Übung 08: UART, SPI, I2C

Hinweis: In den Vorjahren musste jeder Studierende mit dem Mikrocontroller seines Nachbars aus der Übung per SPI kommunizieren. Das geht dieses Jahr nicht. Bei der Aufgabe 2 genügt es deshalb, wenn Sie sich das in der Theorie überlegen, eine gute Übung für die Klausur. Wer mit einem Kommilitonen zusammenarbeiten kann, umso besser. Dann können Sie die Aufgabe 2 auch praktisch umsetzen.

Hardware:

- Basis: Arduino,
- *Optional für Aufgabe 2*: Zweiter Arduino

Informationen:

- Datenblatt ATmega2560: Kapitel 21 (SPI), 22 (UART), 24
- Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560
- Interrupts: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

Aufgabe 1: UART per AVR-Libc

Bislang erfolgte der Zugriff auf die Konsole immer über die Bibliothek `Serial`. Physikalisch ist dieser Zugriff aus folgendem Grund möglich: Der `ATmega2560` hat intern 4 `U(S)ART`-Einheiten. Dabei wird das ***`UART0`***-Ausgangssignal des ATmega2560 durch einen weiteren `ATmega16U2`[1] auf dem Arduino Board in ein USB-Signal umgewandelt und als COM Port im PC verfügbar.

Wir verwenden im folgenden `U(S)ART0`, da dieser als COM Port im PC verfügbar ist. Wir nutzen den ***Asynchronous Normal Mode***, der dem klassischen UART entspricht.

a) Welcher Wert muss im `UBRRn` Register gesetzt werden, damit sich eine Baudrate von 9600 Baud bei einem Systemtakt von $f_{osc} = 16\,MHz$ ergibt?
Hinweis: Datenblatt Kapitel 22.3.1, S. 202/203, Abschnitt 22.3.1 und Abbildung 22-2.

b) Schreiben Sie ein Programm, das **ohne** die Bibliothek `Serial` den String „Hallo“ ausgibt:

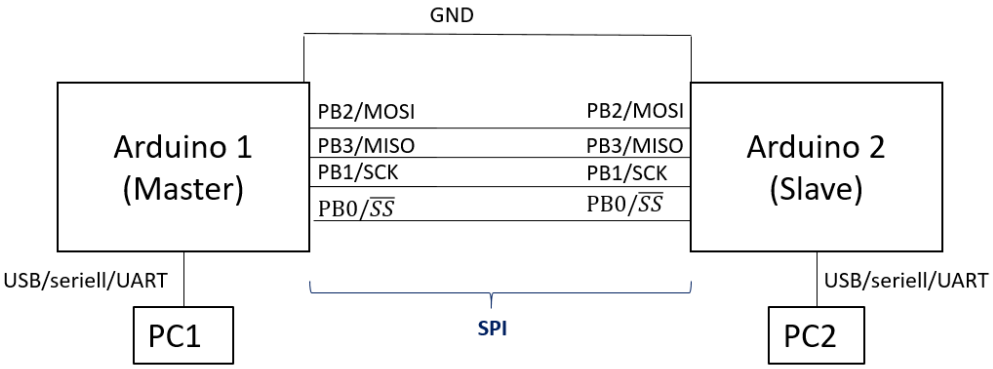
- Ergänzen Sie rechts die Funktion `setup()` um die Initialisierung der UART-Schnittstelle. → Datenblatt S. 218ff
  - `UBRRn`: Baudrate konfigurieren
  - `UCSRnB`: Transmitter aktivieren
  - `UCSRnC`: Frame mit 8 Datenbits und 1 Stoppbit .
- Implementieren Sie die Methode **`void`** `uart_putchar(`**`char`**` c)`. Diese soll 1 Zeichen / Byte senden. *Tipps*:
  - Abfrage ob Übertragungspuffer leer, siehe Codebeispiel, S. 207
  - `UDRn` Register mit zu sendenden Daten befüllen.

```
void setup() {
   // TODO
}

void uart_putchar(char c) {
  // TODO
}

void loop() {
  char text[] = "Hallo";
  for (int i = 0; i < sizeof(text);i++)
  {
      uart_putchar(text[i]);
  }
  delay(2000);
}
```

Aufgabe 2: SPI

Das Ziel ist, dass 2 Mikrocontroller sich gegenseitig per SPI Nachrichten senden auf der Konsole anzeigen. Beide Arduinos werden über USB (von einem PC) mit Strom versorgt. Ggfs. nur auf dem Papier lösen, siehe Anmerkungen oben!

[1] Siehe Stromlaufplan: https://www.arduino.cc/en/uploads/Main/arduino-mega2560-schematic.pdf



a) Welche Buchsen des Arduino Mega Boards haben die Funktionalität `MOSI`, `MISO`, `SCK` und `SS`?

b) Bauen Sie die Schaltung nach! [*praktische Umsetzung optional*]

c) Schreiben Sie ein Programm, das von Master zu Slave und ***gleichzeitig*** in der Gegenrichtung (Vollduplex) Strings sendet. Ergänzen Sie hierzu ***jeweils*** für Master und Slave die Funktion `spi_transceive(.)` und `setup(.)` [*praktische Umsetzung optional*]

Hinweise:

- Master Init: `MOSI` und `SCK` im `DDR`-Register als Ausgang (S. 96ff), SPI aktivieren (S. 197), Master Mode konfigurieren (S.197) aktivieren und beliebigen Takt konfigurieren (S. 198).
- Slave Init: `MISO` im `DDR` Register als Ausgang, SPI aktivieren.
- Senden und Empfangen: S. 191 (2. und 3. Absatz). Die Codebeispiele auf S. 193 und 194 sind hilfreich, müssen aber abgewandelt werden, um Vollduplex zu erreichen.

```
// MASTER
void setup()  {
  Serial.begin(9600);

  //TODO
}

 // send and receive data
char spi_transceive(char data) {

  //TODO
}


void loop() {
  char text[] = "Hallo Slave!";

  for (int i = 0; i < sizeof(text);i++){
     char received =
         spi_transceive(text[i]);
     Serial.print(received);
  }

  delay(1000);
}
```

```
// SLAVE
void setup() {
   Serial.begin(9600);

   //TODO
}

 // send and receive data
char spi_transceive(char data) {

   //TODO
}


void loop() {
  char text[] = "Hallo Master";

  for (int i = 0; i < sizeof(text);i++){
     char received =
             spi_transceive(text[i]);
     Serial.print(received);
  }

  delay(1000);
}
```

Aufgabe 3: SPI vs. I2C

Vergleichen Sie SPI und I2C bzgl. der folgenden Kriterien:

- Wieviel Prozent der Bandbreite werden für Nutzdaten genutzt, wieviel ist Overhead?
- Anzahl der benötigten Verbindungsleitungen/kabel
- Vollduplex?
- Quittierung für empfangene Daten