Beschreibung spezieller Firmware für AVR Attiny 26x

(u. a. genutzt im I²C Projekt „Hauscomputer“ - http:\\www.haus-computer.de)

Inhaltsverzeichnis

1. Einführung....2
   1.1. Hinweise zur Programmierung....2
1. Analoger 8-Kanal AD Wandler mit 2 PWM Kanälen....3
   1.1. Funktionsbeschreibung....3
   1.2. Pinbelegung:....3
   1.3. Adressierung....3
   1.4. Registerbeschreibung....4
   1.5. Programmierbeispiele unter Turbo Pascal....9
   1.6. Versionsgeschichte....11
2. Binäres 8-Kanal I/O Interface....12
   2.1. Beschreibung....12
   2.2. Pinbelegung:....12
   2.3. Adressierung....12
   2.4. Registerbeschreibung....13
   2.5. Programmierbeispiele unter Turbo Pascal....15
   2.6. Versionsgeschichte....16
3. Interface für Busadapter „Deluxe“....17
   2.1. Beschreibung....17
   2.2. Pinbelegung:....17
   2.3. Adressierung....17
   2.4. Registerbeschreibung....18
   2.5. Programmierbeispiele unter Turbo Pascal....20
   2.6. Versionsgeschichte....21

Programmiert von Herrn Thomas Höschler, Rostock.
Dokumentation: Dipl. Ing. Uwe Behrndt.

1. Einführung

Die nachfolgend beschriebene Firmware wurde für den AVR ATtiny26x (ATtiny26, ATtiny261) entwickelt, um ihn in Systemen mit I2C Bus mit einem möglichst geringen Hardwareaufwand einzusetzen. Die Ansteuerung erfolgt über I2C-Bus entsprechend der Spezifikation für den Low-Speed-Mode. Es wurde bis zu einer SCL Taktfrequenz von 2 kHz getestet, sollte aber auch drüber hinaus funktionieren. Die Firmware enthält eine Watchdog, sowie eine gesicherte Übertragung mit Prüfsummen.
Die Kompilierung des Quellcodes erfolgte für zwei Versionen:

- „Save Mode“: Firmware I2C_Ax.x/Bx.x für das Projekt „Hauscomputer“. Dabei läuft die Firmware im sogenannten sicheren Modus, d.h. nicht benötigte Register können über den I2C Bus nicht verändert werden, nicht benötigte Funktionen sind ebenfalls deaktiviert.
- „Normal Mode“: Firmware, in der alle Funktionen und Register freigeschaltet sind. Nur für die eigene Softwareentwicklung vorgesehen.

Dieses Dokument wendet sich vor allem an Programmierer, die die Firmware mit einem ATtiny26x in eigenen Projekten verwenden wollen. Für die Nutzung der Firmware im Projekt „Hauscomputer“ sind die Angaben im dortigen Programmhandbuch ausreichend. Darüber hinaus kann die Firmware aber auch in anderen Projekten verwendet werden, wenn die nachfolgend beschriebenen Spezifikationen eingehalten werden. Entsprechende Programmbeispiele sind beigefügt.
Für das Erreichen der entsprechenden Funktionalität ist mit einem Programmiertool (z.B. Ponyprog2000) die Firmware in den ATtiny26x zu laden. Es ist dabei egal, welche Version des ATtiny26x zum Einsatz kommt, da die Software immer mit dem internen Takt von 1 Mhz arbeitet.

1.1. Hinweise zur Programmierung

Aufspielen der Firmware:
Zum Programmieren der ATtiny26x genügt das Einspielen der entsprechenden *.hex Datei in den IC mit gängigen Programmen wie z.B. Ponnyprog und eines geeigneten Programmieradapters.

Um ab Version A2.0/B2.0, eine korrekte Funktion des Ladens der Startwerte sicherzustellen, ist es nötig die „Fuse Bits“, im Programm „PonyProg“ auch „Configuration and Security bits“ genannt, BODLEVEL und BODEN programmieren. Dies ist nicht nötig, wenn die Applikation einen eigenen Reset-Baustein enthält. Um die Abwärtskompatibilität zu erhalten ist die EEPROM-Datei „ProjectATtiny26.eep“ in den EEPROM zu laden (PWM Startwert „0x00“ und OutRegA „0x00“), das geht mit PonyProg wie folgt:
Nach dem Öffnen der Datei „ProjectATtiny26.hex“, „File/Open Data (EEPROM) File...“ auswahlen, im Öffnen-Dialog den Dateityp „*.eep“ wählen und „ProjectATtiny26“ öffnen. Scrollt man jetzt im Fenster ganz nach unten, kann man im „blauen Datenbereich“ die ersten beiden Bytes mit 00 sehen. Zum Flashen ist „Command/Write all“ zu wählen.

Das Laden der Startwerte kann man aber auch dem Programm „Hauscomp.exe“ überlassen: die Übertragung in die entsprechenden Register und anschließend in den EEPROM erfolgen vollautomatisch bei Programmstart. Näheres im Programmhandbuch.

1. Analoger 8-Kanal AD Wandler mit 2 PWM Kanälen

Bezeichnung des hex-Files: „I2C_Ax.x“

1.1. Funktionsbeschreibung

Die Firmware wurde für die AD – Wandlung von 8 Kanälen (á 10 Bit) und für die Ansteuerung von 2 PWM – Ausgängen programmiert. Darüber hinaus lassen sich statt des AD-Wandlereingangs alle acht oder auch einzelne Kanäle als diskrete Ein- oder Ausgänge betreiben. Zusätzlich ist ein Kanal für den Betrieb als Zähler vorbereitet. Nach Ansprechen der Watchdog, wenn also ca. 42 s lang keine Abfrage mehr über den Bus erfolgt, schalten beide PWMs auf einen im EEPROM hinterlegten Startwert. Bei Neuprogrammierung ist das „dauerhaft ein“. Dieser Wert kann über die Register 0x28 bzw. 0x29 geändert werden.

