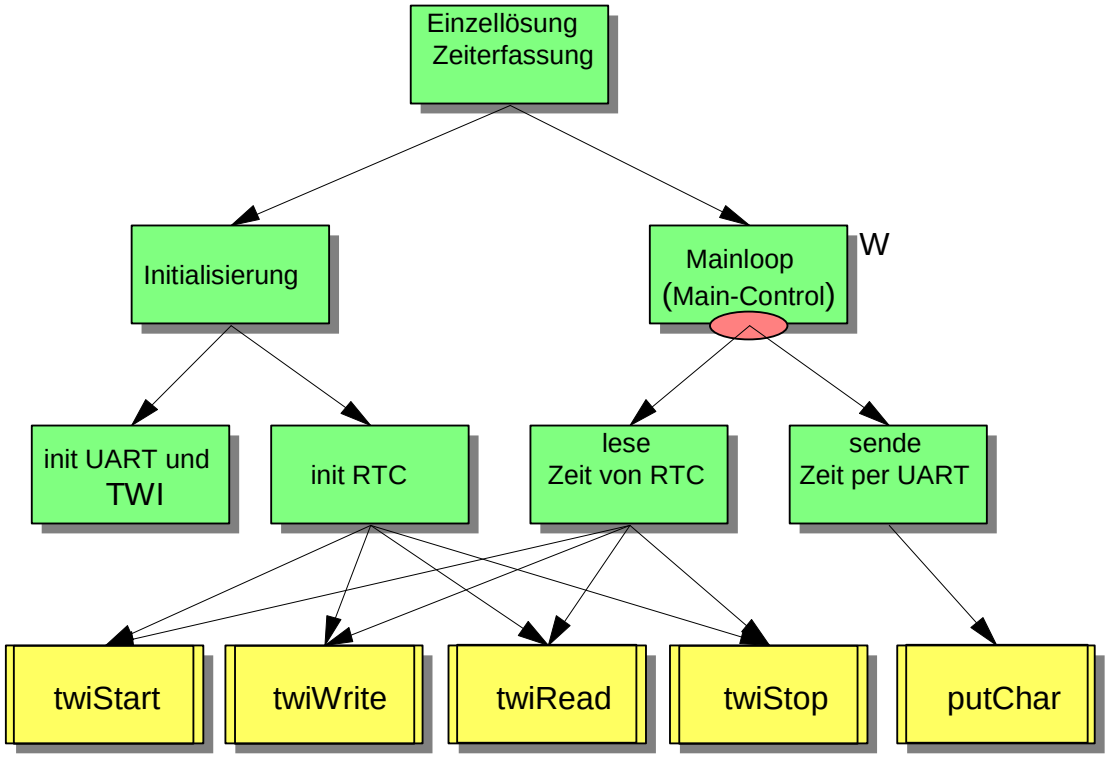


Bild: Grobentwurf der Einzellösung für das Modul Zeiterfassung

5.3 Realisierung der Zeiterfassung

Sie können das Teilprojekt Zeiterfassung mit SiSy AVR oder dem myAVR Workpad realisieren. Da das Teilprojekt und erst recht das Gesamtprojekt eine doch beachtliche Komplexität annehmen wird, ist es empfehlenswert SiSy AVR und dessen Strukturierungsmöglichkeiten zu nutzen.