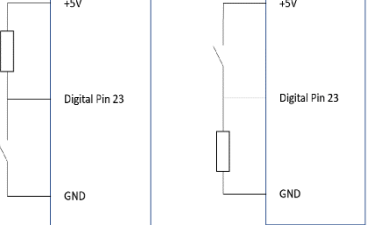

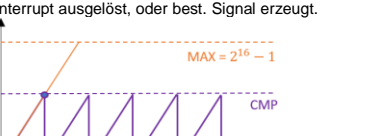
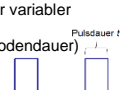
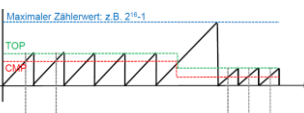
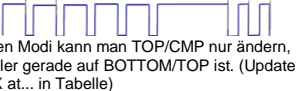
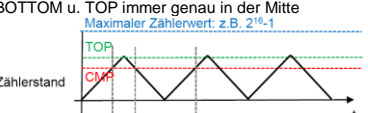
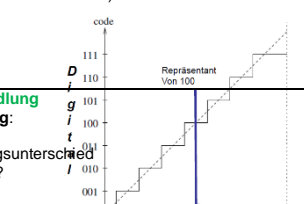


<p><b>Speicher, Digitale Ein- und Ausgabe</b></p> <p><b>DDRx</b> (Data Direction Register): Entsprechendes Bit auf 1 für Ausgang, oder 0 für Eingang</p> <p><b>PORTx</b> (Port Register): Wenn Pin auf Ausgang, dann 1 = 5V und 0 = 0V</p> <p><b>PINx</b> (Port Input Register): Wenn Pin auf Eingang, dann 1 = HIGH liegt an und 0 = LOW liegt an</p> <p>pinMode(13, OUTPUT); digitalWrite(13, HIGH); digitalRead(13); ... if (digitalRead(13) == HIGH)...</p> <p>Serial.begin(9600);</p> <p>Serial.println("Eingabe ist: " + Serial.readString());</p> <p>Serial.available() : Anz. Bytes, die zum Lesen verfügbar sind. Evtl. in while Schleife.</p>	<pre>void setup() {   DDRB  = (1 &lt;&lt; DDB4);   //pinMode(pin, OUTPUT);   DDRB  = ~(1 &lt;&lt; DDB4);   //pinMode(pin, INPUT);    PORTA  = ~(1 &lt;&lt; PA4);   //digitalWrite(pin, LOW); }  void loop() {   if (PINA &amp; (1 &lt;&lt; PINA0)) {     //if (digitalRead(sw)) {     PORTA  = (1 &lt;&lt; PA4);     delay(2000);     PORTA  = ~(1 &lt;&lt; PA4);     //digitalWrite(pin, LOW);   } }</pre> <p><b>Mikrocontroller Bestandteile:</b> MicroProzessor, Timer, Schnittstellen, Speicher, AD-Wandler</p> <p><b>Entwicklerboard:</b> Arduino Mega, Mikroprozessor: Atmega2560</p> <p><b>Cross-Compilation:</b> Programm wird nicht auf Zielplattform (Mikrocontroller), sondern auf anderer Plattform übersetzt</p> <p><b>Flashen:</b> hex-Datei von PC an Entwicklerboard senden</p> <p><b>Harvard-Architektur:</b> Daten- und Instruktionsspeicher getrennt</p> <p><b>Instruktionsspeicher:</b> Nicht flüchtiger Flash Speicher</p> <p><b>Daten:</b> in flüchtigem SRAM</p> <p><b>SRAM und DRAM sind flüchtig.</b> Rest nicht flüchtig.</p>	<p><b>Nichtflüchtige Speicher:</b></p> <p>ROM</p> <p>OTPROM</p> <p>EEPROM: Begrenzte Anzahl an Schreib/Lesezyklen.</p> <p>Konfigurationsdaten, Kalibrierungsdaten</p> <p>Flash: Programm- Daten/Code</p> <p>SRAM(flüchtig): Arbeitsspeicher, Register, Stack usw</p> <p><u>Einzelne Bits setzen:</u> x  = (1 &lt;&lt; Bitnummer);</p> <p><u>Einzelne Bits löschen:</u> x &amp;= ~(1 &lt;&lt; Bitnummer)   (1 &lt;&lt; Bitnummer2));</p> <p><u>Testen ob Bit auf 1:</u> if (DDRC &amp; (1 &lt;&lt; Bitnummer)) { }</p> <p><u>Testen ob Bit auf 0:</u> if (!(DDRC &amp; (1 &lt;&lt; Bitnummer))) { }</p> <p><u>Alle Bits umdrehen:</u> x = 0xFF ^ x</p> <p><u>LED toggeln:</u> PINA  = (1 &lt;&lt; PINA2);</p> <p>Vorwiderstand berechnen:</p> $R_v = \frac{U_{ges} - U_F}{I_F}$	 <p><b>Pull-Up / Active Low</b> Bei offenem Taster wird Spannung am Pin auf HIGH gezogen.</p> <p><b>Pull-Down / Active High</b> Bei offenem Taster wird Spannung am Eingang auf LOW gezogen.</p> <p><b>Entprellung</b> Einmaliges betätigen eines Schalters führt evtl zu mechanischen Vibrationen. SW-Lsg: Künstliche Wartezeit nach Zustandswechsel – Bis Schalter eingeschwungen.</p>
