Projekt Wetterstation, Designmanual

Inhalt

[1. Überblick 3](#_Toc170022020)

[2. Verwendete Bauteile 4](#_Toc170022021)

[2.1. Zentrale Steuereinheit: Conrad C-Control ATmega32 Experimentierplatine 4](#_Toc170022022)

[2.2. Datenspeicher: STMicroelectronics M24C64 5](#_Toc170022023)

[2.3. Feuchtigkeits- und Temperatursensor: SENSIRION SHT71 6](#_Toc170022024)

[2.4. Lichtintensitätssensor: LDR 10](#_Toc170022025)

[2.5. Bustreiber: SGS-Thomson Microelectronics SH74HC125 10](#_Toc170022026)

[3. Hardwarelayout 12](#_Toc170022027)

[4. Kompletter C-Control Code: (Hardcopy) 13](#_Toc170022028)

[5. Kommentierter C-Control Code: 28](#_Toc170022029)

[5.1. Subprogramm zur Kommunikation mit SHT71 zur Feuchtemessung: 28](#_Toc170022030)

[5.2. Subprogramm zur Kommunikation mit SHT71 zur Temperaturmessung: 31](#_Toc170022031)

[5.3. Subprozedur portschreiben(): 33](#_Toc170022032)

[5.4. Auslesen des Wertes des Lichtsensors: 33](#_Toc170022033)

[5.5. Umrechnen der erhaltenen Daten auf brauchbare Umweltwerte: 34](#_Toc170022034)

[5.6. Schreiben der Daten in den Speicher 35](#_Toc170022035)

[5.7. Auslesen der Daten aus dem Speicher: 37](#_Toc170022036)

Dieckmann, Saulder

# Überblick

Diese Wetterstation hat die Aufgabe eigenständig Daten aufzunehmen und in einem externen Speicher abzulegen. In, vom Benutzer gewählten Abständen, misst die Station Umgebungsparameter wie Lufttemperatur, Luftfeuchtigkeit, Lichtintensität, Niederschlagsmenge und Windrichtung.

Zu einem beliebigen Zeitpunkt können dann die, über einen längeren Zeitraum gemessenen Daten, per USB oder serieller Schnittstelle aus dem Speicher abgerufen werden.

* Als zentrale Steuereinheit wird ein ATmega32 Chip aus der AVR-Reihe von ATMEL verwendet. Dieser bietet eine beachtliche Rechenkapazität mit 16 MIPS (verglichen mit 54MIPS eines Intel 486DX) und verfügt über einen 8kanäligen 10-bit ADC.
* Der Programmspeicher der Wetterstation ist als Flash-Speicher ausgelegt, er hält seine Daten selbst bei Zusammenbruch der Spannungsversorgung, ein Neubespielen ist im Normalfall nie nötig.
* Der Datenspeicher ist als EEPROM ausgelegt, auch sein Inhalt übersteht damit Spannungszusammenbrüche, ein Datenverlust ist damit unter normalen Betriebsbedingungen praktisch ausgeschlossen.

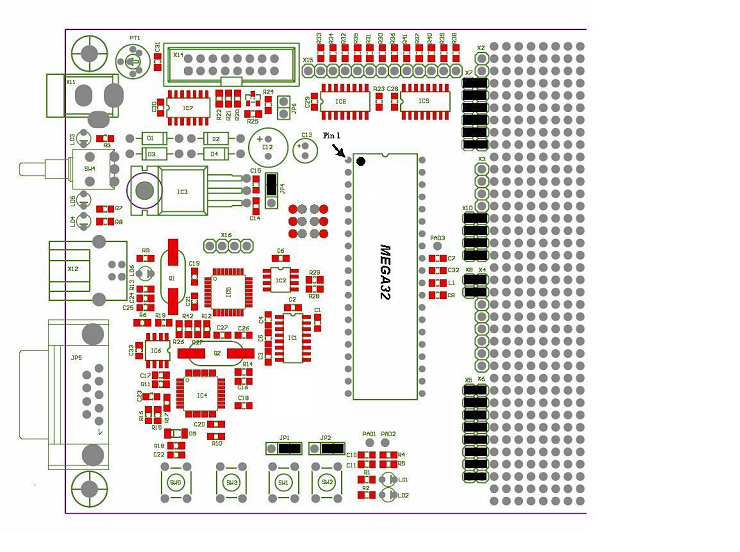
* Zur Aufnahme der Temperatur- und Feuchtigkeitswerte ist ein Präzisionssensor der Firma Sensirion mit 14- bzw. 12-bit Auflösung verwendet welcher zusätzlich durch seine geringen Ausmaße (5mm x 2cm incl. Pins) und seine geringe Stromaufnahme (0,3μA im Standby und 550μA während der Messung) als ideales Bauelement für diesen Einsatz bestimmt wurde.
* Als Lichtsensor wird ein lichtabhängiger Widerstand verwendet. Dieser zeichnet sich durch seine robuste Ausführung, die kostengünstige Anschaffung und leichte Verfügbarkeit, bei ausreichender Genauigkeit aus. Da die Messungen typischerweise höchstens im Sekundentakt oder gar im Stundentakt erfolgen sind hier keine dynamischen Bauteile nötig, die Trägheit des LDR fällt nicht ins Gewicht.

