Wolles Elektronikkiste

Die wunderbare Welt der Elektronik

Der ATmega328PB ist eine Weiterentwicklung des guten alten ATmega328P, den ihr von diversen Boards, wie dem Arduino UNO R3, dem klassischen Arduino Nano V3 oder dem Arduino Pro Mini her kennt. Im Vergleich zum ATmega328P ist der ATmega328PB sicherlich keine Revolution, gleichwohl hat er einige sinnvolle Erweiterungen.

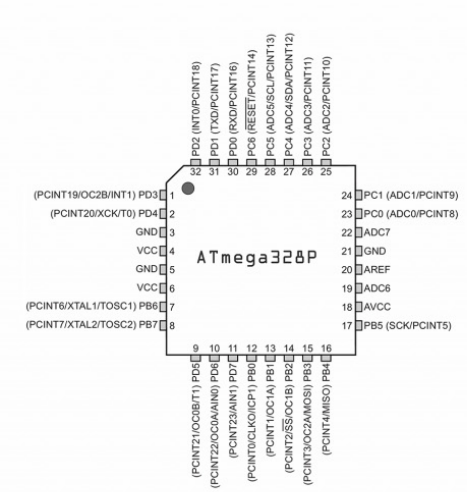
ATmega328PB vs. ATmega328P – Überblick

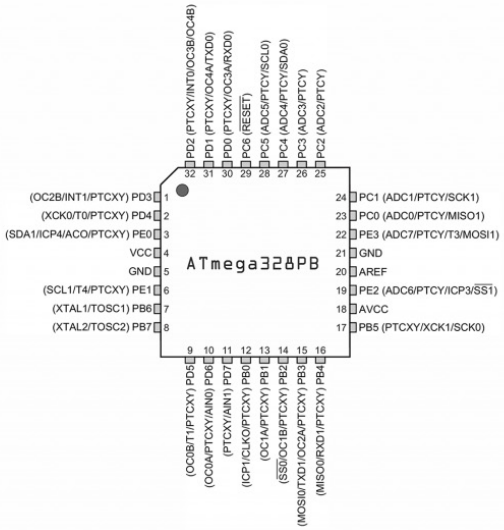
Der ATmega328PB ist abwärtskompatibel zum ATmega328P. Das heißt Sketche, die ihr für den ATmega328P und darauf basierende Boards geschrieben habt, funktionieren auch auf dem ATmega328PB. Ein Datenblatt zum ATmega328PB findet ihr **hier** bei Microchip.

Das sind die aus meiner Sicht wichtigsten Änderungen:

- Jeweils eine zusätzliche USART (Serial), TWI (I2C) und SPI Schnittstelle.
- Zwei zusätzliche 16-Bit Timer/Counter (Timer/Counter 3, Timer/Counter 4).
- Vier zusätzliche I/O Pins (PE0, PE1, PE2, PE3).
- Für PE0 und PE1 wurden ein GND und ein VCC geopfert.
- Die Pins für ADC6 und ADC7 haben jetzt auch I/O-Pin Funktion (PE2 und PE3).
- Output Compare Modulator: kombiniert die Ausgänge von Timer/Counter 3 und Timer/Counter 4 an PD2.
- Analog Comparator Ausgang an PE0.
- PTC: Peripheral Touch Controller. Der PTC ermöglicht es euch, kapazitive Berührungssensoren an den I/O-Pins auszuwerten. Allerdings muss man dafür den Atmel Start QTouch® Configurator verwenden. Ich gehe nicht darauf ein.

Eine vollständige Zusammenfassung der Unterschiede gibt es in der Application Note **AN42559** von Microchip.





„Nano-Stil“ Board am Beispiel ARD-NanoV4-MC

Ich werde am ARD-NanoV4-MC zeigen, wie ihr die zusätzlichen Funktionen, die der ATmega328PB bietet, nutzt. Ihr findet das Board zum Teil auch unter der Bezeichnung „ARD NANO V4 (MINICORE)“. Ich nenne es kurz „NanoV4-MC“.

Auf der **Joy-IT Website** findet ihr die vorbildliche Dokumentation des NanoV4-MC mit Datenblatt und Anleitung in verschiedenen Sprachen.

An dieser Stelle noch eine Anmerkung: ich werde weder von Joy-IT, noch von Pololu oder Watterott in irgendeiner Weise für die Vorstellung ihrer Produkte entlohnt.

