Prüfungsleistung „gut+": 12 v 15 P
30. Jan. 07 Rinschede

17

Name, Vorname, Matr.Nr.: Christiansen, Frank, 812585 24. Jan. 2007 1

HAW Hamburg - Prüfungsklausur Mikroprozessortechnik - WS 2006/2007					
Aufgabe	1	2	3	4	Summe
Punkte	10+3+8+3=24	3+4+4=11	3+3+3=9	8+4+18+14+12=56(+10)	100 (+10)
	5+3+8+3=23	2+1+1=4	9	3+3+12+10+9=37	73 (+7)

80

Aufgabe 1: Programmanalyse

9 v10

a) Kommentieren Sie das nachfolgende Programm. [1]

```
#include <mpp1.h>          /* symbolische Registernamen aus Headerfile */
void main(void)
{
  TPU_TCNT2 = 0x0000;      /* Der Counter des 3. Timers wird auf 0 gesetzt */

  TPU_TCR2  = 0x27;        /* Der Counter wird auf 0 gesetzt wenn TGRA erreicht ist (rising edge
                              und es wird ein 1024tel des CLK zum zählen genutzt */

  TPU_TSR2 &= 0xFC;        /* Das TGFB und das TGFA Flag werden auf 0 gesetzt, die anderen bleiben */

  TPU_TGR2A = 10799;       /* TGRA = 10800 also Zählt der Timer 10800 Schritte */

  TPU_TGR2B = 5399;        /* TGRB = 5400 */                    } Zählung beim 5399 -1

  P2DDR = 0x07;            /* An Port 2 werden P2(0), P2(1) und P2(2) als Ausgang gesetzt,
  P2DR = 0x07;             /* An den 3 Ausgängen werden 1en ausgegeben */   rest ist Eingang */
                                                                TPU ch.3
  TPU_TSTR  = 0x04;        /* Der 3. Timer wird gestartet */

  while((TPU_TSR2 & 0x02) == 0x00); /* Warten solange TGFB nicht gesetzt ist */
  P2DR = 0x06;             /* P2(0) auf 0 setzen P2(1) und P2(2) bleiben 1 */

  while((TPU_TSR2 & 0x01) == 0x00); /* Warten solange TGFA nicht gesetzt ist */
  TPU_TSR2 &= 0xFC;        /* TGFA und TGFB zurücksetzen auf 0 */

  P2DR = 0x04;             /* P2(1) auf 0 setzen, P2(0) bleibt 0 und P2(2) bleibt 1 */

  while((TPU_TSR2 & 0x01) == 0x00); /* Warten bis TGFA wieder gesetzt ist */
  P2DR = 0x00;             /* P2(2) auf 0 setzen, P2(0) und P2(1) bleiben 0 */

  TPU_TSTR  = 0x00;        /* Timer stoppen */
}
```

