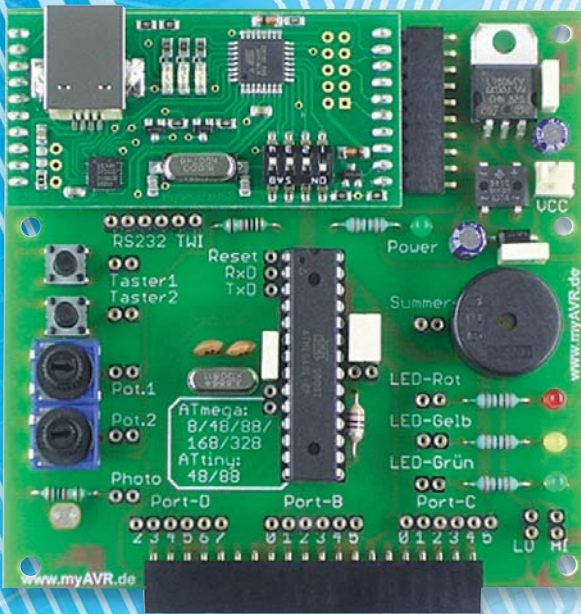




Mikrocontroller-Einstieg

Teil 14: 1-Wire



```
BASCOM-AVR IDE [2.0.7.5] - [C:\user\BASCOM-Programme\Blinker_attiny13.bas]
Datei Editieren Anzeigen Programmieren Werkzeuge Optionen Fenster Hilfe
Blinker_attiny13.bas
Sub
  BASCOM-Programm
  Einfacher Blinker
  In: -
  Out: LED mit Vorwiderstand an Portb.4

  $regfile = "attiny13.dat"
  $crystal = 1200000
  $hwstack = 4
  $swstack = 4
  $framesize = 10

  Config PORTB.4 = Output

  Do
    PORTB.4 = 1
    Waitms 500
    PORTB.4 = 0
    Waitms 500
  Loop
End |

'Verwendeter Chip
'Verwendete Frequenz
'Rücksprungadressen (je 2), Registersicherungen (32)
'Parameteruebergaben (je 2), LOCALs (je 2)
'Parameter (Daten-Laenge), Rechenbereich Funktionen
'B.4 als Ausgang definieren

'Schleifenbeginn
'B.4 auf 1
'Warteschleife 500 ms
'B.4 auf 0
'Warteschleife 500 ms
'Schleifenende
'Programmende
```



mit **BASCOM-AVR**

Nachdem in den Teilen 11 bis 13 unserer Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR“ verschiedene Möglichkeiten mit dem I²C-Bus – der wegen der benötigten Daten- und Taktleitung auch Two-Wire-Interface genannt wird – gezeigt wurden, geht es im vorliegenden Teil um den 1-Wire-Bus. Der Name 1-Wire-Bus weist darauf hin, dass bei Verwendung dieses Busses nur eine Datenleitung (und keine Taktleitung) benötigt wird. Exemplarisch wird die Verwendung des weit verbreiteten Temperatursensors DS18B20 behandelt.

Aufbau des 1-Wire-Busses

1-Wire (Eindraht-Bus) ist eine serielle Schnittstelle der Firma Maxim Integrated (vormals Dallas Semiconductor) mit nur einer Datenleitung. Es gibt keine Taktleitung (asynchron). Außer der Datenleitung wird noch eine gemeinsame GND-Leitung und die Spannungsversorgung benötigt. Die Datenleitung kann bei den meisten 1-Wire-Bausteinen sogar zur Spannungsversorgung eines angeschlossenen Slave-Bausteins benutzt werden. Man hat dann nur noch GND und die Datenleitung und spricht von „parasitärer Spannungsversorgung“ im Gegensatz zu „externer Spannungsversorgung“.

Den Busaufbau mit externer Spannungsversorgung der Busbausteine zeigt [Bild 1](#). Es gibt einen Mikrocontroller als Master, der auch die 1-Wire-Befehle initiiert, und es gibt eine Vielzahl von 1-Wire-Slave-Bausteinen. Der Master und die Slave-Bausteine haben eine gemeinsame GND-Leitung. Der Master und jeder Slave haben jeweils eine Spannungsversorgung und der Master und die Slaves sind durch die Datenleitung (DQ) verbunden. Über die Datenleitung können die Daten in beide Richtungen übertragen werden (bidirektional). Wichtig ist, dass ein 1-Wire-Bus (genau) einen Pull-up-Widerstand (RPU) für die Datenleitung hat. Wie in [Bild 2](#) zu sehen, arbeitet jeder Busteilnehmer mit einem Open-Drain-Ausgang, d. h. jeder Busteilnehmer ist in der Lage, das Potential der Datenleitung nach GND zu ziehen. Damit ein High-Pegel auf der Datenleitung entstehen kann, muss der erwähnte Pull-up-Widerstand vorhanden sein, der das Spannungspotential zur positiven Spannungsversorgung zieht, wenn kein Open-Drain-Ausgang aktiv ist.