<p><b>Interrupts</b></p> <p><b>sei()</b>: Interrupts global aktivieren (oder: SREG  = 128)</p> <p><b>cli()</b>: Interrupts global deaktivieren</p> <p><b>SREG</b> (AVR Status Register): Bit 7 auf 1 = sei();</p> <p><b>EIMSK</b> (External Interrupt Mask Register): Speziellen Interrupt de-/aktivieren</p> <p><b>EICRA</b> (External Interrupt Control Register A) <b>EICRB</b> (External Interrupt Control Register B): ISCn0 und ISCn1. Falling oder Rising Edge. n ist die Interrupt Nummer.</p> <p><b>EIFR</b> (External Interrupt Flag Register): Wenn Interrupt ausgelöst: Bit ist 1</p> <p><b>ISR</b> (INT0_vect) { }</p> <p>attachInterrupt(digitalPinToInterrupt(21), count, RISING);</p>	<pre>volatile int counter = 0; volatile unsigned long time = 0;  void setup() {   Serial.begin(9600); }  sei(); EIMSK  = (1 &lt;&lt; INT2); //pin 19 EICRA  = (1 &lt;&lt; ISC20)   (1 &lt;&lt; ISC21); // rising edge }  void loop() {   Serial.println("Zählerstand: ");   Serial.println(counter);   delay(3000); }  ISR (INT2_vect) {   if (millis() - time &gt;= 250) {     time = millis(); //Entprellung     counter++;   } }</pre>	<p><b>Busy Waiting:</b> while (DDRC &amp; (1 &lt;&lt; DDC3));</p> <p><b>Polling:</b> periodisches Abfragen, ob Ereignis eingetreten</p> <p><b>Interrupt:</b> Kurze Unterbrechung des laufenden Programms um einen anderen zeitkritischen, kurzen Vorgang zu bearbeiten. Hardware prüft dauernd parallel, ob Ereignis eingetreten ist.</p> <p>Wenn auf ein seltenes Ereignis schnell reagiert werden muss.</p> <p><b>Trap:</b> Art von Interrupt, die aber synchron und reproduzierbar ist. z.B. System Call, Div durch 0 ...</p> <p><b>Interrupt Request:</b> Interruptereignis – [InterruptController] – über IRQ Eingang Unterbrechungsanforderung an CPU – CPU unterbricht Programm und startet Unterbrechungsroutine</p> <p><b>Interrupt Vector Table:</b> Welches Interruptereignis gehört zu welcher ISR? Jede Vectornummer hat eine zugehörige Programmadresse. ISR ist selbst nicht unterbrechbar (1 Bit SREG)</p>	<p><b>Externe Interrupts</b> Controller tastet zu Beginn jedes Taktzyklus ab. Falls Interrupt aktiviert, Aufruf der ISR. Probleme: Leichte Verzögerung, „Prellung“</p> <p><b>Interne Interrupts</b> Timer, A/D-Wandler Bei Auslauf eines Timers unterbricht HW Ausführung der normalen Software</p> <p><b>Volatile</b> Variable wird vor jedem Lesen aus SRAM gelesen und nach jedem Schreiben in SRAM geschrieben <b>!!Globale Variablen die in ISR vorkommen immer volatile!!!</b></p>
<p><b>Timer</b></p> <p>n: Timer 1-5</p> <p><b>TCCRnA</b> (Timer/Counter n Control Register A): PWM</p> <p><b>TCCRnB</b> (Timer/Counter n Control Register B): Prescaler; Starten des Timers; Input Capture, CTC</p> <p>Beide TCCRn erst auf 0x00 setzen. Auch wenn keinen Prescaler will, muss man setzen</p> <p><b>TCNTn</b> (Timer Counter n, 16 Bit): Aktueller Zählerstand. Anfangs auf 0 setzen.</p> <p><b>OCRnA, OCRnB, OCRnC</b> (Output Compare Register, 16 Bit): Wert gegen den Zählerstand verglichen werden kann</p> <p><b>ICRn</b> (Input Capture Register): Bei Input Capture erfasster Wert wird gespeichert</p> <p><b>TIMSKn:</b> Aktivieren/Deaktivieren der Timer Interrupts</p> <p><b>TIFRn:</b> Timer bezogene Interrupt Flags</p> <p><b>CTC Beispiel:</b> TCCR4B  = (1 &lt;&lt; WGM42); ISR(TIMER4_OVF_vect) { } Interrupt bei Timer 4 Overflow ISR(TIMER4_COMPA_vect){}: Timer 4 compare A</p>	<pre>// counts number of overflows since last second volatile unsigned int overflow_counter = 0;  void setup() {   DDRB  = (1 &lt;&lt; DDB2);    // initialize timer4   TCCR4A = 0x00;   TCCR4B = 0x00;   TCNT4 = 0x00;   // activate clock, but don't use a prescaler p157   TCCR4B  = (1 &lt;&lt; CS40);   // enable timer overflow interrupt   TIMSK4  = (1 &lt;&lt; TOIE4);    sei(); // enable all interrupts }  // ISR, called when timer overflow occurs ISR(TIMER4_OVF_vect) {   // 16 MHz / 2^16 = 244,1 -&gt;   // during 1 second approx. 