Roboter-Nachrichten 02 / 2014

Vorwort:

Liebe Leser, dies ist die zweite Ausgabe der Roboter-Nachrichten. Es haben sich ausreichend Leser gefunden, die uns Autoren motiviert haben, weitere Newsletter zu erstellen. Ich möchte mich an dieser Stelle recht herzlich über die vielen Emails bedanken, die ich von Ihnen erhalten habe. Das spornt natürlich an und wie in der letzten Ausgabe versprochen, gibt es neben den Korrekturen zum Buch auch ein paar hilfreiche Tipps im Umgang mit Mikrocontrollern.
Das Motto dieser Ausgabe lautet: „Daten richtig einlesen und anschließend weiter-verarbeiten“. Zuerst das Verarbeiten von digitalen Daten und anschließend von analogen Daten.

Es kam die Frage auf was für einen Roboter ich in der ersten Ausgabe in der Hand halte? Es handelt sich hier um den Hexabot Roboter mit Zusatzplatine und Tastsensoren.

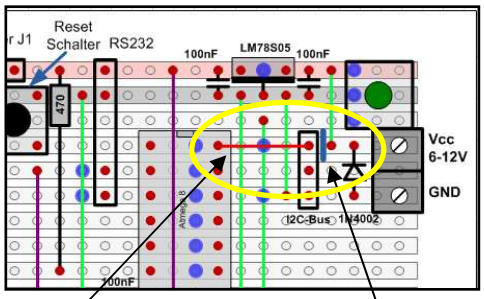
Inhaltsangabe:

Noch ein Hinweis Test- und Programmierplatine:

Dank an Herrn Wolfgang Grandl, der mich nicht nur auf den ersten Fehler aufmerksam gemacht hat, noch viel besser, er hat mir eine Möglichkeit beschrieben, die bereits fertige Platine zu korrigieren, ohne kompliziertes Umsetzen des Festspannungsregler LM78S05.

Hier nun die Lösung:

Eine kleine Anregung für die Leute, die die Platine bereits aufgebaut haben. Es ist wesentlich einfacher, die Leiterbahn, die an Vcc angeschlossen ist, direkt vor dem i²C-Stecker aufzutrennen (blauer Strich), mit einen z.B. Teppichmesser und dann eine Drahtbrücke (rot) zum Pin28 einzulöten. Damit erspart man sich, den Spannungswandler zu versetzen.



Drahtbrücke Leiterbahn auftrennen

Formelfehler im Elektronikgrundkurs:

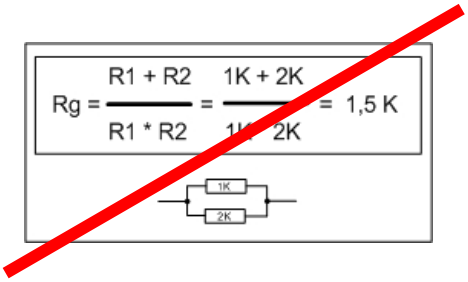
Originaltext aus dem Buch:

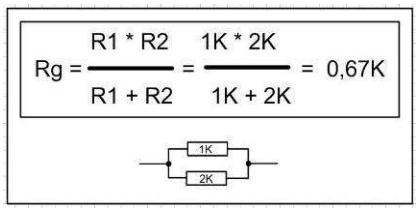
Bei einer Parallelschaltung von Widerständen ist der neue gemeinsame Widerstand immer kleiner als der größte Einzelwiderstand.

Das ist leider falsch. Wieder ein aufmerksamer Leser, der mich informiert hat!
Korrekt muss es heißen:

Bei einer Parallelschaltung von Widerständen ist der neue gemeinsame Widerstand immer kleiner als der **kleinste** Einzelwiderstand.

Als ob das nicht schon peinlich genug wäre, habe ich auch noch die Formel falsch dargestellt.





