Einführung in AVR-Assembler

Easterhack 2008

Chaos Computer Club Cologne

Stefan Schürmans, BlinkenArea

stefan@blinkenarea.org

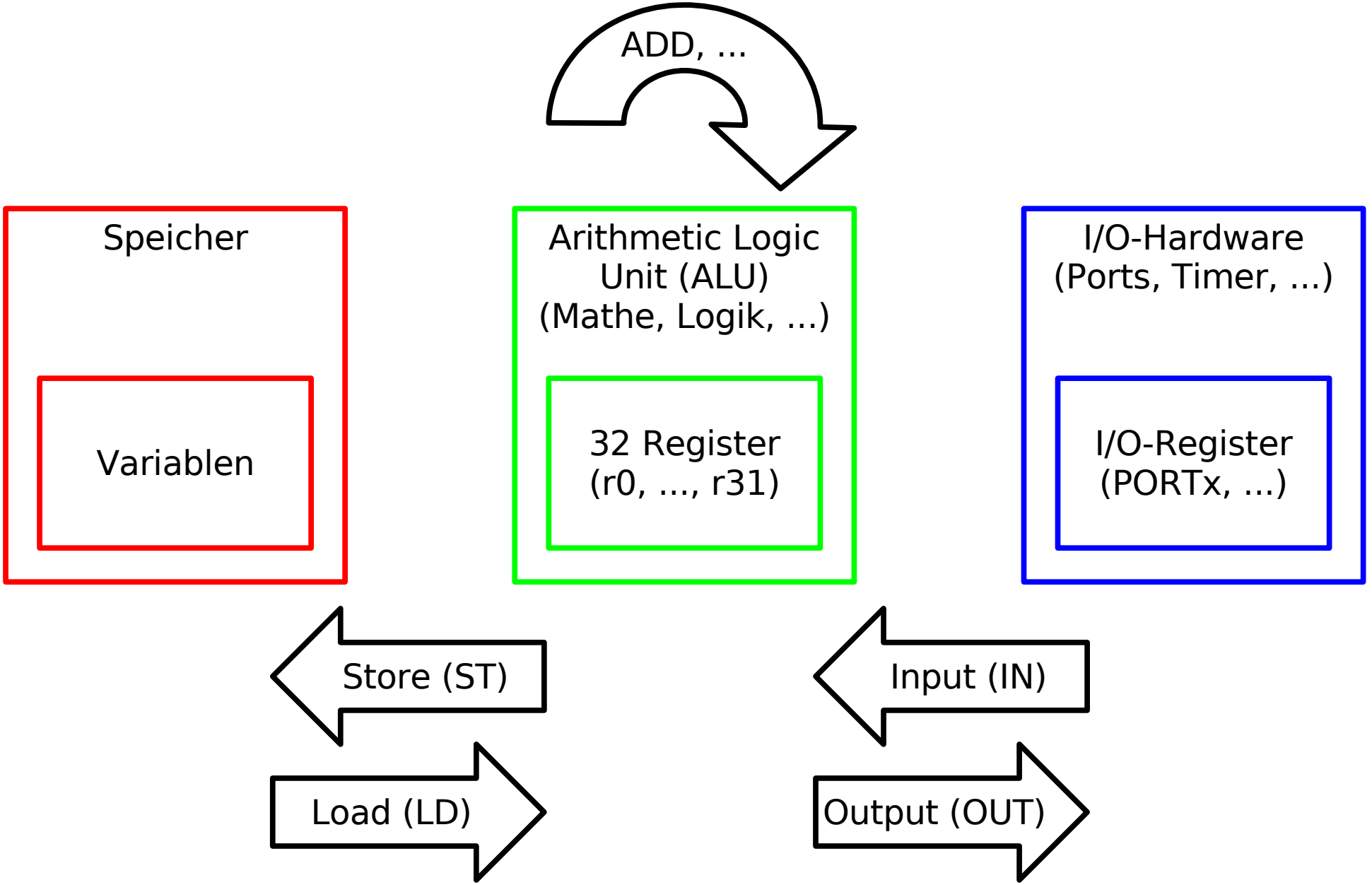
Version 1.0.4

Inhalt

- Vorstellung der AVR-Architektur
- Überblick über wichtigste AVR-Befehle
- Aufgabe: Elektronischer Würfel
- erstes AVR-Programm
- schrittweiser Ausbau bis zum Würfel
- Weiter gemäß Ideen der Teilnehmer

Fragen bitte jederzeit stellen. Wir haben genug Zeit...