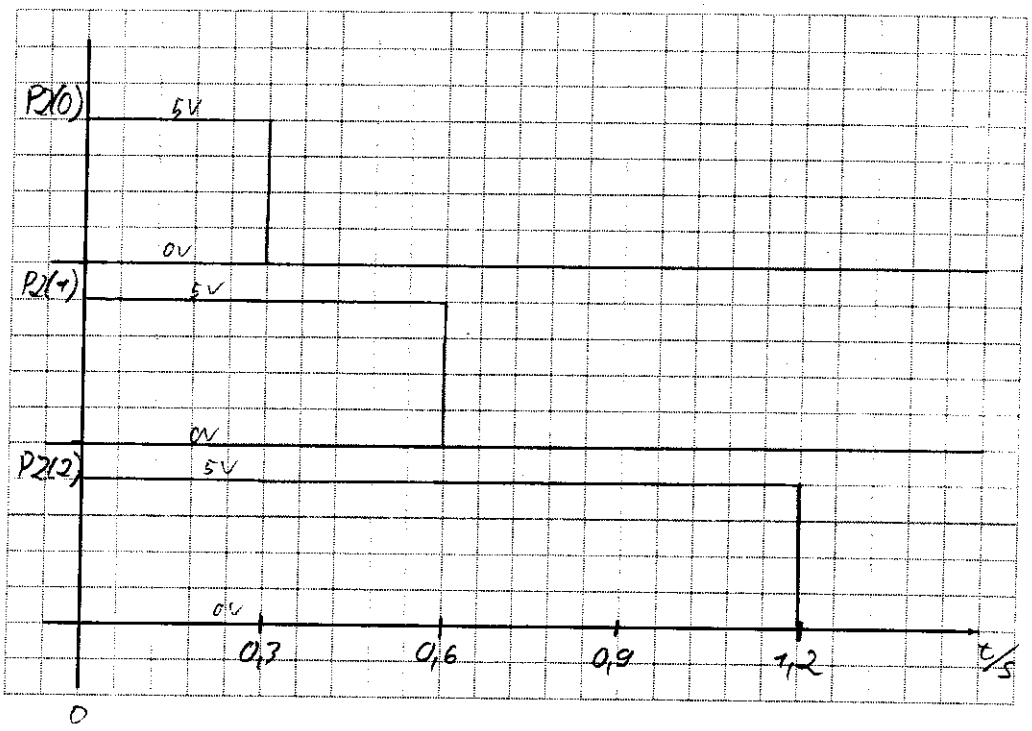
5 v3

b) Nennen Sie die im Programm zur Ein- oder Ausgabe genutzten Port-Pins.

P2(0), P2(1) und P2(2) als Ausgabe

[1] Falls notwendig, verweisen Sie auf ein gesondertes Blatt.

c) Skizzieren Sie den zeitlichen Verlauf der Signale bei den Pins aus Aufgabe b).



8 V8

d) Geben Sie die Zeitdauer für jedes Signal aus Aufgabe c) an, in denen es den Wert logisch '1' besitzt.

P2(0) hat 0,3 Sekunden lang ein 1 am Ausgang, P2(1) 0,6 Sekunden und P2(2) 1,2 Sekunden lang. ✓

3 V3

23 V24

Aufgabe 2: Adressdecoder

Ein ROM-Chip und ein RAM-Chip sollen an einen Controller H8S/2357 angeschlossen werden. Beide Chips haben 4 Megabyte Kapazität. Der ROM-Chip soll mit den untersten Adressen angesprochen werden, der RAM-Chip mit den nächsthöheren Adressen.

a) Geben Sie **die Anzahl und die Bezeichnungen der Adressleitungen** des Controllers, des RAM-Chip und des ROM-Chip an.

Controller: Der Controller hat 3 Exter ...

RAM-Chip: siehe Extrablatt !

ROM-Chip: 2P. vom Blatt

2 V3

b) Geben Sie die **Gleichung** für den Adressdecoder des ROM-Chip an.

$\overline{CS_{ROM}} = \overline{A_{23}} \wedge A_{22}$ 1P. vom Blatt

1 V4

c) Geben Sie die **Gleichung** für den Adressdecoder des RAM-Chip an.

$\overline{CS_{RAM}} = \overline{A_{23}} \wedge \overline{A_{22}}$ 1P. vom Blatt

1 V4

4 V11

Name, Vorname, Matr.Nr.: Christiansen, Frank, 1812585 24. Jan. 2007 3

Aufgabe 3: Serial Interface

In einem Gerät mit Batterieversorgung wird der Controller H8S/2357 mit nur **1.2288 MHz** Systemtakt (clk) betrieben. Einer der Channels des SCI wird genutzt.

a) Geben sie die **maximale Bitrate** an !

Br_{max}= 38400 $\frac{bit}{s}$

3 v. 3

c) Geben Sie den dafür gewählten Prescaler-Faktor p und die beiden Bits in SCIx_SMR an.

p = 1

CKS0 = 0

CKS1 = 0

3 v. 3

d) Geben Sie den dafür gewählten Eintrag im Register SCIx_BRR an.

SCIx_BRR = 0x00 = 0

3 v. 3

9 9

Aufgabe 4: Programmieraufgabe

Nehmen Sie an, an der Eingangstür der HAW soll ein einfaches Codeschloss eingebaut werden. Ein überzähliger Controller H8S/2357 aus dem MP-Labor wird benutzt. Eine 3 x 3 Tastatur-Matrix ist an Port 1 angeschlossen.