# Verwendete Bauteile

## Zentrale Steuereinheit: Conrad C-Control ATmega32 Experimentierplatine

  
*Abb. 1: ATmega32 mit Conrad-Metallkühlkörper*

Rund um den leistungsstarken ATmega32 Chip wurde vom Elektronikdistributor Conrad eine Experimentierplatine gefertigt welche über Anschlüsse für eine Folientastatur, ein 2zeiliges LCD, sowie über eine USB Schnittstelle und eine serielle Schnittstelle verfügt. Zusätzlich wird der I2C Standard unterstützt.

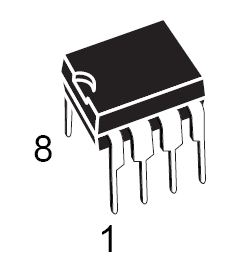
  
*Abb. 2: Conrad Experimentierplantine C-Control Pro Mega 32*

I2C DATA

I2C CLK

Pin 1 ADC0

## Datenspeicher: STMicroelectronics M24C64

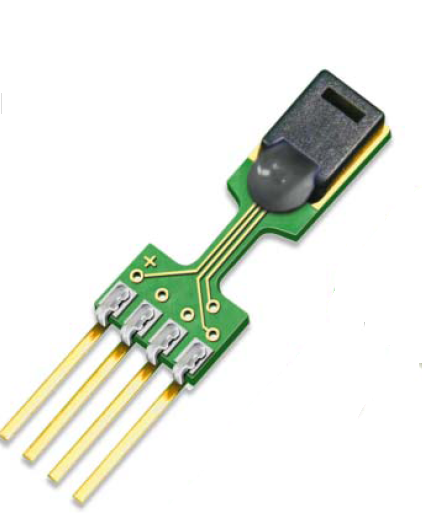
  
*Abb. 3: M24C64 Speicherbaustein*

Dieser 64kbit EEPROM unterstützt das I2C Format (bis zu 400kHz) und ist für eine Versorgungsspannung von 4,5 V bis 5,5V ausgelegt. Er kann Daten bis zu 40 Jahren halten und über eine Million Mal neu beschrieben werden. Das Schreibverfahren kann dabei byteweise oder im PAGE-Modus erfolgen. Gelesen kann sowohl mit Random Access als auch im sequentiellen Modus werden.

|  |
| --- |
| Adressierung: |
| Die ersten 4 bit seiner I2C Adresse sind festgelegt, sie lauten 1010; mit 3 seiner Pins kann man die nächsten 3 bit wählen. Setzt man sie auf high entspricht dies einem 1. |
| E:\Dokumente und Einstellungen\p\Desktop\Elektronik 2\Projekt\m24belegung.bmp  Adresspins: Legt man alle auf +5V so hat der Chip die Adresse 1010111;  sind alle auf 0V hingegen 1010000 |

|  |
| --- |
| Beschaltung: |
| VCC ist mit 5V zu versorgen und VSS gegen Masse zu legen.  Die Pins SCL und SDA sind der I2C Clock und I2C Daten Kanal. Der Pin WC kann als Write Control verwendet werden. Er ist in dieser Wetterstation immer gegen Masse geschaltet, so ist der Speicher immer bereit beschrieben zu werden. |

## Feuchtigkeits- und Temperatursensor: SENSIRION SHT71



*Abb. 4: SENSIRION SHT71 Feuchtigkeits- &Temperatursensor*

Der SHT71 vereint einen Feuchtigkeits- und Temperatursensor gemeinsam mit einem Analog-Digital-Wandler und einem Timing-Logikschaltkreis auf einem Chip. Er verfügt über 4 Pins wobei 2 zur Spannungsversorgung dienen (+5V & 0V) und die anderen 2 zum Datentransfer dienen. Er verwendet ein dem I2C Protokoll verwandtes System zum Übertragen von Daten jedoch kann er nicht an I2C Ports angeschlossen werden sondern muss händisch ausgelesen werden. Er verfügt über eine große Genauigkeit von 0,03% relativer Luftfeuchtigkeit und 0.01K bei der Temperaturmessung.

|  |  |
| --- | --- |
| *Beschaltung:* | |
| *Abb. 5: Abmessung(in mm) und Pinbelegung des SHT71* | Pin1: Serieller Clock  Pin2: Versorgungsspannung  Pin3: Masse  Pin4: Serielle Daten    Pin1 und Pin4 sollten (zusätzlich zur Verbindung mit dem Mikrokontroller) mit einem Pullup Widerstand an die Versorgungsspannung angeschlossen werden. |

|  |
| --- |
| Timing Diagramm: |
| *E:\Dokumente und Einstellungen\p\Desktop\Elektronik 2\Projekt\timing.bmp*  *Abb. 6: Beispiel eines Timingdiagrams zum Auslesen des Luftfeuchtigkeitswertes*  Die SCK (Clock) Leitung wird allein vom Mikrokontroller (ATmega32) gesteuert; über einen Digitalausgangspin werden hier nach Bedarf 0V und 5V angelegt.  Die DATA Leitung hingegen ist bidirektional. Fett gezogene Linien symbolisieren Daten vom SHT71, wohingegen dünne Linien Daten vom Mikrokontroller darstellen.   1. Der erste Schritt um eine Kommunikation zwischen dem Mikrokontroller und dem Sensor zu starten besteht in der Zeichenabfolge „Transmission Start“:  Dazu muss bei SCK=1 DATA von 1 auf 0 fallen, anschließend muss SCK=0 und wieder =1 gesetzt werden, wonach DATA wieder =1 gesetzt werden muss. Durch diese Codeabfolge signalisiert der Mikrokontroller dem Sensor aus dem „sleep state“ in dem er durchschnittlich nur 0,3μA aufnimmt, aufzuwachen. 2. Als nächstes folgt die Adressierung des Sensors.  Um eine Feuchtigkeitsmessung durchzuführen sendet man: 00000101 Um eine Temperaturmessung durchzuführen sendet man: 00000011 Der Sensor bestätigt dies durch ein Nullsetzen des Datenkanals. 3. Nun beginnt der Sensor mit der Messung. Dies kann bis zu 242ms (bei 14bit Genauigkeit) dauern. 4. Ist die Messung beendet zieht der Sensor die Datenleitung auf 0V. 5. Nun folgen die Daten bitweise, mit jedem Clockimpuls des Mikrokontrollers sendet der Sensor ein neues Datenbit. Das Most Significant Bit wird als erstes gesendet. Nach jeweils 8bit muss der Kontroller den Empfang des Bytes durch Nullsetzen der Datenleitung bestätigen. 6. Am Ende der Datenübertragung kann durch abermaliges Nullsetzen der Datenleitung durch den Mikrokontroller noch eine Checksumme angefordert werden um die Integrität der empfangenen Daten zu überprüfen. |