Fehler in den Programmcodes

Korrektur des Testprogrammes auf Seite 42

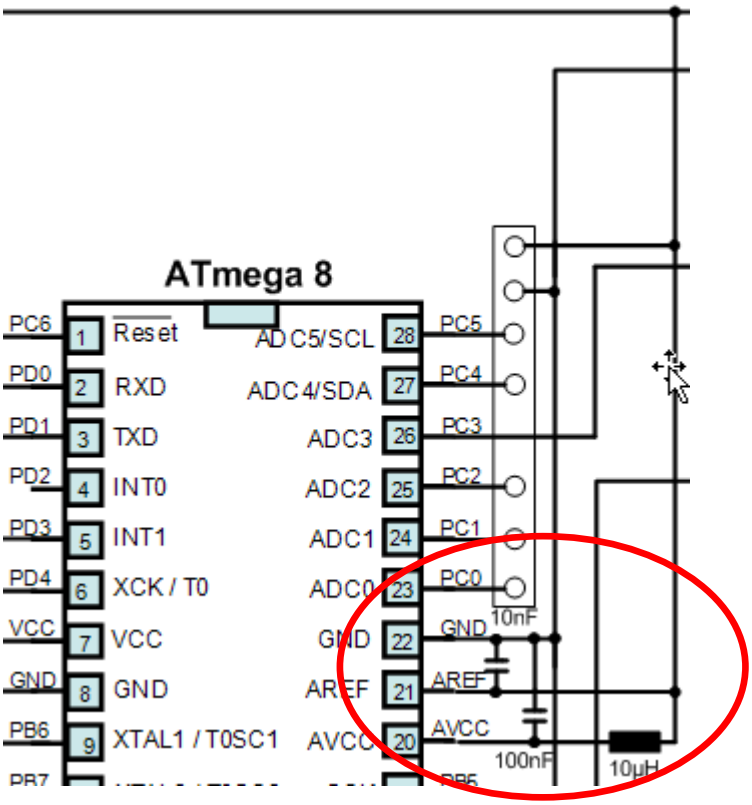
```
1:  #include <avr/io.h>
2:  #include <avr/interrupt.h>
3:  #include <util/delay.h>
4:
5:  // ------------------------------------------------------------------
6:  // FUNKTION  delay ( Millisekunden )  Wartezeit
7:  void  delay  (int loop)
8:  {
9:    int     i;
10:   for (i=0; i < loop; i++)
11:    _delay_ms (1);
12: }
13:
14:
15: // ------------------------------------------------------------------
16: // ************** Hauptprogramm ***************
17: int main (void)
18: {
19:    // Setzt das Richtungsregister von Port B u. D →Alle PINs Ausgang
20:
21:    DDRB = 0xFF;
22:    DDRD = 0xFF;
23:
24:    /* Setzt Port B auf 0x03,Bit 0 und Bit 1 auf High sonst alle Null*/
25:
26:    while (1)
27:    {
28:        PORTB |=  (1 << PB0); // LED an Port B0 gesetzt +5V
29:        delay (1000);        // Zeitschleife mit Verzögerung von 1000ms
30:
31:        PORTB &= ~(1 << PB0);    // LED aus Port B0 zurück auf 0 V
32:        delay (1000);
33:
34:
35:        PORTD |=  (1 << PD6);
36:        delay (1000);
37:
38:        ~~PORTD &= ~(1 << PD7);~~
           PORTD &= ~(1 << PD6);
39:        delay (1000);
40:
41:
42:        PORTD |=  (1 << PD7);
43:        delay (1000);
44:
45:        PORTD &= ~(1 << PD7);
46:        delay (1000);
47:
48:     }  // Ende Endlos Schleife
49:
50:     return (0);
51:
52: }  // Programmende
```

Daten einlesen in den Mikrokontroller und weiterverarbeiten

Digitale Daten über die I0 - Ports einlesen und ausgeben - das hatten wir schon im Buch behandelt. Siehe auch Seite 70 bis 72. Zusammenfassend werden die beiden folgenden Befehle benötigt:
PORTx = Setzen / Ausgeben von Daten an den Ports (Pins/ Beinchen) des Mikrocontrollers.

