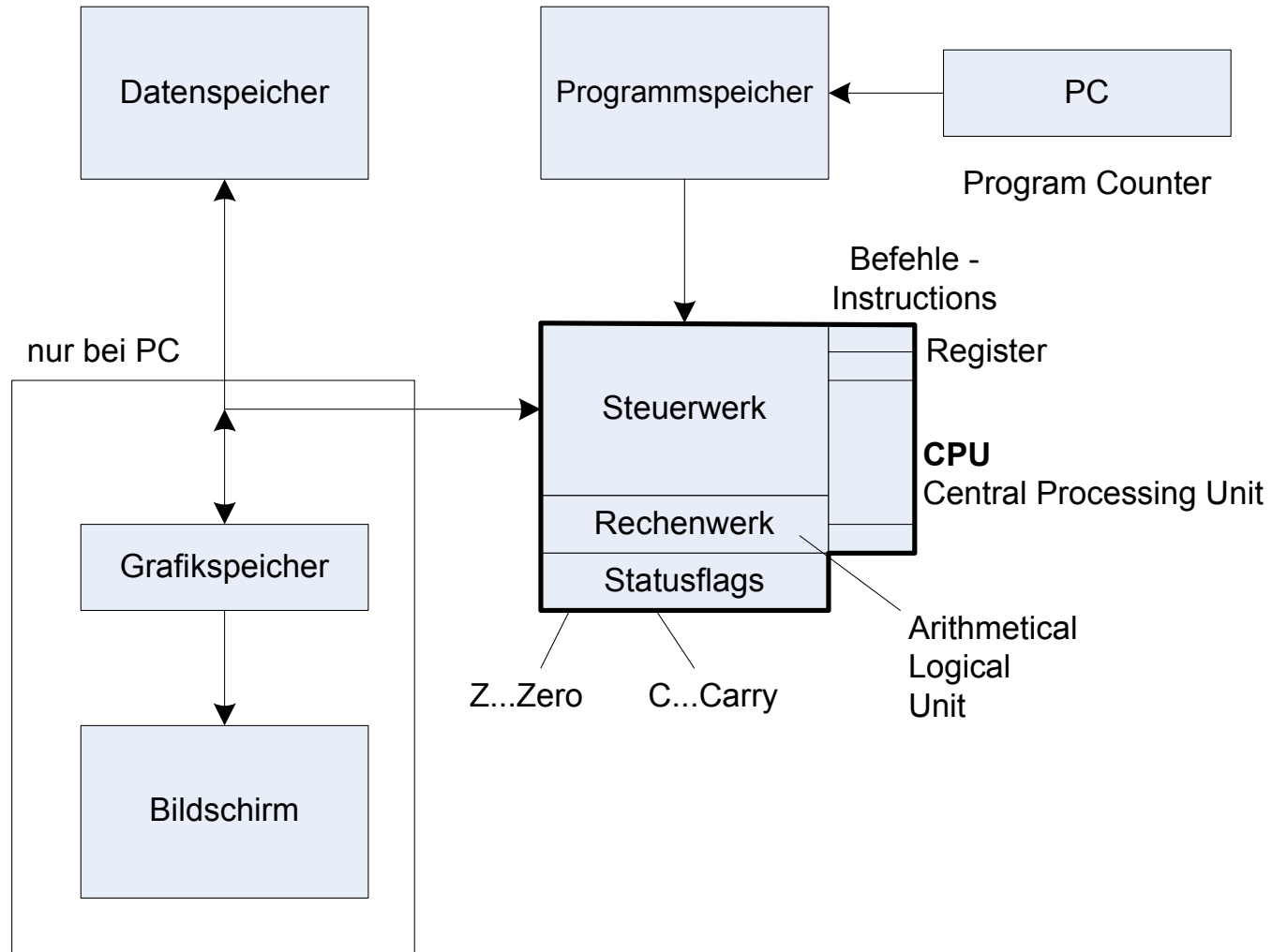




Aufbau eines μ Controllers

Bernhard Wintersperger

Überblick





Befehlssatz

Jeder μC ($\mu\text{Controller}$) verfügt über Befehle verschiedener Arten.

Alle zusammen nennt man den Befehlssatz.

Befehlsarten:

- Arithmetisch-Logische Befehle
- Verzweigungen
- Daten-Transfer Befehle
- Bit-Test Befehle
- Kontrollbefehle



Befehlssatz

Arithmetisch-Logische Befehle

Erklärung:

Wie der Name schon sagt handelt es sich hierbei um Arithmetische oder Logische Befehle um entsprechende Operationen innerhalb des μC auszuführen.

Mit ihnen lassen sich beispielsweise Register addieren oder über zwei Register eine Boolesche AND-Operation ausführen.

Beispiel:

- ADD R30, R31
Addiere Register30 und Register31
- INC R28
Erhöhe Register28 Inkrementell um 1



Befehlssatz Verzweigungen

Erklärung:

Verzweigungen (Branches) dienen dazu um innerhalb des Programms an andere Stellen zu Springen um somit Beispielsweise eine Verzweigung zu realisieren.

Die Stellen an die der μC springen kann nennt man Sprungmarken.

Beispiel:

- `CPI R28, 0x1E`
Vergleiche Register28 mit dem Wert 0x1E
- `BRNE jump1`
Falls R28 ungleich 0x1E springe zu Sprungmarke „jump1“.



Befehlssatz

Daten-Transfer-Befehle

Erklärung:

Dienen dazu Daten in Register zu laden oder diese zu verschieben.

Beispiel:

- LDI R31, 0x01
Lade den Wert 0x01 ins Register31
- LD R20, Y
Lade den Wert der auf der zusammengesetzten Adresse aus den Werten von R28 und R29 steht.



Befehlssatz

Bit-Test Befehle

Erklärung:

Dienen dazu um bestimmte Bits oder Flags innerhalb des μC zu setzen.

Beispiel:

- SBI 0x1B, 0x00
Setze PORTA auf 0x01



Befehlssatz Kontrollbefehle

Erklärung:

Einige wenige
Befehle für
spezielle Vorgänge
im μC .

Beispiel:

- NOP
Tue Nichts
- SLEEP
Wechsle in den
Schlafmodus.



Adressierungsarten

- Registeradressierung:
 - Ziel ist ein Register
 - Register wird im Befehl direkt eingegeben
- Unmittelbare (immediate) Adressierung:
 - Quelle ist der Programmspeicher
 - Adressen werden direkt eingegeben
- Direkte (absolute) Adressierung:
 - Ziel ist ein Speicher im Hauptspeicher
 - Adresse wird direkt eingegeben
- Indirekte (relative) Adressierung:
 - Wert der im Register steht wird zur Adresse auf den der Pointer verweist



Assembler

- Assembler ist eine Programmiersprache die die Maschinensprache für einen bestimmten Prozessor in einer für Menschen lesbaren Form darstellt.
- Jede Prozessorarchitektur hat somit einen anderen Assemblerbefehlssatz.



Assembler Beispielprogramm

```
#include <mega8535.h>
```

```
void main()  
{
```

```
    #asm //assembler starts here
```

```
        LDI R30, 0x00        ;R30 wird definiert  
        LDI R31, 0x01        ;R31 wird definiert  
        ADD R30, R31        ;R30 = R30 + R31
```

```
        LDI R28, 0x01       ;R28 wird definiert  
        LDI R29, 0x00       ;R29 wird definiert
```

```
jump1:        ;jump-marke  
        INC R28            ;R28++  
        CPI R28, 0x1E      ;if(R28 == 0x01), Y=001E  
        BRNE jump1        ;else(goto jump1)
```

```
        LD R20, Y          ;R20 = Wert aus Adresse 001E(R30)  
        CPSE R20, R30      ;if(R20 == R30) ueberspringe naechsten befehl (NOP)  
        NOP                ;No Operation, tuhe nichts  
        SBI 0x1B, 0x00     ;PORTA wird auf 0x01 gesetzt
```

```
    #endasm //assembler ends here
```



Speicheraufbau

Register:

- schnellster Speicher
- kleinster Speicher

Datenspeicher:

- langsamer Speicher
- größter Speicher

Programmspeicher:

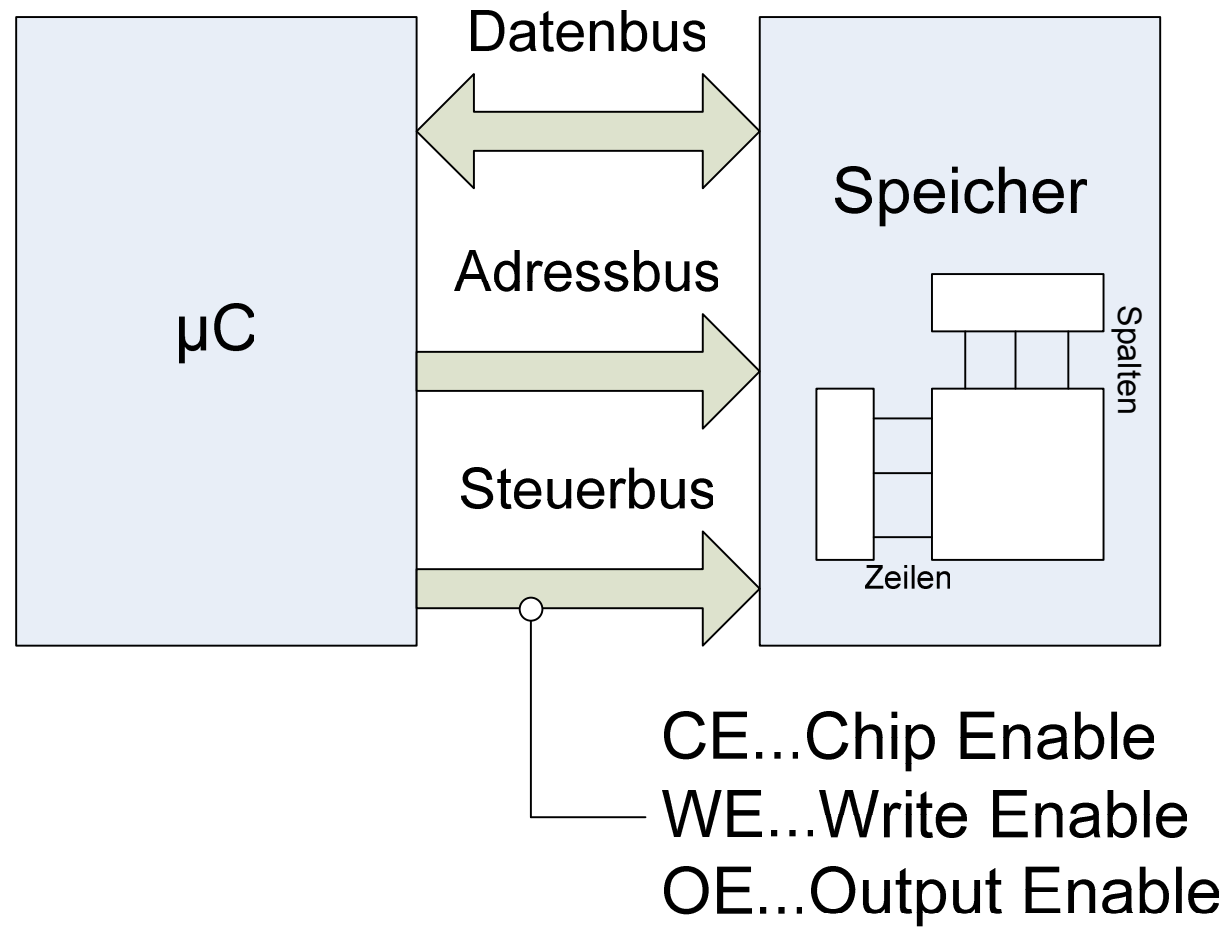
- Speicher für Programmzeilen

Register

Sfr (Special Functions)
I/O - Bereich

Datenspeicher

Bus



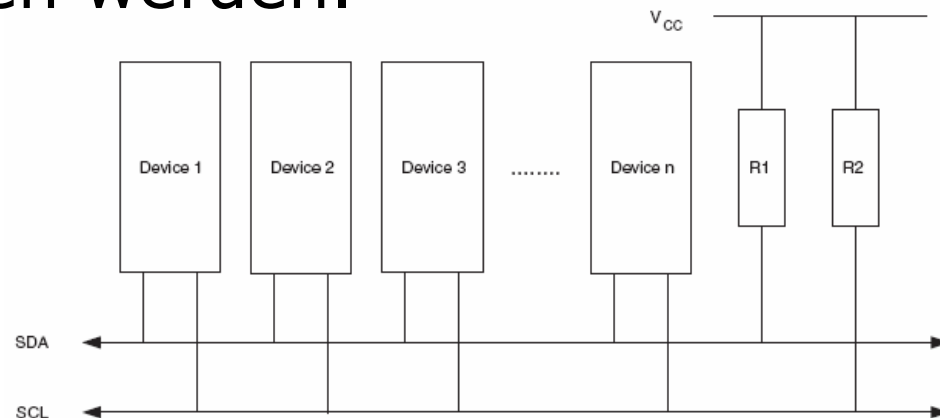


Daten-, Adress- Steuerbus

- Bus: Zusammenfassung von einzelnen digitalen Signalen.
- Datenbus: Überträgt Daten zwischen den Speichern.
- Adressbus: Zum Unterschied zum Datenbus überträgt der Adressbus nur Speicheradressen. Die Busbreite bestimmt wie viel Speicher direkt adressiert werden kann.
 n Adressleitungen = 2^n Speicherstellen
- Steuerbus: Ein Teil des Bussystems. Übernimmt die Steuerung des Bussystems. Zu seinen Aufgaben zählen unter anderem:
Buszugriffssteuerung, Taktung

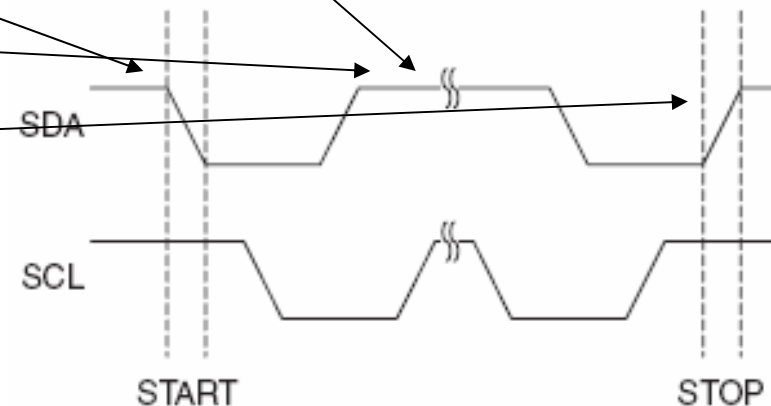
Serielle Schnittstelle

- Ein externer Datenbus für AVR μ Controller wäre beispielsweise das **I²C** System
- An 2 Bidirektionalen Leitungen können bis zu 128 Peripheriegeräte angeschlossen werden
- Die Geräte können wie unten dargestellt angeschlossen werden.



Serielle Schnittstelle TWI

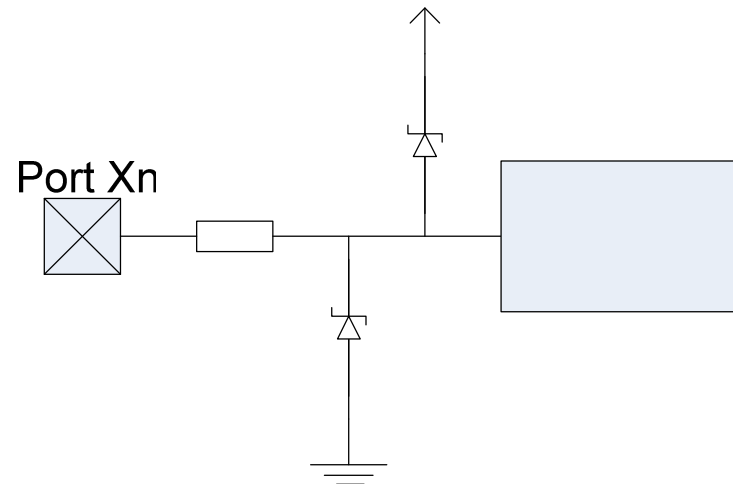
- Sehr ähnlich dem I²C Bus von Phillips
- Die Kommunikation sollte nach folgendem Schema ablaufen:
 - Start
 - Sende Zieladresse (slaveID)
 - Sende Daten
 - Stopp



I/O-Ports

Schutzbeschaltung

- Beim Einschalten des μC sind die Ports auf Ausgang geschaltet um sie gegen eventuell anliegende Spannungen zu schützen.
- Die Ports des μC sind mit Schottky-Dioden Richtung Versorgung und Masse gesichert.





I/O-Ports

Digitale Ansteuerung eines Port Pins

○ Output:

- C: PORTA = 0x00;
- Ass: SBI 0x1B, 0x00

○ Input:

- C: PINA
- Ass: 0x1B

○ Wählen der Datenrichtung (Data Direction)

- C: DDRA = 0x00;
- Ass: SBI 0x1A, 0x00

0x1B (0x3B)	PORTA
0x1A (0x3A)	DDRA
0x19 (0x39)	PINA
0x18 (0x38)	PORTB
0x17 (0x37)	DDRB
0x16 (0x36)	PINB
0x15 (0x35)	PORTC
0x14 (0x34)	DDRC
0x13 (0x33)	PINC
0x12 (0x32)	PORTD
0x11 (0x31)	DDRD
0x10 (0x30)	PIND



Timer/Counter

Timer als Zähler:

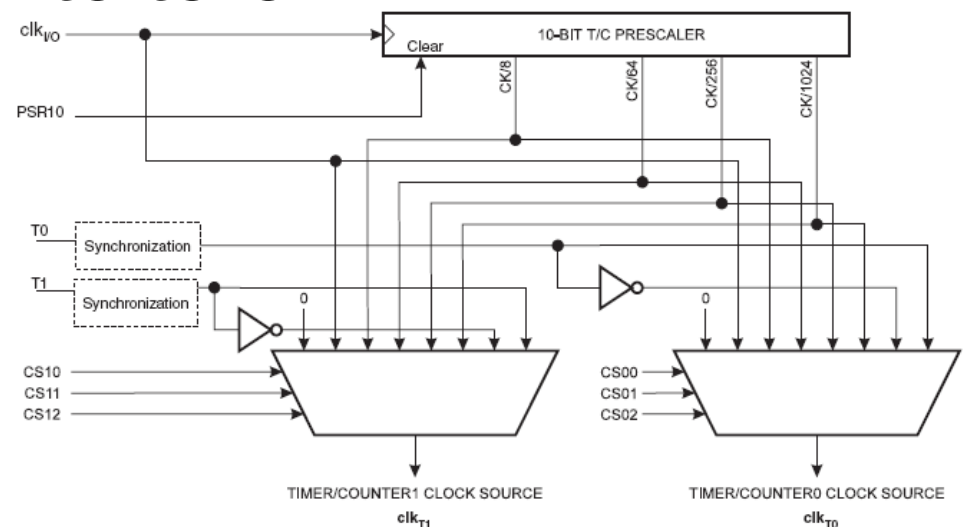
- Zählt mit bestimmter Frequenz bis zu einem bestimmten Wert unabhängig von den restlichen Vorgängen im μC

Timer als Counter:

- Wenn an bestimmten Ports eine Flanke anliegt wird das Counter Register erhöht
- Dies geschieht ebenfalls unabhängig von allen anderen Vorgängen innerhalb des μC .

Timer/Counter Prescaler

- Teilt die Frequenz für den Zähler
- Über das Register TCNTx lässt sich einstellen wie viel mal die Frequenz geteilt wird
- Kann die Frequenz auf $f/8$, $f/64$, $f/256$ oder $f/1024$ herunterteilen.





Timer/Counter

Beispielprogramm Timer als Zähler

Timer Einstellungen:

```
// Timer/Counter 1 initialization
// Clock source: System Clock – Timer Frequenz = Quarzfrequenz.
// Clock value: 4000,000 kHz – 4MHz
// Mode: CTC top=OCR1A – Wenn Timer OCR1A entspricht soll er
// stoppen und rückgesetzt werden.
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x0F;
OCR1AL=0xA0;
OCR1BH=0x00;
OCR1BL=0x00;
```



Timer/Counter

Beispielprogramm Timer als Zähler

Interrupt:

// Timer 1 output compare A interrupt service routine – Wird ausgelöst wenn Timer1 OCR1A entspricht

```
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
    static int i;    //statische Variable (enthaltener Wert wird
                    //gespeichert)

    i++;
    if(i==100)
    {
        PORTC = ~PORTC;    //PORTC = PORTC invertiert
        i=0;
    }
}
```




Timer/Counter

Beispielprogramm Timer als Counter

```
void init(void);
int siebenseg(int dual);

void main(void)
{
    init();

    while (1)
    {
        int x;
        x = TCNT0;

        PORTC = siebenseg(x);
    };
}
```



Timer/Counter

Beispielprogramm Timer als Counter

```
int siebenseg(int dual)
{
// 2
// 3 1
// 4   belegung des 7segments (bei stk500)
// 5 7
// 6 8

int i;
unsigned char vgl[10]={0x01,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,0x11};
//werte werden fix ins array geschrieben
unsigned char re[10]={0x77,0x41,0x3B,0x6B,0x4D,0x6E,0x7E,0x43,0x7F,0x6F};

    for(i=0;i<=10;i++)
    {
        if(dual==vgl[i]) //wenn die übergebene variable mit dem
                        //aktuellen feld übereinstimmt dann...
        {
            return re[i]; //...soll der aktuelle wert des anderen
                        //feldes als funktionswert zurückgegeben
                        //werden
        }
    }
}
```



Vielen Dank für ihre
Aufmerksamkeit!