Jeder Baustein hat eine eigene eindeutige Adresse in Form einer 64 Bit langen Zahl ([Bild 3](#)). Dabei sind die ersten 8 Bit (LSB = Least Significant Byte) der sogenannte „family code“ (z. B. &h28 für den DS18B20; &h10

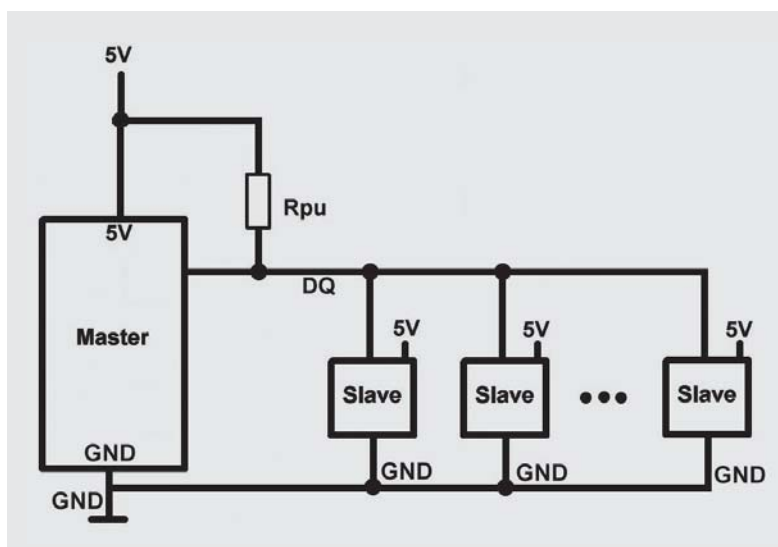


Bild 1: Busaufbau 1-Wire mit externer Spannungsversorgung

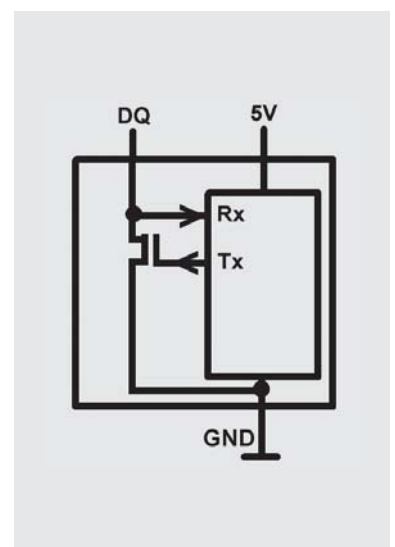


Bild 2: 1-Wire-Slave mit Open Drain

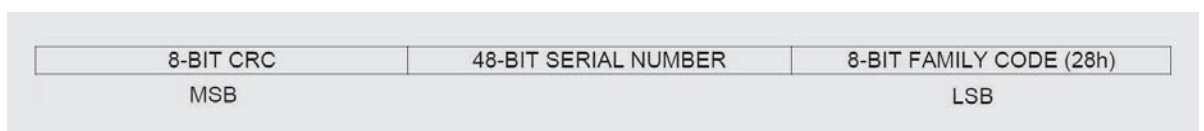


Bild 3: 64-Bit-ROM-Code

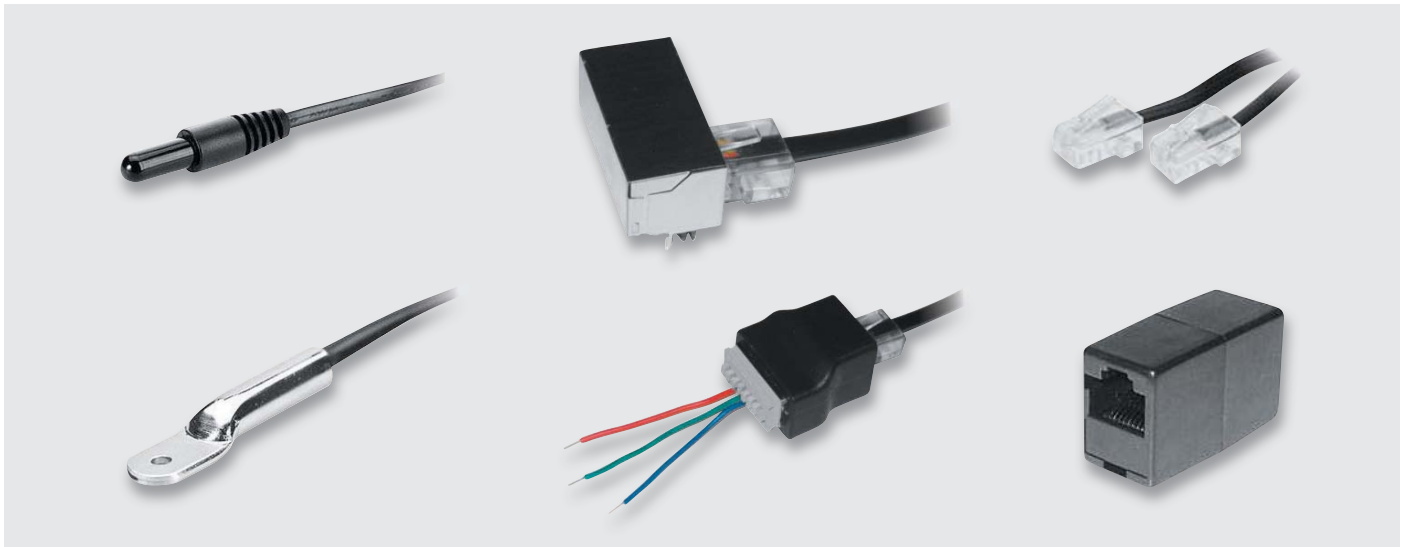


Bild 4: Sensor (Kunststoff bzw. Metall) und Zubehör

für DS18S20 und DS1820) und die letzten 8 Bit (MSB = Most Significant Byte) die Prüfsumme (CRC) der vorausgehenden 7 Bytes. Dazwischen steht die 48 Bit lange Seriennummer. Damit kann man $2 \text{ hoch } 48 = 281.474.976.710.656$ verschiedene Bausteine eindeutig identifizieren.

Es gibt eine Vielzahl verschiedener 1-Wire-Bausteine wie Echtzeit-Uhren, I/O-Bausteine, Batteriemonitore, EEPROM-Speicher und die sehr beliebten Temperatursensor-Bausteine.

