SpartanMC Funktionstest für RTI, Timer und UART.

Nach dem Start erscheint auf der Konsole der 1. Test mit folgende Meldung:

```
Timer RTI und UART Test in C

1. Zeit auf der Konsole anzeigen
   Ende bei jeder Eingabe
CTRL+B Stunden mit + stellen
CTRL+C Minuten mit + stellen
CTRL+D Sekunden mit + stellen
CTRL+E Ende Stellen

 Aktuelle Zeit:    21:30:12  Hallo ich bin SpartanMC 18
```

Die Zeit sollte bei funktionierenden RTI, Timer und UART Modulen mit jeder Sekunde hoch zählen und gleichzeitig sollte auch ein hoch zählen auf den LEDs des Boards zu sehen sein. Die Zeichenkette hinter der Zeit muss aller 4 Sekunden wechseln zwischen „**Hallo ich bin SpartanMC 18**“ und „**SpartanMC 18 TU-Dresden InfMR**“. Die Zeit muss sich mit den Kommandos in der Startmeldung stellen lassen. Mit jeder anderen Eingabe außer den Stellkommandos wird der 1. Test verlassen und in den 2. Test gewechselt.

Der 2. Test ist nur für die UART gedacht. Die eingegebenen Zeichen müssen auf der Konsole erscheinen (CR = ENTER und LF = Ctrl+J). Der Test wird durch Eingabe von ESC beendet. Während des Tests wird im Hintergrund die Uhrzeit weiter hochgezählt und auch die Ausgabe auf die LEDs läuft weiter.

```
2. Alle Eingaben werden auf der Konsole angezeigt
   Ende bei ESC

Hallo, das ist nur ein Test!
Alle Eingaben Anzeigen.
```

Der 3. Test ist ein Dauertest zur Datenübertragung mit einer 2. UART, bei der das Tx und Rx Signal gebrückt sind. Dabei arbeitet die 2. UART im Polling. Im Test wird das Empfangene Zeichen immer mit dem gesendeten verglichen. Auf der Kosole wird immer in der gleichen Zeile das aktuell gesendete Byte, das empfangene Byte und die Zeit ausgegeben.

```
3. Fortlaufend Zeichen an UART 2 im Polling senden und empfangen
```

```
   Ende bei jeder Taste

  Tx Rx   Zeit
  6B 63   21:31:14
```

Kommt es zu einem Fehler bei der Datenübertragung an UART2, dann wird hinter dem empfangenen Byte noch der Sollwerte angezeigt und auf eine neue Zeile gesprungen wie in den folgenden Zeilen zu sehen ist:

```
 Tx  Rx        Zeit
 29  21 22     13:07:48
 2A  22 23     13:07:48
 2B  23 24     13:07:49
```

Bei jeder Eingabe auf der Konsole wird dann der Test verlassen und es folgt ein **4. Test**, bei dem man die Tiefe des installierten Rx-FIFOs von UART2 erkennen kann. Der Test wartet als erstes bis der Tx FIFO leer ist. Dabei wird der Rx FIFO, der mindestens genau so groß sein muss wie der Tx FIFO, gefüllt. Der Rx FIFO wird danach ausgeräumt und es müssen alle Byte von Rx noch angezeigt werden, bis Rx und Tx übereinstimmen. Sind die FIFOs leer, dann wird der Test 5 gleich im Anschluss gestartet.

```
4. Warten bis alle FIFOs leer sind

  Tx Rx  Status
  61 59  02004
  61 5A  00004
  61 5B  00004
  61 5C  00004
  61 5D  00004
  61 5E  00004
  61 5F  00004
  61 60  00004
  61 60  00005
  61 60  02005
  61 61  12004
  61 61  12005

   Fifos sind jetzt leer!
```

Man kann erkennen, das noch 9 Byte empfangen werden. Davon sind 8 im FIFO und eins im Empfangsschieberegister der UART zu dem Zeitpunkt wo der Sendevorgang abgebrochen wurde. Der 5. Test entspricht dem 3. Test, aber die UART wird im Interrupt Mode

betrieben. In diesem Test arbeiten zwei Timer-Interrupts und die beiden UART-Interrupts gleichzeitig. Auch in diesem Test werden die Empfangen Byte ständig mit den gesendeten Byte verglichen und im Fehlerfall angezeigt.

```
5. Fortlaufend Zeichen an UART 2 mit Interrupt senden und empfangen
   Ende bei jeder Taste

  Tx Rx   Zeit
  BF B7   21:31:16
```

Der nächste Test ist eine Einzelzeichen Übertragung mit Interrupts. Wird dieser durch Eingabe von ESC verlassen, dann werden der Test 3 bis 5 mit einem Systemtakt von 256 Hz wiederholt.

```
6. Eingegebenes Zeichen an UART2 senden und empfangen
   Ende bei ESC

  Tx Rx In
  67 67 g
```

