

 **Fix Me!** Sorry, this page is not fully translated, yet.

Proto-Counter



The versatile microcontroller module

- ATtiny4313 controller with 4 kbyte flash memory
- bright 7-segment display with 3 digits
- two pushbuttons
- 5 resp. 7 I/Os for your project
- 8 MHz crystal
- analog knob (optional)
- additional I/Os by using a shift register (optional)
- easy programmierung via ISP port
- runs on 3,3 or 5 volts
- use the Arduino-IDE for programming

Description



pre-assembled PCB


Egal ob Zähler, Timer oder Regler - mit dem Proto-Counter hast du im Handumdrehen deine Mikrocontroller-Anwendung verwirklicht. Er bringt alles mit, was man an Grundfunktionen benötigt: Anzeige, Taster, Drehknopf (optional) und einige I/Os für all die vielen Geräte und Helferlein, die du mal eben schnell realisieren möchtest. Die vorgefertigte Arduino-Library erleichtert dabei die Erstellung eigener Software.

Die Anschlüsse und Befestigungsbohrungen sind im Raster von 2,54 mm (1/10 Zoll) angelegt, so dass sich der Proto-Counter problemlos auf eine Lochrasterplatine aufstecken lässt. Alternativ dazu kann

die Verbindung zu deiner Schaltung auch mit einem Flachbandkabel hergestellt werden.

Aufbau

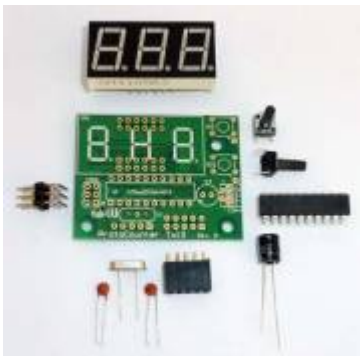



 soldering tools

Durch die bedrahteten Bauteile ist der Aufbau leicht zu bewältigen und schnell erledigt. Alle SMD-Bauelemente sind bereits vorbestückt.

Benötigt werden die üblichen Werkzeuge zum Löten von Elektronikschaltungen. Grundlegende Fähigkeiten und Kenntnisse im Löten und Bestücken sind erforderlich.

Bauteile



 Proto-Counter-Komponenten

Der Bausatz enthält:

- 1 vorbestückte Proto-Counter-Platine
- 1 7-Segment-Display, 3-stellig
- 2 Taster
- 1 Mikrocontroller (ATtiny4313)
- 1 Quarz (8 MHz)
- 2 Keramikkondensatoren (15 pF)
- 1 Elektrolytkondensator (100 μ F)
- 1 Stiftleiste 2×3-polig, gewinkelt
- 1 Buchsenleiste 2×5-polig, gerade

Bauanleitung

1. Zuerst werden der Quarz und die beiden Keramikkondensatoren bestückt. Die Keramikkondensatoren kommen in Pin1+2 bzw. Pin3+4 von CON3. Wer keine quarzgenaue Zeitbasis braucht, lässt stattdessen Quarz und Kondensatoren einfach weg und verwendet den internen RC-Oszillator. Hierdurch stehen an CON3 nun zwei weitere I/Os (PA0 und PA1) für eigene Zwecke zur Verfügung.



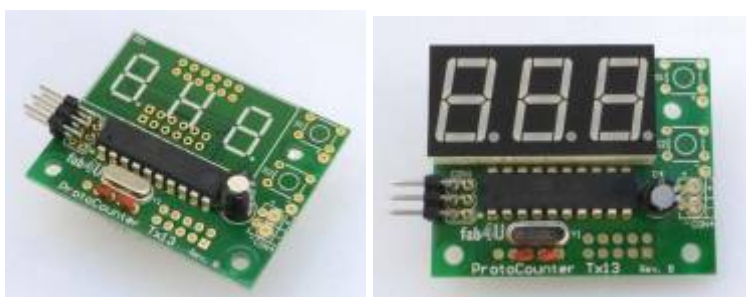
2. Danach folgt die gewinkelte Stiftleiste. Sie dient später zur Programmierung (Atmel ISP).