Da das Prinzip bei 1-Wire immer gleich ist, wird es am Beispiel des Temperatursensors DS18B20 erläutert. Es handelt sich um einen Temperatursensor, der den Temperaturwert auf 0,5 Grad genau digital über die 1-Wire-Schnittstelle ausgibt [1]. Der Baustein ist unter der Best.-Nr. J7-10 93 37 bzw. J7-10 27 83 bei ELV erhältlich. Achtung: Der hier behandelte DS18B20 unterscheidet sich im Detail vom ebenfalls verbreiteten DS18S20 bzw. dessen Vorgänger DS1820! Für neue Anwendungen wird der DS18B20 empfohlen und daher hier betrachtet [2].

ELV bietet den Sensor in einer für Flüssigkeiten geeigneten Version (Best.-Nr. J7-10 27 83) und in einer Metallversion mit Befestigungsglasche (Best.-Nr. J7-10 93 37) an (Bild 4 links). Die Sensoren werden von ELV mit Anschlusskabel und RJ45-Stecker geliefert. Als Zubehör gibt es ein Verlängerungskabel (Best.-Nr. J7-10 65 35) mit Kupplung (Best.-Nr. J7-10 65 34) (Bild 4 rechts). Mit Hilfe einer Einbaubuchse für Printmontage (Best.-Nr. J7-11 21 47), eines Klemmadapters (Best.-Nr. J7-11 52 88) oder direkt per Kabelanschluss kann ein Sensor elegant in eigene Projekte eingebunden werden (Bild 4 Mitte). Beim Klemmadapter ist die Belegung: 3 = Plus, 4 = Datenleitung, 5 = GND.

Bild 5 zeigt das Blockbild des DS18B20. Außerhalb des eigentlichen Sensors sieht man auch hier wieder den notwendigen Pull-up-Widerstand, der einmal pro 1-Wire-Bus vorhanden sein muss. Dessen Wert sollte in der Größenordnung von 4,7 k Ω liegen. Im linken Block im Bild 5 sieht man die „Parasite Power Circuit“. Diese Einheit erkennt, ob an VDD eine positive Spannung angeschlossen ist (wie in Bild 1). Wenn das nicht der Fall ist, erzeugt die Parasite Power Circuit aus der Datenleitung DQ die positive Spannungsversorgung, die durch einen eingebauten Kondensator gepuffert wird. Da dieser eingebaute Kondensator sehr klein ist, reicht diese Art der Spannungsversorgung allerdings nicht für alle Betriebsmodi des Sensors. Details dazu siehe Datenblatt des Sensors [1].