1.2. Pinbelegung:

Projekt	Atmel			Atmel	Projekt	
SDA	MOSI	1	20	PA0	ADC0	
PWM A	MISO	2	19	PA1	ADC1	
SCL	SCK	3	18	PA2	ADC2	
PWM B/(INT)		4	17	PA3	AREF	
+5V	VCC	5	16	GND	GND	
GND	GND	6	15	AVCC	+5V	
A0	PB4	7	14	PA4	ADC3	
A1	PB5	8	13	PA5	ADC4	
ADC9/TO	PB6	9	12	PA6	ADC5	
RST	RST	10	11	PA7	ADC6	

1.3. Adressierung

Als I2C Adresse wurde 0xA0 festgelegt:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	0	A1	A0	R/W

Über die Anschlüsse A0 bis A1 können 4 der IC auf dem Bus adressiert werden, für 1-Pegel ist kein externer Pull-Up nötig. Die Basisadresse ist demzufolge 0xA0, es sind maximal 4 verschiedene IC adressierbar.

Ansteuerung:

Die Ansteuerung erfolgt Registerweise, wobei das erste nach der Adresse übertragene Byte die Registernummer ist, welchem dann zwei weitere Bytes mit dem Inhalt folgen.
Schreiben: Es sind nach der Adresse 1 oder 3 Bytes zu senden. Um das Register für das nachfolgende Lesen zu adressieren ist 1 Byte zu senden. Soll der Inhalt des Registers beschrieben werden, ist zuerst die Registernummer dann das Datenbyte und zum Schluss die Prüfsumme zu senden. Nur wenn die Prüfsumme stimmt, wird ein Achnowledge (vom Slave) generiert und das Register gesetzt.
Die Prüfsumme berechnet sich aus der Summe von Adresse, Registernummer und Datum erhöht um 1, ohne Übertrag (1 Byte). Bsp.: CHS := 0xFF and (Adress + Register + Datum + 1)
Bsp.:

START	
W0xA0	(Adresse 0xA0, write)
ACK	(Slave)
W0x21	(Registerwahl 0x21)
Ack	(Slave)
W0x80	(PWM A Value 50%)
Ack	(Slave)
W0x42	(Checksum)
Ack	(Slave)
STOP	

Lesen: Zunächst muss die Registernummer geschrieben werden, wobei hier als Ausnahme (s.o.) die Übertragung nur eines Bytes erlaubt ist. Danach wird erneut adressiert und es können 1- 3 Bytes gelesen werden: HiByte, LoByte und Prüfsumme.
Bsp.: ADC 9 (Pin 9) Lesen (Vin = 1,5V, Vref = 2,5V, ADC =614)

START	
W0xA0	(Adresse, write)
ACK	(Slave)
W0x09	(Register 0x01)
Ack	(Slave)
STOP	
START	
W0xA1	(Adresse, read)
ACK	(Slave)
R0x66	(ADC- Hi-Byte)
Ack	(Master)
R0x4E	(ADC-Lo-Byte with Checksum)
Ack	(Master)
R0x5F	(Checksum)
STOP	

Zu beachten ist, bei den Registern 0x20 bis 0x2F die nur 1 Byte breit sind, wird das Low-Byte auf 0xFF gesetzt, danach folgt wie gehabt die Prüfsumme.

1.4. Registerbeschreibung

Register 0 bis 31 (0x1F) bilden die AD-Wandler- Eingange ab. Die Registernummer wird nicht übertragen! Es wird zuerst das H-Byte und dann das L-Byte übertragen.
Die ungenutzten 6 MSB des HiByte ist die Prüfsumme, bestehend aus der Summe von Adresse, Registernummer, LoByte, HiByte und 1, ohne Übertrag, multipliziert mit 4.
Bsp.:

```
CHS  := $FF and ((Adress + Register + (HReg and $03) + LReg + 1) SHL 2)
```

Die folgende Prüfsumme berechnet sich wie gehabt aus der Summe von Adresse, Registernummer, HiByte, LoByte (so wie übertragen) und 1, ohne Übertrag.

Es wurde die gesamte Eingangskanal- und Verstärkungskodierung verwendet (nachzulesen im

Datasheet ATtiny 26). Sie ist allerdings mit der Firmware, aufgrund der anderweitig benutzten Pins 7-8 (ADC7-8), nicht vollständig sinnvoll zu benutzen. Auf diese Register kann daher nur lesend zugegriffen werden.

Registernummer	MUX4..0	Einfacher Anschluss	Positiver Differenzanschluss	Negativer Differenzanschluss	Verstärkung
00	0000	ADC 0			
01	0001	ADC 1			
02	0010	ADC 2			
03	0011	ADC 3			
04	0100	ADC 4			
05	0101	ADC 5			
06	0110	ADC 6			
09	1001	ADC 9			
11	01011		ADC 0	ADC1	20x
12	01100		ADC 0	ADC 1	1x
13 (1)	01101		ADC 1	ADC 1	20x
14	01110		ADC 2	ADC 1	20x
15	01111		ADC 2	ADC 1	1x
16	10000		ADC 2	ADC 3	1x
17 (1)	10001		ADC 3	ADC 3	20x
18	10010		ADC 4	ADC 3	20x
19	10011		ADC 4	ADC 3	1x
20	10100		ADC 4	ADC 5	20x
21	10101		ACD 4	ADC 5	1x
22(1)	10110		ADC 5	ADC 5	20x
23	10111		ADC 6	ADC 5	20x
24	11000		ADC 6	ADC 5	1x
30	11110	1,18V (V_{BG})			
31	11111	0V (GND)			

