|  |  |  |  |
| --- | --- | --- | --- |
| Speicher, Digitale Ein- und Ausgabe DDRx (Data Direction Register):  Entsprechendes Bit auf 1 für Ausgang, oder 0 für Eingang  PORTx (Port Register):  Wenn Pin auf Ausgang, dann 1 = 5V und 0 = 0V  PINx (Port Input Register):  Wenn Pin auf Eingang, dann 1 = HIGH liegt an und 0 = LOW liegt an  pinMode(13, OUTPUT);  digitalWrite(13, HIGH);  Serial.begin(9600);  Serial.println("Eingabe ist: " + Serial.readString());  Serial.available() |  | Nichtflüchtige Speicher:  ROM  OTPROM  EEPROM: Begrenzte Anzahl an Schreib/Lesezyklen.  Konfigurationsdaten, Kalibrierungsdaten  Flash: Programm- Daten/Code  SRAM(flüchtig): Arbeitsspeicher, Register, Stack usw  *Einzelne Bits setzen:* x |= (1 << Bitnummer);  *Einzelne Bits löschen:*  x &= ~((1 << Bitnummer) | (1 << Bitnummer2));  *Testen ob Bit auf 1:*  If (DDRC & (1 << Bitnummer)) { }  *Testen ob Bit auf 0:*  If ( ! (DDRC & (1 << Bitnummer))) { }  *Alle Bits umdrehen:*  x = 0xFF ^ x  *LED togglen:*  PINA |= (1 << PINA2);  Vorwiderstand berechnen: | **Pull-Up** / Active Low **Pull-Down** / Active High  Bei offenem Taster wird Bei offenem Taster wird  Spannung am Pin auf Spannung am Eingang  HIGH gezogen. Auf LOW gezogen.  Entprellung  Einmaliges betätigen eines Schalters führt evtl zu mechanischen Vibrationen.  SW-Lsg: Künstliche Wartezeit nach Zustandswechsel – Bis Schalter eingeschwungen. |
| **Mikrocontroller Bestandteile**: MicroProzessor, Timer, Schnittstellen, Speicher, AD-Wandler  **Entwicklerboard**: Arduino Mega,  **Mikroprozessor**: Atmega2560  **Cross-Compilation**: Programm wird nicht auf Zielplattform (Mikrocontroller), sondern auf anderer Plattform übersetzt  **Flashen**: hex-File von PC an Entwicklerboard senden  **Harvard-Architektur**: Daten- und Instruktionsspeicher getrennt  Instruktionsspeicher: Nicht flüchtiger Flash Speicher  Daten: in flüchtigem SRAM  SRAM und DRAM sind flüchtig. Rest nicht flüchtig. |
| Interrupts sei(): Interrupts global aktivieren  cli(): Interrupts global deaktivieren  *SREG* (AVR Status Register):  Bit 7 auf 1 = sei();  *EIMSK* (External Interrupt Mask Register):  Speziellen Interrupt de-/aktivieren  *EICRA* (External Interrupt Control Register A)  *EICRB* (External Interrupt Control Register B):  ISCn0 und ISCn1. Falling oder Rising Edge.  n ist die Interrupt Nummer.  *EIFR* (External Interrupt Flag Register):  Wenn Interrupt ausgelöst: Bit ist 1  *ISR* (INT0\_vect) { }  attachInterrupt(digitalPinToInterrupt(21), count, RISING); | //Entprellung | **Busy Waiting:** while (DDRC & (1 << DDC3));  **Polling**: periodisches Abfragen, ob Ereignis eingetreten  Interrupt: Kurze Unterbrechung des laufenden Programms um einen anderen zeitkritischen, kurzen Vorgang zu bearbeiten. Hardware prüft dauernd parallel, ob Ereignis eingetreten ist.  Wenn auf ein seltenes Ereignis schnell reagiert werden muss.  **Trap**: Art von Interrupt, die aber synchron und reproduzierbar ist. z.B. System Call, Div durch 0 …  Interrupt Request:  Interruptereignis – [InterruptController] – über IRQ Eingang Unterbrechungsanforderung an CPU – CPU unterbricht Programm und startet Unterbrechungsroutine  Interrupt Vector Table:  Welches Interruptereignis gehört zu welcher ISR? Jede Vectornummer hat eine zugehörige Programmadresse.  ISR ist selbst nicht unterbrechbar (I Bit SREG) | Externe Interrupts  Controller tastet zu Beginn jedes Taktzyklus ab. Falls Interrupt aktiviert, Aufruf der ISR.  Probleme: Leichte Verzögerung, „Prellung“  Interne Interrupts  Timer, A/D-Wandler  Bei Auslauf eines Timers unterbricht HW Ausführung der normalen Software  Volatile  Variable wird vor jedem Lesen aus SRAM gelesen und nach jedem Schreiben in SRAM geschrieben  !!Globale Variablen die in ISR vorkommen **immer volatile**!!! |
| Timer **n**: Timer 1-5  *TCCRnA* (Timer/Counter n Control Register A):  PWM  *TCCRnB* (Timer/Counter n Control Register B):  Prescaler; Starten des Timers; Input Capture, CTC  Beide TCCRn erst auf 0x00 setzen.  Auch wenn keinen Prescaler will, muss man setzen  *TCNTn* (Timer Counter n, 16 Bit):  Aktueller Zählerstand. Anfangs auf 0 setzen.  *OCRnA, OCRnB, OCRnC* (Output Compare Register, 16 Bit):  Wert gegen den Zählerstand verglichen werden kann  *ICRn* (Input Capture Register):  Bei Input Capture erfasster Wert wird gespeichert  *TIMSKn*:  Aktivieren/Deaktivieren der Timer Interrupts  *TIFRn*:  Timer bezogene Interrupt Flags  **CTC Beispiel**: TCCR4B |= (1 << WGM42);  ISR(TIMER4\_OVF\_vect) {}:  Interrupt bei Timer 4 Overflow  ISR(TIMER4\_COMPA\_vect){}: Timer 4 compare A |  | Atmega2560 Systemtakt = 16Mhz  16Bit Timer => Timer läuft nach über  Prescaler  Fallende Flanke des Prescalers an Bit Qn triggert Counter.  Vorteil großer Prescaler: Messen langer Zeiten möglich.  kleinstes messbares Zeitintervall ohne Prescaler:  mit f/1024 Prescaler:  Nachteil: Schlechtere Auflösung.   * immer kleinstmöglichen Prescaler!   Welchen Prescaler für 3s Intervalle mit 16 Bit Timer? Takt: 1 MHz (Bei 16MHz bis 3\*1000000) | Input Capture  Bei externen oder internen Signalen/Ereignissen wird aktueller Zählerstand in ICRn gespeichert    Output Compare  Bei erreichen eines konfigurierten Zählerstandes wird Interrupt ausgelöst, oder best. Signal erzeugt.  hier mit CTC  CTC Mode (Clear Timer on Compare Match)  TOP Wert in OCRnA oder ICRn konfiguriert.  Zähler bei erreichen des Zählerstandes automatisch auf 0. |
| Pulsweitenmodulation TCCRnA  Compare Output Mode  Fast PWM usw  TCCRnB  Fast PWM; Prescaler  OCnA, OCnB, OCnC (Output Compare Pins):  PWM-Ausgang  Inverting oder non-Inverting Mode  Output Compare Pins müssen **als Ausgang im DDR** Reigster konfiguriert sein!  OCRnX (Output Compare Register):  Vergleichswert (Schwellwert) muss gesetzt werden, der jeweils PWM-Ausgang OCnX beeinflusst. |  | Signal mit konstanter Periode, aber variabler Pulsdauer wird erzeugt.  Duty Cycle: t/T (= Pulsdauer / Periodendauer)  TOP: ICRn Register (oder andere  siehe S145 Tabelle)  CMP: OCRnX Register    In manchen Modi kann man TOP/CMP nur ändern, wenn Zähler gerade auf BOTTOM/TOP ist. (Update of OCRnX at... in Tabelle) | Inverting u. Non-Inverting Mode  Non-Inverting: siehe Links.  Inverting: PWM Ausgang genau andersrum  TCCRnX Register  Up-Down-Counter  doppelte Periodendauer, geringere Auflösung  BOTTOM u. TOP immer genau in der Mitte    Tabelle S145 TCCRnA u TCCRnB Fast PWM(Up-Counter), PWM(Up-Down Counter) |
| Analoge Ein-/Ausgabe ADMUX  oReferenzspannung wählen  oAnaloge Eingangspins für A/D Umsetzung wählen  ADCSRB  oAnaloge Eingangspins für A/D Umsetzung wählen  oSingle Ended oder Differential Conversion  oFree Running Mode oder manuelles Triggern  ADCSRA  oAktivieren und Starten der A/D Umsetzung  oPrescaler  oInterrupts  ADCL, ADCH  oSpeichert Ergebnis der A/D Umsetzung  oErst ADCL, dann ADCH lesen (atomarer Zugriff) |  | A/D Wandlung  Fehlerquellen |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Watchdog, Energiesparmodus, Reset |  |  |  |
| Kommunikationsschnittstellen |  |  |  |
| Peripherie |  |  |  |
| SW-Download / Debugging |  |  |  |
| Automaten |  |  |  |