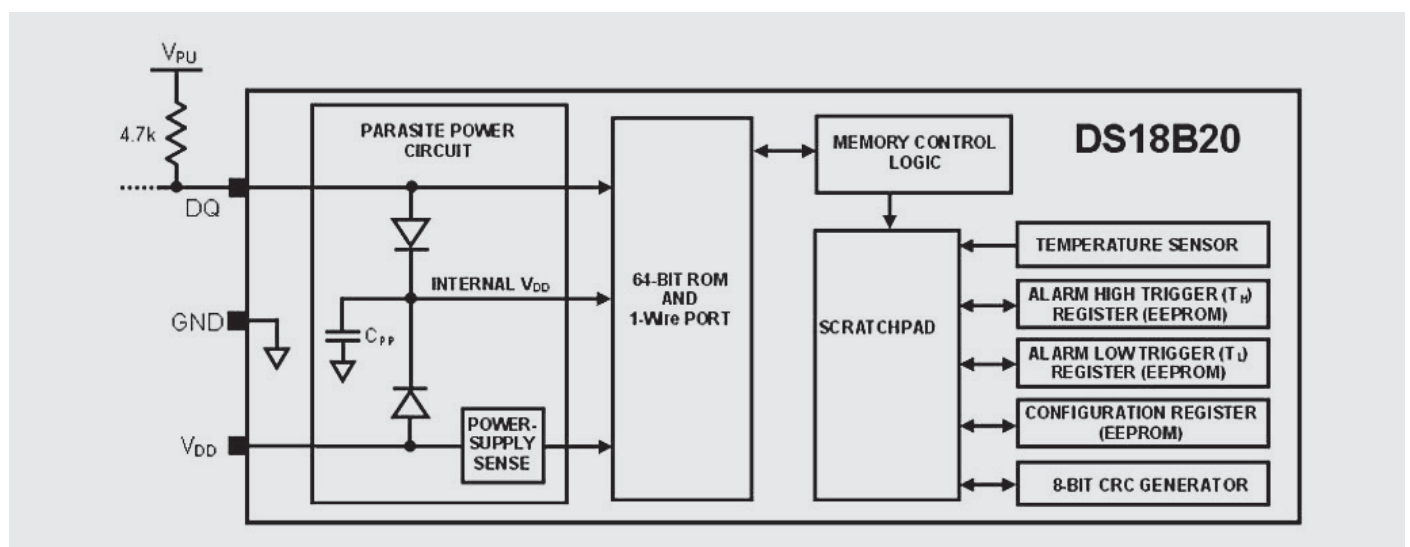


Bild 5: Aufbau des Temperatursensors DS18B20

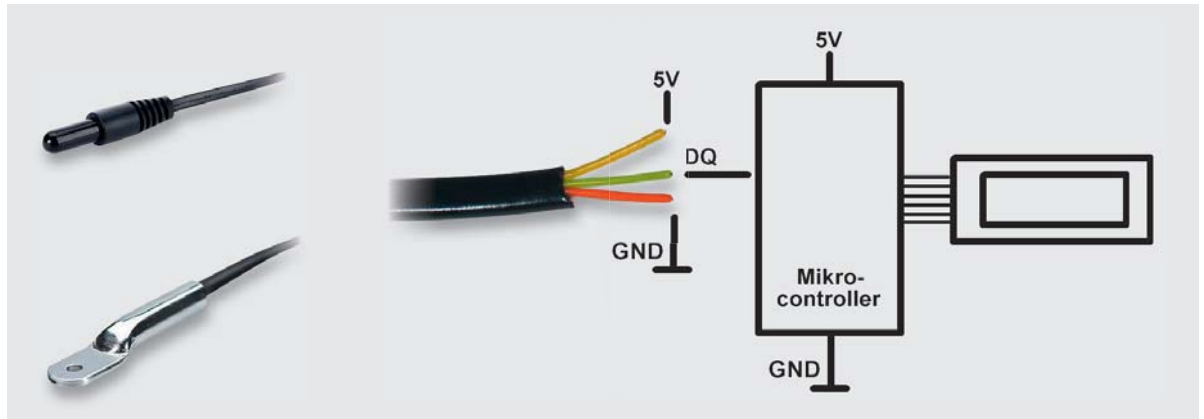


Bild 6: Anschluss des Sensors

Programmierung

Der Ablauf bei der 1-Wire-Programmierung mit BASCOM ist prinzipiell immer gleich. Zunächst wird der Pin definiert, der für 1-Wire verwendet werden soll:

0.) Konfiguration

Config 1wire = Portx.y ' Pin als 1-Wire-Pin definieren.

Im laufenden Programm wird der Bus jeweils mit 1wreset initiiert, dann folgt ein ROM-Kommando, mit dem festgelegt wird, welcher Baustein angesprochen werden soll, gefolgt von einem Funktionskommando, welches dem Compiler mitteilt, was getan werden soll.

1.) Initialisierung

1wreset Bus initialisieren/zurücksetzen.

2.) ROM-Command (gefolgt von benötigten weiteren Parametern)

Z. B. **1wwrite** &h55 für Match ROM oder **1wwrite** &hCC für Skip ROM
(Ggf. Adresse mit for i = 1 to 8 : 1wwrite Adresse(i) : next I)

Beispiele für ROM-Commands:

&hCC Skip ROM Alle Slaves/Sensoren ansprechen
&h55 Match ROM Einen bestimmten Slave/Sensor ansprechen

3.) Function-Command (gefolgt von benötigtem Datenaustausch)

Z. B. **1wwrite** &h44 für Messung anstoßen

Ggf. Bytes einlesen mit Daten(1) = **1wread**(9)

Beispiele für Function-Commands:

&h44 Convert Temperature Temperaturmessung anstoßen
&hBE Read Scratchpad Werte aus Sensor auslesen
&h4E Write Scratchpad Werte schreiben: Byte 2, 3 und 4 = TH, TL und Config

BASCOM stellt außerdem Befehle wie 1wsearchfirst(), 1wsearchnext() und 1wcount() zur Verfügung, mit denen die erste bzw. die nächste Bausteinadresse bzw. die Anzahl angeschlossener Slaves ermittelt werden kann.

1-Wire-ROM-Code-Scanner (ROM-Code lesen)

Wenn mehr als ein 1-Wire-Baustein am Bus angeschlossen werden soll oder wenn geprüft werden soll, ob ein bestimmter Baustein angeschlossen ist, muss man zunächst den eindeutigen ROM-Code des Bausteins ermitteln. Dafür kann man sich ein kleines BASCOM-Programm schreiben, welches als 1-Wire-ROM-Code-Scanner bezeichnet werden kann. Ein ELV-Temperatursensor wird gemäß Bild 6 am Mikrocontroller angeschlossen. Die grüne Ader kennzeichnet die Datenleitung DQ. Gelb bzw. Orange kennzeichnen die 5-V- bzw. GND-Leitung. Andere 1-Wire-Bausteine können natürlich ebenso benutzt werden.

```
' BASCOM-Programm
'
' 1-Wire-Scanner mit ATmega88
' Slaveadresse/ROM-Code EINES Sensors auslesen.
'
' In: PC.0: 1-Wire Wichtig: 4,7k Widerstand nach Plus
' Out: LCD an D2 bis D7
```

```
$regfile = „M88def.dat“
$crystal = 3686400
$hwstack = 40
$swstack = 40
$framesize = 60
```

```
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
```

```
ROM-Code: Fam:28
CF000004A373BD28
```

```
'Verwendeter Chip
'Verwendete Frequenz
'Rücksprungadressen (je 2), Registersicherungen (32)
'Parameteruebergaben (je 2), LOCALs (je 2)
'Parameter (Daten-Laenge), Rechenbereich Funktionen
```



Cursor Off
Cls

Config Iwire = Portc.0

Dim Rom_code(8) As Byte
Dim I As Byte

Lcd "ROM-Code-Leser"
Wait 3
Cls

```
'ROM-Codes der 1-Wire-Devices
' Am ersten Byte (LSB) des ROM-Codes (Family-Code)
' ist zu erkennen, um was für ein Gerät es sich handelt.
' Beim DS18B20 ist es z. B. &H28 und beim DS18S20 &H10.
' Da beide unterschiedliche Formate beim Temperaturwert haben,
' kann die Berechnung anhand des Family-Code angepasst werden.
```

```
Do
Rom_code(1) = Iwsearchfirst()
If Err = 0 Then
  Locate 1, 1 : Lcd "ROM-Code:"
  Lcd " Fam:" ; Hex(rom_code(1))
  Locate 2, 1
  For I = 8 To 1 Step -1
    Lcd Hex(rom_code(i));
  Next
Else
  Upperline : Lcd " Kein Slave! "
  Lowerline : Lcd Spc(16)
End If
Loop
End
```

'Pin fuer IWire festlegen

'Array für ROM-Code. LSB ist Family-Code

'ROM-Code vom ersten (und hier einzigen) Sensor lesen.
'Gibt es ein Gerät?