(1) Für Offset – Messung

$$ADC = \frac{Vin * 1024}{Vref} \qquad ADC = \frac{(Vpos - Vneg) * Gain * 1024}{Vref}$$

Register 0x20 (Mode 1) 1 Byte (im SaveMode nur lesen)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
CS13	CS12	CS11	CS10	OCR1C Bit 7	OCR1C Bit 6	PWM1A	PWM1B	
TCCR1B (3..0)						TCCR1A (1..0)		
1	1	0	1	1	0	1	1	Nach Reset

Mit den Bit 7..4 (CS1x) wird der Vorteiler der PWM Einheit eingestellt, wobei 0 den Teiler stoppt, 0001 dem Teilerfaktor 1, 0010 Faktor 2, 0011 Faktor 4 und 1111 einem Faktor von 16384 entsprechen. Damit ist ein Bereich von ca. 3,9 kHz bis 0,2 Hz abzudecken. Mit Bit 3..2 auf 0 kann die Frequenz um den Faktor 4 erhöht werden. Dafür begrenzt sich aber die „Auflösung“ des PWM von 8 auf 7 oder 6 Bit. Bit 1..0 deaktivieren den jeweiligen PWM-Kanal (0-deaktiviert) und der Pin wird hochohmig. Ist also 0 oder 1 Pegel gewünscht, ist der zugehörige PWM-Wert auf 0x00 bzw. 0xFF zu setzen und PWM1x auf 1.

$$F_{PWM} = \frac{F_{CK}}{\text{Vorteiler} * OCR1C} \qquad F_{CK} = 1\ \text{MHz}$$

OCR1C ist entsprechend o.g. Register Bit 2:3 64, 128, 191, 256

Voreinstellung ist f = 0,96 Hz, T=0,78s
PWMB nur im SaveMode nutzbar.

Register 0x21
Wert ist der für PWMA (OCR1A).

Register 0x22
Wert ist der für PWMB (OCR1B).
PWMB nur im SaveMode nutzbar.

Register 0x23 **(Mode 2)** (im SaveMode nur lesen)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
ADEN	REFS 1	REFS 0	ADLR	WdIe	WdEe	n.b.	TCExt PB6	
ADCS R	ADMUX							
1	0	1	0	1	1	0	0	Nach Reset

REFS1	REFS0	Referenzspannungs- Auswahl
0	0	AVCC
0	1	AREF (Pin 17, PA3)
1	0	Interne Referenz (2,56V) AREF nicht angeschlossen
1	1	Interne Referenz (2,56V) AREF angeschlossen, mit externen Kondensator zu versehen

ADLR (Bit 4) Wenn dieses Bit gesetzt ist erscheint das Ergebnis links ausgerichtet, dass heißt nur die beiden niederen Bit befinden sich im Lo-Byte, anderenfalls nur die beiden höchsten Bits im Hi-Byte.
WdEe (Bit 2) aktiviert „externen“ Watchdog, d.h. Ohne zyklische Abfrage über den Bus werden die Port-Bits rückgesetzt und die PWM's deaktiviert.
WdIe(Bit 3) Funktionalität nicht gegeben interner Watchdog ist immer an.
TCExt PB6 (Bit 0) Nicht im Savemode aktiviert den Pullup für PB6 eingeschaltet.

Register 0x24 (im SaveMode nur lesen)
Wert ist der von DDRA.

Register 0x25 (im SaveMode nur lesen)
Wert ist der von PortA.

Die Zuordnung des jeweiligen Bitpaares der Port-Pins zeigt die Tabelle.

DDR Bit x	Port Bit x	Funktion
0	0	Eingang, hochohmig (Tri-State)
0	1	Eingang, Pullup-Widerstand intern
1	0	Ausgang, 0-Pegel
1	1	Ausgang, 1-Pegel

Um die Eingänge sinnvoll als AD-Wandler-Eingang zu benutzen sind beide Bits auf 0 zu setzen.
Nach Reset sind beide Bits auf 0 gesetzt.

Register 0x26 (Mode 3) (im SaveMode nur lesen)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
				WdIe	WdEe	n.b.	TCExt PB6	
0	0	0	0	1	1	0	0	Nach Reset

WdEe (Bit 2) aktiviert „externen“ Watchdog, d.h. Ohne zyklische Abfrage über den Bus werden die Port-Bits rückgesetzt und die PWM's deaktiviert.
WdIe(Bit 3) Funktionalität nicht gegeben interner Watchdog ist immer an.
TCExt PB6 (Bit 0) Nicht im Savemode aktiviert den Pullup für PB6 eingeschaltet.

Hinweis:
Nach dem Beschreiben der Register 0x28 – 0x29 wird der jeweilige Wert in den EEPROM geschrieben. Das dauert typischer Weise 8,3ms. In dieser Zeit reagiert das Interface nicht. Diese Werte fungieren als Startwerte, die nach Reset oder bei Auslösen des Watchdog eingenommen werden.

Register 0x28 (PWMA Startwert) (Lesen und Schreiben)

Register 0x29 (PWMB Startwert) (Lesen und Schreiben)

Register 0x30 (nur lesen)
Nach der Registernummer wird zuerst der Zustand Port-Eingangsregister A und dann B übertragen.

Register 0x31 (nur lesen)
Nach der Registernummer wird das Statusbyte übertragen

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
SM	n.b.	n.b.	n.b.	n.b.	n.b.	WDint	WDext	

WDint - Power On Reset, interner Watchdog hat ausgelöst (Firmware hängt)
WDext- Watchdog hat ausgelöst
Nach dem Auslesen werden die o.g. Bits auf 0 gesetzt.
(Siehe auch Register 0x88)
SM- Save Mode entsprechend genannte Register können nicht mehr geändert werden. Nach Reset und Auslösung Watchdog ändern sich bestimmte Startparameter. Die Funktionalität des Interrupt am I2C-Bus (PB3) ist deaktiviert.

