2.8 Atmega-Peripherie/ADC

2.8.1 Einlesen von Tasten per ADC

Auf dem RN-Board werden die vorhandenen Tasten (außer der Reset-Taste) alle über einen einzigen Eingang eingelesen. Das funktioniert, weil der betreffende Eingang PA7 als Analog-Digital-Wandler arbeiten kann. Die Widerstände selbst sind über ein Netzwerk miteinander verbunden (Abbildung 1). Im Ruhezustand liegt der Eingang über den Pull-Up-Widerstand R_X am positiven Betriebsspannungsanschluss. Bei jedem Drücken einer Taste wird eine bestimmte Spannung an den Eingang gelegt.

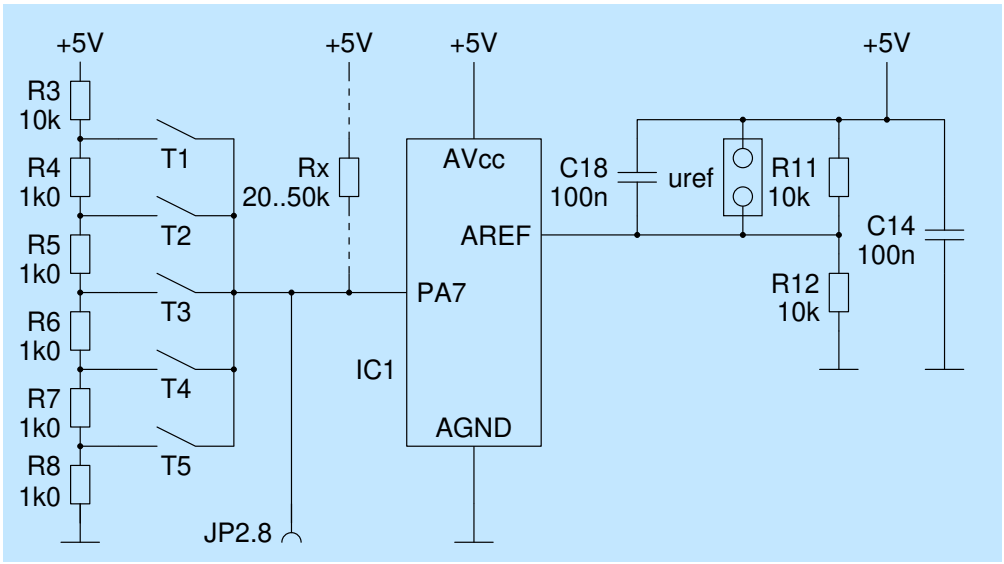

Abbildung 1: Anschluss der Taster am RN-Board

2.8.2 Register des eingebauten ADCs

In den ATmega-Controllern ist ein 10-Bit-Analog-Digital-Umsetzer eingebaut. Er wird über verschiedene Register angesteuert und ausgelesen.

Mit dem ADMUX-Register kann man den gewünschten ADC-Kanal auswählen und die Art der Referenzspannung bestimmen (Abbildung 1).

- Die Bits MUX0 bis MUX3 bilden eine Dualzahl. Sie ist die Nummer des Kanals. Beim ATmega32 gibt es acht Kanäle. Dabei entspricht PA0 dem Kanal 0, PA1 entspricht Kanal 1 usw. Beim ATmega8 gibt es nur sechs Kanäle[1]. Sie liegen an den Anschlüssen PC0 (Kanal 0) bis PC5 (Kanal 5).
- Die obersten beiden Bits im Register (REFS1 und REFS0) geben an, welche Referenzspannung benutzt werden soll. Schließlich gibt der ADC einen Wert aus nach der Formel:

$$\frac{x_{ADC}}{1024} = \frac{U}{U_{ref}} \tag{1}$$

  Folgende Referenzspannungen sind möglich:

  a) REFS1=0, REFS0=1: die Betriebsspannung von etwa 5 V an Anschluss AVcc
  b) REFS1=0, REFS0=0: eine extern eingespeiste Spannung an Anschluss AREF (Grenzwerte beachten!)

[1] zumindest bei den Typen im DIP-Gehäuse

c) REFS1=1, REFS0=1: eine intern erzeugte Spannung von 2,56 V

- Das Bit mit dem Namen ADLAR bestimmt, ob der 10-Bit-Ausgabewert in den Datenregistern rechts- oder linksbündig ist (siehe unten).

Name									Adresse
	Bit	7	6	5	4	3	2	1	0	
ADMUX		REFS1	REFS0	ADLAR		(MUX3)	MUX2	MUX1	MUX0	SFR 7
	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
	Init.	0	0	0	0	0	0	0	0	

Tabelle 1: ADC-Multiplexregister ADMUX

Zur weiteren Steuerung gibt es das kombinierte Kontroll- und Statusregister ADCSR (Abbildung 2).

