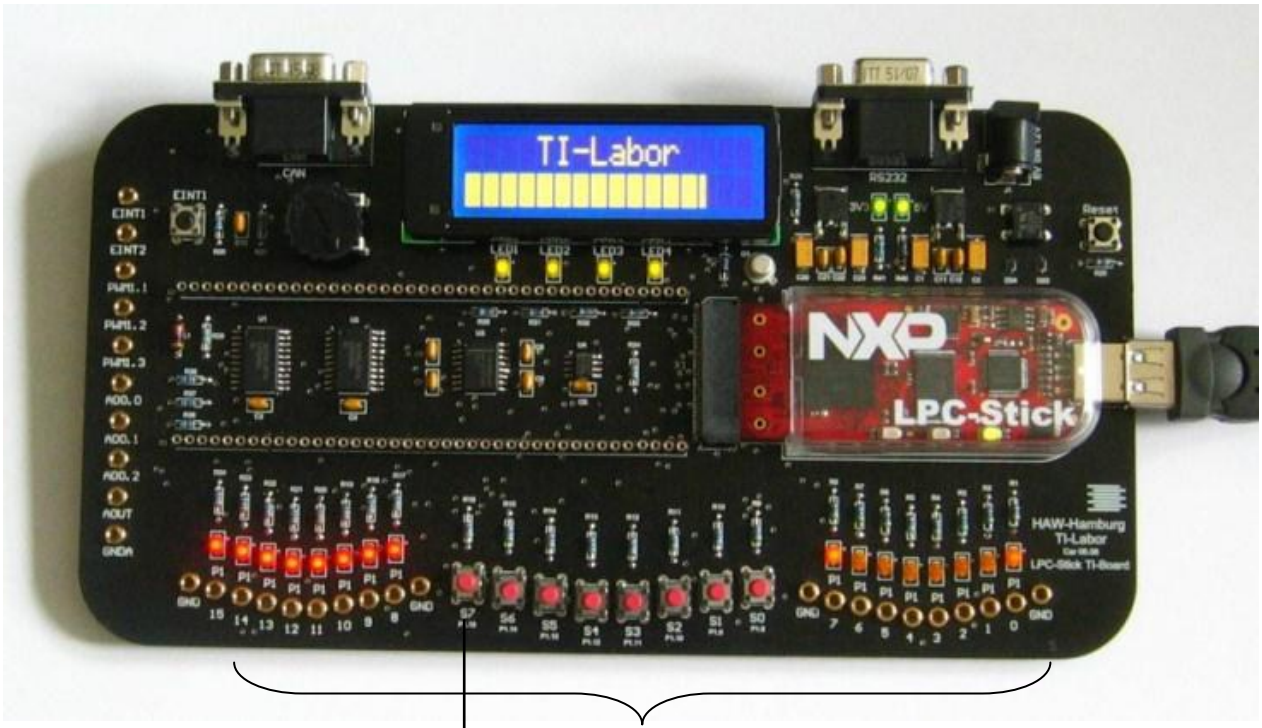
Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Aufgabenstellung:

Mit Hilfe der IO-Ports des ARM-Boards soll ein Reaktionstester aufgebaut werden. Der Reaktionstester besteht aus einer aus 15 Leuchtdioden (LED: Port 1.0 ….. 1.14) bestehenden Zeile sowie einem Taster (S7 = Port 1.15).



Das Programm soll wie folgt arbeiten (siehe Lösungsskizze: letzte Seite):

- Nach dem Start des Programms soll ein Starttext auf („Zum Starten - Taste S7 druecken“) dem LCD-Display angezeigt werden (Zustand: **START_WAIT**).
- Nach Drücken von Taste S7 soll das LCD-Display die Meldung „Achtung !!!“ ausgeben und das Programm soll 3s anhalten (Zustand: **START_DELAY**).
- Danach sollen die 15 LED der Reihe nach und im zeitlichen Abstand von ca. 20ms aufleuchten (Zustand: **RUN**).
- Sobald die Taste gedrückt wird oder die letze LED aufleuchtet (Spieler hat schlechte Reaktion), soll dieser Vorgang für 3s angehalten werden (Zustand: **SHOW_RESULT**). Das LCD-Display soll jetzt „Stopped“ anzeigen.
- Danach sollen die LED erlöschen und das Programm erneut beim Zustand **START_WAIT** beginnen.

Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Realisierungshinweise:

Zur Erhöhung der Übersichtlichkeit soll das Programm in mehrere Unterprogramme unterteilt werden, z.B.

***ConfigurePorts**:*

Setzt Port1.0 …. 1.14 als Ausgang (LED) und Port 1.15 als Eingang (Taste S7).

***TestIfPushButtonPressed**:*

Gibt über r0 eine 1 zurück wenn die Taste gedrückt ist, sonst 0.

***OutputLEDBar**:*

Der Wert in r0 entspricht der anzuzeigenden Balkenlänge.

***LEDBarEndReached**:*

Gibt über r0 eine 1 zurück wenn alle 15 LEDs an sind, sonst 0.

***SafeDelay**:*

Hält den Programmfluß für n * 1/10 Millisekunden (in r0) an und verändert die Register (r1…r4) nicht. Es kann die C-Funktion „delay“ verwendet werden (s. Hinweise).

a) Die realisierten Unterprogramme sollen die Register des aufrufenden Programms nicht ändern (d.h. Sichern und Restaurieren) !

b) Die Parameterübergabe soll über Register (nicht über den Stack) erfolgen.

c) Das zu verwendende Unterprogramm *delay* hält das Programm für eine bestimmte Anzahl von 1/10 ms an (Parameterübergabe über r0). **Aber Achtung**, das Unterprogramm delay (Aufruf: **bl** delay) verändert die Register r1 … r4.

d) Der delay-Timer muss zu Programmbeginn mit init_delay (Aufruf: **bl** init_delay) initialisiert werden.

e) Die u.a. Lösungsskizze (s.u.) ist zunächst so umzuändern, dass sie strukturiert darstellbar ist, d.h. in Form eines Struktogramms (Nassi-Shneiderman-Diagramm) bzw. durch Pseudocode.
   - Schreiben Sie ein entsprechendes Pseudocode-Programm.
   - Realisieren Sie das Assemblerprogramm strukturiert, d.h. mit Strukturierungslabels.

Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Aufgabenbearbeitung:

Die Aufgabe ist spätestens bis zum nächsten Versuchstermin vorzuführen.

- Der funktionierende Reaktionstester ist vorzuführen.
- Der Pseudocode ist zu zeigen und zu erläutern.
- Der Programmcode ist zu erläutern.