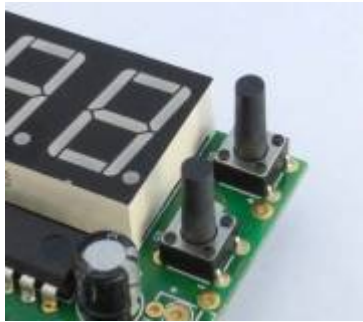
3. Nun kommt der Prozessor an die Reihe. Damit er sich leichter in die Platine einsetzen lässt, empfiehlt es sich, die Beinchen vorher entsprechend dem Lochabstand etwas zusammenzubiegen. Dies ist mit einem Pin-Abrichtwerkzeug schnell erledigt. Beim Bestücken bitte auf die Polarität achten. Die Kerbe am IC muss mit dem Bestückungsaufdruck auf der Platine übereinstimmen.



4. Danach folgt der Elektrolytkondensator. Hier bitte ebenfalls auf die Polarität achten. Der Minuspol ist am Bauteil durch einen weissen Balken markiert. Auch beim Display ist die richtige Orientierung wichtig. Die Dezimalpunkte liegen beim Prozessor.



5. Schließlich werden noch die beiden Taster neben dem Display eingelötet.



6.



Fertig aufgebauter ProtoCounter

Zum Schluss wird auf der Rückseite die Buchsenleiste bestückt und von der Vorderseite verlötet. Wer den ProtoCounter über ein Flachbandkabel mit seiner Applikation verbinden will, kann die Buchse auch durch einen 10-poligen Wannenstecker ersetzen.



Und schon ist der ProtoCounter fertig! Jetzt musst du nur noch die Versorgung anschliessen (5 V=) und den Prozessor programmieren.

Hardware

Anschlüsse

ISP-Anschluss

Der ISP-Anschluss entspricht der bei Atmel-AVR-Prozessoren üblichen Belegung.

< 16em 6em 2em 2em 6em >			
Signal	Pin	Pin	Signal
MISO	1	2	VCC

SCK	3	4	MOSI
RESET	5	6	GND

Applikations-Port

Hier sind die I/O-Pins von Port B herausgeführt. Es sind auch die Pin-Bezeichnungen unter Arduino angegeben. Die Pins PB5-7 werden für die Ansteuerung des Displays und die Abfrage der Taster benötigt.

< 28em 6em 6em 2em 2em 6em 6em >					
Arduino	Signal	Pin	Pin	Signal	Arduino
D9	PB0	1	2	PB1	D10
D11*	PB2	3	4	PB3	D12*
D13*	PB4	5	6	PB5	D14
D15	PB6	7	8	PB7	D16
	GND	9	10	VCC	

* = PWM port

Zusatz-Port

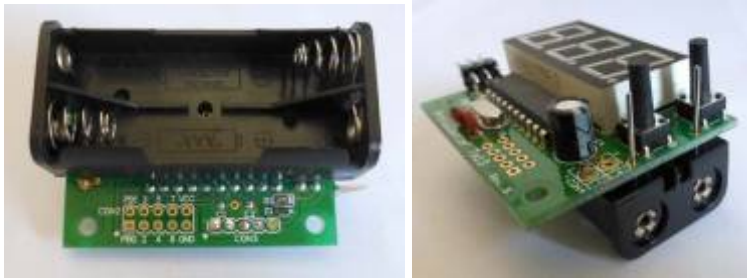
Betreibt man den Prozessor über den internen RC-Oszillator, braucht man den Quarz und die beiden Keramik Kondensatoren nicht zu bestücken. Es stehen an CON3 dann zwei weitere I/Os zur Verfügung.

< 14em 2em 6em 6em >		
Pin	Signal	Arduino
1	GND	
2	PA0	D3
3	GND	
4	PA1	D2
5	VCC	

Versorgung

Die Versorgung kann mit einer geregelten Gleichspannung von 3,3 V oder 5 V erfolgen. Es existieren mehrere Anschlussmöglichkeiten:

- über CON 4: Hier kann ein Stecker im 2,54 mm-Rastermaß oder eine kleine Schraubklemme mit 3,5 mm Pinabstand eingelötet werden.
- Versorgung durch die Applikationsschaltung über die Buchsenleiste CON2
- Per Batteriekasten auf der Rückseite (2 AAA-Batterien)



Die Helligkeit der LED-Anzeige variiert mit der Höhe der Versorgungsspannung. Als Ausgleich lässt sich daher in der Software die Helligkeit in drei Stufen anpassen ([Parameter DIMMING](#) in ProtoCounter.h).

Programmierung

Der ProtoCounter wird über den 6-poligen ISP-Anschluss programmiert. Hierzu ist ein entsprechendes Programmiergerät erforderlich wie z. B. der [AVRISP mkII](#) von Atmel oder der [USBasp](#) von Thomas Fischl.

