Probeklausur TI2 - Grundlagen systemnahes Programmieren

Name	**Matrikelnummer**

Hinweise:

- Verwenden Sie Ihr login.
- Tragen Sie auf jedem Blatt Ihren Namen und Ihre Matrikelnummer ein.
- In jedem Modul, dass Sie erstellen, müssen diese Angaben ebenfalls vorhanden sein.
- Die Klausur besteht aus 6 Aufgaben. Aufgabe 1 und 2 sind „theoretische“ Fragen. In den letzten vier Aufgaben entwickeln Sie schrittweise ein Programm.
- Geben Sie die ersten beiden Aufgaben nach ihrer Bearbeitung ab. Anschließend starten Sie mit dem praktischen Teil.
- Bearbeitungszeit: 180 min.
- Sobald Sie eine der Aufgaben 3 bis 6 gelöst haben, lassen Sie die korrekte Funktionsweise von der Aufsicht abnehmen. Drucken Sie anschließend die Lösung aus, heften und packen Sie diese zur Lösung, die am Schluss abgegeben wird.
- **Erlaubte Hilfsmittel:**
  - 1 Blatt (DIN A4), beidseitig (persönlich) handschriftlich beschrieben, das Blatt muss mit Namen und Matrikelnummer beschriftet sein.
  - Sonst keine weiteren Hilfsmittel (wie zum Beispiel Handy's).
- Es dürfen nur folgende Programme verwendet werden:
  - Keil Entwicklungsumgebung
  - Terminalprogramm
- Insbesondere wird die Benutzung von anderen Programmen, der Zugriff auf das Internet oder die Nutzung Ihres Workspace als Täuschungsversuch gewertet.

Übersicht zur Bewertung der Aufgaben			
Aufgabe	**Punkte**	**Aufgabe**	**Punkte**
1 (100 Punkte)		4 (200 Punkte)	
2 (100 Punkte)		5 (100 Punkte)	
3 (200 Punkte)		6 (300 Punkte)	
Σ			

Aufgabe 1 Fehlersuche (100 Punkte)

(a) Bitte beschreiben Sie den Fehler in folgender syntaktisch korrekten Funktion.[1]

Was bewirkt dieser Fehler?

Wie kann man den Fehler beheben?

```
typedef struct list {
    int elem;  // das Element
    struct list * next;
} LIST, *P_LIST;

P_LIST listAdd(P_LIST l, int elem){
    LIST new_elem ;
    new_elem.next = l;
    new_elem.elem = elem;
    return &new_elem;
}
```

[1] Wenn man von Warnungen des Compilers absieht.

Name Matrikelnummer

(b) Das folgende Programm wird mit einer Linux Entwicklungsumgebung übersetzt.

```
/*
 * This program sleeps for 10 seconds.
 */

#include <stdlib.h>
#define SLEEPTIME       10

int main( int argc, char **argv ){
    sleep(SLEEPTIME);
    return EXIT_SUCCESS;
}
```

Der Compiler liefert folgende Warnung:

warning: implicit declaration of function 'sleep'

Was bedeutet diese Warnung?

Wie können Sie diese Warnung beseitigen?
(Zur Erinnerung: Auf der nächsten Seite steht der Manualeintrag der Funktion sleep).

Manual Eintrag zur Funktion sleep:

Suspend a thread for a given length of time

Synopsis:

```
#include <unistd.h>

unsigned int sleep( unsigned int seconds );
```

Arguments:
Seconds: The number of realtime seconds that you want to suspend the thread for.

Library:
libc

Description:
The sleep() function suspends the calling thread until the number of realtime seconds specified by the seconds argument have elapsed, or the thread receives a signal whose action is either to terminate the process or to call a signal handler. The suspension time may be greater than the requested amount, due to the scheduling of other, higher priority threads by the system.

Returns:
0 if the full time specified was completed; otherwise, the number of seconds unslept if interrupted by a signal.

Errors:
EAGAIN No timer resources were available to satisfy the request.