'Family-Code anzeigen und ..
' ... in zweiter Zeile ..
'... ROM-Code hexadezimal anzeigen.

'wenn es keinen Slave gibt ...

'16 Leerzeichen in 2. Zeile

Erläuterungen:

Mit Config Iwire = Portc.0 wird im Programm der Pin 0 des Ports C als Pin für 1-Wire festgelegt.

Im Anschluss daran wird ein 8 Byte langes Byte-Array Rom_Code(8) für die Aufnahme des ROM-Codes dimensioniert.

In der DO-LOOP wird mit Iwsearchfirst() der ROM-Code des ersten (und hier einzigen) angeschlossenen 1-Wire Bausteins gelesen. Danach steht in der Systemvariablen Err, ob das Lesen erfolgreich war (0 = 1-Wire-Baustein angeschlossen) oder nicht.

Wenn das Lesen des ROM-Codes erfolgreich war, wird der ROM-Code auf dem LC-Display ausgegeben. Diesen einzigartigen ROM-Code kann man sich nun notieren, um ihn später gezielt benutzen zu können.

Thermometer mit nur einem DS18B20-Sensor (Scratchpad lesen)

Ein DS18B20-Temperatursensor gibt den Temperaturwert mit einer Genauigkeit von 0,5 °C digital aus (vgl. „Elektronikwissen“). Im Sensor werden die Daten in einem Speicherbereich gespeichert, der als Scratchpad bezeichnet wird und aus 8-Byte-Registern besteht (Bild 7). In Register 0 und Register 1 wird der Temperaturwert gespeichert.

SCRATCHPAD	
Byte 0	Temperature LSB
Byte 1	Temperature MSB
Byte 2	T _H Register or User Byte 1
Byte 3	T _L Register or User Byte 2
Byte 4	Configuration Register
Byte 5	Reserved (FFh)
Byte 6	Reserved
Byte 7	Reserved (10h)
Byte 8	CRC

Bild 7: DS18B20-Memory-Map (Scratchpad)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2 ³	2 ²	2 ¹	2 ⁰	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2 ⁶	2 ⁵	2 ⁴

S = SIGN

Bild 8: Temperaturregister-Format



Bild 9: Temperaturwerte (Beispiele)

Der Temperaturwert ist in diesen beiden Registern als Zahl im Zweierkomplement gespeichert (Bild 8 und „Elektronikwissen“). Bit 11 bis 15 stellen das Vorzeichen der Zahl dar. In den anderen Bits steht der Wert der Zahl. Bild 9 zeigt Beispielwerte für die Temperatur und deren Darstellung in den beiden Temperaturregistern. Da Integerzahlen auch im AVR-Mikrocontroller im Zweierkomplement abgelegt werden, erfolgt die Umrechnung negativer Zahlen implizit.

```
' BASCOM-Programm
'
' 1-Wire-Thermometer mit ATmega88
' Thermometer mit EINEM DS18B20-Sensor
'
' In: PC.0: 1-Wire Wichtig: 4,7k Widerstand nach Plus
' Out: LCD an D2 bis D.7
```

```
$regfile = „M88def.dat“
$crystal = 3686400
$hwstack = 40
$swstack = 40
$framesize = 60
```

```
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Cursor Off
Cls
```

```
Config lwire = Portc.0
```

```
Dim Scratchpad(9) As Byte
Dim Wert_aus_ds18b20 As Integer
Dim Vorkomma As Integer
Dim Nachkomma As Byte
Dim Temperatur_single As Single
```

```
'Initialisierung (nur für DS18B20!):
lwreset
lwwrite &HCC
lwwrite &H4E
lwwrite &H00
```

```
'Verwendeter Chip
'Verwendete Frequenz
'Rücksprungadressen (je 2), Registersicherungen (32)
'Parameteruebergaben (je 2), LOCALs (je 2)
'Parameter (Daten-Laenge), Rechenbereich Funktionen
```

```
'Pin fuer lWire festlegen
```

```
'Temperaturregister zusammengefasst in einer Variablen
'Ganzzahlige Vorkomma-Temperatur
```

```
'z. B. 19.5
```

```
'skip rom = an alle
'Write Scratchpad
'T_H (Byte 2) schreiben
```

TEMPERATURE (°C)	DIGITAL OUTPUT
+125	0000 0111 1101 0000
+85	0000 0101 0101 0000
+25.0625	0000 0001 1001 0001
+10.125	0000 0000 1010 0010
+0.5	0000 0000 0000 1000
0	0000 0000 0000 0000
-0.5	1111 1111 1111 1000
-10.125	1111 1111 0101 1110
-25.0625	1111 1110 0110 1111
-55	1111 1100 1001 0000

Zweierkomplement

Das Zweierkomplement ist eine übliche Methode, positive und negative Binärzahlen in Mikrocontrollern abzubilden. Das höchstwertige Bit zeigt an, ob eine positive Zahl (0) oder eine negative Zahl (1) vorliegt.

Beispiel: &b00000001 entspricht +1 in Dezimalschreibweise. Das höchstwertige Bit ist 0.

Beispiel: Eine dezimale -1 wird im Zweierkomplement dargestellt als &b11111111. Das höchstwertige Bit ist eine 1. Die Zahl ist also negativ.