PINx = Einlesen von Daten die an den Ports (Pins/Beinchen) des Mikrocontrollers anliegen

Was aber nun, wenn man ein analoges Signal einlesen möchte in den Mikrocontroller. Der von mir benutzte ATmega8 bietet hierzu die Ports C0 bis C5 an. Googelt man etwas im Internet, so wird man schnell fündig, wie man die Ports korrekt konfiguriert und auswertet. Wichtig ist allerdings, dass eine entsprechende Beschaltung des ATmega8 vorhanden ist, die als Reverenzspann.veränderte und deutlichere Darstellung, um die Anfänger nicht zu verwirrenng dient. Eine genaue Erklärung gibt es auf Seite 58. Die folgende Abbildung zeigt die externe Beschaltung des Mikrocontrollers. Aus dem Schaltplan der Steuerplatine entnommen, siehe Seite 54.



Bei der Konvertierung der Daten ist eine Besonderheit zu beachten, die einzelnen Ports liefern einen Wertebereich von 0 – 1023. Das heißt, mit einer Variable, die als „uint8_t“ definiert ist, kommen wir nicht aus. Wir benötigen hier eine Variable vom Type „uint16_t“, als Variable mit zweimal 8 Bit, die uns einen Wertebereich von 0 – 65535 liefert. Die eigentliche Messung an den Ports habe ich in eine Funktion „verbannt“, die dann auf bedarf, aus dem Hauptprogramm aufgerufen werden kann. Wichtig, nicht zu vergessen, dass Setzen des Datenrichtungsregisters (DDRC) für Port C im Hauptprogramm.

Hier nun der Programmcode für die Funktion, die die Messung übernimmt und deren Aufruf aus dem Programm heraus.:

```
// Setzen der globalen Variable in der der Wert der AD-Wandlung gespeichert wird
uint16_t     result_ADC = 0;


// setzen des Datenrichtungsregister für Port C
DDRC  = 0x00;
```

```
// ***** Funktion Analog / Digital Wandlung *****
// *****
uint16_t Read_AD_Channel  (uint8_t  mux)          // mux entspricht dem gewaehlten AD-Channel
{
  uint8_t     i;
  uint16_t    result;

  ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);  // Freqenzteiler setzen auf 32
                                                              // und ADC aktivieren
  ADMUX  = mux;                                               // Kanal waehlen  Bit 0  bis  5
  ADMUX &= ~( (1<<REFS1) | (1<<REFS0) );                      // Loeschen der BITs

  // Nach dem Aktivieren des ADC wird ein "Dummy-Readout" empfohlen, man liest
  // also einen Wert und verwirft diesen, um den ADC "warmlaufen zu lassen
  ADCSRA |= (1<<ADSC);                                        // Eine ADC Wandlung
  while ( ADCSRA & (1<<ADSC) )                                // Warten bis Wandlung fertig
  {
     ;  // warten bis ADC Ready
  }
  result = ADCW;                                          // ADCW muss einmal gelesen werden
                                                          // sonst wird Ergebnis der naechsten
                                                          // Wandlung nicht uebernommen
  // Start der eigentlichen Messung
  result = 0;
  for (i=0; i<4; i++)
  {
     ADCSRA |= (1<<ADSC);                          // Eine Wandlung "single conversion"
     while ( ADCSRA & (1<<ADSC) )                  // Es werden vier Wandlungen durchgefuehrt
     {                                             // und dann der Durchschnitt errechnet
        ;  // warten bis ADC Ready
     }
     result = result + ADCW;                       // Wandlungsergebnisse aufaddieren
  }

  ADCSRA &= ~(1<<ADEN);                        // ADC deaktivieren
  result  = result / 4;                        // Summe durch vier teilen = arithm. Mittelwert

  return  result;

}  // END  **** A/D - Wandlung ****
```

Abfrage des AD-Portes z.B. C0 im Hauptprogramm durch Aufruf der Funktion „Read_AD_Channel“:

```
// Auslesen des Channels  "0"
result_ADC = Read_AD_Channel (0x00);
```

Kolumne von Klaus Wellmann

Eine kleine „Motivationsspritze“ oder der Weg ist das Ziel....

