

1 Einführung

- Projektaufgabe:
Eingebettetes System eigener Wahl zum Laufen bringen
- Projektaufgabe überlegen

Beispiele:

- Motorsteuerung für Rasenmähroboter
- Differential GPS
- 3d-Indoor-Positioning-System für Quadrocopter
- Verbesserungen an Flugzeugkabinensimulatoren
- **Sampler-Grabber für MIDI**
- **„Tastsinn“ für Hexapoden**
- ...

Aufgaben

Aufgabe 1: Sieb des Eratosthenes
möglichst viele Primzahlen berechnen

Aufgabe 2: Kalender
Jahreszahl eingeben,
Kalender mit Feiertagen ausgeben
Hinweis: „Osterformel“

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char a[] = "Test";
```

Zeiger:

```
char *p = "Test";
```

- In beiden Fällen wird ein Array von ganzen Zahlen (5 **char**-Variable mit den Werten 84, 101, 115, 116 und 0) im Speicher angelegt.
- Links heißt das Array **a**; rechts ist es „anonym“.
- Rechts wird zusätzlich ein Zeiger **p** im Speicher angelegt, der auf das (anonyme) Array zeigt.
- **&a** ist dasselbe wie **a**, nämlich die Adresse des Arrays.
- **&p** ist die Adresse des Zeigers.
- **p** ist der Wert des Zeigers, momentan also die Adresse des (anonymen) Arrays.

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char *a[] = { "Dies", "ist", "ein", "Test" };
```

Zeiger:

```
char **p = a;
```

- Array von Zeigern auf **char**-Variable
- Zeiger auf das Array = Zeiger auf Zeiger auf **char**-Variable
- Schleife durch äußeres Array mit **p++** möglich

2 Arrays und Zeiger für Fortgeschrittene

Array:

```
char a[][5] = { "Dies", "ist", "ein", "Test" };
```

Zeiger:

```
char *p = a[0];
```

- zweidimensionales Array von **char**-Variablen
- Zeiger auf Array-Komponente
= Zeiger auf eindimensionales Array
= Zeiger auf **char**-Variable
- Schleife durch äußeres Array mit Zeiger-Arithmetik ~~nicht~~ möglich


nur mit Trick: p += 5

2 Arrays und Zeiger für Fortgeschrittene

```
typedef char string5[5];  
string5 a[] = { "Dies", "ist", "ein", "Test" };  
string5 *p = a;
```

- Array von Array von **char**-Variablen
= zweidimensionales Array von **char**-Variablen
- Zeiger auf zweidimensionales Array
- Schleife durch äußeres Array mit **p++** möglich

→ Fazit:
Ein Hoch auf **typedef**!

2 Arrays und Zeiger für Fortgeschrittene

```
typedef char string5[5];  
string5 *p = { "Dies", "ist", "ein", "Test" };
```

- ~~• anonymes Array von Array von **char**-Variablen
= anonymes zweidimensionales Array von **char**-Variablen~~
- ~~• Zeiger auf zweidimensionales Array~~
- ~~• Schleife durch äußeres Array mit **p++** möglich~~

Das Konstrukt { "Dies", "ist", "ein", "Test" }
steht für ein Array von 4 Zeigern auf **char**-Variable.

string5 *p hingegen erwartet einen Zeiger auf ein Array von 5 **char**-Variablen.

Es bekommt die Adresse von "Dies" zugewiesen.

Durch das Erhöhen von **p** (um 5) zeigt es danach *zufällig* auf das "ist".
Bei nochmaligem Erhöhen zeigt es auf das "in" von "ein".

(Auch ohne Optimierung werden die Strings "ist", "ein" und "Test"
u. U. wegoptimiert.)

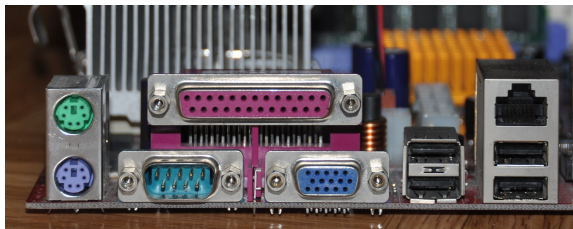
3 Bus-Systeme

Standard-Computer:

- Einsteckkarten: PCI (und Vorgänger)
- Festplatten: SATA (und Vorgänger)
- USB, FireWire, ...
- Ethernet, CAN-Bus, ...
- WLAN, BlueTooth, IR, ...
- PS/2, RS-232, Centronics

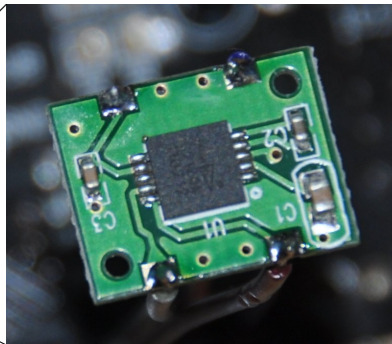
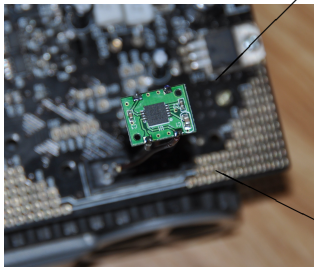
Minimal-Hardware:

- RS-232
- I²C (TWI)
- SPI



3 Bus-Systeme

- I²C: seriell, synchron, mit Adressierung

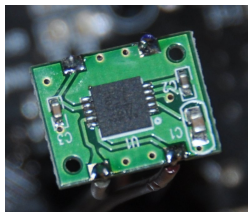


- RS-232: seriell, asynchron, Punkt-zu-Punkt
- RS-485, USB, CAN: seriell, asynchron, mit Adressierung
- SPI: seriell, synchron, Punkt-zu-Punkt oder mit Adressierung