Name Matrikelnummer

Aufgabe 2 Funktionalität (100 Punkte)

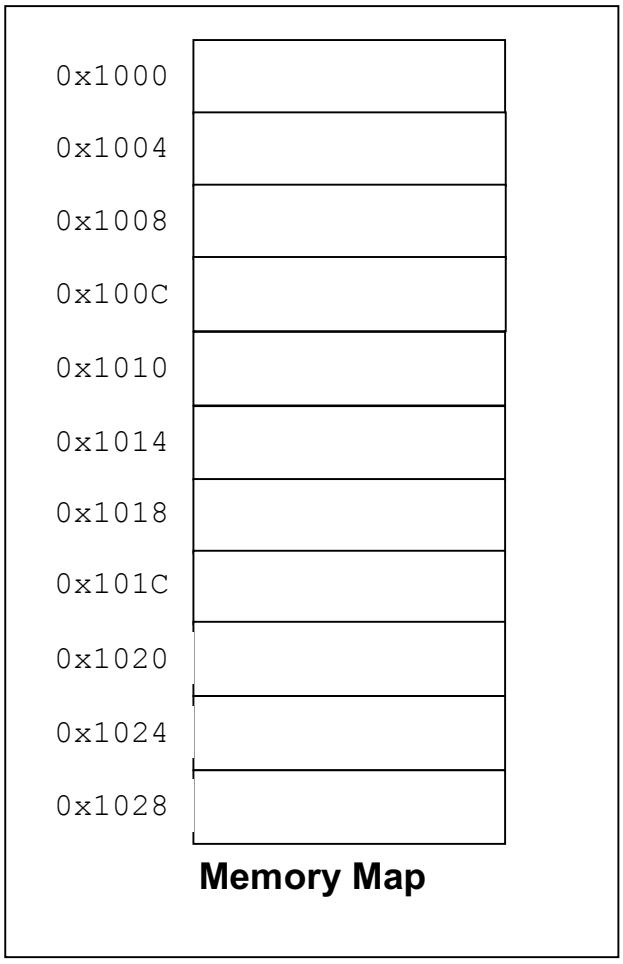
Auf einem 32 Bit System ist folgendes C Programm gegeben:

```
#include <stdio.h>

int a = 9;                    // a liegt ab Adresse 0x1000 in Speicher
int b[]  = {1, 2, (int) &a}; // b liegt hinter a im Speicher
int *p[] = {b,b+2,&a };       // p  liegt hinter b im Speicher
int **pp = p;                 // pp liegt hinter p im Speicher

int main(){
   printf("%x  %x \n",     (int) a,     (int) &a );
   printf("%x  %x  %x\n", (int) b[2], (int) *(int *)(b[2]),
          (int) &b[2] );
   printf("%x  %x  %x\n", (int) p[1], (int) *p[1],
          (int) &p[1] );
   printf("%x  %x \n",
          (int)**pp+2,
          (int) (*(pp+1)- 1) );
   return 0;
}
```

Tragen Sie die Werte von a, b, p und pp in die nebenstehende Memory Map ein.

Geben Sie die Ausgabe des Programms an.

Anmerkung: Beachten Sie, dass %x die Ausgabe **hexadezimal** darstellt.



Praktischer Teil: Aufgaben 3 bis 6

In den nächsten 4 Aufgaben wird ein Reaktionstester entwickelt. In die Bewertung der Aufgaben 3 bis 6 fließen folgende Punkte ein:

- Erreichte Funktionalität
- Dokumentation
- Struktur des Programms
- Berücksichtigung von Fehlersituationen
- Einhaltung der Coding-Rules

Sobald Sie eine der Aufgaben 3 bis 6 gelöst haben, lassen Sie die Funktionsweise des Programms von der Aufsicht abnehmen. Drucken Sie anschließend die Lösung aus, heften und packen Sie diese zur Lösung, die am Schluss abgegeben wird. Die Aufsicht notiert die Abgabe der Aufgabe.

Die Funktionsweise eines Programms wird wie folgt überprüft:

- Sie zeigen der Aufsicht, dass das Programm ohne Warnungen und Fehlermeldungen übersetzt wird.
- Zu jeder Aufgabe gibt es einen kleinen Test. Zeigen Sie der Aufsicht, dass dieser Test fehlerfrei durchläuft.