Installation des empfohlenen Boardpaketes

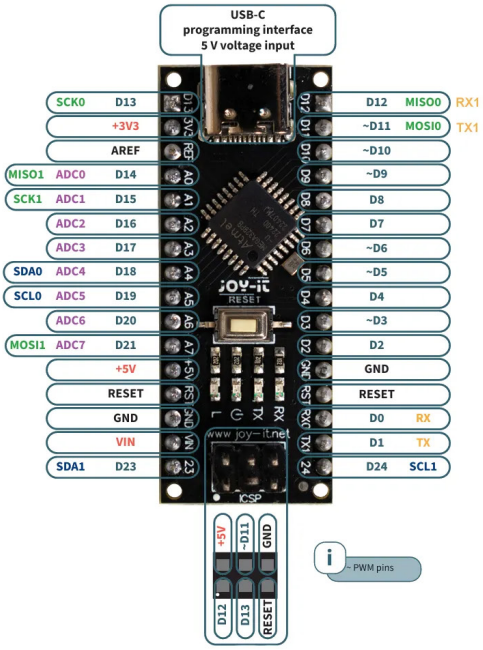
Joy-IT hat für den NanoV4-MC das Boardpaket MiniCore von **MCUdude** vorgesehen, das ihr **hier** auf GitHub findet.

Auf das Installieren von Boardpaketen in der Arduino IDE gehe ich nicht im Detail ein. Bei Bedarf schaut in die Anleitung von Joy-IT. Hier die Kurzversion:

Geht in der Arduino IDE auf Datei → Einstellungen.
Klickt auf das Icon neben „Zusätzliche Boardverwalter-URLs“.
Ergänzt die Zeile: „https://mcudude.github.io/MiniCore/package_MCUdude_MiniCore_index.json“ (ohne Anführungsstriche), dann 2x das „OK“ bestätigen.
Klickt am linken Rand der IDE auf das Icon für die Bordverwaltung und gebt „MiniCore“ als Suchbegriff ein.
Installiert MiniCore von MCUdude.
Um das richtige Board einzustellen, wählt ihr unter „Werkzeuge → Boards“ den Eintrag „MiniCore → ATmega328“. Als Variante müsst ihr noch „328PB“ festlegen und als „Clock“ die Einstellung „External 16 MHz“ vornehmen.

Pinout

Hier das Pinout (mit freundlicher Genehmigung von Joy-IT):



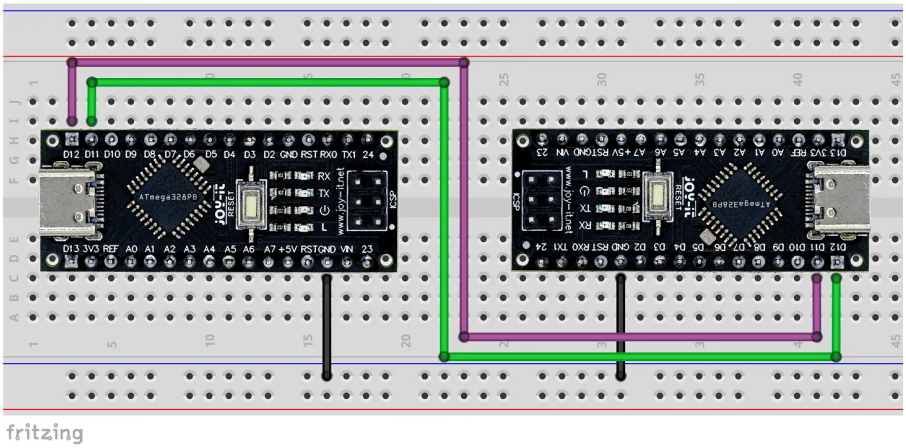
Auf den ersten Blick sieht das gute Stück wie ein Arduino Nano aus. Die wesentlichen Unterschiede sind:

- Die zusätzlichen Pins D23 und D24.
- An D23 und D24 ist die zweite I2C Schnittstelle (Wire1) ausgeführt.
- A6 und A7 sind als GPIOs nutzbar (D20/D21).
- Die zweite SPI-Schnittstelle ist an den Pins A0, A1 und A7 ausgeführt.
- Serial1 ist an den Pins D11 und D12 ausgeführt.

Serial1 nutzen

Da der Programmupload auf den Arduino Boards per USART (Serial) vorgenommen wird, ist die Verwendung von Serial für andere Zwecke manchmal problematisch. Deswegen setzen die meisten in solchen Fällen SoftwareSerial ein, was aber mehr Ressourcen kostet als HardwareSerial.