Eigentlich soll Konfuzius den Spruch „ Der Weg ist das Ziel“ ausgesprochen haben. Gilt das auch für uns Hobbybastler? Wie lange quält man sich herum bis endlich das Programm läuft? Nehmen wir als Beispiel mal eine beliebige Lichtschaltung einer Verkehrsampel. Fundstelle: Internet / Programmer: Bascom / Version : 2009

Gemäß einer alten Empfehlung, verwendete ich die Bascom-Version V.1.11.9.1. Sie, verehrter Leser/in werden jetzt gelangweilt fragen, na und wo ist das Problem ???? Ganz einfach, es funktioniert nämlich nicht? Ach, und warum? Ich habe doch alles korrekt eingegeben und immer wieder erscheint die Fehlermeldung :

```
Error 202:1;EQU not found, probably using funktions that are
not supportet by the selected Chip (WDTCR) in file C:
Programm/MCS .....usw.
```

Die Lösung des Rätsels: Man lade sich die neuere Bascom-Version V.1.11.9.8 (1996 bis 2009), dann funktioniert es !!! (LEDs blinken gesteuert wie eine Ampel auf)

Mein Ratschlag für andere „Anfänger-User“. Versuche herauszufinden, wann das Programm geschrieben worden ist und prüfe, ob die installierte Bascom-Version auf dem PC zeitlich auch zum Programm paßt... Für diejenigen Leser, denen mein Tipp „ zu banal“ klingt, schiebe ich gleich noch die Erklärung eines Spezialisten hinterher:

…..nach der Suche im Netz und Hinweisen im Bascom Forum habe ich mir das Datenblatt Deines fabrikneuen ATtiny24 noch mal angeschaut und bin auf der Seite 229 fündig geworden. Dort wird mit "update" der Name "wdtcr" in "wdtcsr" umbenannt. in Programmen mit der Bascom-Version 1.11.9.1 fragt der Syntax Check nach "wdtcr" und findet ihn nicht, bei der Attiny 24 in Tiny24.dat . Dort heißt das Register nämlich "wdtcsr" und hat die Adresse $21. Ändert man diesen Eintrag in einen Editor in "wdtcr=$21 und löscht den Eintrag "wdtcsr" = $21 und löscht den Eintrag "wdtcsr=$21, dann wird nach dem Test und der Compilierung keine Fehlermeldung mehr ausgegeben. Dies gilt nur für diese Version, da in der Version Bascom V1.11.9.8 dies schon korrigiert ist, denn dort tritt der Fehler nicht auf.....

→ Wären „Sie“ darauf gekommen?

Nachtrag:
Als die harte Nuss endlich geknackt war und der Autor des Programmes sich überzeugen konnte, dass man sich wirklich ernsthaft mit dem Thema beschäftigt hat, kam noch folgende Antwort, die letztlich meine Freunde und mich bestätigt, hinterher:

Feedback:
Ja - natürlich ist es immer wieder einmal möglich, dass durch Updates an verschiedenen Ecken (AVR, BASCOM, Windows,..) die Welt unnötig kompliziert wird und man dann lange sucht…
Genau an diesen Stellen im Leben des Hobbybastler ist es wichtig, dass man das Prinzip verstanden hat. An der Stelle will ich nicht versäumen, auch auf die neue Version BASCOM-