Inhalt: Der Taster S7 (angeschlossen an PE7) dient als Eingabe für den Reaktionstester. Die acht blauen LEDs (PG8 bis PG15) und das Display werden zur Ausgabe verwendet. Der Reaktionstester arbeitet wie folgt:

- Nach dem Start des Programms wird auf dem TFT-Display der Text

  Zum Starten - Taste S7 druecken

  angezeigt.
- Nach Drücken von Taste S7 gibt TFT-Display die Meldung

  ! ! ! E S G E H T L O S ! ! !

  aus.
- Das Programm wartet bis der Taster mindestens 3 Sekunden lang nicht gedrückt wurde. Danach startet der Reaktionstest.
- Im Reaktionstest sollen die 8 LEDs der Reihe nach im zeitlichen Abstand von ca. 50ms aufleuchten.
- Sobald der Taster gedrückt wird oder die letzte LED aufleuchtet (Spieler hat schlechte Reaktion), wird die Zeitmessung des Reaktionstesters gestoppt.
- Falls alle 8 LEDs leuchten und der Spieler somit zu langsam war, wird der Text

  Leider zu langsam

  ausgegeben. Ansonsten wird die Reaktionszeit ausgegeben.
  Die Zeit wird in Mikrosekunden ausgegeben. Die Ausgabe der Zeit soll zeichenweise erfolgen – also Formatierungsfunktionen wie snprintf, sprintf, itoa dürfen nicht verwendet werden.
- Nach einer Wartezeit von 5 Sekunden geht der Reaktionstester wieder in den Startzustand über.

Name Matrikelnummer

Aufgabe 3 Projekt erstellen & GPIO Ansteuerung (200 Punkte)

Erstellen Sie ein neues Projekt mit den Namen reaktest.

Fügen Sie ein Modul gpio.c (mit der Header Datei gpio.h) in das Projekt ein. Es stellt folgende Funktionen zur Verfügung:

```
/**
 * @brief  Diese Funktion initialisiert die verwendeten GPIOs
 *         (falls notwendig)
 * @param  None
 * @retval None
 */
void initGpio(void);

/**
 * @brief  Diese Funktion setzt die 8 blauen LEDs, die an PG8 bis PG15
           angeschlossen sind.
 * @param  value Bitmaske mit den Werten der LEDs.
 * @retval None
 */
void setBlueLeds(unsigned char value);

/**
 * @brief  Die Funktion überprüft, ob S7 gedrueckt ist.
 * @param  NONE
 * @retval Der Return Wert ist 0 genau dann wenn S7 nicht gedrückt ist.
 */
unsigned char buttonPressed(void);
```

Rufen Sie in main folgende Testfunktion auf:

```
#include "TI_Lib.h"
void testGpio(void) {
       uint8_t v = 0xFF;
       while (1) {
                  setBlueLeds(v);
                  Delay(1000);
                  v = v << 1;
                  if (buttonPressed()) {
                            v = 0xFF;
                     }
          }
}
```

Zeigen Sie der Aufsicht die Ausgabe der Testfunktion, drucken Sie die Dateien gpio.c, gpio.h und main.c aus, heften Sie diese und legen Sie das Papier für die Abgabe beiseite.

Hinweis: Informationen zur Ansteuerung der GPIOs.

- Input Data Register

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Output Data Register

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit Reset Register

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSRRH	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

- Bit Set Register

Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSRRL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Name Matrikelnummer

Aufgabe 4 TFT Ansteuerung (200 Punkte)

Fügen Sie ein Modul display.c (mit der Header Datei display.h) in das Projekt ein. Es stellt folgende Funktionen zur Verfügung:

```
/**
 * @brief  Diese Funktion initialisiert das TFT Display und setzt
 *         das Ausgabefenster.
 * @param  None
 * @retval None
 */
void initDisplay(void);

/**
 * @brief  Diese Funktion loescht das Display und gibt den als
 *         Parameter übergebenen String aus.
 * @param  str Der String, der ausgegeben werden soll.
 * @retval None
 */
void printStr(char * str);

/**
 * @brief  Diese Funktion loescht das Display und gibt den die
 *         Reaktionszeit zusammen mit einen passende Text auf
 *         auf dem Display aus.
 * @param  time Reaktionszeit in Mikrosekunden
 * @retval None
 */
void printReactionTime(int time);
```

Hinweis: Im Modul tft finden Sie die Funktionen zur Ansteuerung des TFT Displays.

