

MIKROKONTROLLER & I²C BUS



www.makerconnect.de

<https://www.makerconnect.de/resource>

Attiny 841 - ein IC und
5 verschiedene Module
Teil 6 - ADC

I²C Bus und der
Attiny 841



Copyright

Sofern nicht anders angegeben, stehen die Inhalte dieser Dokumentation unter einer „Creative Commons - Namensnennung-NichtKommerziell-Weitergabe unter gleichen Bedingungen 3.0 DE Lizenz“



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbeachtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung / Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfwerkstätten ist das Betreiben dieser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle einer Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versorgungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsversorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehler muss das Gerät unverzüglich außer Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Gefahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

Attiny 841 – ADC (A/D-Wandler)

Was macht der ADC?

Wenn es darum geht, Spannungen zu messen, wird der Analog-/Digital-Wandler (kurz: A/D-Wandler) oder englisch **Analog Digital Converter** (ADC) benutzt. Er konvertiert eine elektrische Spannung in eine Digitalzahl. Prinzipiell wird dabei die Messspannung mit einer Referenzspannung verglichen. Die Zahl drückt daher das Verhältnis der Messspannung zu dieser Referenzspannung aus. Sie kann in gewohnter Weise von einem Mikrocontroller weiterverarbeitet werden.

Elektronische Grundlagen

Die ADC-Versorgungsspannung (**AVCC**) darf maximal um $\pm 0,3V$ von der Versorgung des Digitalteils (**VCC**) abweichen, jedoch nicht 5,5V überschreiten. Die externe Referenzspannung **V_{REF}** darf nicht kleiner als die im Datenblatt unter ADC Characteristics als **VREFmin** angegebene Spannung und nicht größer als **AVCC** sein.

Der Eingangswiderstand des ADC liegt in der Größenordnung von einigen Megaohm, so dass der ADC die Signalquelle praktisch nicht belastet. Des weiteren enthält der Mikrocontroller eine sog. **Sample&Hold** Schaltung. Dies ist wichtig, wenn sich während des Wandlungsvorgangs die Eingangsspannung verändert, da die AD-Wandlung eine bestimmte Zeit dauert. Die Sample&Hold-Stufe speichert zum Beginn der Wandlung die anliegende Spannung und hält sie während des Wandlungsvorgangs konstant.

Referenzspannung

Der ADC benötigt für seine Arbeit eine Referenzspannung. Dabei gibt es 2 Möglichkeiten:

- interne Referenzspannung
- externe Referenzspannung

Interne Referenzspannung

Mittels Konfigurationsregister können verschiedene interne Referenzspannungen eingestellt werden. Dies umfasst beim Attiny 841 die Versorgungsspannung Vcc sowie eine vom AVR bereitgestellte interne Spannung von 1,1V, 2,2V oder 4,096V. Es ist jedoch zu beachten, dass die interne Referenzspannung ca. $\pm 10\%$ vom Nominalwert abweichen kann. Genaue Angaben dazu bitte dem Datenblatt entnehmen.

Externe Referenzspannung

Wird eine externe Referenz verwendet, so wird diese an **A_{REF}** angeschlossen. Aber aufgepasst! Wenn eine Referenz in Höhe der Versorgungsspannung benutzt werden soll, so ist es besser, dies über die interne Referenz zu tun. Außer bei anderen Spannungen als 5V bzw. 2,56V gibt es eigentlich keinen Grund, an AREF eine Spannungsquelle anzuschließen. In Standardanwendungen fährt man immer besser, wenn die interne Referenzspannung mit einem Kondensator an **A_{REF}** benutzt wird.

Ein paar ADC-Grundlagen

Der ADC ist ein 10-Bit ADC, d.h. er liefert Messwerte im Bereich 0 bis 1023. Liegt am Eingangskanal 0V an, so liefert der ADC einen Wert von 0. Hat die Spannung am Eingangskanal die Referenzspannung erreicht, so liefert der ADC einen Wert von 1023. Unterschreitet oder überschreitet die zu messende Spannung diese Grenzen, so liefert der ADC 0 bzw. 1023. Wird die Auflösung von 10 Bit nicht benötigt, so ist es möglich die Ausgabe durch ein Konfigurationsregister so einzuschränken, dass ein leichter Zugriff auf die 8 höchstwertigen Bits möglich ist.