AVR 2.0.7.7 hinzuweisen. Es bietet neben einer langen Liste von Bug-Fixes und neuen Definitions-Files für neue Mikrocontroller sogar neue Features. Besonders hilfreich ist der verbesserte Code-Explorer. Wenn dieser aktiviert ist (Default-Einstellung) und man bei gedrückter Shift-Taste mit der Maus über seinen Code streicht, dann wird bei jeder Variable ein kleines Popup angezeigt, das die Variablen-Definition enthält. Beim Beispiel-Foto wird darauf hingewiesen, dass die Variable „Voltage“ als Single definiert wurde. Bei Port- oder Pin-Aliases wird z. B. angezeigt, welcher Controller-Pin dem Alias zugewiesen wurde. Allein das ist bei längeren Code-Abschnitten mit vielen Variablen sehr hilfreich, da man nicht immer hin- und her-scrollen muss, bloß weil man vergessen hatte, wie eine Variable definiert war. Alle Besitzer einer BASCOM-Lizenz und auch alle, die lediglich die Demo-Version für gelegentliche kleinere Projekte auf der Platte haben, tun daher gut daran, sich das neue Update unter www.mcselec.com herunterzuladen.

Fazit:
Der phantasielose Betrachter wird sagen, so viel Arbeit investiert nur für das Aufleuchten ein paar Lämpchen? Solche Leute, verehrter Leser/verehrte Leserin werden wir nie als Modellbaufreunde gewinnen können. Wenn darüber hinaus, solche Leute dann auch noch meinen, so etwas wie Blinklichter, Fernsteuerungen, Verstärker usw. gibt es doch fertig zu kaufen, dann spürt man an dieser Stelle ist „Hopfen und Malz“ verloren.

Es gibt natürlich auch das Mittelding, also die gemäßigtere Beurteilung meiner Frau: Ja, das ist wirklich toll, wie Du das alles gemacht hast, aber „Liebling“, wo drin liegt da der praktische Nutzen? Dem nicht genug. Ich werde zum Beispiel, nach langem Basteln mit vielen vergeblichen und auch erfolgreichen Versuchen gefragt, sag mal: Warum suchst Du Dir nicht ein Hobby aus, was funktioniert?

Natürlich kann man jemandem, der nicht vom Virus der Faszination von Mikrocontrollern infiziert ist, nicht erklären, worin der praktische Nutzen einer endlich funktionierenden Ampelschaltung liegt. Nur der/die begeisterte Modellbauer/in kennt die Freude, wenn zum Beispiel der Roboter endlich funktioniert oder die LEDs endlich das machen, was sie sollen. Nämlich im Takt zu leuchten. Die Freude und Befriedigung aus eigenen Stücken etwas geleistet zu haben, wobei die angenommene Hilfe von Dritten nichts Schlechtes sein muss, das ist das (Hoch-)Gefühl, was das Leben ausmacht!

Mein Plädoyer für alle Freunde der Programmierung, sucht Euch Projekte heraus, die Ihr realistischer Weise schaffen könnt. Habt Geduld, sucht Euch in Büchern oder beim Herausgeber direkt Hilfe. Aber vergesst nicht darüber eure „Bessere Hälfte“, denn die hat auch ein Recht darauf, beachtet zu werden. Oder es könnte sein, dass ein Blitz und Donnerschlag bei Euch zu Hause nicht nur die Schaltung mit Hochspannung, sondern auch die Beziehung verstört.....

Klaus Wellmann

Ausblick auf die nächste Ausgabe des Newsletters

- Weiter Korrekturen des Buches

- Vorstellen des laufenden Projektes: Weg suche durch ein Labyrinth

- Vorbereitungen „Hardware“ für die Modellbaumesse in Mainz an der wir teilnehmen wollen

Freuen würde ich mich über etwas Feedback von Ihnen und vielleicht einen Bericht oder Projekt, welches Sie den anderen Lesern vorstellen möchten. Z.B. ein Bericht über den Aufbau und Programmierung des Hexabots bzw. den Schwierigkeiten beim Zusammenbau wären sicherlich hilfreich.

Mit freundlichen Grüßen
Dipl. –Ing. Peter Schneider