# SPiC-Aufgabe #3: Kommunikationsmodul

## (15 Punkte, in Zweier-Gruppen)

Das SPiCboard soll um ein Kommunikationsmodul erweitert werden, damit es in der Lage ist, mit einer weiteren Plattform (hier: einem anderen SPiCboard) über eine serielle Schnittstelle Daten auszutauschen. Dabei soll `PD1` zum Senden (`TX`) und `PD0` für den Empfang (`RX`) verwendet werden – die zugehörigen Anschluss-Pins befinden sich rechts oben beim SPiCboard.

Halten Sie sich bei Ihrer Implementierung genau an die in `com.h` vorgegebene Schnittstelle und implementieren Sie die Schnittstelle in einer Datei `com.c`.

## Teilaufgabe a: Initialisierung (5 Punkte)

Implementieren Sie die zwei modulinternen Hilfsfunktionen `sb_com_getRx()` und `sb_com_setTx()`, die jeweils den Wert des `RX` Pins auslesen bzw. den `TX` Pin auf logisch `0` oder `1` setzen.

- `uint8_t sb_com_getRx()`: Gibt `0` zurück, wenn an `RX` ein `low` Pegel anliegt und und `1`, wenn ein `high` Pegel anliegt
- `void sb_com_setTx(uint8_t bit)`: Setzt den `TX` Pin auf einen `high` Pegel, wenn eine `1` übergeben wird und auf einen `low` Pegel, wenn eine `0` übergeben wurde

Implementieren sie außerdem die modulinterne Initialisierungsfunktion `sb_com_init()`, die beim Aufruf der ersten Schnittstellenfunktion die Initialisierung der Hardware sicherstellt:

- `RX`-Pin (`PD0`) als Eingang konfigurieren und (integrierten) Pull-up Widerstand aktivieren
- `TX`-Pin (`PD1`) als Ausgang konfigurieren
- Sendeleitung `TX` mit Hilfe von `sb_com_setTx()` auf High-Pegel setzen
- Stellen Sie sicher, dass die Hardwareinitialisierung nur bei dem ersten Aufruf von `sb_com_init()` erfolgt.

Achten Sie bei der Implementierung der modulinternen Hilfsfunktionen auf eine korrekte Sichtbarkeit.

## Teilaufgabe b: Sendefunktionalität (5 Punkte)

Implementieren Sie die Funktion `sb_com_sendByte()`. Der Versand eines Bytes soll wie folgt ablaufen:

- Im Leerlauf soll an `PD1` / `TX` ein High-Pegel (logisch `1`) anliegen.
- Ein Startbit (logisch `0`) wird gesendet – welches den Pegel an `PD1` / `TX` nach unten zieht. Die Dauer eines Symbols (hier Bits) wird durch die Baudrate bestimmt, hierfür kann das vorgegebene Makro `sb_com_wait(1)` eingesetzt werden, welches die Zeit $T$, die der Pegel (zur Übertragung des Symbols) an der Leitung anliegen soll, durch aktives Warten überbrückt.
- Anschließend werden die Nutzdaten – ein Byte – bitweise übertragen, angefangen beim geringwertigsten Bit (LSB, also quasi *rückwärts*). Hierfür wird für jedes Bit der Pegel entsprechend gesetzt und die entsprechende Dauer wie beim Startbit gewartet.
- Um Übertragungsfehler erkennen zu können, wird ein gerades Paritätsbit aus den Nutzdaten errechnet. Dieses beträgt logisch `0`, wenn die Anzahl der gesetzten Bits in den Nutzdaten gerade ist und logisch `1`, wenn die Anzahl der gesetzten Bits ungerade ist. Die Übertragung des Paritätsbits erfolgt analog zu den vorherigen Bits.
- Abschließend wird mit dem Stoppbit (logisch `1`) der Pegel wieder nach oben gezogen – die Leerlaufspannung ist somit wiederhergestellt und muss mindestens für die Übertragungsdauer eines Symbols anliegen.