AVR-Architektur



AVR - Register

Z = ZH:ZL = r31:r30
Y = YH:YL = r29:r28
X = XH:XL = r27:r26

indirekter Speicherzugriff
(Adressen, Zeiger)

obere Register
r16, ..., r31
je 8 Bit

Kopieren, Rechnen, Logik
Befehle mit Konstanten

untere Register
r0, ..., r15
je 8 Bit

Kopieren: MOV, IN, LD, ...
Rechnen: ADD, SUB, ...
Logik: OR, CP, ...

AVR - I/O-Register

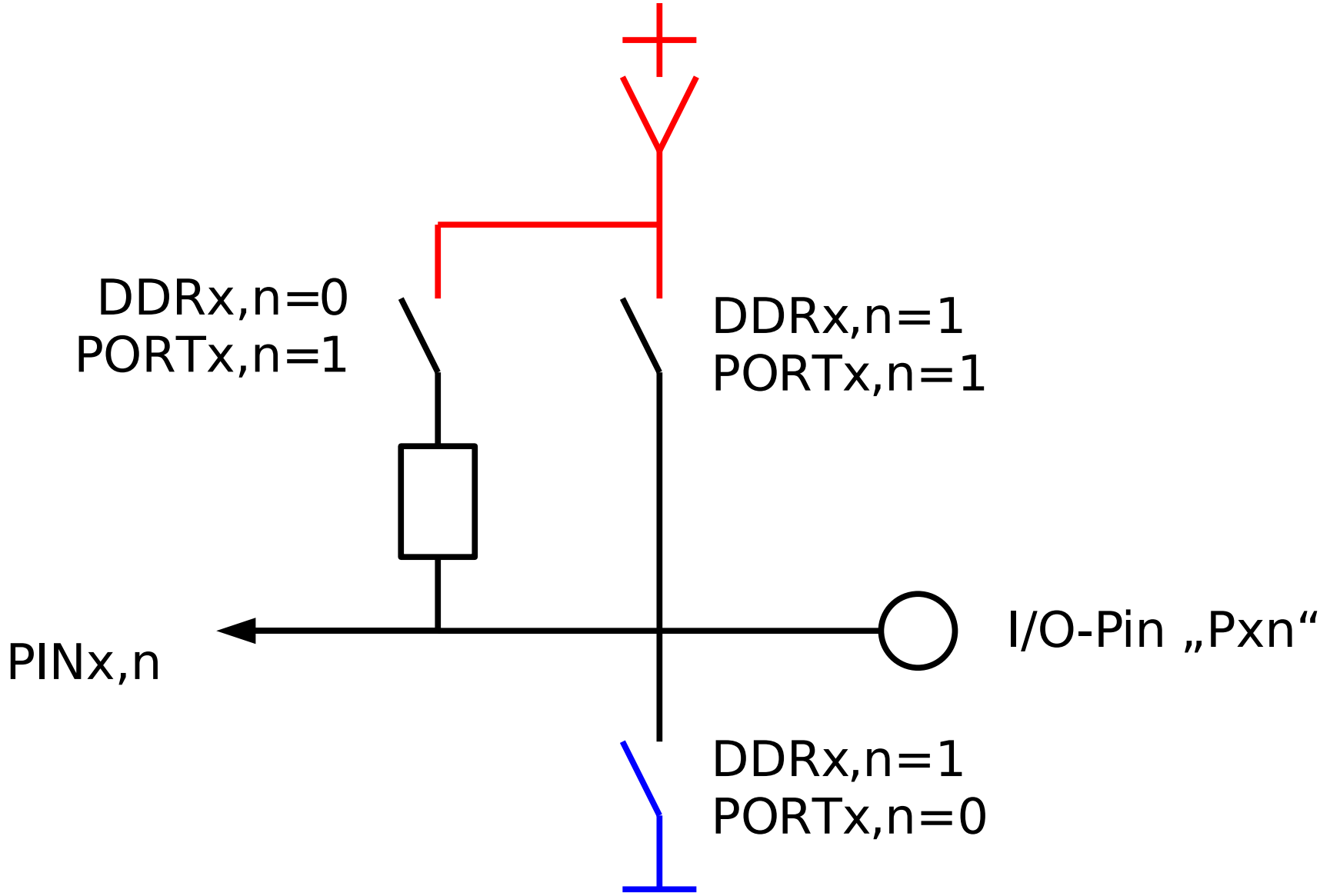
AVR-Kern-Register	
Program-Counter: PCH:PCL	aktueller Befehl
Stack-Pointer: SPH:SPL	Spitze des Stacks
Status-Register: SREG	Status-Flags (Übertrag, ...)
I/O-Port-Register	
Data-Direction-Reg.: DDRx	Kofiguration: Ein-/Ausgang
Port-Register: PORTx	Daten ausgeben
Pin-Register: PINx	Daten einlesen
Spezial-Hardware-Register	
Timer,	je nach vorhandener
serieller Port	Spezial-Hardware
...	in Prozessor