Wie bei vielen analogen Schaltungen, unterliegt auch der ADC einem Rauschen. Das bedeutet, dass man nicht davon ausgehen sollte, dass der ADC bei konstanter Eingangsspannung auch immer denselben konstanten Wert ausgibt. Ein "Zittern" der niederwertigsten 2 Bits ist durchaus nicht ungewöhnlich. Besonders hervorgehoben werden soll an dieser Stelle nochmals die Qualität der Referenzspannung. Diese Qualität geht in erheblichem Maße in die Qualität der Wandler Ergebnisse ein. Die Beschaltung von **AREF** mit einem Kondensator ist die absolut notwendige Mindestbeschaltung, um eine einigermaßen akzeptable Referenzspannung zu erhalten. Reicht dies nicht aus, so kann die Qualität einer Messung durch *Oversampling* erhöht werden. Dazu werden mehrere Messungen gemacht und deren Mittelwert gebildet.

Betriebsarten des ADC

Der **ADC** kann in zwei verschiedenen Betriebsarten verwendet werden:

Einfache Wandlung (Single Conversion)

- In dieser Betriebsart wird der Wandler bei Bedarf vom Programm angestoßen für jeweils eine Messung.

Frei laufend (Free Running)

- In dieser Betriebsart erfasst der Wandler permanent die anliegende Spannung und schreibt diese in das **ADC Data Register**.

Umrechnung des ADC Wertes in eine Spannung

Der Attiny 841 verfügt über einen ADC von 10 Bit. Daraus ergibt sich die folgende Rechnung:

$$\text{Bereichsbreite} = \frac{\text{Referenzspannung}}{\text{Maximalwert} + 1}$$

Bereichsbreite - maximale Auflösung

Referenzspannung - verwendete Referenzspannung, Beispiel 5V

Maximalwert - bei 10 Bit ergibt sich ein Bereich von 0-1023

$$\text{Bereichsbreite} = \frac{5V}{1024} = 0,004883 V = 4,883 mV$$

Bei einem ADC von 10 Bit und einer Referenzspannung von 5V ergibt sich eine maximale Auflösung von 4,883 mV.

Beispiel:

Die gemessene Spannung ergibt sich aus den folgenden Werten:

$$\text{Spannung} = \text{ADC Wert} \times 4,883 mV$$

Spannung - gemessene Spannung

ADC Wert - Angabe des ADC

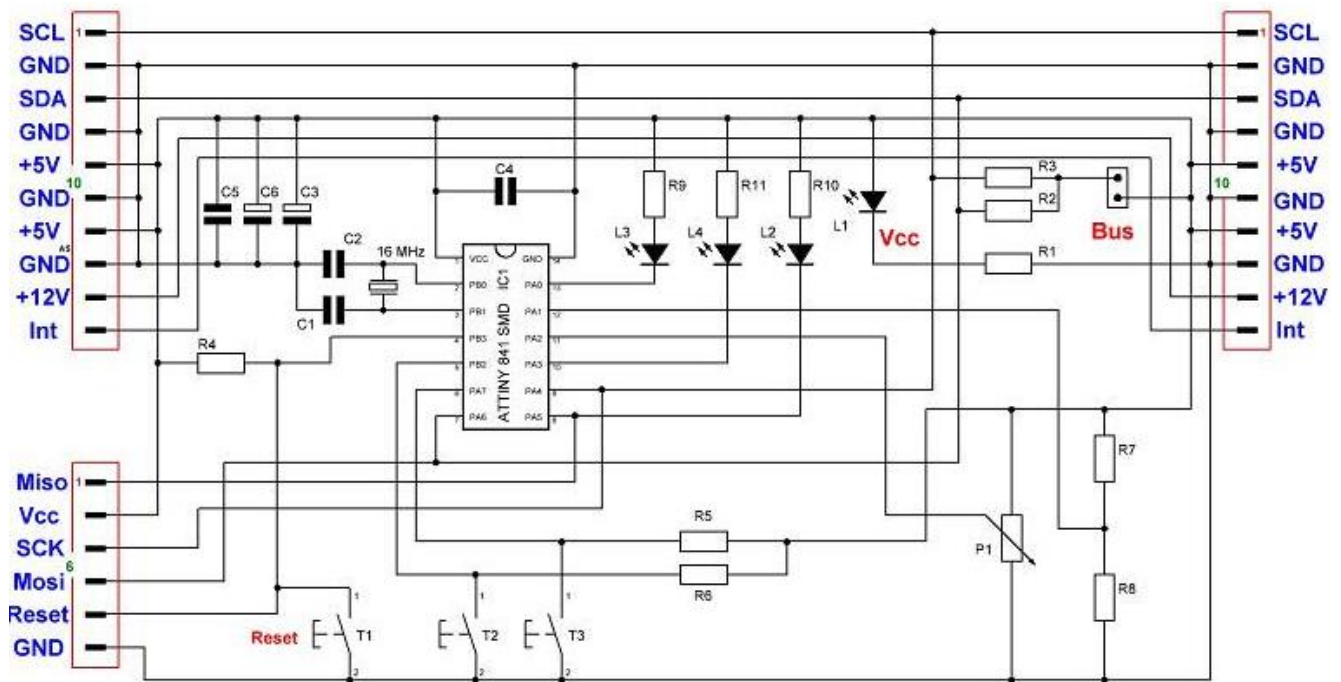
Diese Berechnung gilt bei einer Auflösung von 10 Bit und einer Referenzspannung von 5V. Als Beispiel verwende ich wieder meine Platine 173 und die folgenden Ein- bzw. Ausgänge:

Eingang - PA2 --> Regler (0 bis 5V)

Ausgang - PA5 --> LED 2

PA0 --> LED 3

PA3 --> LED 4



Platine P173 mit 3 x LEDs und Regler

Als nächste werde ich die verschiedenen Einstellungen und Register beschreiben. Das komplette Programm kommt am Ende.

```
ADMUXB &=~((1<<REFS2)|(1<<REFS1)|(1<<REFS0)); // Register ADMUXB
// REF-Auswahl Referenzspannung, auf Vcc(5V)
ADCSRA |=((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)); // Register ADCSRA
// ADPS - Teilungsfaktor 16MHz/128=125kHz
ADCSRA |= (1<<ADEN); // ADC aktivieren
ADCSRA |= (1<<ADSC); // eine ADC-Wandlung "single conversion"
while (ADCSRA & (1<<ADSC)) // auf Abschluss der Konvertierung warten
```

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	REFS2	REFS1	REFS0	—	—	—	GSEL1	GSEL0	ADMUXB
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Im Register **ADMUXB**, Bit 5, 6 und 7, wird die Referenzspannung ausgewählt. Beim Aufruf des Registers sind die Bits im Grunde schon auf „0“ gesetzt. Habe sie trotzdem noch mal auf „0“ gesetzt um eine andere Auswahl zu zeigen.

```
ADMUXB &=~(1<<REFS2); // Register ADMUXB, REFS2=0
ADMUXB |=((1<<REFS1)|(1<<REFS0)); // Register ADMUXB, REFS1=1, REFS0=1
// REF-Auswahl Referenzspannung auf 4,096 V

ADMUXB &=~((1<<REFS2)|(1<<REFS1)); // Register ADMUXB, REFS2=0, REFS1=0
ADMUXB |= (1<<REFS0); // Register ADMUXB, REFS1=1
// REF-Auswahl Referenzspannung auf 1,1 V
```

In der Tabelle 16-4 des DB sind alle Angaben zur Einstellung der Spannungen angegeben.