Register 0x32 (nur lesen)
Prozessor-Register TCCR1A, TCCR1B.

Register 0x33 (nur lesen)
Prozessor-Register OCR1A, OCR1B.

Register 0x34 (nur lesen)
Prozessor-Register OCR1C, kein zweites Byte.

Register 0x35 (nur lesen)
Prozessor-Register ADCSR, ADMUX.

Register 0x36 (nur lesen)
Lesen des IntRegH und IntRegL, Register für die Auswertung des PinChangeInterrupts.
Bei der Initialisierung *und nach dem Auslesen*, wird IntRegH auf 0x00 und IntRegL auf 0xFF gesetzt. Wird ein PinChangeInterrupt über PA7:6, PA3, PB6 ausgelöst, dass heißt der Eingangspegel an den o.g. Pins ändert sich (beide Flanken), wird der jeweils anliegende Pegel des Port A gespeichert, damit nicht in Echtzeit der Pegel gelesen werden muss. Das funktioniert wie folgt:
das IntRegH ist auf 0x00 initialisiert und wird mit dem PortInputRegister (PinA) ODER verknüpft, so dass alle aufgetretenen 1-Pegel im Register gespeichert werden,
das IntRegL ist auf 0xFF initialisiert und wird mit dem PortInputRegister (PinA) UND verknüpft, so dass alle aufgetretenen 0-Pegel im Register gespeichert werden.
Achtung es wird zwar der Interrupt von PB6 bearbeitet, aber der Zustand nicht gespeichert, PA0:2 und PA4:5 werden bei Interrupt Auslösung gespeichert, lösen aber keinen Interrupt aus. Wenn der AREF Eingang, wie initialisiert aktiviert gelassen wird, ist PA3 auch nicht nutzbar.

Register 0x37 (nur lesen)
Lesen des TimerCounter0 (TCNT0) Register. Bei aktivierten Bit in Mode 2 werden die fallenden Flanken an PB6 gezählt. Der Zähler ist frei laufend und wird nicht zurückgesetzt. Die Software im Master muss die Auswertung der Impulsanzahl vornehmen.

Register 0x3A (nur lesen)
Abfrage der Firmware Version, 1 Byte liefert z.B. „0xA2“ und 2. Byte „0x00“ für Version A2.0

Register 0x88 (nur lesen)
Durch Lesen des Registers wird eine Endlosschleife im Programm ausgelöst, dass der Watchdog auslöst und somit die Funktion getestet werden kann.
Nicht im SaveMode

Interupt-Ausgang:
Ist die Firmware nicht für sog. SaveMode compiliert (s.a. Reg 0x25) dann wird bei Auslösen

PinChangeInterrupt (s.a. Reg 0x30 und Internes) PB3 auf 0 gelegt. Bei erfolgreichem Lesen des PinchangeInterruptRegister (0x2B), wird dieser PB3 wieder auf hochohmig (TriState) gesetzt. In dieser Betriebsart ist PWMB nicht nutzbar.

Watchdog:
Die Firmware enthält einen Watchdog-Timer der, wenn der Baustein ca. 120s (42s ab A2.4) vom Master nicht adressiert wird, ein Reset auslöst und den PWMs werden die Startwerte zugewiesen. Beim analogen Modul wird A2.4 jetzt mittels statischem 0-Pegel am Pin 20 – PA0 der Status des Watchdog ausgegeben (ausgelöst/nach Reset → 0-Pegel). Sonst ist der PIN wie gehabt auf Tristate.

Internes:

PWM: Der interne Zähler zählt bis zum erreichen des Wertes OCR1x, dann wird der Ausgang auf 0 gesetzt, ist der Wert von OCR1C erreicht wird der Ausgang wieder auf 1 gesetzt.
Im Register TCCR1A (101000xx) sind COM1x1 und COM1x0 auf 10 gesetzt, d.h. nur die nicht invertierenden PWM- Ausgänge sind verbunden.
ADC: Das Register ADCSR (x1100111)ist wie folgt bestimmt ADEN (variabel) ADSC auf 1, ADFR auf 1 (free running mode), ADPS 2..0 auf 111 (bei 1MHz Systemtakt ca. 600 Wandlungen pro Sekunde).
PinChangeInterrupt im GIMSK ist PCIE1 gesetzt, was die Portbits PA7,6,3; PB7:4 selektiert, wobei PB5:4 für A1:0 genutzt wird. Ist AREF selektiert, ist PA3 ebenso nicht verfügbar.