AVR - I/O-Ports

I/O-Port: je 8 I/O-Pins, Zugriff als Byte

	Lesen	Schreiben
DDRx,n	0 = Eingang 1 = Ausgang	0 = Eingang 1 = Ausgang
PORTx,n (DDRx,n = 1)	0 = Ausgang LOW 1 = Ausgang HIGH	0 = Ausgang LOW 1 = Ausgang HIGH
PORTx,n (DDRx,n = 0)	0 = Pin hochohmig 1 = Pin mit Pull-Up	0 = Pin hochohmig 1 = Pin mit Pull-Up
PINx,n	0 = Eingang LOW 1 = Eingang HIGH	0 = keine Änderung 1 = Ausgang ändern

AVR - I/O-Port-Schaltung



AVR – Befehle - allgemein

- LDI r16, 42
  - r16 := 42
- MOV r0, r1
  - r0 := r1
- CLR r0
  - r0 := 0
- NOP
  - keine Operation
- RJMP LABEL
  - Sprung zu LABEL
- RCALL PROC
  - Unterprogramm PROC aufrufen
- RET
  - Ende des Unterprogramms

AVR – Befehle - Rechnen/Logik

- INC r0
  - r0 := r0 + 1
- DEC r0
  - r0 := r0 - 1
- ADD r0, r1
  - r0 := r0 + r1
- SUB r0, r1
  - r0 := r0 - r1
- AND r0, r1
  - r0 := ro & r1
- OR r0, r1
  - r0 := r0 | r1
- EOR r0, r1
  - r0 := r0 ^ r1
- LSL r0
  - r0 := r0 << 1

AVR – Befehle – Speicher, I/O

- LDS r0, VAR
  - r0 := VAR
- STS VAR, r0
  - VAR := r0
- LD r0, X
  - r0 := [X]
- ST X, r0
  - [X] := r0
- IN r0, PINB
  - r0 := PINB
- OUT PORTB, r0
  - PORTB := r0
- SBI PORTB, 5
  - PORTB, 5 := 1
- CBI PORTB, 5
  - PORTB, 5 := 0

AVR – Befehle – Vergleichen

- CP r0, r1
  - Vergleich r0 mit r1
  - setze Flags in SREG
- CPI r16, 42
  - Vergleich r16 mit 42
  - setze Flags in SREG

- BREQ LABEL
  BRNE LABEL
- Sprung zu LABEL falls (un)gleich
  - gem. Flags in SREG
- BRLO LABEL
  BRSH LABEL
  - Sprung zu LABEL falls (nicht) kleiner
  - gem. Flags in SREG