- Pins 0, 1, 2 von Port 1 dienen als Ausgang für die Spalten.
- Pins 4, 5, 6 von Port 1 dienen als Eingang von den Zeilen. Sie sind über Pull-up Widerstände auf logisch '1' gesetzt (default), wenn keine Taste gedückt ist.
- Pin 3 von Port 1 betätigt das Türöffner-Modul. Das Türschloss ist offen, solange dort eine logische '1' ausgegeben wird.
- Pin 7 von Port 1 betätigt einen Tongeber (Summer). Es wird einen Ton erzeugt, solange dort eine logische '1' ausgegeben wird.

	P1(0)	P1(1)	P1(2)
P1(4)	7	8	9
P1(5)	4	5	6
P1(6)	1	2	3

Abb 4.1: Tastenanordnung der 3x3 Tastaturmatrix

Funktionsweise:

- Nach jedem Tastendruck wird für eine kurze Wartezeit mit dem Tongeber einen Summerton erzeugt.
- Der Öffnungscode lautet '1','5','9'. [2] Wenn dieser Öffnungscode vollständig eingegeben ist, dann wird für eine mittlere Wartezeit der Türöffner betätigt. Danach kann erneut ein Öffnungscode eingegeben werden.
- Wenn ein falscher Öffnungscode (bereits eine falsche Taste reicht aus) eingegeben wird, dann wird für eine etwas längere Wartezeit ein Summerton ausgegeben. Danach kann erneut ein Öffnungscode eingegeben werden.

[2] Der Öffnungscode ist konstant. Sollte er später einmal geändert werden, wird ein neues Programm geladen.

Arbeitshinweise:

- Bitte auf jedes Blatt den Namen und die Matrikelnummer schreiben.
- Nennen Sie die bearbeitete Teilaufgabe [4a), 4b) ...].
- Kommentieren Sie ausführlich und inhaltlich aussagefähig, auch im Struktogramm.
- Arbeiten Sie die folgenden Teilaufgaben möglichst nacheinander ab.

a) Zeichnen Sie das passende Blockschaltbild. 3 v8

b) Programmieren Sie eine einfache Wartefunktion: `void wait_loop(int time)` 3 v4

- Nutzen Sie Warteschleifen.
- Der Parameter `time` soll die Wartezeit festlegen.
- Nehmen Sie ganz grobe Schätzwerte[3] für die drei oben genannten Wartezeiten an.
- Legen Sie diese drei Wartezeiten mit Hilfe von Präprozessordirektiven fest, um sie später beim Aufruf der Funktion zu verwenden. fast

c) Programmieren Sie eine Funktion: `int key(void)` 12 v18

- Die Funktion wartet zunächst bis eine Taste gedrückt wird.
- Dann wird ermittelt, welche Taste gedrückt wurde.
- Erst wenn die Taste losgelassen ist, wird die Funktion beendet.
- Der Rückgabewert ist eine Ganzzahl vom Typ `int`. Der Wert der Ganzzahl entspricht der Tastenbezeichnung 1 bis 9.

d) Der Öffnungscode soll im Hauptprogramm geprüft werden. Zeichnen Sie ein Struktogramm dafür. 10 v14

e) Schreiben Sie das Hauptprogramm: `void main(void)` 9 v12

37 v56

.....

f) Zusatzaufgabe:
Maximal 10 Zusatzpunkte, welche fehlende Punkte ersetzen können.

Realisieren Sie mit Hilfe der TPU eine Wartefunktion: `void wait_timer(int ms)`
- Der Parameter `ms` soll die Wartezeit in Millisekunden vorgeben.
- Die benötigten Wartezeiten sind etwa 200ms, 1000ms und 1500ms.
- Nutzen Sie den Channel 3 der TPU. Col ? Gar nicht gesehen aber markt ja sicher ;) v. 10

[3] Sie können stark vereinfacht annehmen: 1 Mio. Schleifenzyklen benötigen 1s Zeit

Frank Christiansen
Mat.-Nr. 1812585

Zu 2:

Der Controller hat 3 Externe adressbusleitungen:

A: A23 .. A16
B: A15 .. A8
C: A7 .. A0

} insgesamt 24 Leitungen

1 Punkt a)

und 2 externe Datenbusse

D: D15 .. D8
E: D7 .. D0

Vom RAM und ROM aus 8bit breiten Datenbus haben

⇒

4 MB = 4194304 Byte = 33554432 Bit

$\frac{\text{4194304 Byte}}{\text{8 bit}}$ = 4194304 Adressen werden benötigt je Chip...