|  |
| --- |
| *Auswertung der Daten:* |
| Um den Sensorwert für Luftfeuchte und Temperatur in ein brauchbares Format umzuwandeln sind folgende Formeln zu verwenden:  Temperatur:    Formel 1, Tabelle 1&2: *Konversionsformel der Temperatur und Konversionskoeffizienten*  Wobei SOT für den vom Sensor gelieferten Dezimalwert der Temperatur steht.    Luftfeuchte:  E:\Dokumente und Einstellungen\p\Desktop\Elektronik 2\Projekt\rh.bmp  *Formel 2, Tabelle 3: Konversionsformel der Luftfeuchtigkeit und Konversionskoeffizienten*  Wobei SORH für den Sensorwert als Dezimalzahl steht. Und Trilinear für den Luftfeuchtigkeitswert in % relativer Luftfeuchte.    Korrekturfaktor zur Luftfeuchte:    *Formel3, Tabelle4: Korrekturformel zur Verbesserung des Wertes der Luftfeuchtigkeit*  Unterscheidet sich die Temperatur stark von 25°C so hat dies auch einen Einfluss auf den gemessenen Wert der Luftfeuchtigkeit. Um diese Abhängigkeit rückzurechnen wird diese Formel verwendet. |
| Technische Daten*:* |

*Elektrische Daten:*

*Versorgungsspannung: 2,4V bis 5,5V DC  
Stromaufnahme Sleepmode typisch: 0,3µA  
Stromaufnahme Messbetrieb typisch: 550µA  
Maximaler Ausgangsstrom: 4mA*

*Kommunikationsdaten:*

*Definition des low-Zustandes für hereinkommende Daten: 0-20% Versorgungsspannung  
Definition des high-Zustandes für hereinkommende Daten: 80-100% Versorgungsspannung  
Definition des low-Zustandes für ausgehende Daten: 0-20% Versorgungsspannung  
Definition des high-Zustandes für hereinkommende Daten: 75-100% Versorgungsspannung*

*Thermische Daten:*

*Selbstaufheizung: Unter 0,1°C bei weniger als 2Messungen/s mit 12bit Genauigkeit*

*Sensordaten (Luftfeuchte/Temperatur):*

*Auflösung: 0,03% rel. Luftfeuchte / 0,01°C  
Unsicherheit: <±2%rel. Luftfeuchte (bei 10%-90% rel. Luftfeuchte) / <±2K (bei -40°C bis 120°C)  
Messbereich: 0%-100% rel. Luftfeuchte / -40°C bis 123,8°C*

## Lichtintensitätssensor: LDR



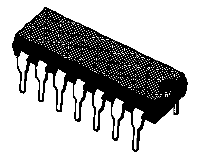
Abb. 7: Gehäuselayout des LDR

Fotowiderstände ändern je nach Lichteinfall ihren Widerstand. Bei unserem Modell variiert dies zwischen 50Ω bei voller Sonneneinstrahlung (≈1000W/m²) und zig MΩ bei Dunkelheit.  
Auf Änderungen der Lichtintensität reagiert er verhältnismäßig langsam, Photodioden oder eigens konstruierte Lichtsensoren agieren hier viel schneller, was jedoch für die Anwendung in einer Wetterstation die eventuell nur alle paar Stunden Daten aufnimmt unnötig ist.

Für den Widerstand gilt:

Wobei const1 und const2 Materialkonstanten sind und I die Beleuchtungsstärke in lux ist.

## Bustreiber: SGS-Thomson Microelectronics SH74HC125

**

*Abb. 8: Gehäuse des SH74HC125*

Der 75HC125 ist ein Hochgeschwindigkeitsbustreiber welcher es ermöglicht einzelne Leitungen durchzuschalten oder zu unterbrechen. In der Wetterstation wird er genutzt um den bidirektionalen Datenkanal der SENSIRION Feuchtigkeits- & Temperatursensors zu verwalten.

|  |  |
| --- | --- |
| *Beschaltung:* | |
| *Abb. 9 : Beschaltungsplan des Bustreibers*  0V  5V | *A1: Eingang für Signal*  *Y1: Ausgang für Signal*  *G1: Outputenable (auf 0V gesetzt werden A1 und Y1 verbunden)  Vcc: Spannungs- versorgung  GND: Masse* |

|  |
| --- |
| Technische Daten*:* |

Elektrische Daten:

Versorgungsspannung (empfohlen): 2V bis 6V  
Leistungsaufnahme: 500mW  
Signalspannung(empfohlen) : 0V bis Versorgungsspannung