1.5. Programmierbeispiele unter Turbo Pascal

Zur Programmierung von Software für die Kommunikation mit der Firmware des ATtiny26x über den I2C Bus gibt es viele Möglichkeiten. Die folgenden Beispiele sind Auszüge aus dem Quellkode „Hauscomp.exe“ und spiegeln die verschiedenen Funktionen wieder:

```
function ADeinlesen26(var Err:Byte;adr8,Kanal:byte):integer;
                                          {adr8:0-7,Kanal:0-3}
  var h,l,k,cs,tt:byte;w:integer;
  begin
    kanal:=kanal and 3;                   {Anpassung Adress+Kanalbereich}
    if((adr8 AND 1)=1)then begin
      adr8:=adr8-1;kanal:=kanal+4; end;
    adr8:=adr8 div 2;                     {adr8:0-3,Kanal:0-7}
    Start;
    if (kanal=7)then kanal:=9;
    ausgeben(Err,(80+adr8)*2);      {Adresse adr8, schreiben}
    Ausgeben(Err,Kanal);            {Kanaladressierung}
    Stop;
    ATtiny_ADpause;                 {Wandlung abwarten}
    Start;
    ausgeben(Err,(80+adr8)*2+1);    {Adresse adr8, lesen}
    h:=Einlesen;                    {HI Byte lesen}
    Acknowledge;
    l:=Einlesen;                    {Low Byte lesen}
    KeinAcknowledge;
    Stop;
    asm                             {Low+Hi Byte zusammensetzen}
      mov Dh,h
      mov Dl,l
      mov w,Dx
    end;
```

```
    cs:=h AND $FC;                  {checksumm aus oberen 4 Bit lesen}
    w:=w AND 1023;                  {10 Bit AD Wert}
                                    {Prüfsumme errechnen}
    tt:=(33+(adr8 shl 1)+kanal+(h AND 3)+1) shl 2 ;
    Adeinlesen26:=w;
                                    {Prüfsumme mit checksumm vergleichen}
    if((cs-tt=0)AND(err=0))then err:=0 else err:=1;
  end;
```

```
function Zaehler_einlesen26(var Err:Byte;adr8:byte):word;     {adr8:0-7}
  var h,l,k,cs,adr:byte;w,tt:word;
  begin
    adr8:=adr8 div 2;                       {adr8:0-3}
    adr:=(80+adr8)*2;
    Start;
    ausgeben(Err,adr);              {Adresse adr8, schreiben}
    Ausgeben(Err,$037);             {Zählregister adressieren}
    Stop;
    Start;
    ausgeben(Err,(80+adr8)*2+1);    {Adresse adr8, lesen}
    h:=Einlesen;                    {ab jetzt Zählregister auslesen}
    Acknowledge;
    l:=Einlesen;
    Acknowledge;
    cs:=Einlesen;                   {checksumm einlesen}
    KeinAcknowledge;
    Stop;
    asm                             {16 Bit Wert zusammensetzen}
      mov Dh,h
      mov Dl,l
      mov w,Dx
    end;
    tt:=(adr+$037+h+l+1)AND 255;    {Prüfsumme errechnen}
    Zaehler_einlesen26:=w;
                                    {Prüfsumme mit checksumm vergleichen}
    if((cs-tt=0)AND(err=0))then err:=0 else err:=1;
  end;
```

```
procedure DAaus26(var Err:Byte;adr8,wert:Byte); {adr8:0..7 PWM Ausgabe}
  var adr,kanal:byte;CHS:word;
  begin
    if((adr8 AND 1)=1)then begin    {Anpassung an Adressierung Firmware
ATtiny}
      adr8:=adr8-1;kanal:=$22; end
    else kanal:=$21;
    adr8:=adr8 div 2;               {adr8:0..3 PWMA:$21, PWMB:$22}
    if(wert>191)then wert:=191;     {Begrenzung PWM-Wert}
    adr:=(80+adr8)*2;               {=$A0+(adr8*2)}
    Start;
    ausgeben(Err,adr);              {Adresse adr8, schreiben}
    Ausgeben(Err,Kanal);            {Kanaladressierung}
    Ausgeben(Err,Wert);             {PWM Wert ausgeben}
    CHS:=(adr+Kanal+Wert+1)AND 255;
    Ausgeben(Err,CHS);              {Prüfsumme ausgeben}
    Stop;
  end;
```

1.6. Versionsgeschichte

I2C_A1.0 - Erste voll funktionsfähige Version. Einschränkung: Bei zu schnellem Abfragen der analogen Kanäle[1] wird allerdings noch der Analogwert des vorherigen Kanals übertragen, ohne dass ein Fehler ausgegeben wird.

I2C_A1.1 - Die Zeit für die A/D Wandlung wurde auf ein Viertel reduziert. Wird der IC trotzdem noch zu schnell abgefragt[1], so wird über eine Manipulation der Prüfsumme der A/D Wert vom Hauptprogramm ignoriert.

I2C_A2.0 - Einführung eines Startwertes für die PWM Kanäle, der im EEPROM hinterlegt ist. Bei Busausfall wird dieser nach der Watchdog- time- out- Zeit automatisch aktiviert.[2]

I2C_A2.2 - Nach Ansprechen der Watchdog und einer gleichzeitigen korrekten Busabfrage in einem engen Zeitfenster konnte es in extrem seltenen Fällen in den bisherigen Versionen zum dauerhaften „Low“ des SDA Signals kommen. Das ist hiermit behoben.

I2C_A2.4 - Die Watchdog ist nun nach außen geführt und mit dem Pin 20 (PA 0) verbunden. Im normalen Betrieb wird dieses Pin wie bisher als Sensoreingang abgefragt. Wird die Watchdog aber aktiv, so wird das Pin auf „low“ bzw. 0V gelegt. Die Zeit bis zum Ansprechen der Watchdog nach Ausfall des I2C-Busses beträgt jetzt 42s. Die PWM Schaltfrequenz beträgt 0,8s (ideal für Pumpen ohne Elektronik).

I2C_A2.4_0.2s – Wie Version I2C_A2.4; Die PWM Schaltfrequenz ist aber von 0,8s auf 0,2s reduziert (besser für thermische Geräte mit Heizdrähten).

I2C_A2.4_0.1s – Wie Version I2C_A2.4; Die PWM Schaltfrequenz ist aber von 0,8s auf 0,1s reduziert (für Ansteuerung von E-Patronen bei Eigenverbrauch von Solarstrom).

[1] im Programm „Hauscomp.exe“ kann die Zeitspanne der Abfrage (Dauer von „Attiny_ADpause“) mit den Scriptbefehlen „Bustakt=“ und „XFaktor=“ manipuliert werden. Außerdem beeinflussen andere als „DOS“ Betriebssysteme die Abfrage.

[2] wird vom Programm „Hauscomp.exe“ ab Version 6.0.32 unterstützt (Scriptbefehl „Analog=“).