An einem einfachen Anwendungsbeispiel möchte ich zeigen, wie ihr Serial1 verwendet. Dazu verbindet ihr zwei NanoV4-MC Boards wie folgt:



Dann ladet ihr den folgenden Sketch auf beide NanoV4-MC Boards (am besten mit unterschiedlichen Namen, damit ihr zwei serielle Monitore öffnen könnt):

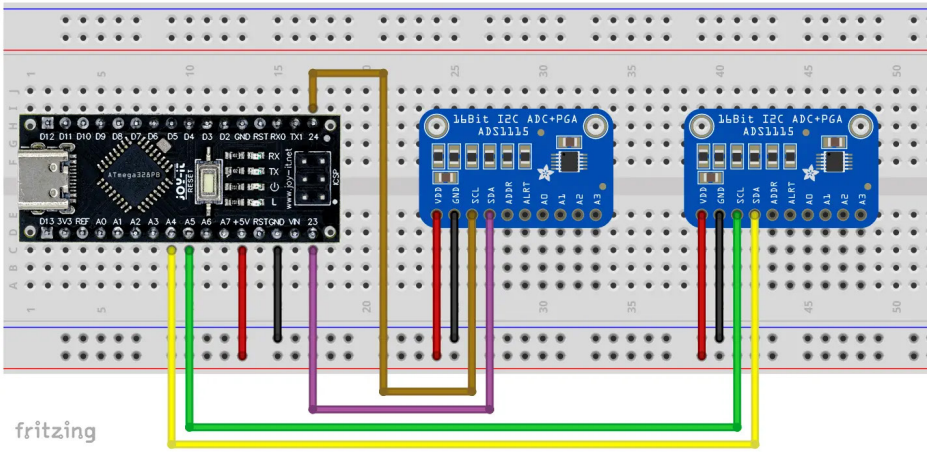
```
void setup() {
        Serial.begin(9600);
        Serial1.begin(9600);
}
void loop() {
        if (Serial1.available()) {
                Serial.write(Serial1.read());
        }
        if (Serial.available()) {
                Serial1.write(Serial.read());
        }
}
```

Jetzt könnt ihr Nachrichten über den seriellen Monitor austauschen. Wenn ihr nur einen NanoV4-MC habt, dann könnt ihr auf der anderen Seite auch ein anderes Board verwenden und dort SoftwareSerial einsetzen. Wichtig ist, dass auf beiden Seiten dieselbe Baudrate eingestellt ist. Falls ihr auf der anderen Seite ein 3.3 Volt basiertes Board verwendet, müsst ihr noch Spannungsteiler oder einen Levelshifter dazwischen setzen. Mehr zu Serial und SoftwareSerial findet ihr **hier**.

Wire1 nutzen

Viele externe Bauteile werden per I2C (Wire) angesteuert. Problematisch wird es, wenn ihr mehrere Bauteile mit derselben I2C-Adresse habt. Sofern es nur zwei Bauteile mit derselben Adresse sind, könnt ihr das eine per Wire und das andere per Wire1 ansteuern. Voraussetzung ist dabei, dass bei Verwendung von Bibliotheken diese die Übergabe von Wire-Objekten erlauben. Wie ihr Wire und Wire1 gleichzeitig einsetzt, zeige ich exemplarisch am ADS1115.

Hier die Schaltung:



Und hier der zugehörige Sketch:

```
#include<ADS1115_WE.h>
#include<Wire.h>
#include<Wire1.h>
#define I2C_ADDRESS 0x48
ADS1115_WE adc_0 = ADS1115_WE(&Wire, I2C_ADDRESS);
ADS1115_WE adc_1 = ADS1115_WE(&Wire1, I2C_ADDRESS);
void setup() {
        Wire.begin();
        Wire1.begin();
        Serial.begin(9600);
        if(!adc_0.init()){
                Serial.println(„ADS1115 0 not connected!“);
        }
        else{
                Serial.println(„ADS1115 0 successfully connected!“);
        }
        if(!adc_1.init()){
                Serial.println(„ADS1115 1 not connected!“);
        }
        else{
                Serial.println(„ADS1115 1 successfully connected!“);
        }
}
void loop() {}
```

Erwartungsgemäß lautet die Ausgabe:

```
ADS1115 0 successfully connected!
ADS1115 1 successfully connected!
```

SPI1 nutzen

Prinzipiell könnt ihr SPI1 zwar nutzen, aber die Übergabe an Objekte funktioniert nicht ohne Weiteres wie bei Wire1. Das Hauptproblem ist, dass SPI1 in der MiniCore Bibliothek als Objekt der Klasse „SPI1Class“ definiert worden ist und die meisten Bauteil-Bibliotheken „SPI1Class“ nicht kennen.