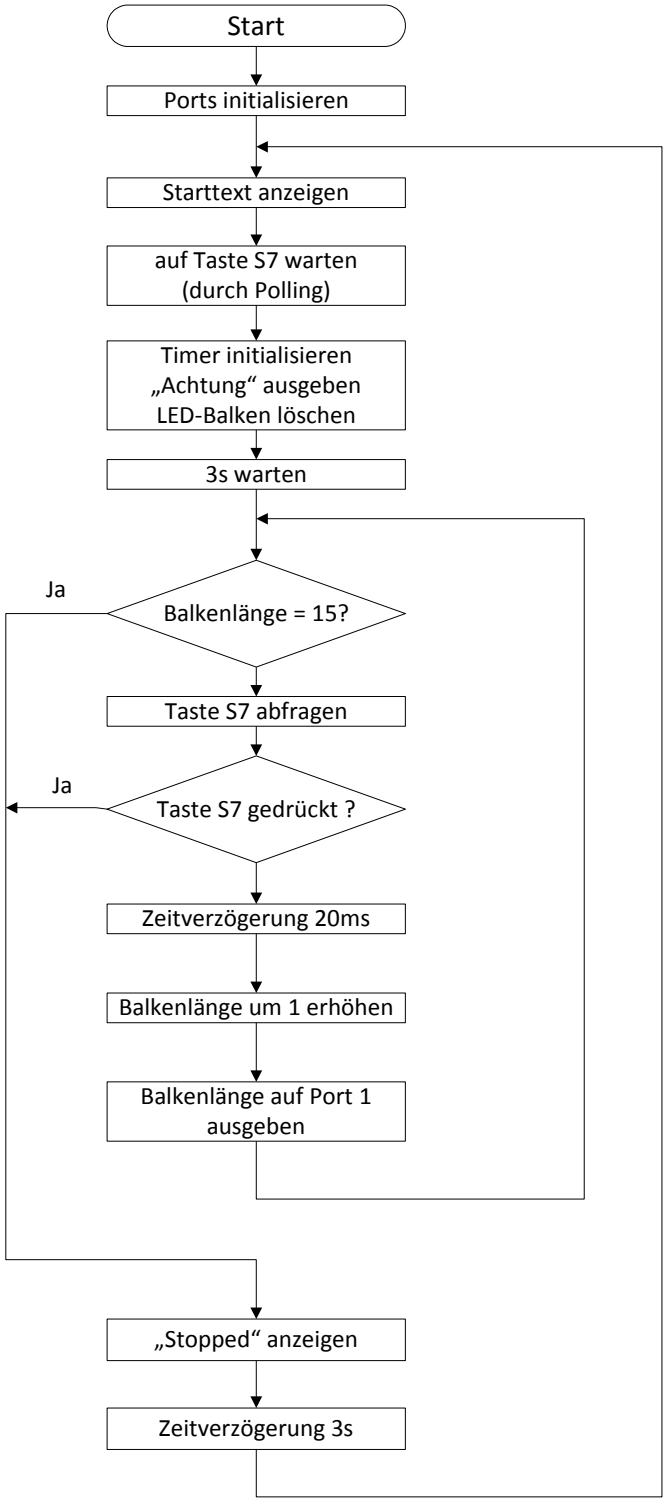
Vorbereitung:

Zu folgenden Themen sollten Sie vorbereitet sein:

* Die Themen von Versuch 1 und 2
* Programmierung der Ports
* Unterprogramme mit Parameterübergabe über Register
* Unterprogramme mit Parameterübergabe über Register
* Call by value, call by reference
* Lösungsskizze (s.u.) ansehen
* Übungsaufgaben ansehen (s.u.)

Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Lösungsskizze "Reaktionstester"



Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Übungsaufgaben:

Gegeben ist folgendes Programmfragment:

```
.section  .data
V1:       .word   15
V2:       .word   368
Tab1:     .word   12, 45, 56, -1

.section  .text
main:     …
          …
          …
          …
loop:
          b loop
#--------------------------------
# Unterprogramme
# --------------------------------
Binom1:   @ Parameterübergabe über Register: r0 ← a, r1 ← b
          @ Rückgabe: r0 ← (a+b)*(a+b)
          …
          …
TabAdd:   @ Parameterübergabe über Reg. r0 ← Adr. der Tabelle,
          @                              r1 ← Elementanzahl
          @ Rückgabe: r0 ← Summe über alle Tabellenwerte
          …
          …
```

1. Das Unterprogramm `Binom1` soll im Hauptprogramm (main) mit den Variablen V1 und V2 aufgerufen werden. Geben Sie den Unterprogrammaufruf (mit Parameterübergabe) an.
2. Das Unterprogramm `Binom1` soll im Hauptprogramm (main) mit den Konstanten 12355 und 12 aufgerufen werden. Geben Sie den Unterprogrammaufruf (mit Parameterübergabe) an.
3. Geben Sie das Unterprogramm `Binom1` an. Alle verwendeten Register und das Linkregister lr sollen gerettet/restauriert werden.
4. Welchen Zweck hat das Linkregister?
5. Das Unterprogramm `TabAdd` soll im Hauptprogramm (main) mit der Tabelle Tab1 und der Elementeanzahl 4 aufgerufen werden. Geben Sie den Unterprogrammaufruf (mit Parameterübergabe) an.
6. Das Unterprogramm `Binom1` soll jetzt so abgeändert werden, dass die Parameterübergabe über den Stack erfolgt. Verwenden Sie den Framepointer fp.
7. Geben Sie den Unterprogrammaufruf (mit Parameterübergabe V1 und V2 über Stack) an. Worauf ist nach der Rückkehr aus dem Unterprogramm zu achten?