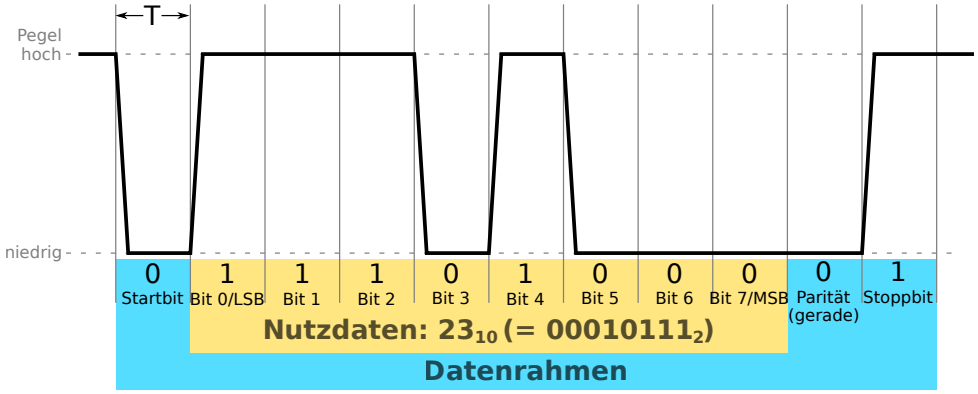


Abbildung 1: Beispiel – Senden eines Bytes mit dem Wert 23

Testen Sie die Funktionalität von `sb_com_sendByte()`:

- Flashen Sie das vorgegebene Binärprogramm `test_receiver.elf` mittels der im selben Ordner liegenden Datei `flash.sh` auf das erste SPiCboard.
- Übersetzen Sie das vorgegebenen Programm `test_sender.c` mit Ihrem `com`-Modul und flashen Sie es auf das zweite SPiCboard.
- Die (periodisch) übertragenen Nutzdaten lassen sich auf Senderseite mit dem Potentiometer einstellen und sollten auf Empfängerseite korrekt angezeigt werden. Tritt ein Empfangsfehler auf, dann wird der zugehörige Fehlercode mit Hilfe der LEDs dargestellt.

## Teilaufgabe c: Empfangsfunktionalität (5 Punkte)

Implementieren Sie nun die Funktion `sb_com_receiveByte()`. Der Empfang eines Bytes soll wie folgt ablaufen:

- Die Empfangsroutine muss gestartet sein, bevor das andere damit verbundene SPiCboard die Senderoutine startet (andernfalls wird ein Datenverlust auftreten)! Während dieser Zeit wird ein High-Pegel als Leerlaufspannung anliegen.
- Zuerst soll überprüft werden, ob der Zeiger für den Empfangspuffer auf einen gültigen Speicherbereich zeigt (d.h. kein Nullpointer ist). Wenn die Überprüfung fehlschlägt, soll der entsprechende Fehlercode zurückgegeben werden.
- Es wird aktiv auf das Startbit (d.h. eine fallende Flanke an `PD0` / `RX`) gewartet. Im Anschluss muss die eineinhalbfache Dauer ($1.5 \times T$) überbrückt werden (entsprechend das Makro aufrufen: `sb_com_wait(1.5)`) – dadurch wird sichergestellt, dass alle nachfolgenden Abfragen bei einem stabilen Pegel (und nicht während einer Pegeländerung) geschehen.
- Die Nutzdaten werden aus den empfangenen Bits (Auslesen des Pegels an `PD0` / `RX`) rekonstruiert. Zwischen den Bits wird jeweils für die Dauer $T$ gewartet, ehe der Pegel erneut abgetastet wird.
- Durch das Vergleichen des empfangenen Paritätsbit mit dem – aus den empfangenen Nutzdaten – errechneten Soll-Paritätsbit kann ein Übertragungsfehler erkannt werden. Tritt ein derartiger Fehler auf wird der Empfang fortgesetzt und der aufgetretene Fehler später über den Rückgabewert an den Aufrufer zurückgegeben.
- Nach dem Paritätsbit sollte ein Stoppbit empfangen werden. Ein fehlendes Stoppbit stellt ebenfalls einen Übertragungsfehler dar, der analog zu einer fehlgeschlagenen Paritätsprüfung behandelt wird.