244 overflow events   if (overflow_counter == 244) {     overflow_counter = 0;     PINA  = (1 &lt;&lt; PINA2); // toggle LED   }   overflow_counter++; }</pre>	<p>Atmega2560 Systemtakt = 16Mhz</p> <p>16Bit Timer =&gt; Timer läuft nach <math>\frac{2^{16}-1}{16\text{ MHz}} = 4\text{ms}</math> über</p> <p><b>Prescaler</b> Fallende Flanke des Prescalers an Bit Qn triggert Counter.</p> <p><b>Vorteil</b> großer Prescaler: Messen langer Zeiten möglich.</p> <p>kleinstes messbares Zeitintervall ohne Prescaler: <math>\frac{1}{16\text{MHz}} = 62\text{ns}</math></p> <p>mit f/1024 Prescaler: <math>\frac{1}{16\text{MHz}/1024} = 64\mu\text{s}</math></p> <p><b>Nachteil:</b> Schlechtere Auflösung. =&gt; immer kleinstmöglichen Prescaler!</p> <p>Welchen Prescaler für 3s Intervalle mit 16 Bit Timer? Takt: 1 MHz (Bei 16MHz bis 3*1000000) <math>\frac{3000000}{x} = 2^{16} - 1 \Rightarrow x = 45,8 \Rightarrow 64\text{Prescaler}</math></p>	<p><b>Input Capture</b> Bei externen oder internen Signalen/Ereignissen wird aktueller Zählerstand in ICRn gespeichert</p>  <p><b>Output Compare</b> Bei Erreichen eines konfigurierten Zählerstandes wird Interrupt ausgelöst, oder best. Signal erzeugt.</p>  <p><b>CTC Mode (Clear Timer on Compare Match)</b> <b>TOP</b> Wert in OCRnA oder ICRn konfiguriert. Zähler bei Erreichen des Zählerstandes automatisch auf 0.</p>
<p><b>Pulsweitenmodulation</b></p> <p><b>TCCRnA</b> Compare Output Mode Fast PWM usw</p> <p><b>TCCRnB</b> Fast PWM; Prescaler</p> <p><b>OCnA, OCnB, OCnC</b> (Output Compare Pins): PWM-Ausgang Inverting oder non-Inverting Mode</p> <p>Output Compare Pins müssen als <b>Ausgang im DDR</b> Register konfiguriert sein!</p> <p><b>OCRnX</b> (Output Compare Register): Vergleichswert (Schwellwert) muss gesetzt werden, der jeweils PWM-Ausgang OCRnX beeinflusst.</p>	<pre>void setup() {   // set pin PB5 (pin5 of port B) to output: this is the PWM pin   // (alternative function: alternative function of PB5: OC4C)   DDRB  = (1 &lt;&lt; DDB5);    // to be safe: initialize timer control registers to zero   TCCR4A = 0x00;   TCCR4B = 0x00;    // Fast PWM mode, counter TOP value taken from ICR,   TCCR4A  = (1 &lt;&lt; WGM41); // WGM bits: "1110", manual p145   TCCR4B  = (1 &lt;&lt; WGM43)   (1 &lt;&lt; WGM42);    // non-inverting mode: clear on compare match: manual p155, Table 17-4   TCCR4A  = (1 &lt;&lt; COM4C1);    // good choice: use clk/8 prescaler -&gt; 16 MHz / 8 = 2 MHz   // counter counts up from 0 to 39999 within 20 ms   TCCR4B  = (1 &lt;&lt; CS41);    ICR4 = 40000; // configure period of PWM, i.e. TOP value in ICR register   // initial pulse width / duty cycle 1,25 ms   // --&gt; (1,25 ms / 20 ms) * 40000   OCR4C = 2500;    delay(3000);    OCR4C = 1088; // duty cycle: min value   delay(3000);    OCR4C = 4800; // duty cycle: max value   delay(3000); }</pre>	<p>Signal mit konstanter Periode, aber variabler Pulsdauer wird erzeugt.