Beginne wir mit den Einstellungen die nur beim Start vorgenommen werden müssen:

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Im Register **ADCSRA**, Bit 2, 1 und 0 werden die Einstellungen für den Prescaler (Vorteiler) angegeben. Beim Einschalten stehen diese auf „0“ und werden auf „1“ gesetzt. Damit ergibt sich ein Vorteiler von 128. Nach Angabe im Datenblatt sollte diese Einstellung zwischen 100 kHz und 200 kHz liegen. Die genauen Angaben stehen in der Tabelle 16-6.

```
ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)); // Register ADCSRA
// ADPS - Teilungsfaktor 16MHz/128=125kHz
```

Zum Aktivieren des ADC muss im Register **ADCSRA**, Bit 7 auf „1“ gesetzt werden

```
ADCSRA |= (1<<ADEN); // ADC aktivieren
```

Wenn im Register **ADCSRA**, Bit 6 auf „1“ gesetzt wird, wird die Single Conversion gestartet

```
ADCSRA |= (1<<ADSC); // eine ADC-Wandlung "single conversion"
```

Mit $(ADCSRA \& (1<<ADSC))$ wird auf den Abschluss der Konvertierung gewartet.

Kommen wir nun zu den Einstellungen die bei jeder Messung gemacht werden:

```
ADMUXA |= (1<<MUX1); // Register ADMUXA
// MUX - Auswahl welcher Eingang - 000010 - ADC2
ADCSRA |= (1<<ADSC); // Register ADCSRA
// ADSC - Start Konvertierung "single conversion"
while (ADCSRA & (1<<ADSC))
{
    // auf Abschluss der Konvertierung warten
}
return ADCW; // ADC mit 10 Bit auslesen und zurückgeben
```

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	–	–	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUXA
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Mit $ADMUXA \& (1<<MUX1)$ wird der Eingangspin ausgewählt. Die mögliche Auswahl ist in der Tabelle 16-3 des DB dargestellt

Mit $ADCSRA \& (1<<ADSC)$ erfolgt wieder die Auswahl „single conversion“

Mit $(ADCSRA \& (1<<ADSC))$ wird wieder auf den Abschluss der Konvertierung gewartet.

Das Ergebnis der Messung steht in ADCH und ADCL zur Verfügung und wird als ADCW zusammengefasst und zurückgegeben. Dieser Wert hat eine Grösse von 10 Bit.

Es ist auch eine Rückgabe von 8 Bit möglich.

```
ADCSRB |= (1<<ADLAR); // bei 8 Bit links angeben
while (ADCSRA & (1<<ADSC))
{
    // auf Abschluss der Konvertierung warten
}
//return ADCW; // ADC auslesen und zurückgeben bei 10 Bit
return ADCH; // 8 Bit mit ADLAR=1
```

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	–	–	–	–	ADLAR	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Wenn eine Umwandlung abgeschlossen ist, befindet sich der gemessene Wert in den Registern ADCL und ADCH.

Einschaltzustand **ADLAR = 0**

Bit	15	14	13	12	11	10	9	8	
0x07 (0x27)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x06 (0x26)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Wird mit **ADCSRB |= (1<<ADLAR)** ADLAR auf „1“ gesetzt ...

Bit	15	14	13	12	11	10	9	8	
0x07 (0x27)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x06 (0x26)	ADC1	ADC0	–	–	–	–	–	–	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

... stehen unsere benötigten Angaben in **ADCH**. Die Rückgabe erfolgt mit **return ADCH**;