Thermische Daten:

Einsatztemperatur (empfohlen): -40°C bis +85°C  
Lagertemperatur: -65°C bis 150°C

# Hardwarelayout

**10kΩ**

**10kΩ**

1

**7V**

ADC 0

**Conrad ATmega32**

**LDR**

**5V**

DA

CL

WC

Vcc

Vss

**M24C64  
Flash  
Speicher**

E2

E1

E0

I2C Daten

I2C Clock

17

16

**5V**

**11kΩ**

**Pullup**

**11kΩ**

Clockkanal für SHT71

Dateneingang von SHT71

Outputenable für Bustreiber  
*(Port21 auf low →  
Bustreiber schaltet durch)*

Datenausgang für SHT71

Y1

G1

A1

**SH74HC125  
Bustreiber**

**5V**

21

20

19

18

4

3

2

1

**SENSIRION  
SHT71  
Feuchtigkeits- &  
Temperatursensor**

# Kompletter C-Control Code: (Hardcopy)

int zeit;

int zz;

int messint;

word address;

int feucht[12];

int temp[14];

int werttemp;

int wertfeucht;

int wertlicht;

float wtemp;

float wfeucht;

float wzwischen;

float wlicht;

byte jalesen;

void main(void)

{

zeit=0;

zz=0;

messint=0;

I2C\_Init(I2C\_100kHz);

ADC\_Set(ADC\_VREF\_VCC,0);

Setupstation();

address=0x10;

feuchte();

temperatur();

licht();

Umrechnung();

schreibenstart();

schreiben(messint);

schreiben(wertfeucht);

schreiben(werttemp);

schreiben(wertlicht);

schreibenstop(8);

Irq\_SetVect(INT\_TIM2COMP,Befehl);

while(true);

}

void Umrechnung(void)

{

wtemp=werttemp;

wzwischen=wertfeucht;

wlicht=wertlicht;

wtemp=(wtemp\*0.01)-40.0;

wfeucht=-(wzwischen\*wzwischen\*2.8\*0.001\*0.001);

wfeucht=wfeucht+(wzwischen\*0.0405)-4.0;

wfeucht=(wtemp-25.0)\*(0.01+0.00008\*wzwischen)+wfeucht;

werttemp=wtemp\*100.0;

wertfeucht=wfeucht\*100.0;

wlicht=wlicht\*5/1024;

wlicht=wlicht/7;

wlicht=(((1-wlicht)/(wlicht\*10))-(1/24));

wlicht=wlicht\*600;

wertlicht=wlicht;

Msg\_WriteInt(wertfeucht);

Msg\_WriteHex(43690);

Msg\_WriteInt(werttemp);

Msg\_WriteHex(43690);

Msg\_WriteInt(wertlicht);

Msg\_WriteHex(43690);

}

void Befehl(void)

{

 int irqcnt;

if (zz==100\*messint)

{

zeit=zeit+1;

zz=0;

feuchte();

temperatur();

licht();

Umrechnung();

schreibenstart();

schreiben(wertfeucht);

schreiben(werttemp);

schreiben(wertlicht);

schreibenstop(6);

}

else

{

zz=zz+1;

}

irqcnt=Irq\_GetCount(INT\_TIM2COMP);

}

void Clock(void)

{

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

AbsDelay(1);

}

void portschreiben(int y)

{

Port\_WriteBit(21,0);

Port\_WriteBit(20,y);

}

void licht(void)

{

wertlicht=ADC\_Read();

}

void feuchte(void)

{

int checken;

int messen;

int i;

Port\_DataDirBit(18,1);

Port\_DataDirBit(19,0);

Port\_DataDirBit(20,1);

Port\_DataDirBit(21,1);

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(0);

AbsDelay(1);

Port\_WriteBit(18,0); //Transmission Start

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,0);

AbsDelay(1);

portschreiben(0);

AbsDelay(1);

Clock();

Clock();

Clock();

Clock();

Clock();

portschreiben(1);

AbsDelay(1);

Clock(); //Code für Feuchtigkeit 00000101

portschreiben(0);

AbsDelay(1);

Clock();

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19);

if (messen==0)

{ //Ack.

Clock();

checken=0;

}

}

AbsDelay(1);

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19); //Warten für Messung

AbsDelay(1);

if (messen==0)

{

checken=0;

}

}

Clock();

Clock();

Clock();

Clock();

for(i=0;i<4;i++)

{

messen=Port\_ReadBit(19); //erste 4Bit

AbsDelay(1);

feucht[i]=messen;

Clock();

}

AbsDelay(1);

portschreiben(0); //Ack.

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

for(i=0;i<8;i++)

{

messen=Port\_ReadBit(19); //nächte 8 bit

AbsDelay(1);

feucht[i+4]=messen;

Clock();

}

AbsDelay(1);

portschreiben(0);

AbsDelay(1); // Verbindung trennen

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

wertfeucht=0;

for (i=0;i<12;i++)

{

//Msg\_WriteInt(feucht[i]);

wertfeucht=wertfeucht+(feucht[i]<<(11-i));

}

}

void temperatur(void)

{

int checken;

int messen;

int i;

Port\_DataDirBit(18,1);

Port\_DataDirBit(19,0);

Port\_DataDirBit(20,1);

Port\_DataDirBit(21,1);

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(0);

AbsDelay(1);

Port\_WriteBit(18,0); //Transmission Start

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,0);

AbsDelay(1);

portschreiben(0);

AbsDelay(1);

Clock();

Clock();

Clock();

Clock();

Clock();

Clock();

portschreiben(1);

AbsDelay(1);

Clock(); //Code für Temp 00000011

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19);

if (messen==0)

{ //Ack.