$2^{16} = 65536$

hm...

$2^{24} = 16777216$

offensichtlich brauchen wir alle 3 Externen Adressbusse...

hm... sieht komisch aus... hm... egal

1. Adressbereich würde von 0 bis 4194303 ✓ 1P. zu b)

2. Adressbereich würde von 4194304 bis 8388607

egtl. ✓ 1P c)

OHNE 4 MB = $\frac{1}{4}$ Adressbereich ist einfach

✓ 1P ⇒ a)

zeit vorbei ☺ daty

Frank Christiansen
Matr.Nr.: 189585

Zu 3:

$$B_t = \frac{pclk}{(BRR+1)\cdot 32}$$ ✓

$\Rightarrow$ B_t ist hoch wenn pclk ↑ und BRR klein ist

$$pclk = \frac{clk}{P}$$

$\Rightarrow$ prescaler muss klein sein

$\Rightarrow$ $CKS0 = 0$, $CKS1 = 0 \Rightarrow P = 1$ ✓

$\Rightarrow$ $B_{t_{max}} = \frac{clk}{32}$ ✓

Punkte $\Rightarrow$ siehe Aufgabe

Frank Christiansen
Mat.-Nr.: 1812585

(4)

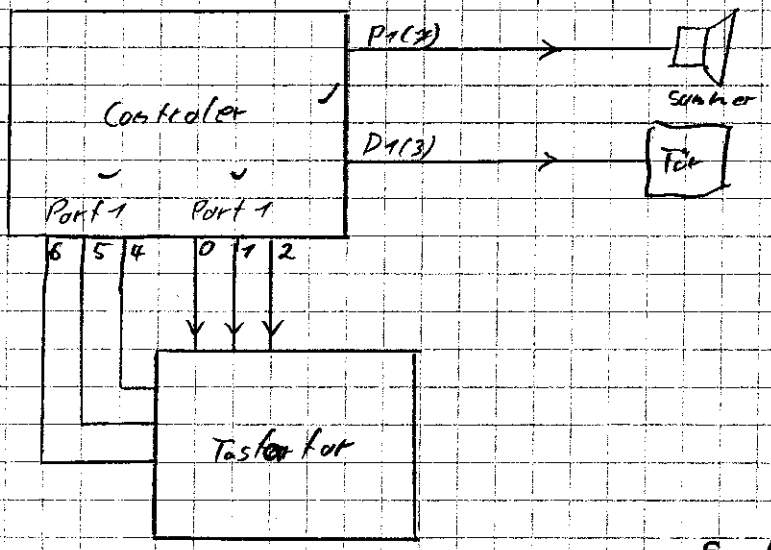
Ausgang:

P1(0), P1(1), P1(2), P1(3) (für), P1(7) (Summer)

Eingang:

P1(4), P1(5), P1(6)

a)



Es fehlt

- Tastatur einverschaltung
- Tasten zuordnung
- Pullup - Widerstände
- Kann zwischen Summer / Türöffner als „high" ablesen

$\sum$ 3 v. 8 P.

vom Einsteiger

Frank Christiansen
Matr.Nr.: 1892585

b)
```
void wait_loop (int time)
{
  int i;
  int j;
  for (i=0; i< time ; i++)
  {
    for (j=0, j<3000, j++) ; /* entspricht nach Labor-
                                messung ca 1ms. */
  }
}
```