3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

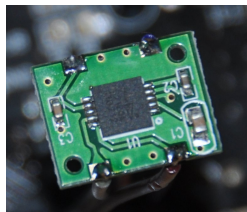


3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```

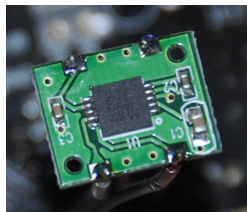


3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{ ... }
```

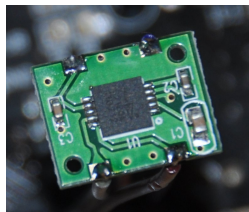
```
int main (void)
{
    uint16_t compass_x, compass_y;
    read_compass (&compass_x, &compass_y);
    writeInteger (compass_x, DEC);
    writeChar ('_');
    writeInteger (compass_y, DEC);
    writeChar ('\n');
    return 0;
}
```



3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

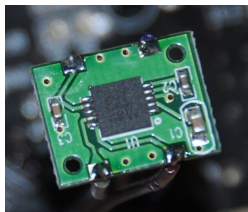


3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

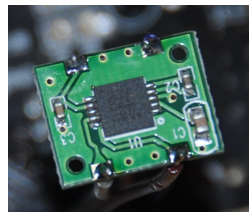
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:
Schreib-Adresse



3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

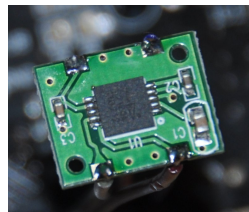


```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:
Schreib-Adresse
Register-Nr.
innerhalb des Geräts

3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

Gerät adressieren:
Schreib-Adresse

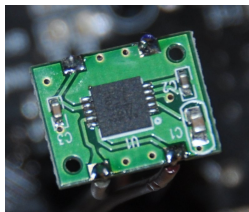
Register-Nr.
innerhalb des Geräts

Befehl: Spule an

3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

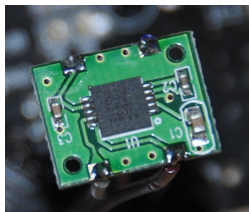
```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



3 Bus-Systeme

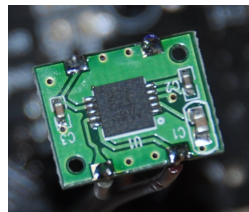
Beispiel: Benutzung des I²C-Busses

```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```



3 Bus-Systeme

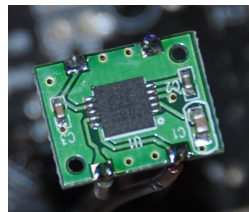
Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses



```
void read_compass (uint16_t *x, uint16_t *y)
{
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x02); // set coil
    mSleep (1);
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x04); // reset coil
    mSleep (5);
    uint8_t result[5];
    I2CTWI_transmit2Bytes (0x60, 0x00, 0x01); // Messung starten
    mSleep (5); // 5ms warten, bis Sensor fertig gemessen hat
    I2CTWI_transmitByte (0x60, 0x01); // Leseindex setzen
    I2CTWI_readBytes (0x61, result, 4); // lesen: msb x, lsb x, msb y, lsb y
    result[0] &= 0b00001111; // Unwichtige Bits vom msb abschneiden
    result[2] &= 0b00001111;
    *x = (result[0] << 8) + result[1]; // Wert berechnen aus msb und lsb
    *y = (result[2] << 8) + result[3];
}
```

3 Bus-Systeme

Beispiel: Benutzung des I²C-Busses

Messung:

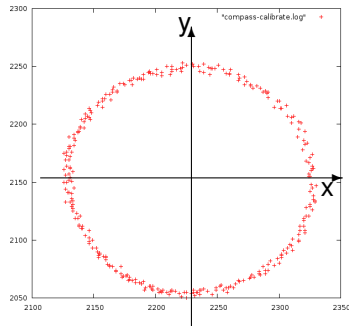
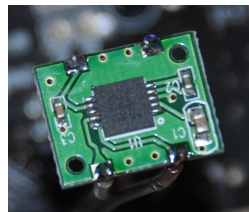
Roboter im Kreis drehen lassen,
x- und y-Werte aufzeichnen
und gegeneinander auftragen

Ergebnis: Ellipse

Anwendung:

Roboter in eine Himmelsrichtung ausrichten

- Achsen normieren:
Mittelwert subtrahieren
- Nulldurchgang einer Achse
markiert eine Himmelsrichtung
- Vorzeichen der anderen Achse
sagt aus, welche Himmelsrichtung



Beispiel: nach Norden ausrichten
Grob drehen, bis y positiv ist, fein drehen, bis $x = 0$ ist

3 Bus-Systeme

Array:

```
char a[] = "Test";
```

Zeiger:

```
char *p = "Test";
```

- In beiden Fällen wird ein Array von ganzen Zahlen (5 **char**-Variable mit den Werten 84, 101, 115, 116 und 0) im Speicher angelegt.
- Links heißt das Array **a**; rechts ist es „anonym“.
- Rechts wird zusätzlich ein Zeiger **p** im Speicher angelegt, der auf das (anonyme) Array zeigt.
- **&a** ist **fast** dasselbe wie **a**, nämlich die Adresse des Arrays **bzw. das Array selbst**, **das zuweisungskompatibel zu einem Zeiger auf Elemente des Arrays ist. & bewirkt hier eine (nicht explizite!) Typumwandlung.**
- **&p** ist die Adresse des Zeigers.
- **p** ist der Wert des Zeigers, momentan also die Adresse des (anonymen) Arrays.