Name									Adresse
	Bit	7	6	5	4	3	2	1	0	
ADCSRA		ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	SFR 6
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	Init.	0	0	0	0	0	0	0	0	

Tabelle 2: ADC-Kontroll- und Statusregister ADCSR

- Das Bit ADEN schaltet den AD-Umsetzer ein oder aus. Mit ADSC=1 wird eine Umsetzung gestartet. Nach Ende der Umsetzung setzt der Controller dieses Bit von selbst wieder auf 0.
- Mit dem ADFR-Bit kann man einen Freilauf-Betrieb ermöglichen, bei dem nach jeder Umsetzung sofort wieder eine neue Umsetzung erfolgt.
- Das Bit ADIF ist ein Flag, mit dem der Controller anzeigt, dass die Umsetzung zu Ende ist und ein neuer Wert vorliegt.
- Das Bit ADIE kann einen Interrupt einschalten, der nach jeder erfolgten Umsetzung auftritt.
- Die drei niederwertigen Bits ADPS2 bis ADPS0 ergeben ein drei-Bit-Wort W, das die Taktfrequenz des Umsetzers f_{ADC} festlegt:

$$f_{ADC} = f_{CPU}/N_{ADC} \text{ mit } N_{ADC} = 2^W \tag{2}$$

  Der minimale Teilerfaktor $N_{ADC,min}$ ist allerdings 2. Die Taktfrequenz, so sagt der Hersteller, soll zwischen $f_{ADC,min} = 50\,\text{kHz}$ und $f_{ADC,max} = 200\,\text{kHz}$ liegen. Damit kann der ADC verwendet werden, sobald die CPU-Taktfrequenz mindestens 100 kHz beträgt.

Schließlich gibt es die Daten-Register ADCL und ADCH, die nach einer Umsetzung mit ihren zusammen 16 Bit das 10-Bit-Datenwort enthalten. Falls das Bit 5 von ADMUX gleich 1 ist, ist das Ergebnis linksbündig untergebracht (Abbildung 3), ansonsten rechtsbündig (Abbildung 4).

2.8.3 Benutzung

Im einfachsten Fall setzt man ADMUX und ADCSRA auf die richtigen Werte im Modus Einzelwandung und setzt das Startbit ADSC. Nun wartet man ab, bis das Startbit wieder zurückgesetzt wurde. Alternativ kann man auch auf das Setzen des Flags ADIF warten. Sobald das Warten beendet ist, kann man das Ergebnis in ADCH und ADCL abholen.

Name									Adresse
	Bit	7	6	5	4	3	2	1	0	
ADCH		ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	SFR 5
	R/W	R	R	R	R	R	R	R	R	
	Init.	0	0	0	0	0	0	0	0	
	Bit	7	6	5	4	3	2	1	0	
ADCL		ADC1	ADC0							SFR 4
	R/W	R	R	R	R	R	R	R	R	
	Init.	0	0	0	0	0	0	0	0	

Tabelle 3: ADC-Datenregister ADCL und ADCH bei ADLAR=1

Name									Adresse
	Bit	7	6	5	4	3	2	1	0	
ADCH								ADC9	ADC8	SFR 5
	R/W	R	R	R	R	R	R	R	R	
	Init.	0	0	0	0	0	0	0	0	
	Bit	7	6	5	4	3	2	1	0	
ADCL		ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	SFR 4
	R/W	R	R	R	R	R	R	R	R	
	Init.	0	0	0	0	0	0	0	0	

Tabelle 4: ADC-Datenregister ADCL und ADCH bei ADLAR=0

```
1  .include "/usr/share/avra/m32def.inc"
2  ; Spannungswert an ADC einlesen und hexadez. ausgeben an Port C
3  ;
4          ldi r16, 0xff  ; PORTC als Ausgang
5          out DDRC, r16
6          sbi PORTA, PA7 ; PA7 als Eingang, Pullup-Widerstand setzen
7  ;
8  ; REFS1=0: Referenz ist AVcc
9  ; REFS0=1
10 ; ADLAR=1: Ergebnis linksbuendig
11 ; MUX=0111: Kanal 7
12 ;
13         ldi r16, (1<<REFS0)|(1<<ADLAR)|(1<<MUX2)|(1<<MUX1)|(1<<MUX0)
14         out ADMUX, r16
15 ;
16 ; Wandler einstellen
17 ; ADEN=1: Wandler ein
18 ; ADSC=1: Start
19 ; ADFR=0: Einzelumsetzung
20 ; ADIF__: Int.-Flag nicht benutzt
21 ; ADIE=0: keine Int.-Freigabe
22 ; ADPS2-ADPS0: fCPU=16 MHz, fADC=125 kHz, NTeiler=128, Bits=0b111
23 ;
24         ldi r16, (1<<ADEN)|( 1<ADSC)|0b111
25         out ADCSRA, r16
26 warte:
```

```
27          sbic ADCSRA, ADSC; warte, bis ADSC zurueckgesetzt wurde
28          rjmp warte
29
30          in r16, ADCH      ; Ausgabe
31          out PORTC, r16
32
33          sbi ADCSRA, ADSC
34          rjmp warte
```