Ihre Referenten:

Dipl.-Ing. (FH) Raimund Lehmann
Dipl.-Ing. (FH) Stefan Staiger

Übersicht 02.04.2008

Hochschule Offenburg
University of Applied Sciences

- Motivation und Ziel
- Ablauf des Kurses
- Was braucht man für den Kurs
- HSO Atmega32 Board
- Programmer USBasp
- Aufbau und Funktion eines Mikrocontrollers

Motivation und Ziel

Hochschule Offenburg
University of Applied Sciences

- Motivation
  - Praktische Vertiefung von theoretischen Grundlagen
    - Regelungstechnik
    - Filterentwurf
- Ziel
  - Spaß beim Programmieren
  - Grundkenntnisse vermitteln

Ablauf des Kurses

Hochschule Offenburg
University of Applied Sciences

- Einführende Vorlesungen
  - Aufbau und Funktion eines Mikrocontrollers
  - WinAVR, AVRdude
  - Fuse Bits
  - Timer, AD-Wandler, Interrupts, I²C,..
  - ...
- Übungen
  - Taster einlesen, LED's setzen
  - ...
- Individuelle Projekte
  - Regelung, Steuerung, ...

Was braucht man für den Kurs

Hochschule Offenburg
University of Applied Sciences

- PC bzw. Laptop
- C-Compiler (AVR-gcc)
- Programmiersoftware (avrdude)
- Board mit einem Atmel Prozessor
- Programmieradapter (z.B. AVR ISP) bzw. Programmer (z.B. USBasp)
- Internetzugang

HSO Atmega32 Board

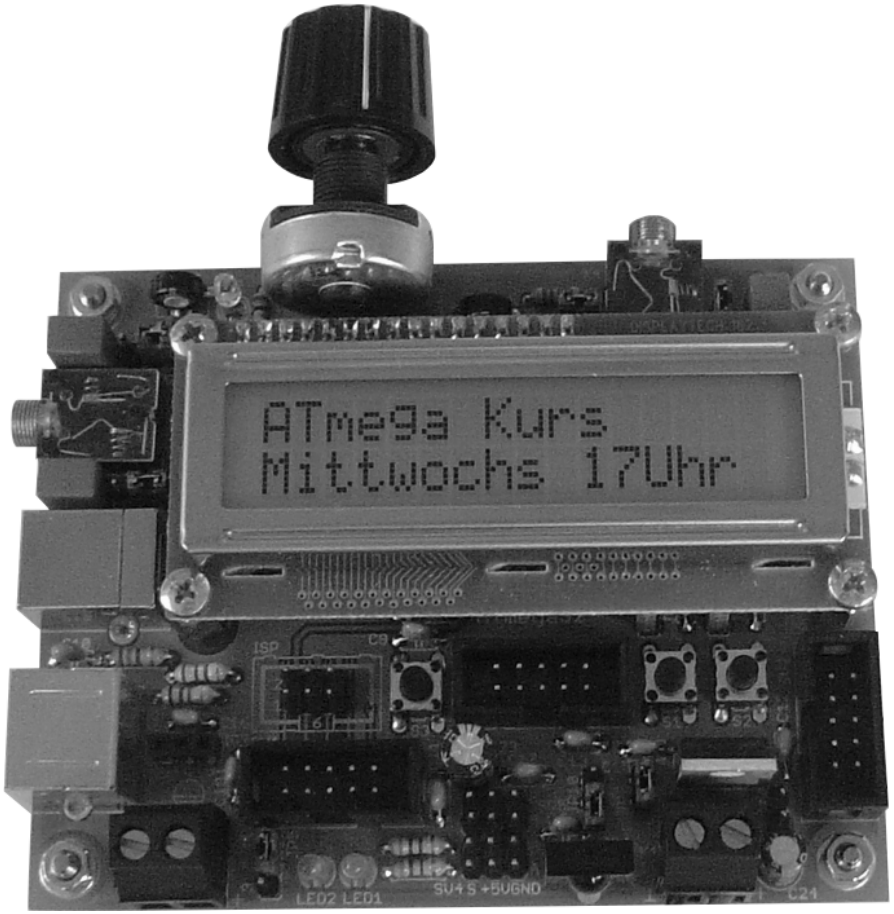
Hochschule Offenburg
University of Applied Sciences

- USB und PS2 Schnittstelle
- 2x16 Zeichen Display
- DA- und AD-Wandler über Klinkenbuchsen
- Temperaturfühler
- Fototransistor
- Infrarot Empfänger für Fernbedienungen
- Taster, LED's
- Schnittstelle für vier Servomotoren
- Spannungsstabilisierung
- Leistungsausgang (MOSFET)
- Erweiterungsmöglichkeit über Wannenstecker

Programmer USBasp

Hochschule Offenburg
University of Applied Sciences

- Plattformen
  - Windows (ab win98)
  - Linux
  - Mac OS X
- USB-Anschluss
- Programmiergeschwindigkeit
  - 5kBytes/sec
- Open Source
- Lange und gute Erfahrungen



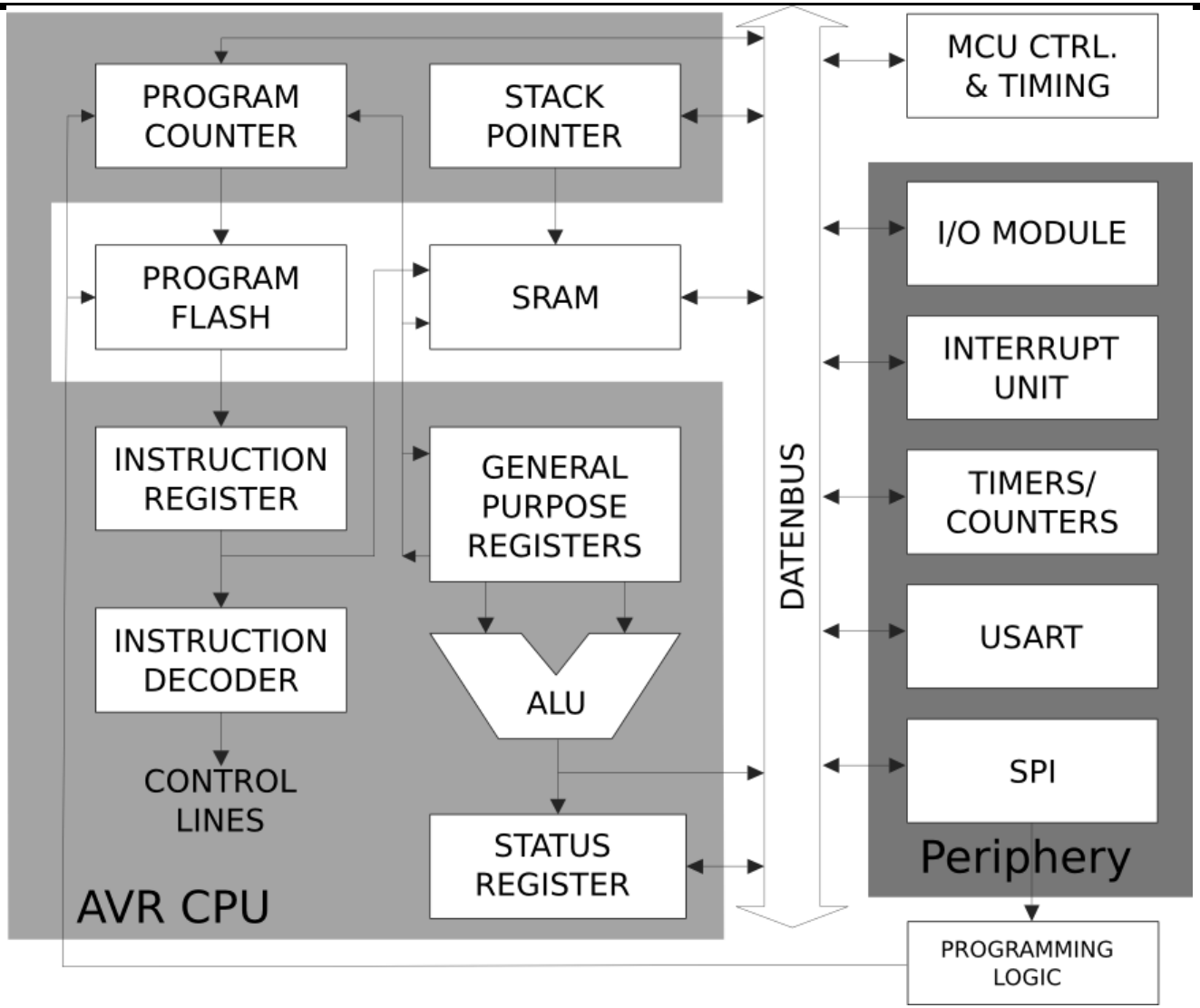
Fragen?

Aufbau und Funktion eines Mikrokontrollers

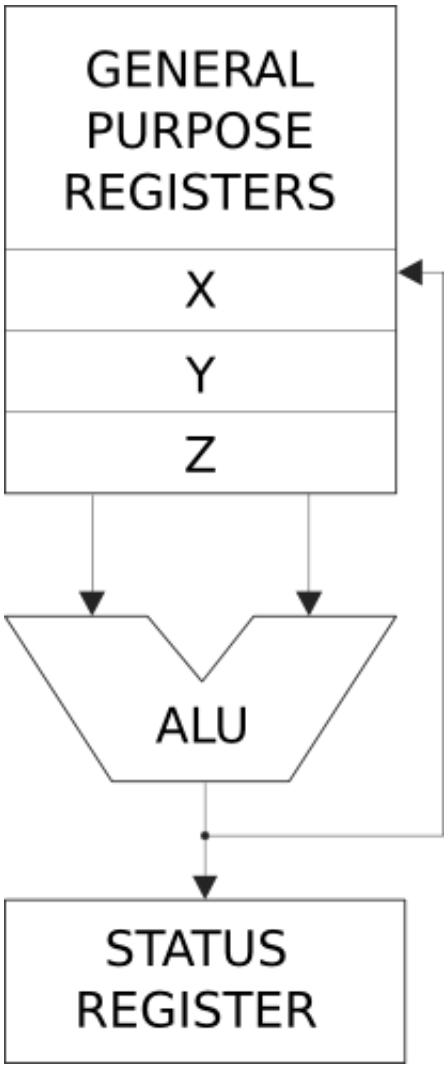
Übersicht

- Aufbau
- Arithmetic Logic Unit
- Befehlsablauf
- Stackpointer
- Speicheradressierung
- Beispiel 1
- Zusammenfassung CPU
- I/O Register
- Beispiel 2

Aufbau

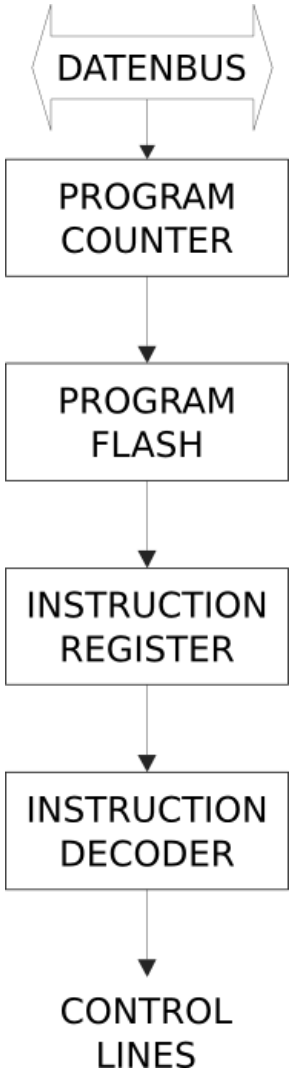

PROGRAM COUNTER
STACK POINTER
PROGRAM FLASH
SRAM
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
INSTRUCTION DECODER
ALU
CONTROL LINES
STATUS REGISTER
AVR CPU
DATENBUS
MCU CTRL. & TIMING
I/O MODULE
INTERRUPT UNIT
TIMERS/ COUNTERS
USART
SPI
Periphery
PROGRAMMING LOGIC

Arithmetic Logic Unit (ALU)



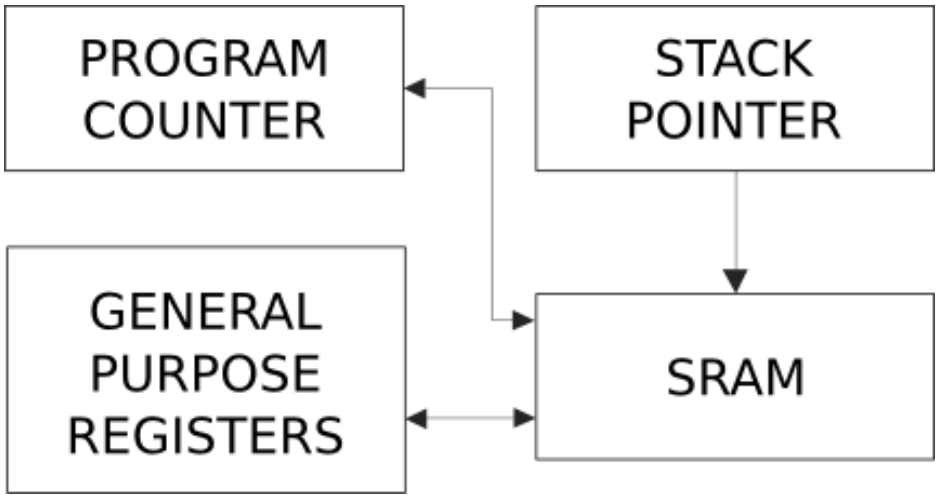
- ALU: Führt arithmetische und logische Operationen durch (z.B. Addition, Subtraktion, Und, Oder...).
- General Purpose Register: Arbeitsregister für die Alu und für Zugriffe auf andere Speicher. Die letzten 6 Register lassen sich als 16-Bit Register ansprechen.
- Status Register: Beinhaltet Flags für das Ergebnis der ALU
  (z.B. Zeroflag, Carryfalg...).
- Assembler Befehle: and, add, mul ...

Befehlsablauf



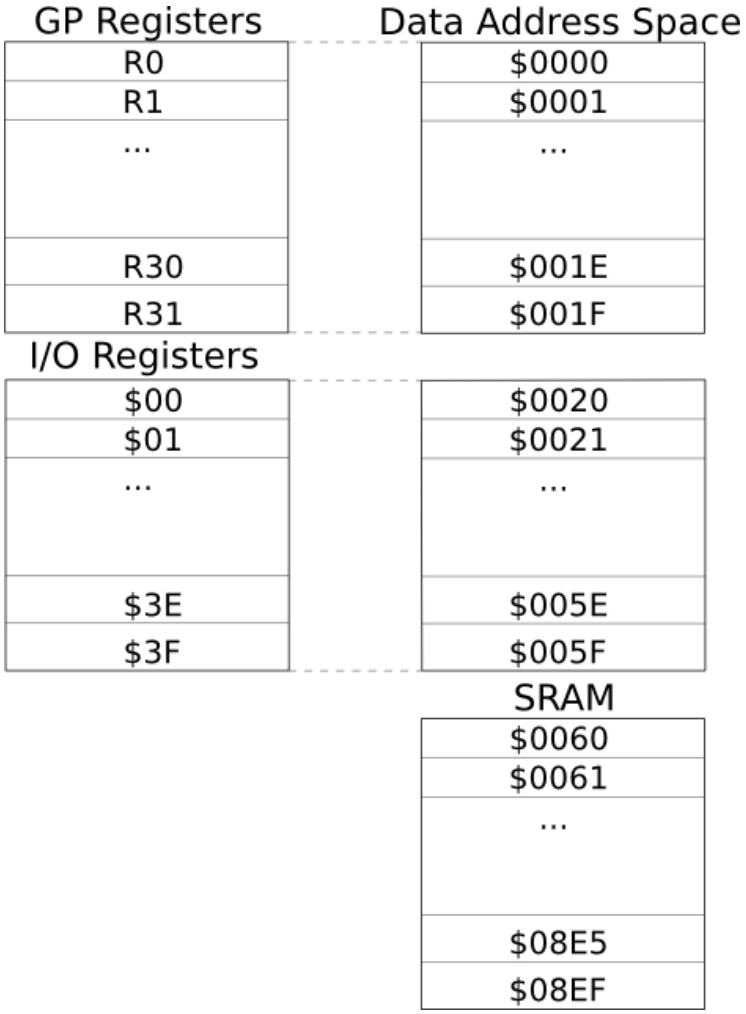
- Program Counter: Zeigt auf den aktuellen Befehl. Wird nach jeder Befehls-ausführung inkrementiert. Sprünge werden durch das Laden eines bestimmten Wertes realisiert.
- Flash: Beinhaltet Programm und Daten.
- Instruction Register. Zwischenspeicher für die Befehle (Pipelining).
- Instruction Decoder: Dekodiert die Befehle.
- Assembler Befehle: jmp, breq, brlo

Stackpointer



- Mit Hilfe des Stackpointers können Registerinhalte auf dem SRAM gespeichert bzw. vom SRAM geladen werden.
- Wichtig für das Aufrufen von Unterprogrammen.
- Assembler Befehle: push, pop, call, ret...

Speicheradressierung



- Speicher ist linear aufgebaut.
- Der Speicher kann auf verschiedene Arten angesprochen werden: direkt, indirekt, indirekt + offset, indirekt mit predekrement und indirekt mit postinkrement.
- Assembler Befehle: mov, ld, st, in, out ...

Beispiel 1

- C:

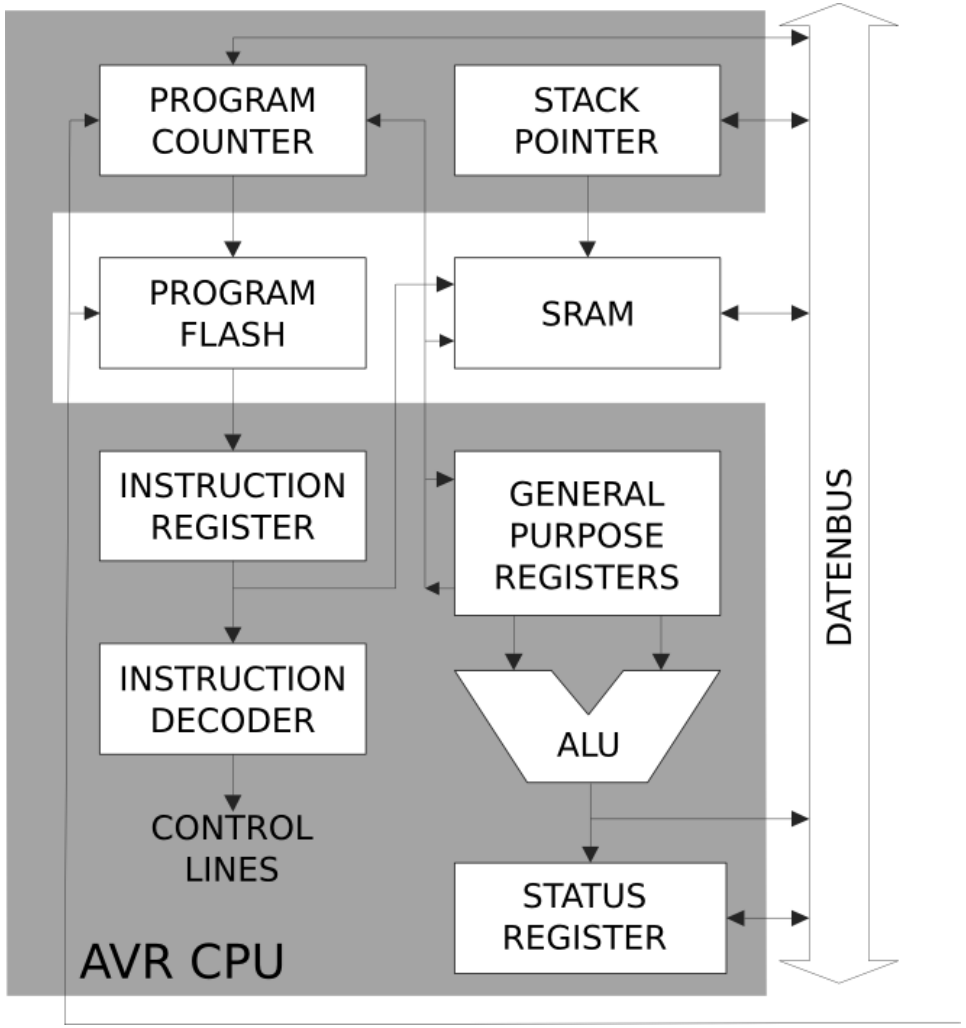
```
...
uint8_t a;
uint8_t b;
a= 0x10;
if(a == 0x12)
{
   b = a+0x02;
}
...
```

- Assembler

```
...
   ldi r4,0x10  ;lade Konstante
   cpi r4,0x12  ;vergleiche mit
                ;Konstante
   brne L2      ;springe wenn
                ;ungleich
   ldi r5,0x2
   add r5,r4    ;Addition
L2: ....
```

Zusammenfassung CPU

- Arithmetische und Logische Befehle
- Transport Befehle
- Sprungbefehle
- Bitweise Operatoren



I/O Register

- Dienen zur Steuerung und zum Auslesen der Peripherie.
- Alle I/O Register könne wie gewöhnliche SRAM Inhalte beschrieben werden.
- Jedes Bit in einem I/O Register hat eine bestimmte Bedeutung.
- Bestimmte Bits werden automatisch gesetzt und/oder gelöscht.
- Beispiel:

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Beispiel 2

Hochschule Offenburg
University of Applied Sciences

```
...
*(volatile uint8_t*)0x3B = 0x55;
...
//oder PORTA= 0x55;
```

```
    ldi r30,0x3B     ;Die Adresse von PortA
    ldi r31,0x00     ;ins Z-Register laden
    ldi r24,0x55     ;Konstante indirekt
    st Z,r24         ;ins den Speicher sichern

    oder:
    ldi r24,lo8(85)  ;Konstante laden
    out 0x1B,r24     ;und in PortA schreiben
```

Leistungsmerkmale des Atmega32

Hochschule Offenburg
University of Applied Sciences

- Taktfrequenz bis zu 16Mhz (16MIPS)
- on-chip 2-Takt Multiplizierer
- Interner Speicher: 32 kB Flash, 1kB EEPROM, 2kB SRAM
- kann direkt im System programmiert werden (ISP)
- JTAG Interface
- Timer/Zähler: 2x8Bit, 1x16Bit
- Schnittstellen: USART, TWI, SPI
- 32 I/O Ports
- 10-Bit Analog/Digital-Wandler