Σ 3 v. 4 P

vom Tutor eingetragen

```
#define KURZ 100 000
#define MITTEL .....
usw.
```

Frank Christiansen
Matr.-Nr.: 1892585

c)

```
int Key (void)
{
  int Key;
  P2DR = 0;  /* Alle Ausgänge auf 0setzen */
  while ((PORT1 | 0x8F) == 0xFF); /* Warten bis irgendeine Taste
                                    gedrückt wurde */
                    06
  P2DR = 0x 0E; /* P1(0) auf 0 rest 1 von 0-3 */
**)
  if ((PORT1 & 0x10) == 0) { Key = 7; }; /* Tasten einzeln
                                            abfragen */
  if ((PORT1 & 0x20) == 0) { Key = 4; };
  if ((PORT1 & 0x40) == 0) { Key = 1; };
                    05
  P2DR = 0x0D; /* P1(1) auf 0 rest 1 von 0-3 */
**)
  if ((PORT1 & 0x10) == 0) { Key = 8; }; /* Tasten einzeln
                                            abfragen */
  if ((PORT1 & 0x20) == 0) { Key = 5; };
  if ((PORT1 & 0x40) == 0) { Key = 2; };

  P2DR = 0x0B; /* P1(2) auf 0 rest 1 von 0-3 */
**)
  if ((PORT1 & 0x10) == 0) { Key = 9; }; /* Tasten einzeln
                                            abfragen */
  if ((PORT1 & 0x20) == 0) { Key = 6; };
  if ((PORT1 & 0x40) == 0) { Key = 3; };

  :
  :
  "Warte auf Tastatur"
  :
  :
  return Key;
}
```