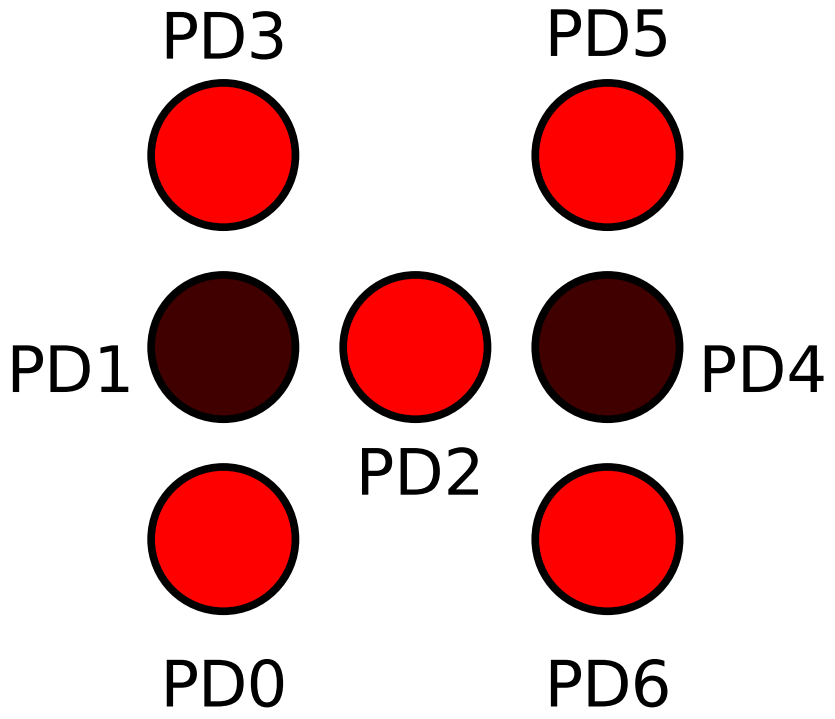
AVR – Befehle - Bit-Test

- SBRC r0, 5
  - wenn r0,5 = 0
    - nächsten Befehl auslassen
- SBRS r0, 5
  - wenn r0,5 = 1
    - nächsten Befehl auslassen
- SBIC PINB, 5
  - wenn PINB,5 = 0
    - nächsten Befehl auslassen
- SBIS PINB, 5
  - wenn PINB,5 = 1
    - nächsten Befehl auslassen

Aufgabe: Elektronischer Würfel

- Mikroprozessor
  - Atmel ATtiny2313
- Start-Knopf
  - PB3
  - gedrückt = 0
- Ausschalter
  - PB4
  - gedrückt = 0

- Ergebnis-Anzeige
  - eingeschaltet = 1



Schaltplan zeigen

Elektronischer Würfel - Prinzip

- Start-Knopf startet Würfel-Vorgang
  - Neustart bei erneutem Drücken
- automatischer Würfel-Vorgang
  - Würfel-Muster „zählen“ durch
    - erst schnell (ca. 10ms)
    - immer langsamer (bis ca. 300ms)
  - Ergebnis zufällig
- Ausschalter schaltet alle LEDs aus
  - egal ob Würfelvorgang aktiv oder nicht

Hello AVR

- Start-Knopf (PB3) steuert LED1 (PD0)
  - von PINB, 3 einlesen
  - auf PORTD, 0 ausgeben
- Ausschalter (PB4) steuert LED2 (PD1)
  - von PINB, 4 einlesen
  - auf PORTD, 1 ausgeben

Programm schreiben

Assemblieren und Flashen

- Assemblieren:
  - avra -l edice.lst edice.asm
- „Fuses“ (Konfig-Bits) setzen:
  - `avrdude -c stk200 -p t2313 -u -U lfuse:w:0x64:m -U hfuse:w:0xD9:m -U efuse:w:0xFF:m`
- Programm flashen:
  - `avrdude -c stk200 -p t2313 -u -U flash:w:edice.hex`

Schaltplan Programmieradapter zeigen

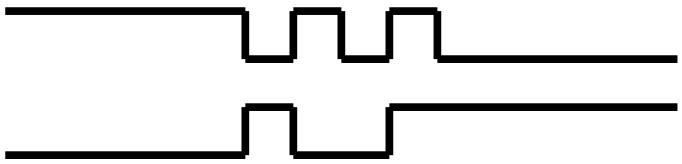
Würfel-Muster

- Start-Knopf (PB3) schaltet LEDs auf nächstes Würfel-Muster, Ausschalter (PB4) schaltet LEDs aus
  - Drücken der Knöpfe erkennen
    - Übergang von „losgelassen“ nach „gedrückt“
  - aktuelles Würfel-Muster merken
    - z.B. 0=aus, 1...6=Nummer des Würfel-Musters
  - Ausgabe der Muster
    - Unterprogramm

Programm erweitern

Entprellung

- Knöpfe „prellen“ beim Drücken/Loslassen
  - Drücken:
  - Loslassen:



- Würfel-Muster werden „übersprungen“
- Korrektur in Software
  - Zustand der Knöpfe z.B. alle 10ms einlesen
  - Knopf erst wirklich gedrückt/losgelassen, wenn z.B. 5 mal hintereinander gleich