Clock();

checken=0;

}

}

AbsDelay(1);

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19); //Warten für Messung

AbsDelay(1);

if (messen==0)

{

checken=0;

}

}

Clock();

Clock();

for(i=0;i<6;i++)

{

messen=Port\_ReadBit(19); //erste 6Bit

AbsDelay(1);

temp[i]=messen;

Clock();

}

AbsDelay(1);

portschreiben(0); //Ack.

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

for(i=0;i<8;i++)

{

messen=Port\_ReadBit(19); //nächte 8 bit

AbsDelay(1);

temp[i+6]=messen;

Clock();

}

AbsDelay(1);

portschreiben(0);

AbsDelay(1); // Verbindung trennen

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

werttemp=0;

for (i=0;i<14;i++)

{

//Msg\_WriteInt(temp[i]);

werttemp=werttemp+(temp[i]<<(13-i));

}

}

void Setupstation(void)

{

byte fertig;

word key\_in;

byte key\_ch;

int pos;

LCD\_Init();

Key\_Init();

LCD\_ClearLCD();

LCD\_CursorOff();

LCD\_CursorPos(0);

LCD\_WriteChar('g');

LCD\_CursorPos(1);

LCD\_WriteChar('e');

LCD\_CursorPos(2);

LCD\_WriteChar('t');

LCD\_CursorPos(3);

LCD\_WriteChar(' ');

LCD\_CursorPos(4);

LCD\_WriteChar('d');

LCD\_CursorPos(5);

LCD\_WriteChar('a');

LCD\_CursorPos(6);

LCD\_WriteChar('t');

LCD\_CursorPos(7);

LCD\_WriteChar('a');

LCD\_CursorPos(0x40);

LCD\_WriteChar('1');

LCD\_CursorPos(0x41);

LCD\_WriteChar('=');

LCD\_CursorPos(0x42);

LCD\_WriteChar('y');

LCD\_CursorPos(0x43);

LCD\_WriteChar('e');

LCD\_CursorPos(0x44);

LCD\_WriteChar('s');

fertig=0;

jalesen=0;

while (fertig==0)

{

key\_in=Key\_Scan();

if (key\_in!=0)

{

key\_ch=Key\_TranslateKey(key\_in);

LCD\_CursorPos(12);

LCD\_WriteChar(key\_ch);

key\_ch=key\_ch-48;

if (key\_ch==1)

{

jalesen=1;

fertig=1;

}

else

{

fertig=1;

}

}

}

if (jalesen==1)

{

LCD\_ClearLCD();

LCD\_CursorOff();

LCD\_CursorPos(0);

LCD\_WriteChar('2');

LCD\_CursorPos(2);

LCD\_WriteChar('=');

LCD\_CursorPos(4);

LCD\_WriteChar('U');

LCD\_CursorPos(5);

LCD\_WriteChar('S');

LCD\_CursorPos(6);

LCD\_WriteChar('B');

LCD\_CursorPos(0x40);

LCD\_WriteChar('3');

LCD\_CursorPos(0x42);

LCD\_WriteChar('=');

LCD\_CursorPos(0x44);

LCD\_WriteChar('C');

LCD\_CursorPos(0x45);

LCD\_WriteChar('O');

LCD\_CursorPos(0x46);

LCD\_WriteChar('M');

fertig=0;

while (fertig==0)

{

key\_in=Key\_Scan();

if (key\_in!=0)

{

key\_ch=Key\_TranslateKey(key\_in);

LCD\_CursorPos(12);

LCD\_WriteChar(key\_ch);

key\_ch=key\_ch-48;

if (key\_ch==2)

{

speicherholen(1);

fertig=1;

}

else

{

if (key\_ch==3)

{

speicherholen(2);

fertig=1;

}

}

}

}

}

fertig=0;

LCD\_ClearLCD();

LCD\_CursorOff();

LCD\_CursorPos(0);

LCD\_WriteChar('M');

LCD\_CursorPos(1);

LCD\_WriteChar('e');

LCD\_CursorPos(2);

LCD\_WriteChar('s');

LCD\_CursorPos(3);

LCD\_WriteChar('s');

LCD\_CursorPos(4);

LCD\_WriteChar('i');

LCD\_CursorPos(5);

LCD\_WriteChar('n');

LCD\_CursorPos(6);

LCD\_WriteChar('t');

LCD\_CursorPos(7);

LCD\_WriteChar('.');

LCD\_CursorPos(0x47);

LCD\_WriteChar('s');

pos=0;

while (fertig==0)

{

AbsDelay(500);

key\_in=Key\_Scan();

if (key\_in!=0)

{

key\_ch=Key\_TranslateKey(key\_in);

LCD\_CursorPos(12);

LCD\_WriteChar(key\_ch);

if (key\_ch!=35)

{

pos=pos+1;

key\_ch=key\_ch-48;

LCD\_CursorPos(0x40+pos);

LCD\_WriteChar('0'+key\_ch);

messint=messint\*hoch(pos-1);

messint=messint+key\_ch;

}

else

{

fertig=1;

}

}

}

LCD\_ClearLCD();

LCD\_CursorOff();

LCD\_CursorPos(0);

LCD\_WriteChar('S');

LCD\_CursorPos(1);

LCD\_WriteChar('t');

LCD\_CursorPos(2);

LCD\_WriteChar('a');

LCD\_CursorPos(3);

LCD\_WriteChar('r');