Rufen Sie in main folgende Testfunktion auf:

```
void testTFT(void) {
   while (1) {
      printStr("Aufgabe 4 geschafft!!\n");   Delay(2000);
      printReactionTime(0);                  Delay(2000);
      printReactionTime(-10);                Delay(2000);
      printReactionTime(123000);             Delay(2000);
   }
}
```

Zeigen Sie der Aufsicht die Ausgabe der Testfunktion, drucken Sie die Dateien display.c, dispay.h und main.c aus, heften Sie diese und legen Sie das Papier für die Abgabe beiseite.

Aufgabe 5 Timer Modul (100 Punkte)

Fügen Sie ein Modul timer.c (mit der Header Datei timer.h) in das Projekt ein. Es stellt folgende Funktionen zur Verfügung:

```
#include <stdint.h>
/**
 * @brief  Diese Funktion initialisiert das Timer-Modul
 * @param  None
 * @retval None
 */
void initTimer(void);

/**
 * @brief  Diese Funktion liefert einen TimeStamp
 *         84 Timer Ticks entsprechen 1 us
 * @param  None
 * @retval Aktuelle TimeStamp
 */
uint32_t getTimeStamp(void);

/**
 * @brief  Diese Funktion berechnet die Zeitspanne zwischen zwei
 *         Time Stamps
 * @param  firstTimeStamp ist der erste (frühere) Zeitstempel
 * @param  secondTimeStamp ist der zweite (spätere) Zeitstempel
 * @retval Die Zeit (in us), die zwischen den beiden Zeitstempeln
 *         vergangen ist.
 */
uint32_t timerDiffToUsec(uint32_t firstTS, uint32_t secondTS);
```

Hinweise:
Folgende Codesequenz initialisiert den Timer:

```
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN; /* Takt fuer Timer 2 einschalten */
TIM2->CR1 = 0;                      /* Timer disabled                */
TIM2->CR2 = 0;                      /*                               */
TIM2->PSC = 0;                      /* Prescaler   (84MHz)           */
TIM2->ARR = 0xffffffff;             /* Auto reload register          */
TIM2->DIER  = 0;                    /* Interrupt ausschalten         */
TIM2->CR1 = TIM_CR1_CEN;            /* Enable Timer                  */
```

Folgende Codesequenz liefert den aktuellen TimeStamp: `TIM2->CNT`

Name Matrikelnummer

Rufen Sie in main folgende Testfunktion auf:

```
void testTimer(void) {
   uint32_t t;
   while (1) {
      t = getTimeStamp();
      Delay(3000);
      printReactionTime(timerDiffToUsec(t, getTimeStamp()));
      Delay(8000);
   }
}
```

Zeigen Sie der Aufsicht die Ausgabe der Testfunktion, drucken Sie die Dateien timer.c, timer.h und main.c aus, heften Sie diese und legen Sie das Papier für die Abgabe beiseite.

Aufgabe 6 Automat zur Steuerung des Reaktionstesters (300 Punkte)

Implementieren Sie nun in der Datei main.c oder in einem separaten Modul die Steuerung des Reaktionstesters. Es könnte hilfreich sein, wenn Sie einen Automaten und eine (eventuell abgewandelte Version) von Direct Digital Control (DDC) in einer Super Loop verwenden.

Bitte demonstrieren Sie der Aufsicht die

- fehler- und warnungsfreie Übersetzung des Programms

und folgende Testfälle:

- Reaktionszeit wird überschritten
- Reaktionszeit wird nicht überschritten
- Nach der Aufforderung „*Zum Starten - Taste S7 druecken*“ wird der Taster ca. 10 Sekunden ohne Unterbrechung gedrückt.
  In diesem Fall sollte der Text „*! ! ! E S G E H T L O S ! ! !*“ sofort auf dem TFT Monitor erscheinen und der Reaktionstest ca. 3 Sekunden **nach dem Loslassen** des Tasters starten.

Drucken Sie die Datei main.c und ggf. die Dateien des Moduls für diese Aufgabe aus und heften Sie diese.

Heften Sie die Lösungen aller Teilaufgaben zusammen und geben Sie diese der Aufsicht ab.