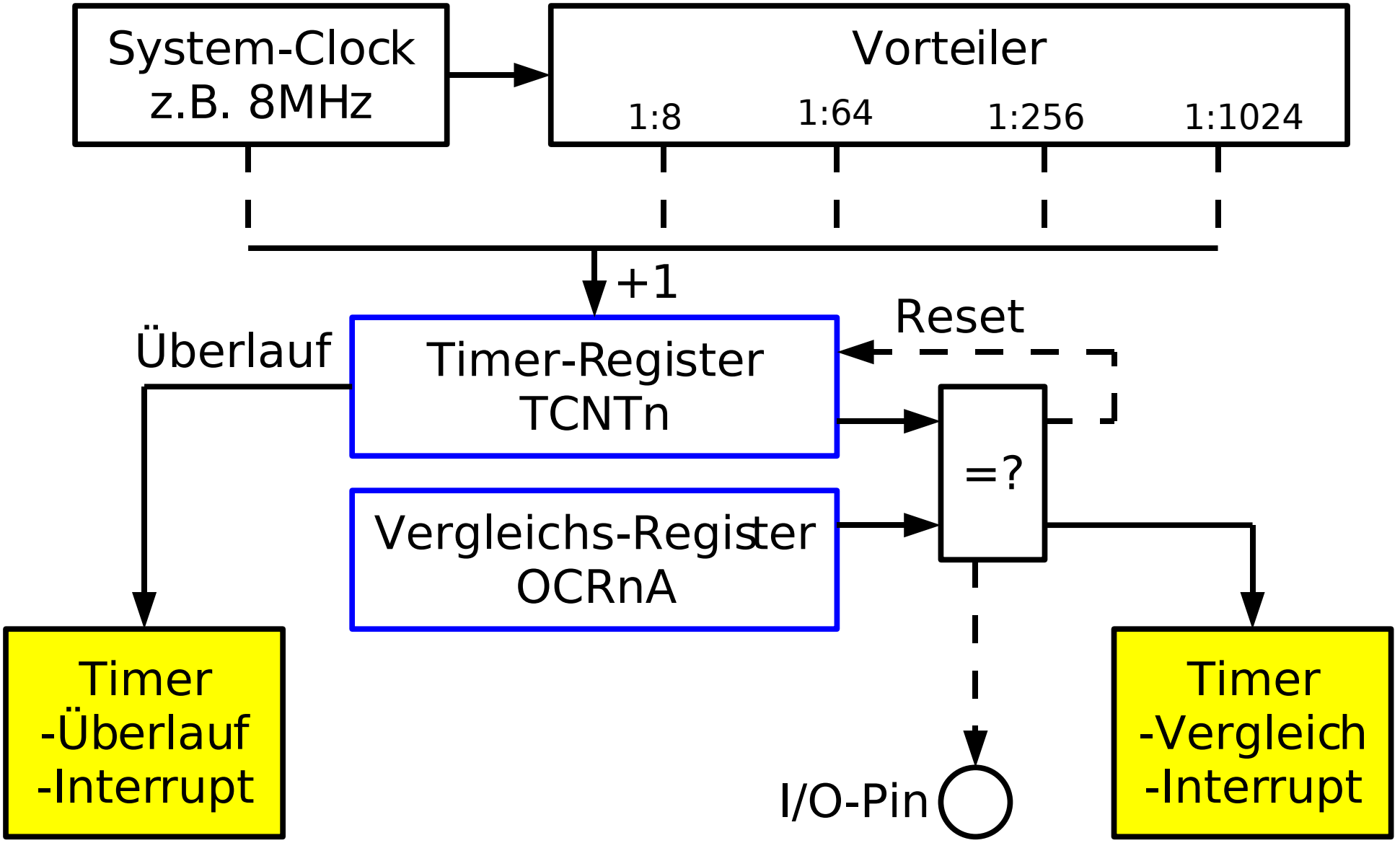
Interrupts

- Haupt-Programm läuft
- Ereignis
  - Haupt-Programm wird unterbrochen
  - Ereignis wird von speziellem Code verarbeitet
  - Haupt-Programm wird fortgesetzt
    - an der Stelle, an der es unterbrochen wurde
- Haupt-Programm läuft weiter
  - Ereignis-Code darf „nichts“ verändert haben
    - insbesondere r0, ..., r31 und SREG