LCD\_CursorPos(4);

LCD\_WriteChar('t');

LCD\_CursorPos(5);

LCD\_WriteChar('e');

LCD\_CursorPos(6);

LCD\_WriteChar('n');

LCD\_CursorPos(0x40);

LCD\_WriteChar('1');

LCD\_CursorPos(0x41);

LCD\_WriteChar('=');

LCD\_CursorPos(0x42);

LCD\_WriteChar('l');

LCD\_CursorPos(0x43);

LCD\_WriteChar('o');

LCD\_CursorPos(0x44);

LCD\_WriteChar('s');

fertig=0;

while (fertig==0)

{

key\_in=Key\_Scan();

if (key\_in!=0)

{

key\_ch=Key\_TranslateKey(key\_in);

LCD\_CursorPos(12);

LCD\_WriteChar(key\_ch);

key\_ch=key\_ch-48;

if (key\_ch==1)

{

fertig=1;

LCD\_ClearLCD();

LCD\_CursorOff();

}

}

}

}

int hoch(int x)

{

int i;

int a;

a=1;

for(i=1;i<x+1;i++)

{

a=a\*10;

}

return a;

}

byte ausgabe1;

byte ausgabe2;

int anzeige;

void speicherholen(byte ziel)

{

if (ziel==2)

{

Serial\_Init(0,SR\_8BIT|SR\_1STOP|SR\_NO\_PAR,SR\_BD19200);

}

address=0x10;

I2C\_Start();

   I2C\_Write(0xA0);

I2C\_Write(address>>8);

I2C\_Write(address);

I2C\_Start();

I2C\_Write(0xA1);

do   {

  ausgabe1=I2C\_Read\_ACK();

ausgabe2=I2C\_Read\_ACK();

anzeige=ausgabe1<<8;

anzeige=anzeige+ausgabe2;

if (ziel==1)

{

if (anzeige!=0)

{

Msg\_WriteInt(anzeige);

Msg\_WriteHex(43690);

}

}

else

{

Serial\_Write(0,ausgabe1);

Serial\_Write(0,ausgabe2);

}

}

while (anzeige!=0);

  ausgabe1=I2C\_Read\_NACK();

ausgabe2=I2C\_Read\_NACK();

   I2C\_Stop();

if (ziel==2)

{

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Disable(0);

}

}

void schreibenstart(void)

{

I2C\_Start();

I2C\_Write(0xA0);

I2C\_Write(address>>8);

I2C\_Write(address);

}

void schreiben(int data)

{

byte a;

a=data>>8;

I2C\_Write(a);

a=data;

I2C\_Write(a);

}

void schreibenstop(int bytezahl)

{

byte a;

a=0;

I2C\_Write(a);

I2C\_Write(a);

I2C\_Stop();

address=address+bytezahl;   
}

# Kommentierter C-Control Code:

## Subprogramm zur Kommunikation mit SHT71 zur Feuchtemessung:

void feuchte(void)

{

int checken;

int messen;

int i;

// Port\_DataDirBit bestimmt die Richtung der Kommunikation am jeweiligen Pin (hier 18,19,20,21). Mittels dem Wert 1 wird der Port zum Schreibeport und mittels 0 zum Leseport deklariert.

Port\_DataDirBit(18,1);

Port\_DataDirBit(19,0);

Port\_DataDirBit(20,1);

Port\_DataDirBit(21,1);

//portschreiben ist eine weiter Subprozedur (siehe 5.3.), sie legt in diesem Fall den Wert 1(=5V) an den Pin 20 und setzt gleichzeitig auch Port 21 auf 0V, so wird der Bustreiber durchgeschaltet

portschreiben(1);

//Die Transmission Start Prozedur. Sie ist nötig um die Kommunikation mit dem Sensor zu beginnen. (Siehe Timingdiagramm auf Seite 6). Zur Erinnerung: Port18 ist der Clockport, Port20 der Data-Schreibeport (wird hier über die Subfunktion portschreiben() angesprochen)

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(0);

AbsDelay(1);

Port\_WriteBit(18,0); //Transmission Start

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,0);

AbsDelay(1);

// Mit der Werteabfolge 00000101 am Datenport signalisiert man dem Sensor dass eine Feuchtigkeitsmessung durchzuführen ist.

portschreiben(0);

AbsDelay(1);

Clock();

Clock();

Clock();

Clock();

Clock();

portschreiben(1);

AbsDelay(1);

Clock(); //Code für Feuchtigkeit 00000101

portschreiben(0);

AbsDelay(1);

Clock();

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

//Port19 ist der Leseport (siehe Hardwarelayout auf Seite 11 für besseres Verständnis). Auf diesem Port wartet nun der Mikrokontroller auf ein „Acknowledge“ in welchem er die DATA Leitung auf 0V setzt. Damit bestätigt der Sensor dass er den Code für die Feuchtigkeitsmessung erhalten hat und beginnt mit der Messung.

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19);

if (messen==0)

{ //Ack.

Clock();

checken=0;

}

}

AbsDelay(1);

//Hier wartet der Mikrokontroller nun auf die Beendigung der Messung durch den Sensor, dies kann über 100ms dauern.

checken=1;

while(checken==1)

{

messen=Port\_ReadBit(19); //Warten für Messung

AbsDelay(1);

if (messen==0)

{

checken=0;

}

}

//Nun werden die 12bit der Feuchtigkeitsdaten der Reihe nach empfangen (das MSB zuerst). Die 12 Bit sind auf 2 Bytes (16bit) aufgeteilt, die ersten 4bit enthalten also keine Information, daher die 4 Clockimpulse.