Testen Sie die Funktionalität von `sb_com_receiveByte()`:

- Flashen Sie das vorgegebene Binärprogramm `test_sender.elf` mittels der im selben Ordner liegenden Datei `flash.sh` auf das erste SPiCboard.
- Übersetzen Sie das vorgegebenen Programm `test_receiver.c` mit Ihrem `com`-Modul und flashen Sie es auf das zweite SPiCboard.
- Die (periodisch) übertragenen Nutzdaten lassen sich auf Senderseite mit dem Potentiometer einstellen und sollten auf Empfängerseite korrekt angezeigt werden. Tritt ein Empfangsfehler auf, dann wird der zugehörige Fehlercode mit Hilfe der LEDs dargestellt.

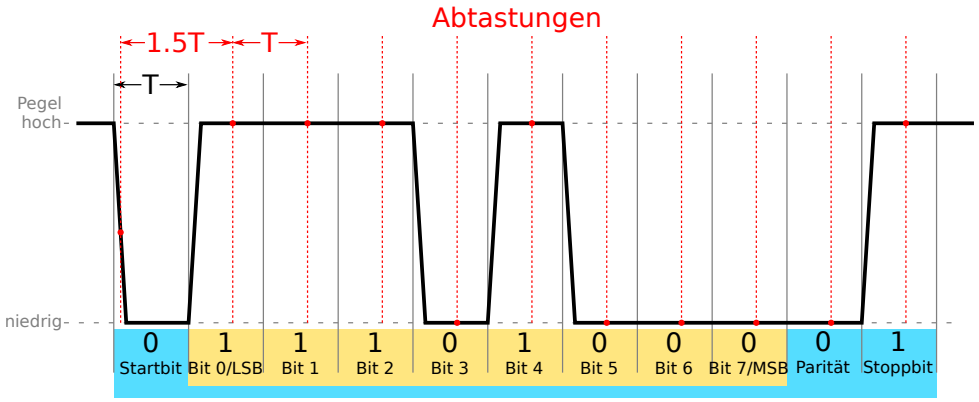


Abbildung 2: Beispiel – Abtastzeitpunkte auf Empfangsseite

### Allgemeine Hinweise

- Die Modulschnittstelle (`com.h`) und die Testprogramme befinden sich unter `/proj/i4spic/pub/aufgabe3`.
- Die Modulschnittstelle ist auch auf der Webseite `libspicboard` Dokumentation (*Serial Communication*) zu finden.
  `https://www4.cs.fau.de/Lehre/WS18/V_SPIC/SPiCboard/libapi.shtml`
- Der Fehlerfall `ERROR_BUFFER_FULL` darf für diese Aufgabe ignoriert werden.
- Funktionen oder globale Variablen, die in der Schnittstelle nicht deklariert werden, sind in ihrer Sichtbarkeit auf das Modul zu beschränken.
- Achten Sie darauf, dass Ihr Programm mit Optimierungen kompiliert und funktioniert; die Abgabe wird mit Optimierungen übersetzt und bewertet.
- Testen sie ihr fertiges Modul mit einem eigenen Testprogramm oder dem vorgegebenen Programm `auf100.c`. (In dem Spiel `auf100` kann abwechselnd eine zufällige Zahl um 1 bis 8 erhöht werden, gewonnen hat der Spieler, der zuerst 100 erreicht.)

## Abgabezeitpunkt

T01 19.11.2018 18:00:00