</p> <p>Duty Cycle: <math>t = \text{Pulsdauer} / \text{Periodendauer}</math></p>  <p><b>TOP:</b> ICRn Register (oder andere siehe S145 Tabelle)</p> <p><b>CMP:</b> OCRnX Register</p>  <p>In manchen Modi kann man TOP/CMP nur ändern, wenn Zähler gerade auf BOTTOM/TOP ist. (Update of OCRnX at... in Tabelle)</p> 	<p><b>Inverting u. Non-Inverting Mode</b> Non-Inverting: siehe Links. Inverting: PWM Ausgang genau andersrum</p> <p>TCCRnX Register</p> <p><b>Up-Down-Counter</b> doppelte Periodendauer, geringere Auflösung</p> <p>BOTTOM u. TOP immer genau in der Mitte</p>  <p>PWM Signal (OCnX)</p> <p>Tabelle S145 TCCRnA u TCCRnB Fast PWM(Up-Counter), PWM(Up-Down Counter)</p>
<p><b>Analoge Ein-/Ausgabe</b></p> <p><b>ADMUX</b> Referenzspannung wählen Analoge Eingangspins für A/D Umsetzung wählen</p> <p><b>ADCSRB</b> Analoge Eingangspins für A/D Umsetzung wählen Single Ended oder Differential Conversion Free Running Mode oder manuelles Triggern</p> <p><b>ADCSRA</b> Aktivieren und Starten der A/D Umsetzung Prescaler Interrupts</p> <p><b>ADCL, ADCH</b> Speichert Ergebnis der A/D Umsetzung Erst ADCL, dann ADCH lesen (atomarer Zugriff)</p>	<pre>void setup() {   // activate serial console   Serial.begin(9600);    // enable ADC functionality   ADCSRA  = (1 &lt;&lt; ADEN);    // use /128 prescaler (ADC requires 50 kHz to 200 kHz, see manual p271,   // but system clock is 16 MHz)   ADCSRA  = (1 &lt;&lt; ADPS2)   (1 &lt;&lt; ADPS1)   (1 &lt;&lt; ADPS0);    // select ADC2 as input pin (there is one AD converter for the 16 AD:   ADMUX  = (1 &lt;&lt; MUX1);    // use reference voltage 5V (Note: AVCC is AREF), manual, p281   ADMUX  = (1 &lt;&lt; REFS0); }  void loop() {   // trigger ADC conversion   ADCSRA  = (1 &lt;&lt; ADSC);    // wait until conversion is finished, see manual p286   while (ADCSRA &amp; (1 &lt;&lt; ADSC));    // read analog value, first LOW then HIGH register   unsigned int read = ADCL + 256 * ADCH;    double result = 5.0 * read / 1024.0;   Serial.println(result);   Serial.println(" Volt"); }</pre>	<p><b>A/D Wandlung</b></p> <p><b>Auflösung:</b> Wie viel Spannungsunterschied pro Stufe?</p> <p><math>V_{ref} / 2^i</math></p> <p><b>Repräsentant</b> eines Intervalls liegt in Intervallmitte um Quantisierungsfehler zu vermeiden</p> <p><b>Analog</b> Erstes Intervall: Repr. Von 000n Stufenbreite 1/2 LSB Letztes Intervall: Stufenbreite 1 1/2 LSB</p> <p><b>Fehlerquellen</b> Quantisierungsrauschen Umsetzungszeit Änderung d. Eingangs während der Umsetzung</p> 	<p><b>Ansätze zur A/D Wandlung</b> Komparator Parallelverfahren, Zählverfahren, Wägeverfahren</p> <p><b>A/D Umsetzung beim Atmega</b> Nur ein interner A/D Umsetzer, aber 16 analoge Eingangspins können an A/D Um. weitergeleitet werden. Konfigurierbar, welcher Eingang an A/D Um. Weitergeleitet wird</p> <p><b>Trigger:</b> Manuelles Auslösen: durch Codeanweisung Free Running Mode: Endlosschleife Auto Trigger: angestoßen durch Timeroverflow, Komparatorausgang etc</p> <p><b>Erkennen, dass A/D Umsetzung beendet wurde:</b> Auswerten eines speziellen Flags, oder durch speziellen Interrupt</p> <p><b>Wertebereich:</b> Single-Ended Conversion: <math>[0, V_{ref}]</math> Differential Conversion: <math>[-V_{ref}/2, V_{ref}/2]</math></p>