Clock();

Clock();

Clock();

Clock();

for(i=0;i<4;i++)

{

messen=Port\_ReadBit(19); //erste 4Bit

AbsDelay(1);

feucht[i]=messen;

Clock();

}

//Nach jedem der Bytes muss der Miktokontroller die DATALeitung auf 0V setzen und damit dem Sensor den Empfang bestätigen

AbsDelay(1);

portschreiben(0); //Ack.

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

//Daraufhin sendet der Sensor das zweite Byte

for(i=0;i<8;i++)

{

messen=Port\_ReadBit(19); //nächte 8 bit

AbsDelay(1);

feucht[i+4]=messen;

Clock();

}

//Diese Abfolge beendet die Kommunikation ohne dass der Sensor eine Checksumme sendet (siehe Timingdiagramm auf Seite 6). Der Sensor kehrt in seinen lowpower sleep Zustand zurück.

AbsDelay(1);

portschreiben(0);

AbsDelay(1); // Verbindung trennen

Port\_WriteBit(18,1);

AbsDelay(1);

portschreiben(1);

//Mit dieser Funktion werden nun die 12 digitalen Bits in einen Dezimalwert umgerechnet

wertfeucht=0;

for (i=0;i<12;i++)

{

//Msg\_WriteInt(feucht[i]);

wertfeucht=wertfeucht+(feucht[i]<<(11-i));

}

}

## Subprogramm zur Kommunikation mit SHT71 zur Temperaturmessung:

void temperatur(void)

{

//Dieses Programm gleicht dem der Feuchtigkeitsmessung bis auf 2 Details:

//1. Der Code um die Temperaturmessung zu starten lautet hier 00000011 :

portschreiben(0);

AbsDelay(1);

Clock();

Clock();

Clock();

Clock();

Clock();

Clock();

portschreiben(1);

AbsDelay(1);

Clock(); //Code für Temp 00000011

portschreiben(1);

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

//2. Die Messung erfolgt in 14bit Auflösung, so sind beim Auslesen der Daten nun 2 „leere“ Clockimpulse nötig um 2Bytes (16bit) zu füllen

Clock();

Clock();

for(i=0;i<6;i++)

{

messen=Port\_ReadBit(19); //erste 6Bit

AbsDelay(1);

temp[i]=messen;

Clock();

}

AbsDelay(1);

portschreiben(0); //Ack.

AbsDelay(1);

Port\_WriteBit(18,1);

AbsDelay(1);

Port\_WriteBit(18,0);

Port\_WriteBit(21,1);

AbsDelay(1);

for(i=0;i<8;i++)

{

messen=Port\_ReadBit(19); //nächte 8 bit

AbsDelay(1);

temp[i+6]=messen;

Clock();

}

//Daraus resultiert natürlich dass auch bei der Umrechnung auf einen Dezimalwert die Schleife für 14bit gestaltet sein muss.

werttemp=0;

for (i=0;i<14;i++)

{

//Msg\_WriteInt(temp[i]);

werttemp=werttemp+(temp[i]<<(13-i));

}

## Subprozedur portschreiben():

Diese Prozedur legt den ihr übergebenen Wert (1 für 5V und 0 für 0V) an Port 20 des Mikrokontrollers an und gleichzeitig den Wert 0 (für 0V) an den Port 21. Port 21 steuert das Outputenable des Bustreibers, schaltet ihn also bei 0V durch. Insgesamt hat diese Prozedur die Funktion den übergebenen Wert an den Datenpin des SHT71 weiterzuleiten.

void portschreiben(int y)

{

Port\_WriteBit(21,0);

Port\_WriteBit(20,y);

}

## Auslesen des Wertes des Lichtsensors:

Conrad C-Control bietet zum Auslesen des Wertes an einem Analog-Digitalwandler Kanal eine fertige Funktion an. Mit ADC\_Read() kann der derzeitige Analogwert in einer Variable gespeichert werden.

void licht(void)

{

wertlicht=ADC\_Read();

}

## Umrechnen der erhaltenen Daten auf brauchbare Umweltwerte:

// Die von den Sensoren erhaltenen Werte werden in Arbeitsvariablen kopiert. Der ursprüngliche Sensorwert bleibt über die Berechnung hinweg unangetastet.

wtemp=werttemp;

wzwischen=wertfeucht;

wlicht=wertlicht;

//Temperatur und Feuchtigkeit werden nach den, von SENSIRION gegebenen Daten, (siehe Seite 7) umgerechnet.

wtemp=(wtemp\*0.01)-40.0;

wfeucht=-(wzwischen\*wzwischen\*2.8\*0.001\*0.001);

wfeucht=wfeucht+(wzwischen\*0.0405)-4.0;

wfeucht=(wtemp-25.0)\*(0.01+0.00008\*wzwischen)+wfeucht;

werttemp=wtemp\*100.0;

wertfeucht=wfeucht\*100.0;

//Der Lichtwert wird mittels der Formel für das Verhältnis des Widerstandes zur Beleuchtung (siehe Seite 9 ) in einen Lux-wert umgerechnet.

wlicht=wlicht\*5/1024;

