Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Übungsaufgaben für die Vorbereitung:

Gegeben ist folgendes Programmfragment:

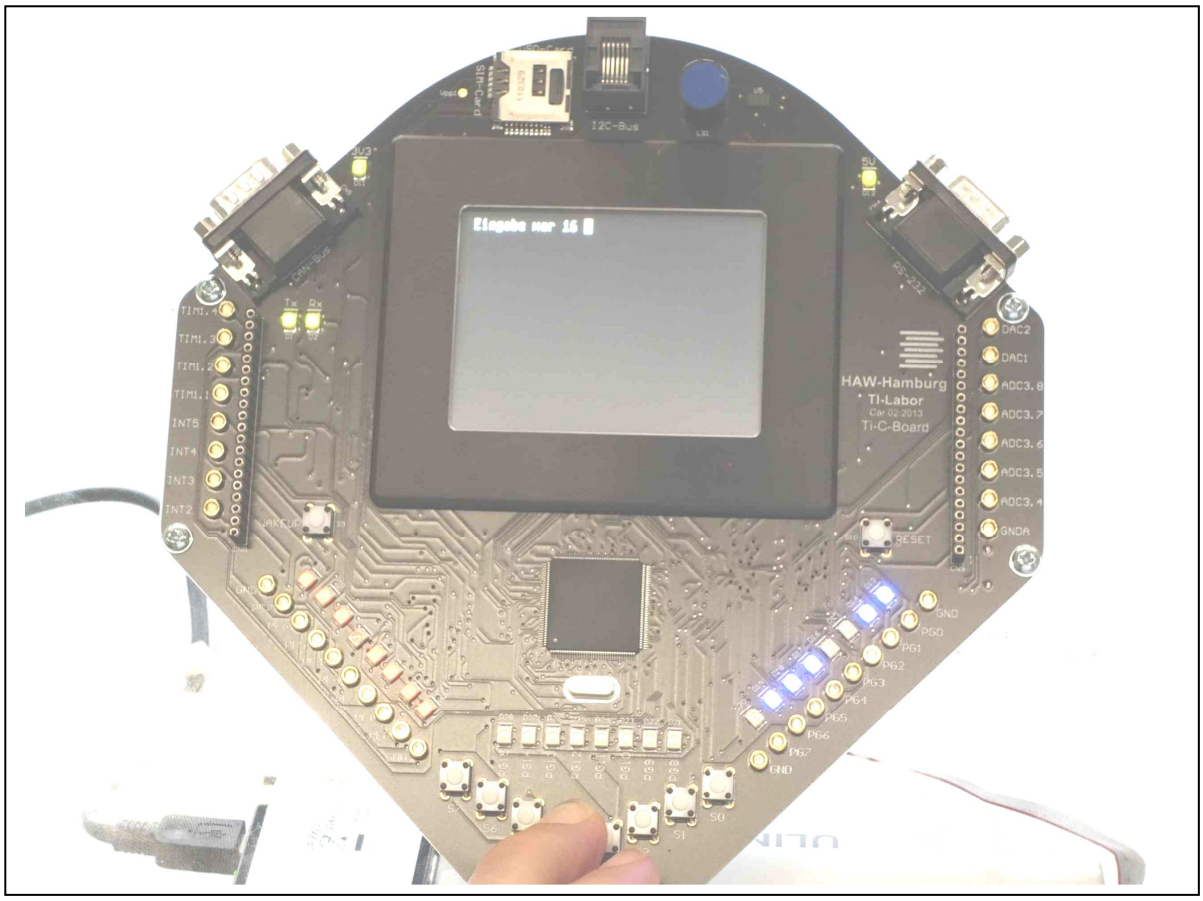
```
    AREA MyData, DATA, align = 2
V1        DCD   15
V2        DCD   368
Tab1      DCD   12, 45, 56, -1


    AREA MyCode, CODE, readonly, align = 2
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?

Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Aufgabenstellung:

Mit den IO-Ports des ARM-Boards soll ein Reaktionstester aufgebaut werden. Der Reaktionstester besteht aus einer 16 Leuchtdioden bestehenden Zeile (LEDs: Port PG0-PG15) sowie einem Taster (S7: Port PE7).



Das Programm soll wie folgt arbeiten (siehe Lösungsskizze S. 5):

- Nach dem Start des Programms soll ein Starttext auf dem TFT-Display angezeigt werden („Zum Starten - Taste S7 druecken“)
(Zustand: **START_WAIT**).
- Nach Drücken von Taste S7 soll das TFT-Display die Meldung „Achtung !!!“ ausgeben und das Programm soll 3s anhalten (Zustand: **START_DELAY**).
- Danach sollen die 16 LEDs der Reihe nach im zeitlichen Abstand von ca. 20ms aufleuchten (Zustand: **RUN**).
- Sobald die Taste gedrückt wird oder die letzte LED aufleuchtet (Spieler hat schlechte Reaktion), soll dieser Vorgang für 3s angehalten werden (Zustand: **SHOW_RESULT**). Das TFT-Display soll jetzt „Stopped“ anzeigen.
- Danach sollen die LEDs 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 Unterprogramme unterteilt werden:

***TestIfPushButtonPressed**:*

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

Achtung: die Taster haben eine invertierte Logik, dh: beim gedrückten Taster liegt am Pin eine 0 an. Achten Sie auf die zum Taster gehörende LED.

***OutputLEDBar**:*

Der Wert in r0 entspricht der anzuzeigenden Balkenlänge.

***LEDBarEndReached**:*

Gibt über r0 eine 1 zurück wenn alle 16 LEDs an sind, sonst 0.

***SafeDelay**:*

Hält den Programmfluß für n 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., alle verwendeten Register einschließlich des Linkregisters müssen gesichert und restauriert werden. Ausnahme: Rückgaberegister, sofern verwendet.