<div>Watchdog, Energiesparmodus, Reset</div> <div>WDTCSR</div> <div>Watchdog Modul Konfiguration</div> <div>Spezielles vorgehen zum Beschreiben des Registers! (Damit nicht ausversehen)</div> <div>MCUSR</div> <div>Informationen über Ursache des Resets (nach Neustart abrufbar)</div> <div>wdt_reset() (in C) (Assembler: WDT)</div> <div>Watchdog Timer zurücksetzen</div> <div>SMCR</div> <div>Energiesparmodus wählen</div> <div>sleep_mode()</div> <div>(Assembler: SE-Bit in SMCR setzen, dann SLEEP-Instruktion)</div> <div>Energiesparmodus aktivieren</div>		<div>Watchdog</div> <div>Timer, der hoch oder runterzählt. Muss vor Überlauf zurückgesetzt werden. Sonst: Interrupt oder Reset.</div> <div>Aufgaben:</div> <div>Überprüfung: Codestellen in vorgegebener Zeit erreicht? SW noch aktiv und nicht abgestürzt?</div> <div>Bei Timeout: Überführen in wohldefinierten Zustand. Neustart oder Interrupt auslösen.</div> <div>Erkennt Probleme, löst sie aber nicht!</div> <div>Prescaler: Beeinflusst Zeit bis Watchdog Timeout</div> <div>Energiesparmodus</div> <div>Energieverbrauch verringern durch: Systakt verlangsamen, Betriebsspannung verringern, abschalten nicht benötigter Module (Energiesparmodi (ESM))</div> <div>ESM unterscheiden sich bzgl. Abgeschalteter Komponenten und aufweckender Ereignisse (Ext Interrupts, Watchdog Interrupt, Speicherzugriff beendet, Timer, Anlegen einer (leeren) ISR und Aktivieren des Interrupts genügt).</div> <div>Aufwachen kann verzögert passieren</div> <div>Energiesparmodi beim Atmega2560:</div> <div>Idle Mode, ADC Noise Reduction Mode, Power Save Mode, Power Down Mode, Standby Mode</div>	<div>Reset</div> <div>System von wohldefiniertem Zustand starten</div> <div>Arten von Resets:</div> <div>Power-On Reset, Brown-Out Reset, External Reset, Watchdog Reset, Internal Reset</div> <div>Sensordaten</div> <div>In bestimmten Bereich linearer Zusammenhang zw. Messgröße (z.B. °C) u. Ausgangsspannung.</div> <div>Beispiel TMP 36: -40°C – 125°C</div> <div>750mV bei 25°C. Output Scale Factor 10mV/°C</div> <div>Min Ausgangsspannung: 100 mV</div> <div>Max Ausgangsspannung: 1750 mV</div> <div>Max Ausgangsspannung sollte möglichst knapp unter Referenzspannung liegen.</div> <div>Binäre Zahl (bei V<sub>ref</sub> = 2,56V):</div> <div>Max: 1,750V / 2,56V * 2<sup>10</sup> = 700</div> <div>Min: 0,100V / 2,56V * 2<sup>10</sup> = 40</div> <div>Binäre Zahl in Messgröße:</div> <div>Lineare Zusammenhang: y = mx + t</div> <div>y: Messgröße, x: binäre Zahl</div> <div>y und x gegeben, m und t bestimmen</div> <div>-40°C = m*40 + t [°C]</div> <div>125°C = m*700 + t [°C]</div>																																			