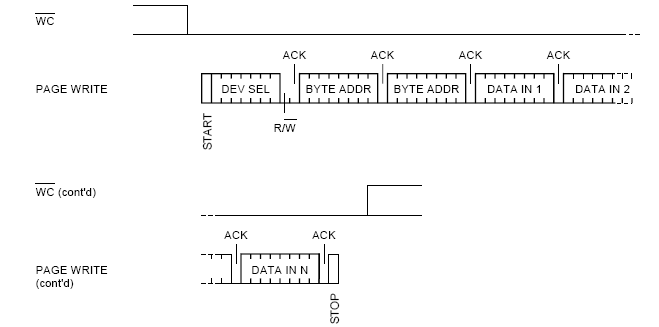
wlicht=wlicht/7;

wlicht=(((1-wlicht)/(wlicht\*10))-(1/24));

wlicht=wlicht\*600;

wertlicht=wlicht;

## Schreiben der Daten in den Speicher



**%**

**#**

**\***

*Abb. 10: Timing Diagramm zum Datenschreiben*

// Als erstes wird der I2C Bus initialisiert (mittels I2C\_Start). Über seine Adresse (0xA0) wird der Speicherchip am I2C Bus adressiert. Er steht nun in Kommunikation mit dem Mikrocontroller.

void schreibenstart(void)

{

I2C\_Start();

I2C\_Write(0xA0);  // (\* im Timingdiagramm)

//address ist der Ort im Speicher in dem die Daten abgelegt werden.

I2C\_Write(address>>8);          // (# im Timingdiagramm)

I2C\_Write(address);

}  
//Nun ist das Schreiben initialisiert, der Speicher ist ausgewählt, der Zielort für die Daten ist festgelegt. Nun kann der Datenstrom beginnen.

void schreiben(int data)

{

//Die Funktion schreiben() schreibt die ihr übergebenen integer Zahlen in 2 separate Bytes zerlegt hintereinander in den Speicher.

byte a;

a=data>>8;

I2C\_Write(a); // (% im Timingdiagramm)

a=data;

I2C\_Write(a);

}

void schreibenstop(int bytezahl)

{

//Nachdem die gewünschten Daten in den Speicher geschrieben wurden werden 2 beendende Nullen hinzugefügt um Datensätze leicht voneinander trennen zu können. Der I2C Bus wird deaktiviert und der Wert der Variable address wird um die gerade geschriebene bytezahl erhöht. So werden die gerade geschriebenen Daten beim nächsten Schreiben nicht überschrieben, sondern es wird hinter ihnen fortgesetzt.

byte a;

a=0;

I2C\_Write(a);

I2C\_Write(a);

I2C\_Stop();

address=address+bytezahl;   
}

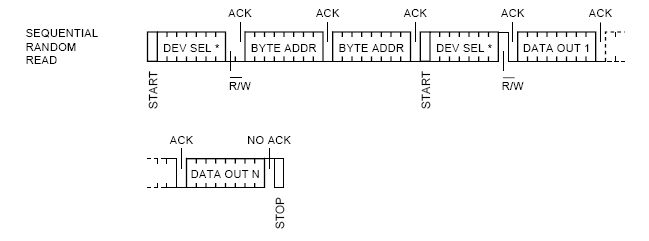
## Auslesen der Daten aus dem Speicher:

**%**

**$**

**#**

**\***



*Abb. 11: Timing Diagram zum Datenlesen*

*//Der Speicherinhalt kann auf Wunsch entweder direkt im C-Control Programmierinterface dargestellt werden oder über die serielle Schnittstelle gesendet werden. Via LCD wird der Benutzer gefragt welche Ausgabemethode er wünscht.*

void speicherholen(byte ziel)

{

// 2 entspricht der Wahl das Benutzers die COM (serielle) Schnittstelle anzusteuern.

if (ziel==2)

{

//Die C-Control eigene Funktion Serial\_Init initialisiert die serielle Schnittstelle. Der erste Parameter (=0) wählt die erste Schnittstelle aus (die zweite wäre die USB Schnittstelle). Anschließend wird das Protokoll festgelegt, 8 Datenbits, danach je ein Stoppbit, kein Paritybit. Als letzter Parameter wird noch die Baudrate mit 19200 festgelegt.

Serial\_Init(0,SR\_8BIT|SR\_1STOP|SR\_NO\_PAR,SR\_BD19200);

}

//Festlegung des Lesestartortes im Speicher

address=0x10;

I2C\_Start();

 //Auswahl des Chips über seine Adresse

   I2C\_Write(0xA0); // (\* im Timingdiagramm)

I2C\_Write(address>>8);

I2C\_Write(address); //(**#** im Timingdiagramm)

I2C\_Start();

I2C\_Write(0xA1); //(**$** im Timingdiagramm)

do   {

  ausgabe1=I2C\_Read\_ACK(); //(**%** im Timingdiagramm)

ausgabe2=I2C\_Read\_ACK();

anzeige=ausgabe1<<8;

anzeige=anzeige+ausgabe2;

if (ziel==1)

//Mit ziel=1 wählt der Benutzer die Ausgabe auf dem C-Control Entwicklungsbildschirm aus

{

if (anzeige!=0)

{

//Msg\_WriteInt ist die C-Control interne Funktion um integer Werte direkt auf der Entwicklungsumgebung auszugeben. Der Anschließende HexWert 43690 entspricht AAAA, was als Separator zwischen den Datensätzen zur besseren Lesbarkeit dient.

Msg\_WriteInt(anzeige);

Msg\_WriteHex(43690);

}

}

else

{

Serial\_Write(0,ausgabe1);

Serial\_Write(0,ausgabe2);

}

}

while (anzeige!=0);

  ausgabe1=I2C\_Read\_NACK();

ausgabe2=I2C\_Read\_NACK();

   I2C\_Stop();

if (ziel==2)

{

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Write(0,0);

Serial\_Disable(0);

}

}