26

15 Punkte

HAW Hamburg - Prüfungsklausur Computertechnik - SS 2008					
Aufgabe	1	2	3	4	Summe
Punkte	18	13	33	36	100
	14	13	30	34	91

Deutsche Leistung! Rücksicht 20/7/08

Aufgabe 1: Programmanalyse

a) Kommentieren Sie das nachfolgende Programm in **jeder** Zeile genau beschreibend. [1]

```
#include <mpp1.h>

void main(void)
{
  TPU_TIOR1 = 0x33; // Timer1: TIOCB1 toggle at compare match
                    // Timer1: TIOCA1 toggle at compare match

  TPU_TCNT1 = 0x0000; // Counter von Timer1 zurücksetzen

  TPU_TCR1  = 0x21; // Counter Clear by TGRA1-Event
                    // Clock Edge Selection: rising Edge
  TPU_TGR1A = 21599; // Clock source: overflow of TCNT2 in channel 2
                     // 1/18.432 MHz · 2^16 · 2 = 600 Timerticks = 76,8s
  TPU_TGR1B = 7199; // 1/18.432 MHz · 2^16 · 7

  TPU_TSTR |= 0x02; // Timer Channel 1 starten

  while(1); // Endlosschleife

}
```

Siehe Extrablatt ✓ o.k.

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

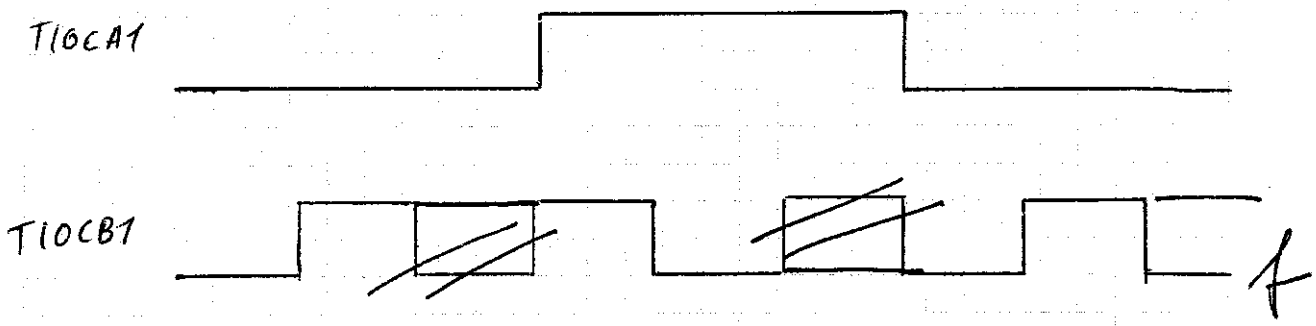
Name, Vorname, Matr.Nr.: Jacobsen, Martin, ~~185 7660~~ 1. Juli 2008 2

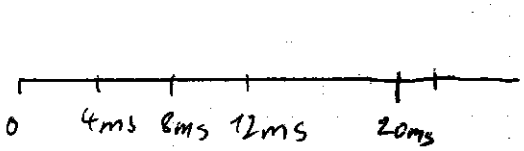
b) Nennen Sie das Ausgabesignal oder die Ausgabesignale im Programm zuvor.

Name[n]:
TIOCB1 (Port1, Pin5) TIOCA1 (Port1, Pin4)

1 v. 1

c) Skizzieren Sie das Ausgabesignal oder die Ausgabesignale. Beginnen Sie mit dem Programmstart. Beschriften Sie die Achsen des Diagramms vollständig. fehlt!





Ja, oben gezeichnet!

7 v. 10

d) Nennen Sie die Periodendauer.

Periodendauer:
TIOCB1: 12,25ms ✓ TIOCA1: 37,5ms ✓

1 v. 2

14 v. 18

Name, Vorname, Matr.Nr.: Jacobsen, Martin, 1857660 1. Juli 2008 3

Aufgabe 2: Prozessorsystem

a) Nennen Sie die Namen der Busse und deren Signalrichtungen in einem gewöhnlichen, einfachen Prozessorsystem.