AVR – Befehle - Interrupt

- PUSH r0
  - r0 auf Stack legen
  - z.B. Sicherung des Register-Wertes
- POP r0
  - r0 vom Stack holen
  - z.B. Wieder-herstellung des ursprünglichen Register-Wertes
- Interrupt-Aufruf
  - PC auf Stack sichern
  - Interrupts deaktivieren
- RETI
  - Ende des Interrupts
  - PC vom Stack wiederherstellen
  - Interrupts wieder aktivieren

AVR – Timer (vereinfacht)



Entprellung mit Timer 0

- Konfiguration Timer 0
  - Ziel: Überlauf ca. alle 8ms
  - Überlauf-Interrupt einschalten
- Bearbeitung Knöpfe in Überlauf-Interrupt
  - Entprellung
    - aktuellen Zustand einlesen
    - mit letzten paar (z.B. 4) Zuständen vergleichen
  - neu gedrückte Taster erkennen
    - Übergang von „losgelassen“ nach „gedrückt“

Programm um Entprellung erweitern

Elektr. Würfel – glob. Variablen

- aktuelles Würfel-Muster
  - 0 = aus
  - 1...6 = Nummer des angezeigten Musters
- Zeit bis zum nächsten Muster
  - in Schritten von ca. 8ms
  - 0 = kein Würfel-Vorgang aktiv
  - 1..40 = Dauer des aktuellen Bildes
- Rest-Zeit des aktuellen Musters
  - in Schritten von 8ms
  - 0 = aktuelles Bild ist stabil
  - 1..40 = Rest-Dauer des aktuellen Bildes

Elektr. Würfel - Ablauf

- Start-Knopf
  - Muster aussuchen
    - erst einmal immer 1
  - Muster anzeigen
  - Zeit := 1
- Ausschalter
  - Muster auf 0
  - LEDs aus
  - Zeit, Rest-Zeit := 0
- alle 8ms
  - Rest-Zeit > 0
    - Rest-Zeit - 1
  - Rest-Zeit wird 0
    - nächstes Muster
    - Muster anzeigen
    - Zeit + 1
    - Rest-Zeit := Zeit
  - Zeit > 40:
    - Zeit, Rest-Zeit := 0

Programm zum Würfel ausbauen

Zufall für Muster

- Zufall benötigt Entropie
  - Idee: Entropie aus Benutzereingaben
    - gemäß Ansatz von Unix
    - Zeitpunkt des Drucks auf Start-Knopf
  - einfache Lösung
    - Zeit seit Einschalten
    - direkt passend von 1 bis 6 zählen
      - Erkennung Druck auf Start-Knopf „nur“ alle 8ms
      - deshalb alle 8ms weiterzählen

Zufall für Muster - Ablauf

- zus. glob. Variable
  - Zufalls-Zähler
    - 1...6
    - mit diesem Muster beginnen, wenn jetzt Start-Knopf gedrückt wird
- alle 8ms
  - Zufalls-Zähler + 1
  - Zufalls-Zähler > 6
    - Zufalls-Zähler := 1
- Start-Knopf
  - Muster aussuchen
    - nach Zufalls-Zähler
  - ...

Zufall in Würfel-Programm einbauen

Watchdog-Timer

- Absturz im Mikrocontroller
  - keine Reaktion mehr
  - I/O-Pins nehmen verbotene Zustände an
    - Hardwareschäden
- Neustart bei Absturz
  - Wiederherstellung eines definierten Zustands
- Absturz-Erkennung
  - durch Ablauf eines Timers (z.B. 64ms)

Watchdog-Timer – Verwendung

- Programmbeginn
  - Watchdog-Timer konfigurieren
    - über I/O-Register WDTCSR
- Haupt-Schleife
  - Watchdog-Timer oft genug zurücksetzen
    - mit Befehl WDR
- Interrupts
  - Watchdog-Timer **nicht** zurücksetzen
    - Interrupt kann bei Absturz noch funktionieren

Watchdog-Timer in Würfel-Programm einbauen

Ideen der Teilnehmer

- Fragen?
- Erweiterungen des Würfels?
  - evtl. direkte Umsetzung
- andere I/O-Bausteine des Prozessors?
  - PWM, USI, USART, ...
- andere AVR-Prozessoren?
  - ATtiny13, ATMEGA8, ATMEGA128, ...
- sonstiges?