Eine direkte Verwendung von SPI1, also per SPI1.transfer() und Co ist hingegen kein Problem. Falls ihr nur SPI1 verwenden wollt, weil ihr die Standard-SPI Pins für andere Zwecke benötigt, dann könntet ihr das Pololu Boardpaket nutzen. Dazu kommen wir später.

Zusätzliche Timer / PWM

Der ATmega328PB besitzt die zusätzlichen Timer/Counter 3 und 4. Die Systematik der zugehörigen Register entspricht der des Timer/Counter 1, den ich hier im Detail beschrieben habe. Die Ausgänge OC3A und OC4A befinden sich an PD0 (RX/D0) und PD1 (TX/D1). Hier ein einfaches Beispiel für ein PWM Signal an OCRA:

```
void setup(){
        noInterrupts();
        // prescaler: 8, top = ICR3, clear at OCR3A, set at top, output at OC3A
        TCCR3A = (1 << COM3A1) | (1 << WGM31);
        TCCR3B = (1 << WGM33) | (1 << WGM32) | (1 << CS31);
        DDRD = (1 << DDD0); // PD0 to OUTPUT
        OCR3A = 9999;
        ICR3 = 39999; // pwm frequency 50 Hz, duty cycle 25%
        interrupts();
}
void loop() {}
```

Und wie immer bei diesem Thema empfehle ich das großartige Tool **Arduino Web Timers** von **David Buezas**. Damit könnt ihr eure Einstellungen mittels Tick-Boxen vornehmen und müsst die Registereinstellungen nicht mühsam aus dem Datenblatt heraussuchen.

Output Compare Modulator

OC3B und OC4B teilen sich den Pin PD2 (D2) und bilden den Output Compare Modulator, mit dem ihr die beiden PWM Signale kombinieren könnt. Dabei gibt es zwei Einstellungen:

1. PD2 ist auf HIGH eingestellt:
- PD2 Level = SignalOC3B | SignalOC4B
- D. h. die Signale sind über ein logisches ODER verknüpft.

2. PD2 ist auf LOW eingestellt:
- PD2 Level = SignalOC3B & SignalOC4B
- D. h. die Signale sind über ein logisches UND verknüpft.

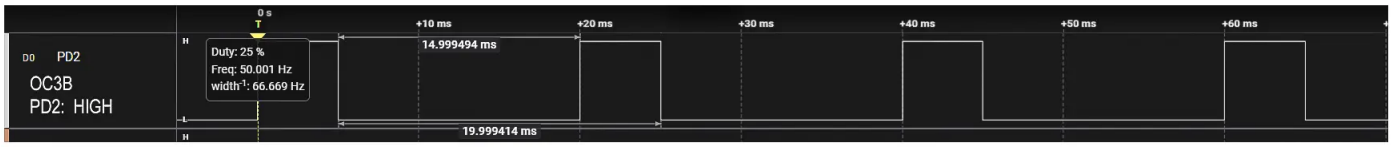
Als Beispiel nehmen wir zwei PWM Signale mit einem Duty Cycle von 25 %. Die PWM-Frequenz von OC4B soll das Zwanzigfache der PWM Frequenz von OC3B betragen.

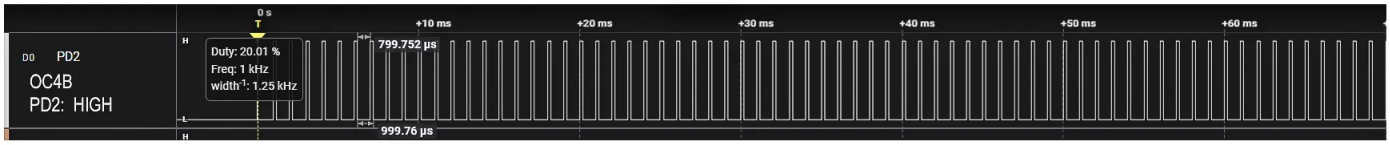
Hier der Sketch dazu:

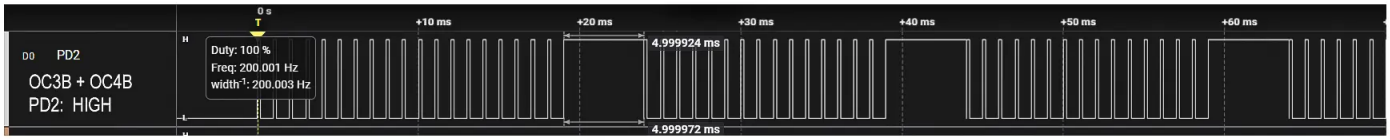
```
1  void setup(){
2          noInterrupts();
3          // OC3B settings:
4          TCCR3A = (1 << COM3B1) | (1 << WGM31);
5          TCCR3B = (1 << WGM33) | (1 << WGM32) | (1 << CS31);
6          OCR3B = 9999;
7          ICR3 = 39999;
8          // OC4B settings:
9                  TCCR4A = (1 << COM4B1) | (1 << WGM41);
10         TCCR4B = (1 << WGM43) | (1 << WGM42) | (1 << CS41);
11         OCR4B = 399;
12         ICR4 = 1999;
13         // PD2 settings
14         DDRD = (1 << DDD2);
15         // PORTD |= (1<<PD2);
16         interrupts();
17 }
18 void loop() {}
```

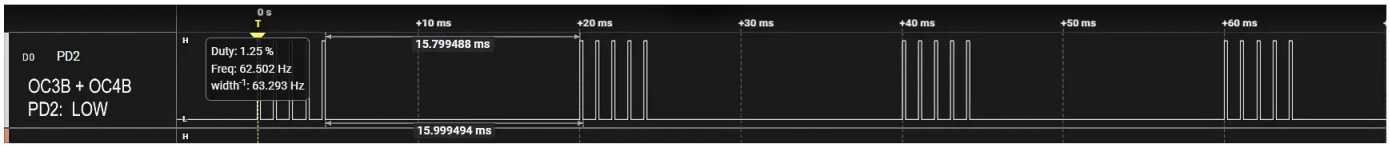
Das Signal an PD2 hängt davon ab, ob ihr Zeile 17 auskommentiert lasst oder entkommentiert. Um die Einzelsignale OC3B und OC4B zu erhalten, müsst ihr Zeile 17 entkommentieren und jeweils die Zeilen 4 bis 7 bzw. 10 bis 13 auskommentieren.

Mit einem **Logic Analyzer** habe ich folgende Ausgaben erhalten:





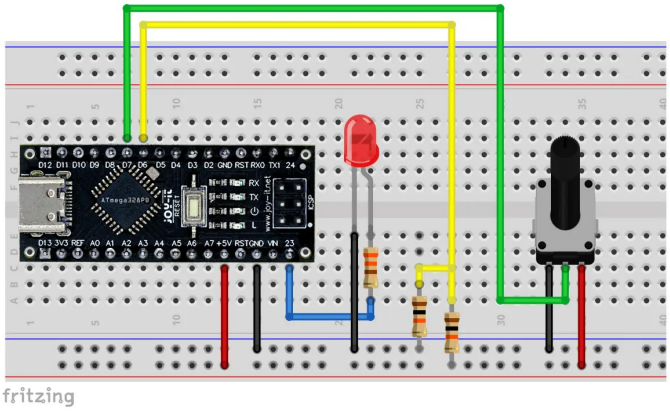




Analog Comparator Output

Der Analog Comparator vergleicht zwei Spannungen und zeigt an, welche der Spannungen größer ist. Beim ATmega328P hat man dazu das ACO Bit (Analog Comparator Output) abgefragt oder einen Interrupt eingerichtet. Der ATmega328PB besitzt hingegen zusätzlich einen ACO Pin (PE0 / D23).

Hier ein kleines Beispiel, in dem wir an AIN0 (PD6 / D6) die halbe Versorgungsspannung einstellen und an AIN1 (PD7 / D7) mit einem Poti eine variable Spannungsquelle hängen. Eine LED am ACO Pin zeigt an, welche Spannung größer ist.



Hier der erstaunlich kurze Sketch dazu:

```
void setup() {
        ACSR = 0x00;
        ADCSRB = 0x00;
        ACSRB = (1 << ACOE); // Analog Comparator Output Enable
}
void loop() {}
```

Wenn die Spannung an D7 höher ist als an D6, dann ist die LED aus. Ist die Spannung an D7 kleiner, dann ist die LED an. Die Eingänge für den Analog Comparator lassen sich auf alle analogen Pins multiplexen. Aber das würde hier jetzt etwas zu weit führen.

Originalbeitrag: **https://wolles-elektronikkiste.de/atmega328pb-basierte-boards**