b) Der Stack des Cortex-Prozessors ist aus Performancegründen auf 8 Bytes ‚aligned‘. Es muss deshalb bei push/pop immer eine **gerade** Anzahl von Registern geschrieben/gelesen werden.

c) Die Parameterübergabe soll über Register und nicht über den Stack erfolgen. Ein Framepointer ist daher nicht zwingend notwendig.

d) Das zu verwendende Unterprogramm *Delay* hält das Programm für eine bestimmte Anzahl von ms an (Parameterübergabe über r0).
Achtung: das Unterprogramm Delay (Aufruf: **bl** Delay) verändert die Register r1…r4. Dies gilt auch für die anderen Programme aus der C-Labor-Bibliothek, zB TFT_cls etc.

e) Die Lösungsskizze (s.u.) ist zunächst so zu ändern, dass sie strukturiert darstellbar ist, d.h. durch Pseudocode.

- Erzeugen Sie ein entsprechendes Pseudocode-Programm.
- Realisieren Sie das Assemblerprogramm mit Strukturierungslabels.

f) Das System läuft in der C-Umgebung ab, deshalb müssen Sie das Projekt mit der Batch-Prozedur ‚Keil C' starten und das main.c mit folgenden Zeilen ergänzen:

```
extern void mainASM(void); vor int main(void)
```

`mainASM();` hinter `Init_TI_Board(); in main()`

Weiter müssen Sie in der Gruppe USER ein **neues** file ‚mainASM.s' hinzufügen und den Code aus dem Template in EMIL: ‚mainASM_fuer_A3_reduziert.s' **einfügen.**

Die vorgegebenen Codeschnipsel sind zu einem vollständigen Programm zu ergänzen und so zu ändern, dass anstelle des Ports S0 der Port S7 abgefragt wird.

Aufgabenbearbeitung:

Die Aufgabe ist am Ende des Praktikumstermins vorzuführen, in Ausnahmefällen bis zum nächsten Versuchstermin.

- **Das Programm muss dem Pseudocode entsprechen. Sonst wird die Lösung nicht anerkannt!**
- 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
* Call by value, call by reference
* Lösungsskizze (s.u.) ansehen
* Übungsaufgaben ansehen (s.u.)

Programmierung der IO-Ports
Aufbau eines Reaktionstesters

Lösungsskizze "Reaktionstester" in strukturiere Form mit Labels umsetzen: