Hochschule Bochum
Bochum University
of Applied Sciences

Grundlagen Rechnertechnik

Prof. Dr. Peter Gerwinski

11. Dezember 2012

5 Hardwarenahe Programmierung

5.1 Bit-Operationen

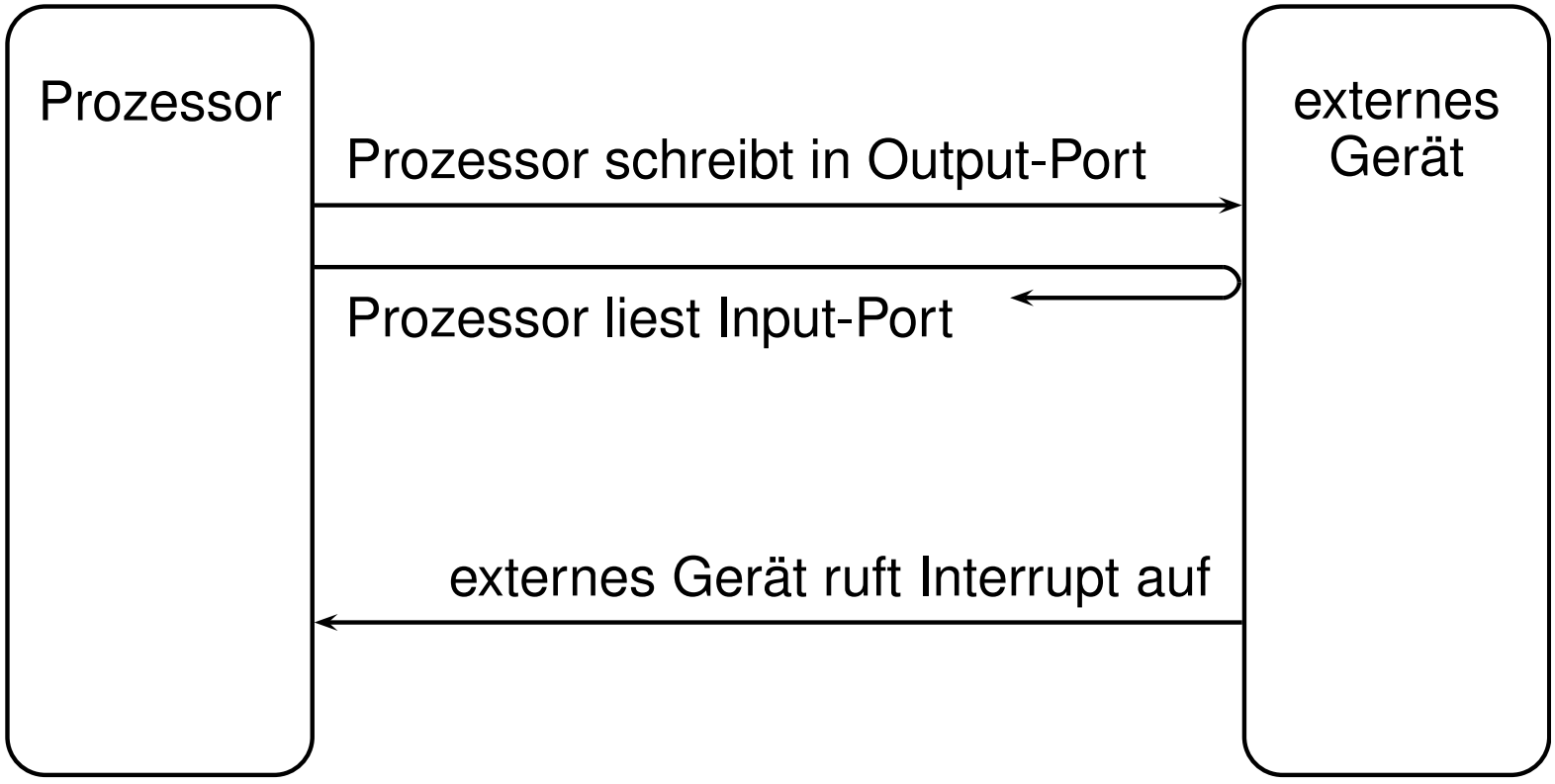
C-Operator	Verknüpfung	Anwendung
&	Und	Bits gezielt löschen
|	Oder	Bits gezielt setzen
^	Exklusiv-Oder	Bits gezielt invertieren
~	Nicht	Alle Bits invertieren
<<	Verschiebung nach links	Maske generieren
>>	Verschiebung nach rechts	Bits isolieren

