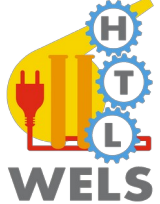


| Arbeitsauftrag | |  |
|-----------------------|---|---|
| Fach: | AIIT 4JG Höhere Elektrotechnik | |
| Thema: | UART Kommunikation | |
| Lernziel: | Daten an PC senden (Anzeige mit Terminal Emulator) Daten von PC empfangen Verwendung von Strings in C | |
| Zeitdauer: | 3 x 2 Einheiten | |

UART

Die UART Schnittstelle ist eine weit verbreitete Kommunikationsschnittstelle, die unter anderem auch für die Datenübertragung zwischen PC und μ C verwendet werden kann.

Die Vorlage beinhaltet Funktionen um Daten mit der UART senden zu können:

- `static void uartSendByte(uint8_t byte);`
- `static void uartSendData(uint8_t *data, unsigned int size);`
- `static void uartSendString(char *buffer);`

Der Interrupt für empfangene Daten ist bereits aktiviert und die ISR implementiert:

```
void USART6_IRQHandler(void) {
    if(USART_GetITStatus(USART6, USART_IT_RXNE)) {
        USART_ClearITPendingBit(USART6, USART_IT_RXNE);
        uint8_t rx = USART_ReceiveData(USART6);
    }
}
```

Strings

Eine Abfolge von Zeichen („Text“) wird in Programmiersprachen als *String* bezeichnet. Ein String in C wird mit doppelten Anführungszeichen (") angegeben, zB:

```
printf("Ergebnis: %3i", result);
```

In C wird ein String als Abfolge von Zeichen, also in einem char array gespeichert:

```
char s[] = "Hello";
```

Die Vorlage beinhaltet bereits die Funktion `uartSendString(char *buffer)`. Diese kann für die Übertragung eines Strings an den PC genutzt werden.

Beachte: Der Funktion wird keine Größe des Strings (des char arrays) übergeben! Dies ist nicht notwendig, da jeder String mit dem Wert 0 (Null) abgeschlossen wird. Das folgende Bild zeigt die Speicherbelegung des obigen Strings s:

| | | | | | | | |
|-----|---|---|---|---|---|---|-----|
| ... | H | E | L | L | O | 0 | ... |
|-----|---|---|---|---|---|---|-----|

- ⇒ Ein String benötigt im Speicher ein Byte mehr als die Anzahl der Zeichen.
Die Anzahl der Zeichen in einem String kann mit der Funktion `strlen()` ermittelt werden.

Um Strings dynamisch zu generieren stehen die Funktionen `sprintf()` und `snprintf()` zur Verfügung. Diese Funktionen werden ähnlich wie die bereits bekannte `printf()` Funktion verwendet, erlauben allerdings das Ergebnis in einem String (char array) speichern, anstatt das Resultat am Display auszugeben. Verwendung:

```
char str[32];
snprintf(str, 32, "Ergebnis: %3i", result);
```

Achtung: Die Verwendung der Funktion `snprintf()` ist sicherer, da sie erlaubt die Größe des char arrays anzugeben. Bei der Verwendung von `sprintf()` (ohne Arraygröße) kann es zu einem Pufferüberlauf kommen.

Aufgaben

Die Aufgaben in dieser Übung umfassen:

- ✓ Konfiguration der UART am μ C (Baudrate, Stopbits, Parität, Wortlänge).
- ✓ Kompatible Konfiguration des Terminal Emulators (TeraTerm).
- ✓ Senden der Werte 64 bis 90 an den PC, anschließend den Wert 13 senden.
- ✓ Senden der Werte 40 bis 63, anschließend zweimal den Wert 13 senden.
- ✓ Mit der Funktion `snprintf()` einen String mit einem Zähler erzeugen. Der Zähler soll jede Sekunde erhöht werden. Der String wird an den PC gesendet.
- ✓ Daten die am μ C empfangen werden, sollen am Display angezeigt werden.