<div>Kommunikationsschnittstellen</div> <div>USART Register:</div> <div>UDR</div> <div>UCSRnA</div> <div>UCSRnB</div> <div>UCSRnC</div> <div>UBRRnL</div> <div>SPI Register:</div> <div>SPCR</div> <div>SPSR</div> <div>SPDR</div>		<div>Klassifizierung</div> <div>Seriell vs parallel   vs     </div> <div>Synchron (meist eigener Takt für Datenleitung) vs asynchron (Empfänger muss Takt d. Senders kennen)</div> <div>Bus (Mehr als zwei Geräte verbunden, erfordert Adressierung) vs Point-to-Point</div> <div>Vollduplex (Datenübertragung in beide Richtungen gleichzeitig möglich, separate Leitungen für Senden u. Empfangen) vs halbduplex</div> <div>Peer-to-Peer vs Master-Slave (Nur Master darf Kommunikation starten)</div> <div>Differential (Spannungsunterschied zw. 2 Leitungen trägt Information) vs Single-Ended (Gemeinsame GND Leitung für alle Datenleitungen)</div> <table><thead><tr><th></th><th>UART</th><th>SPI</th><th>I<sup>2</sup>C</th></tr></thead><tbody><tr><td>Seriell</td><td>Ja</td><td>Ja</td><td>Ja</td></tr><tr><td>Duplex</td><td>Ja</td><td>Ja</td><td>Nein</td></tr><tr><td>Synchron</td><td>Nein</td><td>Ja</td><td>Ja</td></tr><tr><td></td><td>kein Takt</td><td></td><td></td></tr><tr><td>Bus</td><td>Nein</td><td>Jein</td><td>Ja</td></tr><tr><td>Anz. Leitungen</td><td>3</td><td>5</td><td>3</td></tr><tr><td>Datenrate</td><td>BAUD = f<sub>osc</sub> / (16(UBRRn+1))</td><td>f<sub>osc</sub> / 128</td><td>Max 400 kbit/s</td></tr><tr><td>Atmega2560</td><td></td><td>- f<sub>osc</sub> / 2</td><td></td></tr></tbody></table> <div></div> <div>UART (oder SCI)</div> <div>2 Datenleitungen: TxD und RxD</div> <div>Sender u. Empfänger müssen Baudrate kennen</div> <div>Übertragung von UART-Frames D(E)O(N)S</div> <div>Beispiel: 8E1: 8 Datenbits, gerade Parität, 1 Stopbit</div> <div>SPI (hohe Geschwindigkeit)</div> <div>Master-Slave. 4 Datenleitungen:</div> <div>MOSI: Master Out, Slave IN (8 Bit Schieberegister)</div> <div>MISO: Master In, Slave Out (8 Bit Schieberegister)</div> <div>SSK: System Clock, SS: Slave Select</div> <div>I<sup>2</sup>C, TWI (Viele Geräte)</div> <div>Bus mit 7 Bit Adressierung</div> <div>SCL: Serial Clock Line, SDA: Serial Data Line</div> <div>Startbedingung: Fallende Flanke: SDA+SCL == HIGH</div> <div>Adresse anlegen -&gt; R/w: Master spezifiziert, ob Lese oder Schrieبزugriff -&gt; Datentransfer</div> <div>Stoppbedingung: Steigende Fl.: SDA+SCL == HIGH</div>		UART	SPI	I <sup>2</sup> C	Seriell	Ja	Ja	Ja	Duplex	Ja	Ja	Nein	Synchron	Nein	Ja	Ja		kein Takt			Bus	Nein	Jein	Ja	Anz. Leitungen	3	5	3	Datenrate	BAUD = f <sub>osc</sub> / (16(UBRRn+1))	f <sub>osc</sub> / 128	Max 400 kbit/s	Atmega2560		- f <sub>osc</sub> / 2	
	UART	SPI	I <sup>2</sup> C																																			
Seriell	Ja	Ja	Ja																																			
Duplex	Ja	Ja	Nein																																			
Synchron	Nein	Ja	Ja																																			
	kein Takt																																					
Bus	Nein	Jein	Ja																																			
Anz. Leitungen	3	5	3																																			
Datenrate	BAUD = f <sub>osc</sub> / (16(UBRRn+1))	f <sub>osc</sub> / 128	Max 400 kbit/s																																			
Atmega2560		- f <sub>osc</sub> / 2																																				
<div>Peripherie</div>		<div>CGRAM</div> <div>DDRAM</div> <div>Cursor</div> <div>4-Bit Modus</div> <div>Initialisierung Liquid Crystal</div>																																				
<div>SW-Download / Debugging</div>		<div>JTAG</div>																																				
<div>Automaten</div>																																						