Eine mögliche manuelle Umwandlungsmethode von Binärzahl im Zweierkomplement zu Dezimalzahl interpretiert das höchstwertige Bit mit seiner Wertigkeit als negative Zahl und die restlichen Bits entsprechend ihrer Wertigkeiten jeweils positiv. Die Addition ergibt die dezimale Zahl. Am Beispiel &b11111111: $-128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = -1$

Genauigkeit und Auflösung

Genauigkeit gibt an, wie sehr ein angezeigter Wert dem wahren Wert entspricht. Wenn ein Thermometer auf 2 Grad genau anzeigt, dann können bei einer wahren Temperatur von z. B. 4 °C auch 2 Grad weniger oder mehr angezeigt werden. Wenn ein Temperatursensor eine Genauigkeit von 0,5 °C hat, entspricht die angezeigte Temperatur der wahren physikalischen Temperatur bis auf $\pm 0,5$ Grad. Auf die Genauigkeit kommt es an, wenn der angezeigte Wert möglichst gut der Realität entsprechen soll.

Auflösung gibt an, wie fein abgestuft eine analoge Größe digital dargestellt wird. Die Auflösung ist besonders dann wichtig, wenn auch kleine Veränderungen des Wertes relevant sind.

Beispiel: Ein Thermometer mit einer Genauigkeit von 2 Grad zeigt die Temperatur auch mit fünf Nachkommastellen nicht realitätsnäher an. Auch bei einer Anzeige von zum Beispiel 21,00123 kann die wahre Temperatur 2 Grad weniger oder mehr betragen.

Beispiel: Für einen Thermostaten ist es oft nicht relevant, dass die angezeigte Temperatur der wahren Temperatur entspricht (Genauigkeit), aber es sind sehr kleine Temperaturänderungen zu unterscheiden (Auflösung).

Gängige digitale Temperatursensoren haben oft eine Genauigkeit von 0,5 Grad und die Auflösung lässt sich konfigurieren.



```

lwwrite &H00
'lwwrite &B0001_1111
'lwwrite &B0011_1111
'lwwrite &B0101_1111
lwwrite &B0111_1111

Do
lwreset
lwwrite &HCC
lwwrite &H44
Wait 1

lwreset
lwwrite &HCC
lwwrite &HBE
Scratchpad(1) = lwread(9)

Locate 1 , 1
If Scratchpad(9) = Crc8(Scratchpad(1) , 8) Then
  Lcd "CRC ok "
  Test:
  Scratchpad(1) = &B01000010
  Scratchpad(2) = &B00000001
  Wert_aus_ds18b20 = Makeint(Scratchpad(1) , Scratchpad(2))
  Vorkomma = Wert_aus_ds18b20
  Shift Vorkomma , Right , 4 , Signed

  Nachkomma = Scratchpad(1) And &B00001111
  Temperatur_single = Nachkomma * 0.0625

  Temperatur_single = Vorkomma + Temperatur_single
  Locate 1 , 8 : Lcd Fusing(temperatur_single , "#.####" ;

Locate 2 , 1
Lcd Bin(wert_aus_ds18b20 )
Else
Lcd "CRC falsch"
Locate 2 , 1 : Lcd Spc(16)
End If

Loop
End

```

'T_L (Byte 3) schreiben
'Konfigurationsregister schreiben: 9-Bit-Auflösung
'Konfigurationsregister schreiben: 10-Bit-Auflösung
'Konfigurationsregister schreiben: 11-Bit-Auflösung
'Konfigurationsregister schreiben: 12-Bit-Auflösung (Default)

'skip rom = an alle
'Messvorgang anstoßen
'Wartezeit für Temperaturmessung!

'skip rom = an alle
'Auslesen des Scratchpads
'9 Bytes in das Array scratchpad ab Adresse scratchpad(1)

' Wenn crc korrekt..

'LSB Zum Testen.
'MSB Hier z. B. 20,1250 Grad
'Beide Temperaturregister (Byte) zu einer Integerzahl zusammenfassen
'Vorkommazahl ermitteln:
'Unter Erhaltung des Vorzeichens 4 nach rechts schieben = Nachkomma rausschieben

'Nachkommazahl ausmaskieren z. B. 8
'z. B. 8 * 0,0624 = 0.5

'Temperatur als Single = Vorkomma + Nachkomma
'{223}C" 'Temperatur anzeigen

'Temperaturregister (Byte 0 und Byte 1) binär anzeigen

'Zeile 2 leeren

Erläuterungen:

Mit Config 1-Wire wird zunächst wieder der Pin für 1-Wire definiert.

Dann wird ein 9-Byte-Array Scratchpad(9) dimensioniert, in das im Folgenden die Temperaturwerte aus dem Sensor eingelesen werden.

Die Initialisierung des Sensors erfolgt nach dem Standard-1-Wire-Schema:

```

lwreset          'Bus in Grundzustand versetzen
lwwrite &HCC     'ROM Command = skip rom = alle angeschlossenen Sensoren ansprechen.
lwwrite &H4E     'Function Command = Write Scratchpad = In den Speicher des Sensors schreiben.
lwwrite &H00     'T_H (Byte 2) schreiben. Hier nicht weiter benutzt, daher 0.
lwwrite &H00     'T_L (Byte 3) schreiben. Hier nicht weiter benutzt, daher 0.
lwwrite &B0111_1111 'Konfigurationsregister schreiben: 12-Bit-Auflösung (Default)

```

Danach werden alle angeschlossenen Sensoren (hier nur einer vorhanden) angewiesen, eine Temperaturmessung durchzuführen:

```

lwreset          'Bus in Grundzustand versetzen
lwwrite &HCC     'ROM Command = skip rom = alle angeschlossenen Sensoren ansprechen.
lwwrite &H44     'Function Command = Messvorgang anstoßen

```

Die Messung kann je nach gewählter Auflösung bis zu 0,75 Sekunden dauern. Mit einer Wartezeit von einer Sekunde (WAIT 1) ist man also sicher, dass der Sensor die Messung beendet hat.