Register

**Digital IO:**

* DDRx (Data Direction Register):
  + Entsprechendes Bit auf 1 für Ausgang, oder 0 für Eingang
* PORTx (Port Register):
  + Wenn Pin auf Ausgang, dann 1 = 5V und 0 = 0V
* PINx (Port Input Register):
  + 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):
  + Prescaler
  + Starten des Timers
  + Input Capture
* TCNTn (Timer Counter n, 16 Bit):
  + Aktueller Zählerstand
* OCRnA (Output Compare Register A, 16 Bit):
  + Wert gegen den Zählerstand verglichen werden kann
* OCRnB (Output Compare Register B, 16 Bit):
  + Wert gegen den Zählerstand verglichen werden kann
* ICRn (Input Capture Register):
  + Bei Input Capture erfasster Wert wird gespeichert
* TIMSKn:
  + Aktivieren/Deaktivieren der Timer Interrupts
* TIFRn:
  + Timer bezogene Interrupt Flags

**Pulsweitenmodulation:**

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

**Interrupts:**

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

**Analoge IO:**

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

Kochrezepte:

TODO:

* Jeweils für Register relevante Manual Ausschnitte
* Theorie zu jedem Thema. Evtl. 2/3 Seite Theorie und 1 1/3 Seite Register, Kochrezepte usw.
* Bilder mit verschiedenen PWM: TOP/CMP/ und inverting/non-inverting mode