Die Fuses müssen wie folgt eingestellt werden:

```
LOW      = 0xFF
HIGH     = 0xDF
EXTENDED = 0xFF
```

Anzeige

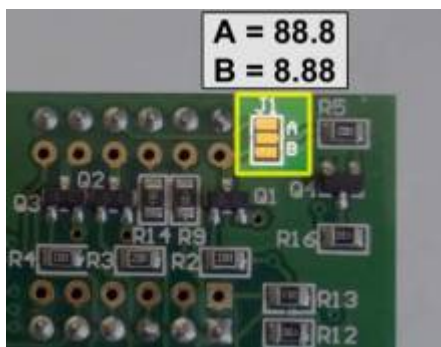
Helligkeit


Die Helligkeit der Anzeige und damit auch der Stromverbrauch des ProtoCounters kann über den Parameter „dimming“ angepasst werden. Sinnvolle Werte für das Dimming liegen zwischen 0 und 9.

```
ProtoCounter pc;    // create a ProtoCounter instance

pc.dimming = 8;     // set dimming (0 = no dimming)
```

Dezimalpunkt



 Jumper für Dezimalpunkt

Der Dezimalpunkt kann über einen Lötjumper fest eingestellt werden. In Position A hat die Anzeige eine, in Position B zwei Nachkommastellen. Bei offenem Jumper ist der Dezimalpunkt ausgeschaltet.

Die eingestellte Anzahl an Nachkommastellen wird dann im Programm festgelegt.

```
ProtoCounter pc; // create a ProtoCounter instance

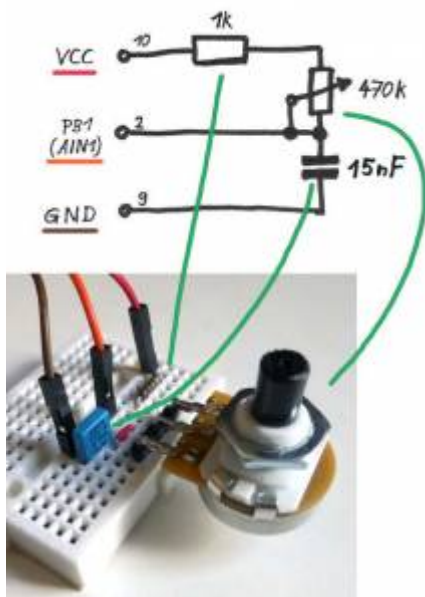
pc.decimal_places = 1; // number of decimal places (range 0..2)
```

Wichtiger Hinweis

Soll der Dezimalpunkt zusammen mit externen Schieberegistern verwendet werden, müssen die Leitungen für die Input- bzw. Outputdaten auf freie Pins (z. B. Pins PB2 und PB3) verlegt werden.

```
/* ProtoCounter.h */
...
#define SH_REG_OUT_BIT 2 // data out bit number
#define SH_REG_IN_BIT 3 // data in bit number
...
```

Drehknopf



Anschluss eines Drehknopfs

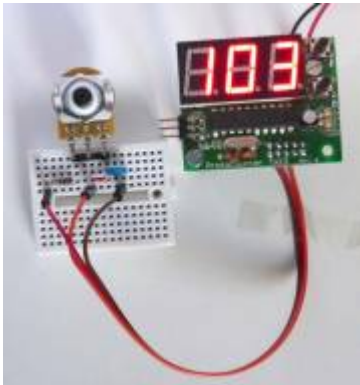
Einstellungen mit einem Drehknopf vorzunehmen ist praktisch und intuitiv. Deshalb bietet auch der ProtoCounter diese Möglichkeit. An Pin PB1 (AIN1) lässt sich ein Potentiometer anschliessen, das zur Eingabe von Werten benutzt werden kann. Der Wertebereich geht von 0 bis maximal 250. Man benötigt folgende Bauteile:

- Potentiometer 470 kΩ
- Widerstand 1 kΩ
- Kondensator 15 nF

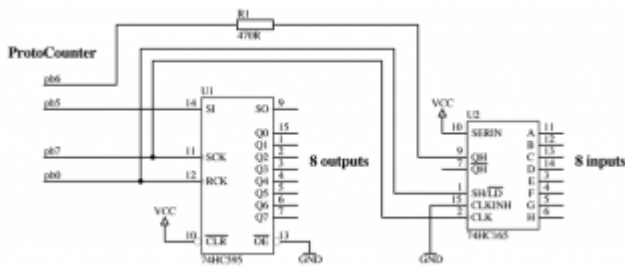
Der Schaltplan rechts zeigt, wie man sie mit dem ProtoCounter verbindet.