Es folgen Test 3 bis 5 mit einem Systemtakt von 256 Hz.

```
7. Fortlaufend Zeichen an UART 2 im Polling senden und empfangen
   Ende bei jeder Taste

  Tx Rx   Zeit
  63 63   21:31:54

8. Warten bis alle FIFOs leer sind

  Tx Rx  Status
  66 66  12005

   Fifos sind jetzt leer!

9. Fortlaufend Zeichen an UART 2 mit Interrupt senden und empfangen
   Ende bei jeder Taste

  Tx Rx   Zeit
  66 66   21:33:00
```

```
10.Eingegebenes Zeichen an UART2 senden und empfangen
   Ende bei ESC

  Tx Rx In
  20 20 a
  61 61 b
```

Zu jeder Zeit kann während der Tests die „Interrupt-Tase“ des Boards betätigt werden. Sie löst einen Interrupt aus, bei dem die Adresse des aktuellem PC und das CC-Bit angezeigt werden solange wie man die Taste drückt.

```
Timer RTI und UART Test in C

1. Zeit auf der Konsole anzeigen
   Ende bei jeder Eingabe
CTRL+B Stunden mit + stellen
CTRL+C Minuten mit + stellen
CTRL+D Sekunden mit + stellen
CTRL+E Ende Stellen

isr_def02 PC= 00345 CC= 0 SpartanMC 18 TU-Dresden InfMR
isr_def02 PC= 00346 CC= 0
isr_def02 PC= 00347 CC= 0
isr_def02 PC= 00348 CC= 0
isr_def02 PC= 00349 CC= 0
isr_def02 PC= 0034A CC= 0
isr_def02 PC= 0034B CC= 0
isr_def02 PC= 0034C CC= 1
isr_def02 PC= 0034D CC= 1
isr_def02 PC= 00351 CC= 1
isr_def02 PC= 00352 CC= 1
isr_def02 PC= 00353 CC= 1
isr_def02 PC= 00354 CC= 1
isr_def02 PC= 00355 CC= 1
isr_def02 PC= 00356 CC= 1
isr_def02 PC= 0035C CC= 1
isr_def02 PC= 0035D CC= 1
isr_def02 PC= 0035E CC= 0
isr_def02 PC= 0035F CC= 0
isr_def02 PC= 00360 CC= 0
isr_def02 PC= 00361 CC= 0
isr_def02 PC= 00362 CC= 0
isr_def02 PC= 00363 CC= 0
isr_def02 PC= 00336 CC= 0
isr_def02 PC= 001C7 CC= 0
isr_def02 PC= 001C8 CC= 0
isr_def02 PC= 001C9 CC= 0
```

Die Adressen mit 2 multipliziert entsprechen den Adressen im Listfile ***.lst** im gleichen Projekt.

```
 68a:     00 03 18 05    seqi r12, 5    ; 0x05
 68c:     00 00 20 15    or   r0, r0
 68e:     00 00 e0 52    bnezc     0x00732   ; 0x732 <L151>
 690:     00 02 48 c0    l18  r4, 0(r6)
 692:     00 02 5c 84    l18  r14, 4(r4)
 694:     00 01 3e 01    movi r15, 0x01 ; 0x1 <SBBOOL_YES>
 696:     00 00 1d ec    sleu r14, r15
 698:     00 00 20 15    or   r0, r0
 69a:     00 00 e0 04    bnezc     0x006a2   ; 0x6a2 <L138>
 69c:     00 03 18 2b    seqi r12, 43   ; 0x2b
 69e:     00 00 20 15    or   r0, r0
 6a0:     00 00 e0 33    bnezc     0x00706   ; 0x706 <L152>

000006a2 <L138>:
 6a2:     00 02 39 a5    s9   5(r13), r12
 6a4:     00 02 5a 86    l18  r13, 6(r4)

000006a6 <L134>:
 6a6:     00 02 58 80    l18  r12, 0(r4)
 6a8:     00 00 3b 98    seq  r13, r12
 6aa:     00 00 20 15    or   r0, r0
 6ac:     00 00 e0 06    bnezc     0x006b8   ; 0x6b8 <L139>
 6ae:     00 00 21 05    jalrs     r8
 6b0:     00 00 20 15    or   r0, r0
 6b2:     00 02 48 c0    l18  r4, 0(r6)
 6b4:     00 02 5e 80    l18  r15, 0(r4)
 6b6:     00 02 7e 86    s18  6(r4), r15

000006b8 <L139>:
 6b8:     00 02 48 82    l18  r4, 2(r4)
 6ba:     00 03 08 01    seqi r4, 1     ; 0x01
 6bc:     00 00 20 15    or   r0, r0
 6be:     00 00 e0 0b    bnezc     0x006d4   ; 0x6d4 <L153>

000006c0 <L140>:
 6c0:     00 02 48 a0    l18  r4, 0(r5)
 6c2:     00 02 18 85    l9   r12, 5(r4)
 6c4:     00 01 99 ff    andi r12, 511  ; 0x1ff
 6c6:     00 00 99 d3    beqz r12, 0x0066c   ; 0x66c <L142>
```

```
0000066c <L142>:
 66c:     00 00 20 e5    jalrs     r7



0000038e <uart_getstat>:
 38e:     00 01 01 fe    addi r0, -2    ; 0x1fe
 390:     00 01 28 00    movi r4, 0x00  ; 0x0
 392:     00 02 28 00    s9   0(r0), r4
```