Aufgabe: Schreiben Sie C-Funktionen, die ein „Array von Bits“ realisieren, z. B.

void set_bit (**int** i);	Bei Index *i* auf 1 setzen
void clear_bit (**int** i);	Bei Index *i* auf 0 setzen
int get_bit (**int** i);	Bei Index *i* lesen

5 Hardwarenahe Programmierung

Kommunikation mit externen Geräten



5.2 I/O-Ports

In Output-Port schreiben = Leitungen ansteuern

Datei: RP6Base/RP6Base_Examples/RP6Examples_20080915/RP6Lib/RP6base/RP6RobotBaseLib.c
Suchbegriff: setMotorDir

```
void setMotorDir(uint8_t left_dir, uint8_t right_dir)
{
  /* ... */
  if(left_dir)
    PORTC |= DIR_L;
  else
    PORTC &= ~DIR_L;
  if(right_dir)
    PORTC |= DIR_R;
  else
    PORTC &= ~DIR_R;
}
```

Manipulation einzelner Bits

Output-Port

5.2 I/O-Ports

In Output-Port schreiben = Leitungen ansteuern

Datei: RP6Base/RP6Base_Examples/RP6Examples_20080915/RP6Lib/RP6base/RP6RobotBaseLib.c
Suchbegriff: setMotorDir

```
void setMotorDir(uint8_t left_dir, uint8_t right_dir)
{
  /* ... */
  if(left_dir)
    PORTC |= DIR_L;
  else
    PORTC &= ~DIR_L;
  if(right_dir)
    PORTC |= DIR_R;
  else
    PORTC &= ~DIR_R;
}
```

Manipulation einzelner Bits

⟶ Steuerung der Motordrehrichtung

Output-Port

5.3 Interrupts

Externes Gerät ruft (per Stromsignal) Unterprogramm auf
Zeiger hinterlegen: „Interrupt-Vektor“

Datei: RP6Base/RP6Base_Examples/RP6Examples_20080915/RP6Lib/RP6base/RP6RobotBaseLib.c
Suchbegriff: ISR

„Dies ist ein Interrupt-Handler.“

Interrupt-Vektor 0 darauf zeigen lassen

Schreibweise herstellerspezifisch!

```
ISR (INT0_vect)
{
  mleft_dist++;
  mleft_counter++;
  /* ... */
}
```

5.3 Interrupts

Externes Gerät ruft (per Stromsignal) Unterprogramm auf
Zeiger hinterlegen: „Interrupt-Vektor“

Datei: RP6Base/RP6Base_Examples/RP6Examples_20080915/RP6Lib/RP6base/RP6RobotBaseLib.c
Suchbegriff: ISR

„Dies ist ein Interrupt-Handler.“

Interrupt-Vektor 0 darauf zeigen lassen

Schreibweise herstellerspezifisch!

```
ISR (INT0_vect)
{
  mleft_dist++;
  mleft_counter++;
  /* ... */
}
```

Aufruf durch Sensor an Encoder-Scheibe
$\longrightarrow$ Entfernungsmessung

5.4 volatile-Variable

```
volatile uint16_t mleft_counter;

/* ... */

volatile uint16_t mleft_dist;

/* ... */

ISR (INT0_vect)
{
   mleft_dist++;
   mleft_counter++;
   /* ... */
}
```

5.4 volatile-Variable

```
volatile uint16_t mleft_counter;

/* ... */

volatile uint16_t mleft_dist;

/* ... */

ISR (INT0_vect)
{
   mleft_dist++;
   mleft_counter++;
   /* ... */
}
```

„Immer lesen und schreiben. Nicht wegoptimieren.“

5.4 volatile-Variable

```
volatile uint16_t mleft_counter;

/* ... */

volatile uint16_t mleft_dist;

/* ... */
```

„Immer lesen und schreiben. Nicht wegoptimieren.“

```
ISR (INT0_vect)
{
   mleft_dist++;
   mleft_counter++;
   /* ... */
}
```

```
int main (void)
{
   int prev_mleft_dist = mleft_dist;
   while (mleft_dist == prev_mleft_dist)
      /* just wait */;
}
```