Die Bauteilwerte sind nur Anhaltspunkte. Insbesondere Potentiometer haben oft einen etwas größeren

Widerstandsbereich als ihr Nennwert angibt. Der Kondensator sollte nicht größer als 33 nF gewählt werden. Der Widerstand muss immer größer als 1 k Ω sein, was durch den 1 k Ω Widerstand in Reihe mit dem Potentiometer gewährleistet wird. Die Zeitkonstante, also das Produkt von Kapazität und Widerstand, sollte ungefähr 10 ms betragen, um den Wertebereich gut auszunutzen.



I/O-Erweiterung



Erweiterung um zusätzliche I/Os

Mit minimalem Aufwand lässt sich der ProtoCounter um bis zu 32 Ausgänge und 32 Eingänge erweitern. Hierfür werden kostengünstige Standard-Schieberegister verwendet. Der Schaltplan zeigt beispielhaft die Erweiterung um jeweils 8 Ein- und Ausgänge. Man benötigt:

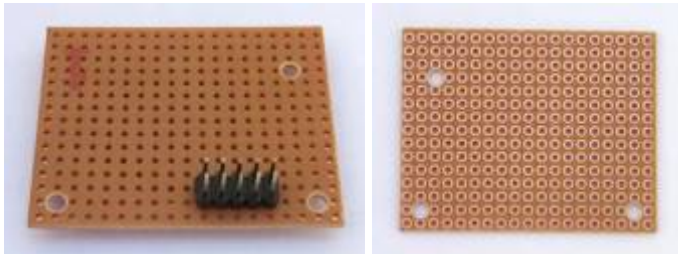
- Schieberegister 74HC595 für jeweils 8 Ausgänge
- Schieberegister 74HC165 für jeweils 8 Eingänge
- Widerstand 470 Ω

Natürlich kann man auch nur Eingänge oder nur Ausgänge hinzufügen. Die gewünschte Anzahl an auszugebenden bzw. einzulesenden Bits wird im Headerfile angegeben.

```
/* ProtoCounter.h */
...
// external shift register
// Data is shifted out with MSB first.
// To disable the external shift register set BITCOUNT to 0
#define SH_REG_IN_BITCOUNT    8    // number of input shift register bits
                                      (range 0..32)
#define SH_REG_OUT_BITCOUNT  8    // number of output shift register bits
                                      (range 0..32)
...
```


Montage

Die Anschlüsse und Befestigungsbohrungen des ProtoCounters sind im Raster von 2,54 mm (1/10 Zoll) angelegt, so dass sich der Proto-Counter problemlos auf eine Lochrasterplatine aufstecken lässt. Genaue Koordinaten in mil (= 1/1000 Zoll) enthält die [Maßzeichnung](#). 12 mm-Distanzbolzen sorgen für den richtigen Abstand. Über 10 mm-Distanzbolzen kann der ProtoCounter mit einer passenden Frontplatte verschraubt werden. Wenn man die Maßzeichnung 1:1 ausdruckt, kann sie als Bohrschablone dienen.



Alternativ kann die Verbindung zu deiner Schaltung auch mit einem Flachbandkabel hergestellt werden. Der ProtoCounter bietet genügend Platz für einen 10-poligen Wannenstecker.

Software

Der ProtoCounter kann mit Hilfe der kostenlosen [Arduino-IDE](#) programmiert werden. Zusätzlich wird noch die [Erweiterung für ATtiny-Prozessoren](#) benötigt. Die Installation ist auf der Github-Seite des Projekts unter der Überschrift „Board Manager Installation“ beschrieben.

Für die Dokumentation der ProtoCounter-Software gibt es eine eigene Seite ([Software-Dokumentation](#)).

Download

Datei	Lizenz
Schematic	CC-BY-SA
Assembly Drawing	CC-BY-SA
Dimensions	CC-BY-SA
ProtoCounter Arduino library	see license.md
ATtiny4313 datasheet	Copyright by Atmel

CC-BY-SA = [Creative-Commons-License](#), attribution, share alike

CC0 = [Creative-Commons-License](#), without restrictions

[Proto-Counter](#), [electronics](#), [kit](#), [7-segment](#), [Arduino](#), [ATtiny4313](#), [microcontroller](#)

From:

<http://www.doku.fab4u.de/> - **fab4U**

Permanent link:

<http://www.doku.fab4u.de/en/kits/protocounter/start>

Last update: **2018/07/03 21:12**