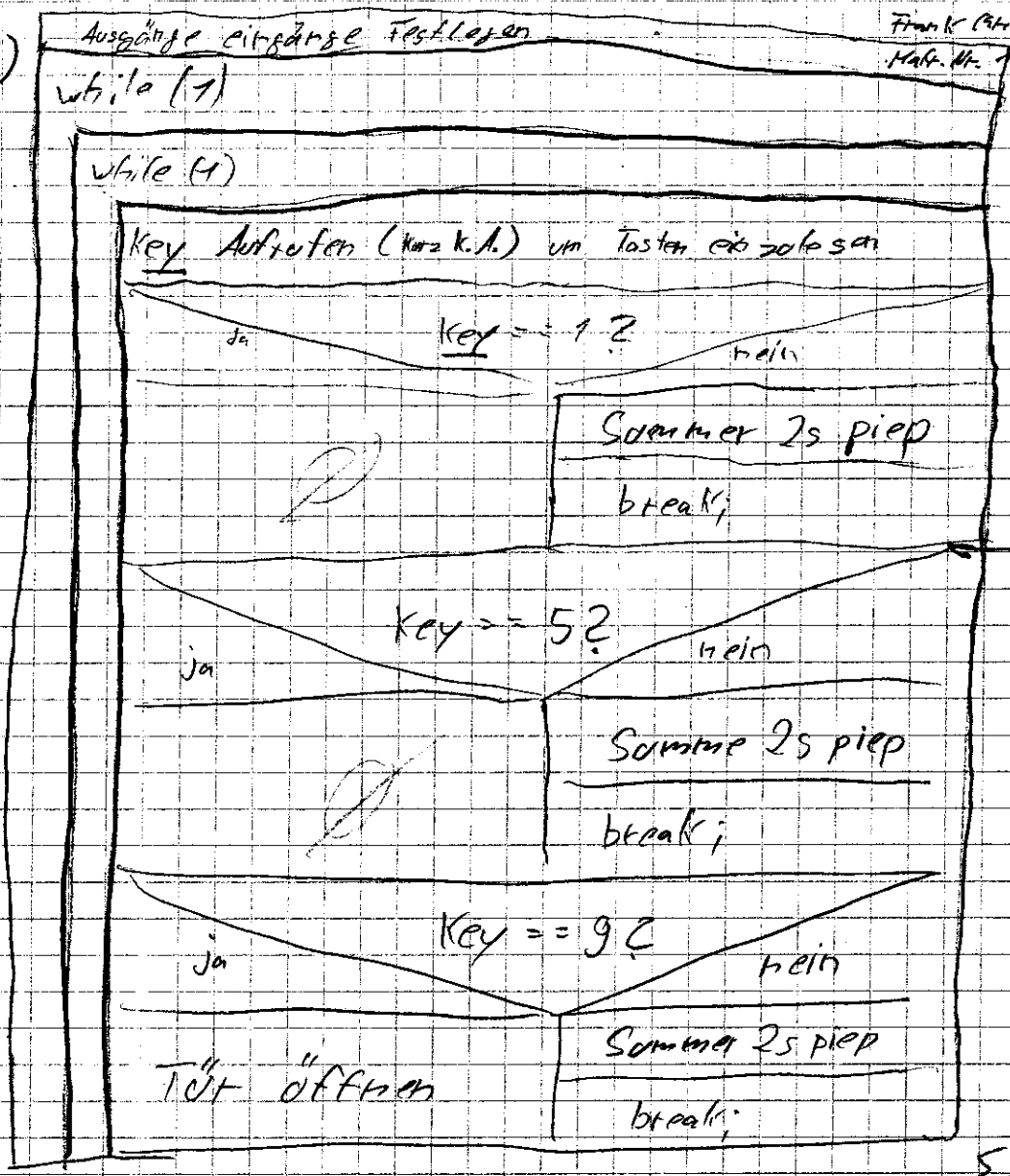
$\sum$ 1hr. 18P
vorm eintragen

* Die mehrfache PORT1 Abfrage ist problematisch, da können Werte sich zwischenzeitlich ändern!

** Hier kurze Warte zeit für Signal auf der Tastaturseite

Frank Christiansen
Matr.-Nr. 7812585

d)



Σ 10v. 14P
⇒ von

- Es fehlt:
- Kurzer Ton nach jeder Taste!
- → Schema der Tür/Summer Verknüpfung
  1) Pin x(y) = 1
  2) warte_loop(....)
  3) Pin x(y) = 0

Eine doppelte Endlosschleife mit „break" ist eine unsaubere Lösung.

Wenn sie später so programmieren, haben sie wichtige Aspekte der Fehlervermeidung durch saubere Strukturen nicht genutzt.....

Frank Christiansen
Matr.Nr.: 4812585

e)
```
void main(void)
{
 P1DDR = 0x8F;

 while (1)
 {
  while (1)
  {

   wait_loop (200);
   if (!(Key()== 1))
   {
    P1DR = 0x80;

    wait_loop (2000);
    P1DR = 0x0F;
    break;
   }
   if (!(Key() == 5))
   {
    P1DR = 0x80;

    wait_loop (2000);

    P1DR = 0x0F;

    break;
   }
   if (!(Key() == 9))
   {
    P1DR = 0x80;

    wait_loop (2000);

    P1DR = 0x0F;

    break;        /* Koennten wir uns schenken */
   else
   {
    /* Tuer oeffnen */
    P1DD = 0xFF;
    wait_loop (1000);
    P1DR&= 0x0F      /* Summer Aus, tür weiteröffnen */
    wait_loop (4000);
    P1DR = 0x00      /* Summer und Tür Aus */
   }
  }
 }
}
```

Kommentar dazu fehlt zuvor

⇒ fast keine Kommentare! −1

← s.u.

← s.u.

← jedesmal die Tür auf! −1P.

nicht exakt lt. Aufgabe auch nicht lt. Struktogramm −1

FSR - Klausurensammlung 11/12 | WS 06 | Semester E4 | Fach MP | Dozent RMS

Σ 8v. 12

11/12

Frank Christiansen
Matr.-Nr.: 1812585

f)

```
void wait_timer (int ms)
{
  TPU_TCNT2 = 0; /* Counter auf 0 setzen */
  TPU_TGR2A = ms * 32; /* t = 18 ! */
  TPU_TCR2 = 0x27; /* Das Counter clear
                      bei TGRA ist
                      eigentlich egal */
  TPU_TIOR2 = 0;
  TPU_TSR2 &= 0xEC; /* TCIV
                       TGFB
                       TGFA
                       auf 0 setzen */
  TPU_TSTR = 0x04; /* den 3. Timer
                      starten */
  while ((TPU_TSR2 & 0x01) == 0);
  /* warten bis TGFA gesetzt ist */
  TPU_TSTR = 0; /* Timer Stoppen */
}
```

max: 2 s

Timer zählt: $2^{16} = 65536$

$\Rightarrow \frac{2\,s}{65536} = 31\,\mu s$

Kleinster möglicher Schritt

$\Rightarrow$ CLK/1024

$\Rightarrow$ 3 Timer

$\frac{1\,ms}{31\,\mu s} = 32$

3

7 v. 10 P.

12/12