2. Binäres 8-Kanal I/O Interface

Bezeichnung des hex-Files: „I2C_Bx.x“

2.1. Beschreibung

Die Firmware dient der Ansteuerung von 8 Kanälen als diskreter Ein- oder Ausgang. Watchdog (Pin 2) liefert High-Pegel, wenn die Watchdog ausgelöst hat (auch unmittelbar nach Reset), sonst Low-Pegel. Hinweis! (Pin 4) liefert ein 1 Hz Signal, wenn auf I2C Adresse 0x40 Bit 3 (PCF8574) eine 0 empfangen wird (Signal Hinweis! Im Projekt Hauscomputer), wird auf der angegebenen Adresse dagegen eine 1 empfangen, so führt Hinweis! Low-Pegel. Beide Pins sind für den Anschluss einer zweifarbigen Duo LED über zwei Widerstände geeignet. Nach dem Auslösen der Watchdog wird der im EEPROM gespeicherte Startwert auf das Ausgangsregister gelegt.

2.2. Pinbelegung:

Projekt	Atmel			Atmel	Projekt
SDA	MOSI	1	20	PA0	Kanal0
Watchdog	MISO	2	19	PA1	Kanal1
SCL	SCK	3	18	PA2	Kanal2
Hinweis!	(INT)		17	PA3	Kanal3
+5V	VCC	5	16	GND	GND
GND	GND	6	15	AVCC	+5V
A0	PB4	7	14	PA4	Kanal4
A1	PB5	8	13	PA5	Kanal5
A2	PB6	9	12	PA6	Kanal6
RST	RST	10	11	PA7	Kanal7

2.3. Adressierung

Als I2C Adresse wurde 0xB0 festgelegt:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	1	A2	A1	A0	R/W

Über die Anschlüsse A0 bis A1 können 4 der IC auf dem Bus adressiert werden, für 1-Pegel ist kein externer Pull-Up nötig. Die Basisadresse ist demzufolge 0xB0, es sind maximal 8 verschiedene ICs und damit 8x8=64 I/O Kanäle adressierbar

Ansteuerung:
Siehe Abschnitt 1.3.

2.4. Registerbeschreibung

Register 0x24 (im SaveMode nur lesen)
Wert ist der von DDRA.

Register 0x25 (im SaveMode nur lesen)
Wert ist der von PortA.Register 0x26 (Mode 3) (im SaveMode nur lesen)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
				WdIe	WdEe	n.b.	TCExt PB6	
0	0	0	0	1	1	0	0	Nach Reset

WdEe (Bit 2) aktiviert „externen“ Watchdog, d.h. Ohne zyklische Abfrage über den Bus werden die Port-Bits rückgesetzt und die PWM's deaktiviert.
WdIe(Bit 3) Funktionalität nicht gegeben interner Watchdog ist immer an.
TCExt PB6 (Bit 0) Nicht im Savemode aktiviert den Pullup für PB6 eingeschaltet.

Register 0x27 (Port A Output Register)
Ist ein Bit auf 0 gesetzt, ist der entsprechende Port-Pin hochohmig (und auf Eingang gesetzt). Ist das Bit auf 1 , dann liegt der Port-Pin auf VCC (5V). Nach Reset ist das Register auf 0x00 gesetzt.
(Außerhalb SaveMode kann der Port auch über Register 0x24 und 0x25 gesetzt werden. Wird Register 0x27 beschrieben wird DDRA immer auf 0x00 gesetzt und PortA entsprechend negiert zum Output Register)

Hinweis:
Nach dem Beschreiben des Registers 0x2A wird der jeweilige Wert in den EEPROM geschrieben. Das dauert typischer Weise 8,3ms. In dieser Zeit reagiert das Interface nicht. Dieser Wert fungiert als Startwert, der nach Reset oder bei Auslösen des Watchdog eingenommen wird.

Register 0x2A (Output Register Startwert) (Lesen und Schreiben)
- kanalweise: 0 - „aus“, 1- „ein“

Register 0x30 (nur lesen)
Port-Eingangsregister A und B

Register 0x31 (nur lesen)
Statusbyte, LoByte bleibt unbestimmt

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
SM	diskret	n.b.	n.b.	n.b.	n.b.	WDint	WDext	

WDint - Power On Reset, interner Watchdog hat ausgelöst (Firmware „hängt“)
WDext- Watchdog hat ausgelöst
Nach dem Auslesen werden die o.g. Bits auf 0 gesetzt.
(Siehe auch Register 0x88)
SM- Save Mode entsprechend genannte Register können nicht mehr geändert werden. Nach Reset und Auslösung Watchdog ändern sich bestimmte Startparameter. Die Funktionalität des Interrupt am I2C-Bus (PB3) ist deaktiviert.

Register 0x32 (nur lesen)
Prozessor-Register TCCR1A, TCCR1B.

Register 0x33 (nur lesen)
Prozessor-Register OCR1A, OCR1B.

Register 0x34 (nur lesen)
Prozessor-Register OCR1C, kein zweites Byte.

Register 0x35 (nur lesen)
Prozessor-Register ADCSR, ADMUX.

