

Register - kleiner schneller Speicherbereich in CPU  
**SRAM(static)**: Flip-Flop, 6 Transistoren -> schneller + teurer, CS=1:Din ändert Dout, CS=0:Din hat kein Einfluss auf Dout  
 Arbeitsspeicher speichert flüchtige Daten während LF Register, Stack  
**DRAM(dynamic)**: 1 Transistor -> langlebig günstig  
**Flash**: Programmdateien/code (Firmware - Code wird selten verändert)  
**EPPROM**: nichtflüchtige Konfiguration, Kalibrierungsdaten

ATMega2560(CM OS) Low <=1.5V, High >=3V

Niemals PINout mit Masse(GND) verbinden und auf 1 setzen!!!

PINS sind bidirekt

jeder PORT hat 8 PINS, werden über Register gesteuert (3Register pro Port)

DDRx(Data). PINx. PORTx

**Busy Waiting** (man wartet und blockiert bis ein Ergebnis eingetreten ist (while(DDRC & (1<<DDC3))

**Polling**(periodisch abfragen, ob Ergebnis eingetreten, zwischendrin weiterarbeiten)

Nachteile: -selte

Ereignisse(Verschwendung der CPU TIME);

-häufige Ereignisse (an mehreren

Codestellen auf eingetret-s Ereignis prüfen

**Interrupt**: vorübergehende Unterbrechung des laufenden Programms, um einen anderen, meist zeitkritischen und kurzen Vorgang zu bearbeiten. HW des mCs prüft

je-ol ob Ereignis einget.

Eigenschaften: async, nicht reproduzierbar

Quellen: -externe(Spannung an Eingang, z.B durch Tastendruck)

-interne(Tastatur, Maus, Drucker, Festplatte, Flash)

**Trap**: sync, reproduzierbar

Quellen: System Call, Unbekannter Befehl,

falsche Rechenoperation(1/0)

Takt erlaubt Sync-tion von Schaltkreisen/FlipFlop

Kenngrößen:

-Frequenz,

-DutyCycle:Dauer von t(high) im Verhältnis zur Periode

-ClockStability:Abweichung von Nominalfrequenz

-ClockJitter:Zufällige Schwankungen in Frequenz

-ClockDrift:Systematische Freq-änderung über Zeit

Aufbau:mC Timer:

-Startwertregister(Reg zum Festlegen eines Startwerte für Zähler)

-Zähler(zählt meist Nullflanken)

-Zählerstandreg(Reg zum Auslesen des Zählerstandes)

-Steuerung Takt und Ausgang(bestimmt

Eingangssignal des Zählers/Reaktion auf Ereignis

-1:n(Prescaler:verringert Takt)

-externer Takt/Ereignisse(steigende Flanke erhöht Zähler um 1)

-Freigabe(Zähler läuft nur bei Freigabe, de- und

clk = 16MHz und 16Bit timer

nach (2^16-1)/16MHz = 4 ms Überlauf

**CTC Mode** mit max Wert  
 Clear Timer on Compare Match Mode

**Pulsweitenmodulation(PWM)** - Modulationsart bei der ein Signal mit konst Periode, aber variabler Pulsdauer, erzeugt wird. Pulsdauer(High-Anteil innerhalb einer Periode == Duty Cycle)

Anwendung: Info-Übertragung, Ansteuern von Gleichstrommotoren, allgemeine Steuerungstechnik (Dimmen von LEDs)

Bei Servomotor wird bestätigt die Position oder Geschwindigkeit kontrolliert

Betriebsmodi beim ATMega2560:

- Normal:Zähle hoch bis zum Overflow, dann beginne mit 0

- CTC: konfig TOP -> Zähler zählt bis TOP, dann wieder automatisch auf 0

- PWM: HW-Unterstützung um direkt PWM-Signale zu erzeugen

**Erzeugung PWM-Signalen**: Timer + OutputCompare + INTs

CMP: OCnX (um zu verkleinen CMP muss auch)

setzt PWM-Signal auf LOW

TOP:Umkehrwert des Zählers , setzt PWM-Signal auf HIGH

ADMAX

o Wahl der Referenzspannung

o Wahl der analogen EingangsPins für A/D Umsetzung

**ADC SRB**

o Wahl der analogen EingangsPins für A/D Umsetzung

o Single-Ended oder Differential Conversion

o Free Running Mode oder manuelle Triggern

fängt mit 0.5, damit Wert

zwischen 0 und Vref/8 nicht als 0 ausgelesen wurde

Welchen Wert von U soll man annehmen?

Wert zwischen 0 und Vref/8 nicht als 0 ausgelesen wurde

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5

0.5