Name des Busses	Signalrichtung
Adressbus	unidirektional: Von der CPU zum Memory I/O
Datenbus	bidirektional: Von der CPU zum Memory I/O und zurück
Controlbus	bidirektional: Von der CPU zum Memory I/O und zurück

- Wie groß ist die Kapazität des Adressraums des Controllers H8S/2357 ? 16 MByte ✓
- Wie breit ist der Adressbus ? 24 Bit ✓
- Wie werden die Leitungen des Adressbusses genannt ? $A_0, A_1, A_2, \ldots, A_{22}, A_{23}$ ✓

4 v. 4

c) An einen Controller H8S/2357 sollen angeschlossen werden:

1. Genau ein ROM-Chip. Der ROM-Chip hat 4 Megabyte Kapazität.
2. Genau so viele RAM-Chips, wie sinnvoll möglich sind. Alle RAM-Chips sind Typen mit 8 Megabyte Kapazität.

- Die gesamte verbleibende Kapazität des Adressraums soll genutzt werden. Es soll keine unbenutzte Lücke in der Adressenbelegung entstehen.

Wie viele RAM-Chips werden Sie anschließen ? Anzahl der RAM-Chips: 2

1 v. 1

d) Diskutieren Sie in kurzen Stichworten das entstehende Problem.

Unser Adressraum hat eine Kapazität von 16 MByte
Mit einem ROM-Chip und 2 RAM-Chips benötigen
wir allerdings eine Kapazität von 20 MByte.
Fazit: 4 MByte des zweiten RAM-Chips können
nicht genutzt werden. ✓

2 v. 2

e) Bieten Sie einen Lösungsvorschlag an.

Der RAM Chip hat die Adressleitungen
$A_0 \ldots A_{22}$, wenn A_{22} dauerhaft auf
Low-Pegel gelegt wird kann man nur die
ersten 4 MByte des Chips benutzen.
Prima Vorschlag!

6 v. 6

13 v. 13

SS / WS	Semester	Fach	Dozent
08	E 4	CT	RMS

Name, Vorname, Matr.Nr.: Jacobsen, Martin, 1857660 1. Juli 2008 4 9/13

Aufgabe 3: Serial Interface

Ein Mobiltelefon mit einer Schnittstelle nach dem RS232-Standard soll die empfangenen SMS-Nachrichten an das Praktikums-Laborsystem weiterleiten. Die Einstellungen des Schnittstelle sind nicht bekannt, daher werden Signale mit einem Speicheroszilloskop am TxD-Pin an der Mobiltelefon-Buchse (außen) aufgezeichnet.

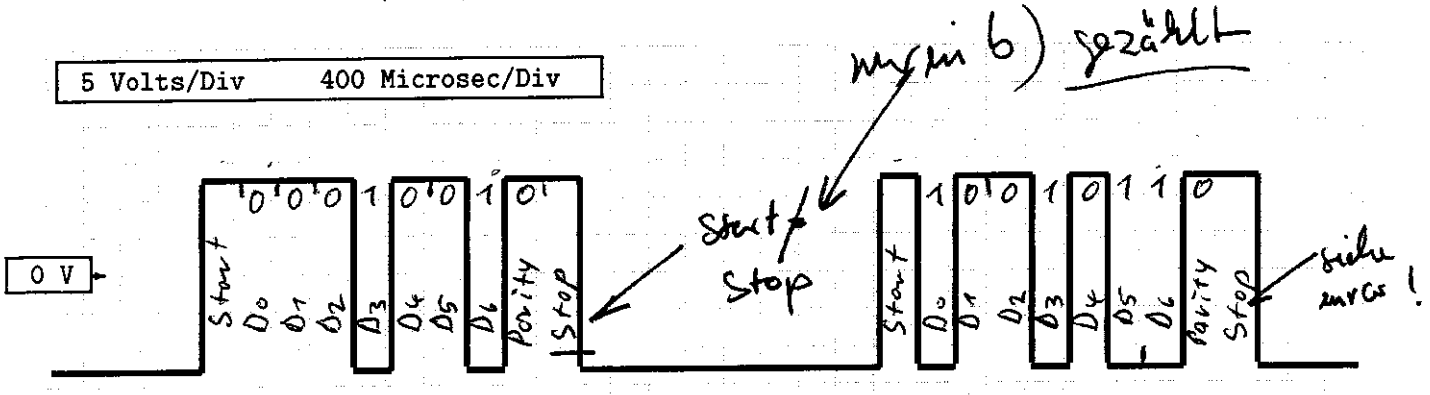


Bild 1: Anzeige des Speicheroszilloskops

a) Benennen Sie alle Bits (Namen und Wert) in der Signaldarstellung im Bild 1.

b) Wieviel Bit sind Nutzdaten ? ...7... Bit

Wieviel Bit gehören zum Protokoll ? ...10... Bit

c) Welche Parameter der seriellen Übertragung sind eingestellt (Kurzbezeichnung genügt)?