Register 0x36 (nur lesen)
Lesen des IntRegH und IntRegL, Register für die Auswertung des PinChangeInterrupts.
Bei der Initialisierung *und nach dem Auslesen*, wird IntRegH auf 0x00 und IntRegL auf 0xFF gesetzt. Wird ein PinChangeInterrupt über PA7:6, PA3, PB6 ausgelöst, dass heißt der Eingangspegel an den o.g. Pins ändert sich (beide Flanken), wird der jeweils anliegende Pegel des Port A gespeichert, damit nicht in Echtzeit der Pegel gelesen werden muss. Das funktioniert wie folgt:
das IntRegH ist auf 0x00 initialisiert und wird mit dem PortInputRegister (PinA) ODER verknüpft, so dass alle aufgetretenen 1-Pegel im Register gespeichert werden,
das IntRegL ist auf 0xFF initialisiert und wird mit dem PortInputRegister (PinA) UND verknüpft, so dass alle aufgetretenen 0-Pegel im Register gespeichert werden.
Achtung es wird zwar der Interrupt von PB6 bearbeitet, aber der Zustand nicht gespeichert, PA0:2 und PA4:5 werden bei Interrupt Auslösung gespeichert, lösen aber keinen Interrupt aus. Wenn der AREF Eingang, wie initialisiert aktiviert gelassen wird, ist PA3 auch nicht nutzbar.

Register 0x3A (nur lesen)
Abfrage der Firmware Version, 1 Byte liefert z.B. „0xA2“ und 2. Byte „0x00“ für Version A2.0

Watchdog:
Die Firmware enthält einen Watchdog-Timer der, wenn der Baustein ca. 120s vom Master nicht adressiert wird, ein Reset auslöst. Watchdog (Pin 2): 1-Pegel wenn der Watchdog ausgelöst hat (auch unmittelbar nach Reset), sonst 0-Pegel. Ab Version 2.4: 45s

Hinweis!:
Hinweis! (Pin 4): 1 Hz Signal wenn auf I2C Adresse 0x40 Bit 3 (PCF8574) eine 0 empfangen wird (Anzeige: Hinweis!), bei 1 geht der Ausgang auf 1-Pegel, dieses Datum wird auch ganz normal mit Acknowledge bestätigt. Zu beachten ist: es wird lediglich auf Adresse 0b01000000 geantwortet (Write), auf 0b01000001 (Read) nicht. Nach Reset hat der Pin 0-Pegel, erst wenn der Watchdog-Ausgang auf 0 geht, wechselt der Ausgang auf 1.

Internes:
Der ADC ist deaktiviert (Mode2 ist auf 0x00 gesetzt)

2.5. Programmierbeispiele unter Turbo Pascal

Auszug aus dem Quellkode „Hauscomp.exe“ zur Ansteuerung des ATtiny26x mit der Firmware I2C_B1.0 über den I2C- Bus:

```
procedure ParallelAus26 (var Err:Byte;adr,Wert : Byte); {Ausgabe 8 Bit}
  var reg:byte;chs:word;
  begin
    reg:=$27;
    adr:=(88+adr)*2;                {=$B0+(adr*2)}
    Start;
    ausgeben(Err,adr);              {Adresse adr, schreiben}
    Ausgeben(Err,reg);              {Registeradresse}
    Wert:=NOT(Wert);
    Ausgeben(Err,Wert);             {bin Wert ausgeben}
    CHS:=(adr+reg+Wert+1)AND 255;
    Ausgeben(Err,CHS);              {Prüfsumme ausgeben}
    Stop;
  end;


function ParallelEin26 (var Err:Byte;Adr:Byte) : Byte; {Eingabe 8 Bit}
  var reg,tt,PortA,PortB:byte;chs:word;
  begin
    reg:=$30;
    adr:=(88+adr)*2;                {=$B0+(adr*2)}
    Start;
    ausgeben(Err,adr);              {Adresse adr,schreiben}
    Ausgeben(Err,reg);              {Registeradresse}
    Stop;
    Start;
    ausgeben(Err,adr+1);            {Adresse adr,lesen}
    PortA:=Einlesen;                {Port A lesen}
    Acknowledge;
    PortB:=Einlesen;                {Port B lesen}
    Acknowledge;
    chs:=Einlesen;                  {checksumm einlesen}
    KeinAcknowledge;
    Stop;
    tt:=(adr+reg+PortA+PortB+1)AND 255;
    testwert1:=PortA;
    testwert2:=PortB;
    testwert3:=chs;
    testwert4:=tt;
    if((chs-tt=0)AND(err=0))then err:=0 else err:=1;
    ParallelEin26:=NOT(PortA);
  end;
```

2.6. Versionsgeschichte

I2C_B1.0 - Erste voll funktionsfähige Version mit 8 x I/O Kanälen, Watchdog und Signal HINWEIS!.

I2C_B2.0 - Einführung eines Startwertes für alle 8 Kanäle, der im EEPROM hinterlegt ist. Bei Busausfall wird dieser nach der Watchdog- time- out- Zeit automatisch aktiviert.[2]

I2C_B2.2 - Nach Ansprechen der Watchdog und einer gleichzeitigen korrekten Busabfrage in einem engen Zeitfenster konnte es in extrem seltenen Fällen in den bisherigen Versionen zum dauerhaften „Low“ des SDA Signals kommen. Das ist hiermit behoben.

I2C_B2.4 - Verkürzung der Ansprechzeit des Watchdogs von 120s auf 45s.

- Diese Version ist nun auch für den ATtiny 261(A) geeignet (Die vorigen Versionen berücksichtigen nicht den veränderten Pin-Change-Interrupt des 261).

- Zur besseren Orientierung des zur Programmierung stehenden Files und später zur Identifizierung eines geladenen Controllers steht ziemlich am Anfang des Programms die Versionsnummer im Klartext.

[2] wird vom Programm „Hauscomp.exe“ ab Version 6.0.32 unterstützt.