# Register

## Digital IO:

- DDRx (Data Direction Register):
  - o Entsprechendes Bit auf 1 für Ausgang, oder 0 für Eingang
- PORTx (Port Register):
  - o Wenn Pin auf Ausgang, dann 1 = 5V und 0 = 0V
- PINx (Port Input Register):
  - o Wenn Pin auf Eingang, dann 1 = HIGH liegt an und 0 = LOW liegt an

## Timer:

- TCCRnA (Timer/Counter n Control Register A):
- TCCRnB (Timer/Counter n Control Register B):
  - o Prescaler
  - o Starten des Timers
  - o Input Capture
- TCNTn (Timer Counter n, 16 Bit):
  - o Aktueller Zählerstand
- OCRnA (Output Compare Register A, 16 Bit):
  - o Wert gegen den Zählerstand verglichen werden kann
- OCRnB (Output Compare Register B, 16 Bit):
  - o Wert gegen den Zählerstand verglichen werden kann
- ICRn (Input Capture Register):
  - o Bei Input Capture erfasster Wert wird gespeichert
- TIMSKn:
  - o Aktivieren/Deaktivieren der Timer Interrupts
- TIFRn:
  - o Timer bezogene Interrupt Flags

## Pulsweitenmodulation:

- OCnA:
- OCnB:
- OCnC (Output Compare Pins):
  - o Inverting oder non-Inverting Mode
  - o Output Compare Pins müssen als Ausgang im DDR Register konfiguriert sein!
- OCRnX (Output Compare Register):
  - o Vergleichswert muss gesetzt werden

## Interrupts:

- sei() (Set Enable Interrupt):
  - o Interrupts global aktivieren
- SREG:
  - o I Bit hier setzen statt sei() möglich
- EIMSK:
  - o De/aktivieren von speziellen Interrupts
- EIFR:
  - o Interrupt Flags
- EICRA:
- EICRB:
  - o Steigende/fallende Flanke?

## Analoge IO:

- ADMUX:
  - o Referenzspannung wählen
  - o Analoge Eingangspins für A/D Umsetzung wählen
- ADCSRB:
  - o Analoge Eingangspins für A/D Umsetzung wählen
  - o Single Ended oder Differential Conversion
  - o Free Running Mode oder manuelles Triggern
- ADCSRA:
  - o Aktivieren und Starten der A/D Umsetzung
  - o Prescaler
  - o Interrupts
- ADCL u. ADCH:
  - o Speichert Ergebnis der A/D Umsetzung
  - o Erst ADCL, dann ADCH lesen (atomarer Zugriff)

## TODO:

- Jeweils für Register relevante Manual Ausschnitte
- Bilder mit verschiedenen PWM: TOP/CMP/ und inverting/non-inverting mode
- Reset Ablauf, Seite 16 bei Watchdog Skript