Sehen wir uns noch die Auswertung an. Habe einfach eine Schleife mit 3 Werten angegeben. Die 3 Werte sind 28, 128 und 228. Diese sind im Bereich von 8 Bit. Beim langsamen drehen des Potis schalten, je nach Grösse von **adcval**, die angegebenen LEDs ein. Am Ende leuchten alle LEDs. Beim langsamen zurückdrehen schalten die LEDs der Reihe nach wieder aus. Wird 10 Bit verwendet können die Werte zwischen 0 und 1023 liegen. Dabei ist 0 eigentlich Sinnlos, da bei einer kleinen Berührung die Led schon leuchtet. Besser ist es einen Wert > 50 zu wählen. Der Wert der gemessenen Spannung wird in der **adcval** angegeben.

```
while(1)
{
    adcval = ADC_Read();
    if(adcval>=28)                                // mach was mit adcval
    {
        PORTA &=~(1<<PINA5);                    // LED2 ein
    }
    else
    {
        PORTA |= (1<<PINA5);                     // aus
    }
    if(adcval>128)                                // mach was mit adcval
    {
        PORTA &=~(1<<PINA0);                    // LED3 ein
    }
    else
    {
        PORTA |= (1<<PINA0);                     // aus
    }
}
```



```

if(adcv<228)                                // mach was mit adcv
{
    PORTA &=~(1<<PINA3);                    // LED4 ein
}
else
{
    PORTA |= (1<<PINA3);                     // aus
}
}

```

Am Ende noch der komplette Code:

```

/* ATB_Ati841_P173ADC_Prg10.c
 * Created: 07.06.2021 14:46:37 Author : hjsee */

#define F_CPU 16000000UL                      // Angabe der Frequenz, wichtig für die Zeit
#include <util/delay.h>                        // Einbindung Datei Pause
#include <avr/io.h>                            // Einbindung Datei Ausgänge
#include <avr/interrupt.h>

void init_ADC()
{
    ADMUXB &=~((1<<REFS2)|(1<<REFS1)|(1<<REFS0)); // Register ADMUXB
    // REF-Auswahl Referenzspannung, auf Vcc(5V)
    ADCSRA |=((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)); // Register ADCSRA
    // ADPS - Teilungsfaktor 16MHz/128=125kHz
    ADCSRA |= (1<<ADEN);                       // ADC aktivieren
    ADCSRA |= (1<<ADSC);                       // eine ADC-Wandlung "single conversion"
    while (ADCSRA & (1<<ADSC))
    {
        // auf Abschluss der Konvertierung warten
    }
}

uint16_t ADC_Read()                          // ADC Einzelmessung
{
    ADMUXA |= (1<<MUX1);                       // Register ADMUXA
    // MUX - Auswahl welcher Eingang - 000010 - ADC2
    ADCSRA |= (1<<ADSC);                       // Register ADCSRA
    // ADSC - Start Konvertierung "single conversion"
    ADCSRB |= (1<<ADLAR);                      // bei 8 Bit links
    while (ADCSRA & (1<<ADSC))
    {
        // auf Abschluss der Konvertierung warten
    }
    //return ADCW;                             // ADC auslesen und zurückgeben bei 10 Bit
    return ADCH;                               // 8 Bit mit ADLAR=1
}

int main(void)
{
    DDRA=0b00101001;                          // Port A auf Ausgang schalten
}

```



```

PORTA=0b00101001;           // Port A auf aus
uint8_t adcval;              // 16 Bit mit ADCW
init_ADC();
while(1)
{
    adcval = ADC_Read();
    if(adcval>=28)             // mach was mit adcval
    {
        PORTA &=~(1<<PINA5); // LED 2 ein
    }
    else
    {
        PORTA |=(1<<PINA5);   // aus
    }
    if(adcval>128)            // mach was mit adcval
    {
        PORTA &=~(1<<PINA0); // LED 3 ein
    }
    else
    {
        PORTA |=(1<<PINA0);   // aus
    }
    if(adcval>228)            // mach was mit adcval
    {
        PORTA &=~(1<<PINA3); // LED 4 ein
    }
    else
    {
        PORTA |=(1<<PINA3);   // aus
    }
}
}

```

Einige Teile des Textes wurden zur besseren Übersicht **farblich** gestaltet.

Die Nutzung erfolgt auf eigenes Risiko.

Ich wünsche viel Spaß beim Bauen und programmieren.

Achim

myroboter@web.de

Quellenangabe:

https://www.mikrocontroller.net/articles/AVR-Tutorial:_ADC

https://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/Analoge_Ein-und_Ausgabe#ADC_28Analog_Digital_Converter.29

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8495-8-bit-AVR-Microcontrollers-ATtiny441-ATtiny841_Datasheet.pdf