3. Firmware für Busadapter „Deluxe“

Bezeichnung des hex-Files: „I2C_Dx.x“

3.1. Beschreibung

Die Firmware dient der Ansteuerung von 4 Kanälen als diskreter Eingang (E0..E3), einem Umschaltrelais zwischen Haupt-/ReservePC 2 LEDs (Hinweis, Watchdog) und einem 3-stellungs Schalter (S1, S2). Watchdog (Pin 2) liefert High-Pegel, wenn die Watchdog ausgelöst hat (auch unmittelbar nach Reset), sonst Low-Pegel. Hinweis! (Pin 4) liefert ein 1 Hz Signal, wenn auf I2C Adresse 0x40 Bit 3 (PCF8574) eine 0 empfangen wird (Signal Hinweis! Im Projekt Hauscomputer), wird auf der angegebenen Adresse dagegen eine 1 empfangen, so führt Hinweis! Low-Pegel. Beide Pins sind für den Anschluss einer zweifarbigen Duo LED über zwei Widerstände geeignet. Die Ansteuerung des Umschaltrelais erfolgt folgendermaßen: werden an Adresse 0x40 2 min und 40s lang keine Signale detektiert und Pin 7 und Pin 9 führen High-Pegel, so schaltet Pin 8 für 9s von Low- auf High-Pegel. Das erfolgt 3 mal. Wurde dann immer noch kein Signal detektiert, so verbleibt Pin 8 dauerhaft auf High-Pegel.

- an Pin 8 ist das NORM/REV Relais angeschlossen, 0-Pegel = NORM, 1-Pegel = REV
- an Pin 7 führt 0-Pegel (Pull up Widerstand ein) zu zwangsweise NORM
- an Pin 9 führt 0-Pegel (Pull up Widerstand ein) zu zwangsweise REV

3.2. Pinbelegung:

Projekt	Atmel			Atmel	Projekt
SDA	MOSI	1	20	PA0	E0
Watchdog	MISO	2	19	PA1	E1
SCL	SCK	3	18	PA2	E2
Hinweis!	(INT)		17	PA3	E3
+5V	VCC	5	16	GND	GND
GND	GND	6	15	AVCC	+5V
S2	PB4	7	14	PA4	E_PWR1
Relais	PB5	8	13	PA5	E_PWR2
S1	PB6	9	12	PA6	E_PWR3
RST	RST	10	11	PA7	E_Reserve

3.3. Adressierung

Als I2C Adresse wurde 0xB0 festgelegt:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	1	0	0	0	R/W

Die Basisadresse ist demzufolge 0x10.

Ansteuerung:
Siehe Abschnitt 1.3.

3.4. Registerbeschreibung

Siehe Abschnitt 2.4.

Zusätzlich:

Alle Eingangs-Pins (Port A) zählen die 0-1 Pegelwechsel in einem 32 Bit Register. Diese Zählerstände werden durch o.g. Verfahren dauerhaft gesichert und bei Start wieder zurück geladen. Alternativ kann das das Abspeichern im EEPROM auch über den Bus durch Beschreiben des **Registers 0x66** über Bus ausgelöst werden.

Hinweis: in dieser Zeit reagiert der Attiny nicht auf den Bus.

Ein weiterer 32 Bit Zähler zählt die Anzahl der „Power on Vorgänge“ und noch einer die Einschaltzeit seit dem letzten „Power on“ mit ca. 1,5Hz.

Hinweis: Beim Flashen wird auch der EEPROM gelöscht und die Zähler starten mit dem Zählerstand 0xFFFFFFFF.

Die Zählerstände werden über den Bus ab Registeradresse 0x80 aufwärts ausgelesen. Pro Adresse 2 Byte, also sind für einen Zähler zwei Lesevorgänge nötig.

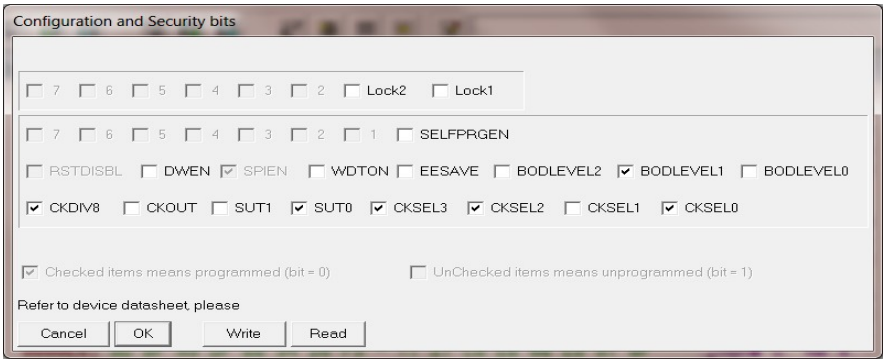
Hinweis: Der Wert ist nicht konsistent. Wenn sich der Zählerstand während des Auslesens ändert, kann das ausgelesene Ergebnis falsch sein.

Gelesen wird ab **Register 0x80,** zuerst Byte 1 dann Byte 0, anschließend an Adresse 0x81 Byte 3 und Byte 2 vom Zähler des Pin PB 0. Dann folgt an Adresse 0x82 Byte 1 und Byte 0 vom Zähler PB 1 u.s.w.

Register 0x90 + 0x91(nur lesen)
Hier kann der Zähler der „Power on“ Zahler ausgelesen werden.

Register 0x92 + 0x93(nur lesen)
Hier kann der Zähler der Einschaltdauer nach dem „Power on“ ausgelesen werden.

Programmierung der BrownOut-Detection auf 2,7V:



Register 0x3B(nur lesen)
Hier kann der Interface – Typ ausgelesen werden:

0	-	analoges Interface
1	-	digitales Interface
3	-	neues Interface Basisadresse 0xB0
7	-	neues Interface Basisadresse 0x10
15	-	Busadapter „Deluxe“

3.5. Programmierbeispiele unter Turbo Pascal

- Baustelle -

2.6. Versionsgeschichte

I2C_D2.6 - Erste voll funktionsfähige Version [1]

[1] wird vom Programm „Hauscomp.exe“ ab Version 6.0.41 unterstützt.