Das eigentliche Auslesen des Temperaturwertes erfolgt mit:

```

lwreset          'Bus in Grundzustand versetzen
lwwrite &HCC     'ROM Command = skip rom = alle angeschlossenen Sensoren ansprechen.
lwwrite &HBE     'FUNCTION Command = Auslesen des Scratchpad-Speichers
Scratchpad(1) = lwread(9) '9 Bytes in das Array scratchpad ab Adresse scratchpad(1)

```

Danach erfolgt die Aufbereitung und die Anzeige des Temperaturwertes.

Thermometer mit mehreren Sensoren

In praktischen Projekten reicht natürlich oftmals nicht die Auswertung EINES Temperatursensors. Oft sollen mehrere Temperaturwerte gemessen und angezeigt bzw. weiterverarbeitet werden. Dafür lassen sich an einen einzigen 1-Wire-Bus viele Temperatursensoren anschließen (siehe Bild 1). Die Sensoren werden wie beschrieben durch die jeweils einzigartige ROM-Adresse unterschieden – siehe oben. Dadurch ist die Zuordnung von eingelesenem Temperaturwert zu Temperatursensor/Messstelle möglich. Im folgenden Beispiel werden die (durch den 1-Wire-Scanner ermittelten) ROM-Codes von drei Sensoren im Data-Bereich des BASCOM-Programms hinterlegt. Die Temperaturwerte der drei Sensoren werden nacheinander ermittelt und angezeigt.

```

' BASCOM-Programm
'
' 1-Wire-Thermometer mit ATmega88
' Thermometer mit mehreren DS18x20-Sensoren

```



Aussen: 16.5°C

```

'
' In: PC.0: 1-Wire Wichtig: 4,7k Widerstand nach Plus
' Out: LCD an D2 bis D.7

$regfile = „M88def.dat“
$crystal = 3686400
$hwstack = 40
$swstack = 40
$framesize = 60

Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Cursor Off
Cls

Const Timer1startwert = 7936
Config Timer1 = Timer , Prescale = 64
Timer1 = Timer1startwert
On Timer1 Sekunden_isr
Enable Timer1
Start Timer1
Enable Interrupts

Config lwire = Portc.0

Dim Rom_code(8) As Byte
Dim Scratchpad(9) As Byte
Dim I As Byte
Dim Wert_aus_ds18x20 As Integer
Dim Vorkomma As Integer
Dim Nachkomma As Byte
Dim Temperatur_single As Single
Dim Sensornummer As Byte
Dim Sensortext As String * 10
Dim Neuesekunde As Byte
Dim Ds18x20_anzeige_dauer As Byte
Dim Ds18x20_anzeige_restzeit As Byte : Ds18x20_anzeige_restzeit = 0

Restore Sensoren
Do
If Neuesekunde = 1 Then
    Neuesekunde = 0
    If Ds18x20_anzeige_restzeit = 0 Then
        Gosub Sensorbeschreibung_lesen
        If Sensornummer = 99 Then
            Restore Sensoren
            Gosub Sensorbeschreibung_lesen
        End If
        Ds18x20_anzeige_restzeit = Ds18x20_anzeige_dauer
    End If

    Locate 1 , 1
    Lcd Sensortext
    lwverify Rom_code(1)
    If Err = 0 Then
        lwwrite &HBE
        Scratchpad(1) = lwread(9)
        If Rom_code(1) = &H28 Then
            Gosub Temperatur_aus_ds18b20
        ElseIf Rom_code(1) = &H10 Then
            Gosub Temperatur_aus_ds18s20
        End If
        Locate 1 , 9 : Lcd Fusing(temperatur_single , "#.#") ; "{223}C "
    Else
        Locate 1 , 9 : Lcd "-----"
    End If
    'Temperaturmessung anstoßen:
    Gosub Temperaturmessung_starten
End If
' ...
'..hier ggf. weitere Anweisungen..
' ...
Loop
End

Sekunden_isr:
Timer1 = Timer1startwert
Neuesekunde = 1
If Ds18x20_anzeige_restzeit > 0 Then Decr Ds18x20_anzeige_restzeit
Return

Sensorbeschreibung_lesen:
Read Sensornummer
Read Ds18x20_anzeige_dauer
For I = 8 To 1 Step -1 : Read Rom_code(i) : Next
Read Sensortext
Return

Temperaturmessung_starten:
lwreset
lwwrite &HCC
lwwrite &H44
Return

```

```

'Verwendeter Chip
'Verwendete Frequenz
'Rücksprungadressen (je 2), Registersicherungen (32)
'Parameteruebergaben (je 2), LOCALs (je 2)
'Parameter (Daten-Laenge), Rechenbereich Funktionen

```

```

'1-Sekundentimer

```

```

'Pin fuer lWire festlegen

```

```

'Adresse des Sensors
'Datenregister des Sensors

```

```

'Temperaturregister zusammengefasst in einer Variablen
'Ganzzahlige Vorkomma-Temperatur
'Nachkomma-Temperatur
'z. B. 19.5

```

```

'Pointer auf Anfang der DATA-Tabelle mit Sensordaten

```

```

'Anzeigedauer für Sensor zu Ende
'nächste Sensorbeschreibung lesen
'wenn letzter Eintrag ...
'... dann von vorne
'... und erste Sensorbeschreibung lesen
'end if Sensornummer=99

```

```

'end if Ds18x20_anzeige_restzeit = 0

```

```

'Bestimmten Sensor ansprechen
'Wenn der Sensor vorhanden ist ...
'Auslesen des Scratchpads
'9 Bytes in das Array scratchpad ab Adresse scratchpad(1)
'Wenn DS18B20 ...

```

```

'Wenn DS1820/DS18S20 ...

```

```

'Temperatur anzeigen °C
'Wenn der Sensor nicht vorhanden ist ...

```

```

'Neue Temperaturmessung anstoßen
'end if neue Sekunde

```

```

'Jede Sekunde herunterzählen

```

```

'skip rom = an alle
'Neuen Messvorgang anstoßen

```




```

Temperatur_aus_ds18b20:
Wert_aus_ds18x20 = Makeint(scratchpad(1) , Scratchpad(2))
Vorkomma = Wert_aus_ds18x20
Shift Vorkomma , Right , 4 , Signed
Nachkomma = Scratchpad(1) And &B00001111
Temperatur_single = Nachkomma * 0.0625
Temperatur_single = Vorkomma + Temperatur_single
Return

```

```

'Bei Family-Code &28
'Beide Temperaturregister (Byte) zu einer Integerzahl zusammenfassen
'Vorkommazahl ermitteln:
'Unter Erhaltung des Vorzeichens 4 nach rechts schieben = Nachkomma rausschieben
'Nachkommazahl ausmaskieren z. B. 8
'z. B. 8 * 0,0624 = 0.5
'Temperatur als Single = Vorkomma + Nachkomma

```

```

Temperatur_aus_ds18s20:
Wert_aus_ds18x20 = Makeint(scratchpad(1) , Scratchpad(2))
Vorkomma = Wert_aus_ds18x20
Shift Vorkomma , Right , 1 , Signed
If Wert_aus_ds18x20.0 = 1 Then
    Temperatur_single = 0.5
Else
    Temperatur_single = 0.0
End If
Temperatur_single = Vorkomma + Temperatur_single
Return

```

```

'(LSB, MSB)
'Unter Erhaltung des Vorzeichens 1 nach rechts schieben = Nachkomma rausschieben
'Nachkomma
'Temperatur als Single = Vorkomma + Nachkomma

```

```

Sensoren:
' Sensor#, Anzeigedauer, ROM-ID, Text
Data 1 , 2 , &HCF , &H00 , &H00 , &H04 , &HA3 , &H73 , &HBD , &H28 , "Innen: "
Data 2 , 2 , &HB0 , &H00 , &H00 , &H04 , &HA3 , &H71 , &H90 , &H28 , "Aussen: "
Data 3 , 1 , &H79 , &H00 , &H08 , &H01 , &HCA , &HC7 , &H79 , &H10 , "Keller: "
Data 99 , 1 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , &H00 , "Ende"


```

Erläuterungen:

Durch einen Timer wird ein Ein-Sekunden-Takt erzeugt. Jede Sekunde wird in der Timer-ISR ‚Sekunden_isr‘ das Flag Neuesekunde auf 1 gesetzt. In der Hauptschleife wird dieses Flag regelmäßig abgefragt, so dass einmal pro Sekunde eine neue Temperaturanzeige erfolgen kann. In der Sekunden-ISR wird außerdem ein Zähler für die Anzeige-Restzeit für eine Messstelle heruntergezählt. Dadurch ist es möglich, im DATA-Bereich zu hinterlegen, wie viele Sekunden der Temperaturwert einer Messstelle jeweils angezeigt werden soll. Im DATA-Bereich steht die Anzeigedauer je Messstelle. Im Beispiel: zwei Sekunden lang Innentemperatur, zwei Sekunden Außentemperatur und eine Sekunde Kellertemperatur. Dieser Wert wird in der Sekunden-ISR heruntergezählt und wenn null erreicht ist, dann ist die entsprechende IF-Bedingung in der Hauptschleife erfüllt und die Daten für den nächsten Sensor werden mittels der Routine ‚Sensorbeschreibung_lesen‘ aus dem DATA-Bereich eingelesen.

Mit `1wverify Rom_Code(1)` wird der entsprechende Sensor gezielt angesprochen und dann wie bereits oben beschrieben der Temperaturwert dieses Sensors ausgelesen und angezeigt. Wichtig ist, dass auch hier den Sensoren genügend Zeit zum Durchführen der eigentlichen Temperaturmessung gegeben wird, indem am Ende der sekundlich stattfindenden Temperatúrauswertung eine neue Temperaturmessung durch alle angeschlossenen Sensoren angestoßen wird. Bis zur nächsten „Runde“ bleibt den Sensoren dadurch Zeit, die Temperatur zu messen.

Ausblick

Nach der Beschreibung der komfortablen 1-Wire-Schnittstelle in diesem Teil wird im nächsten Teil der Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR“ die SPI-Schnittstelle behandelt. 



Weitere Infos:

- [1] Datenblatt DS18B20: Unter www.elv.de bei der Artikelbeschreibung hinterlegt. Geben Sie dazu bitte einfach die Best.-Nr. J7-10 93 37 im Suchfeld ein.
- [2] DS18B20 vs. DS18S20/DS1820: www.maximintegrated.com/en/app-notes/index.mvp/id/4377
 - Stefan Hoffmann: Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM. Systematische Einführung und Nachschlagewerk mit vielen Anregungen. ISBN 978-3-8391-8430-1
 - www.bascom-buch.de
 - www.mcselec.com
 - www.atmel.com
 - Produktübersicht BASCOM: www.elv.de/bascom.html

Empfohlene Produkte/Bauteile:

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics, www.mcselec.com	–	–
ATmega88	J7-10 07 62	€ 3,95
BASCOM-Buch	J7-10 90 02	€ 54,-
H-Tronic TS 1 Temperatursensor DS18B20	J7-10 93 37	€ 12,95
H-Tronic TS 2 Temperatursensor DS18B20	J7-10 27 83	€ 14,95
Kupplung für RJ45-Stecker	J7-10 65 34	€ 0,95
ISDN-Kabel mit RJ45-Steckern	J7-10 65 35	€ 5,95
Terminalblock-Adapter mit Drucktasten	J7-11 52 88	€ 5,95
Modular-Einbaubuchse 2x RJ45	J7-11 21 47	€ 1,09

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Preisstellung November 2014 – aktuelle Preise im Web-Shop