7 E 1

d) Welche Nutzdaten wurden übertragen? Geben Sie die Daten binär, hexadezimal und als ASCII-Zeichen an.

1. Frame (binär/hex/ASCII): 100 1000 / 48 / H

2. Frame (binär/hex/ASCII): 110 1001 / 69 / i

e) Welche Datenrate wird genutzt ? (Ankreuzen)

☐ 110 bit/s	☐ 300 bit/s	☐ 1200 bit/s	☒ 2400 bits/s	☐ 4800 bits/s
☐ 9600 bit/s	☐ 19200 bit/s	☐ 38200 bit/s	☐ 57600 bit/s	☐ 115200 bit/s

ASCII Tabelle (niederwertige Bits in den Spalten, höherwertige Bits in den Zeilen)

	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0x0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Name, Vorname, Matr.Nr.: Jacobsen, Martin 1857660 1. Juli 2008 5

f) Schreiben Sie ein gut kommentiertes C-Programm für die Anzeige der empfangenen SMS mit dem Praktikums-Laborsystem[2].

- Benutzen Sie den SCI-Channel 2.
- Wählen Sie das passende Übertragungsprotokoll und die Bitrate, wie zuvor in c) und e) bestimmt.
- Nutzen Sie das Headerfile `mpp1.h`.
- Zuerst programmieren Sie eine passende Funktion zur Initialisierung.

```
#include <mpp1.h>
void serial_receiver_init (void)
{
                                    // 01100001 ≙ 0x61
  SCI2_SMR  = 0x61;// 7 Bit Daten, 1 Stop Bit, CLK/4, Parity: Eve
  SCI2_BRR  = 59;  // 2400 Bit/s
  SCI2_SCR  = 0x30; // Transmission & Reception Enable
}                   0x10      7
```

$$// BRR_{theo} = \frac{18{,}432\,MHz}{4 \cdot 32 \cdot 2400} = 60$$

$$// BRR = BRR_{theo} - 1 = 59$$

9

[2] H8S/2357 mit 18.432 MHz

g) Nun schreiben Sie die Funktion `void main(void)`, ebenfalls mit ausführlichen Kommentaren.

- Diese Funktion gibt alle Daten der SMS als Zeichen an das Terminal aus.
- Die Funktion `putchar()` oder `printf()` sind verfügbar, wenn Sie das passende Headerfile einbinden.
- Die SMS wird durch ein Sonderzeichen mit allen Bits gleich 0 abgeschlossen. Dieses Zeichen geben Sie nicht aus. Statt dessen beginnen Sie am Terminal eine neue Ausgabezeile für die nächste SMS. Das ist beispielsweise durch die Ausgabe von der ASCII-Sonderzeichen `CR` und `LF` möglich.

```
#include <stdio.h> // Headerfile für printf
void main(void)
{
char c;

serial_receiver_init();

while(1) {
  while((SCI2_SSR & 0x40)==0); // Warten bis
  // Zeichen empfangen wurde
  c=SCI2_RDR; // Zeichen auslesen
  SCI2_SSR &= 0xBF; // RDRF-Flag auf 0 setzen

  if(c==0x00) { // Ende der SMS
    printf("\n"); // neue Zeile
  }
  else {
    printf("%c", c); // Ausgabe des empfangenen
                     // Zeichens
  }
}
}
```

Aufgabe 4: Schrittmotorsteuerung

Mit dem Praktikums-Laborsystem[3] ist eine einfache Schrittmotorsteuerung zu entwickeln. Der Schrittmotor wird durch die Signale a, b, c, d angesteuert (Bild 2). Er führt in vier Schritten eine Umdrehung aus. Die Signalfolge dafür zeigt die Tabelle 1.

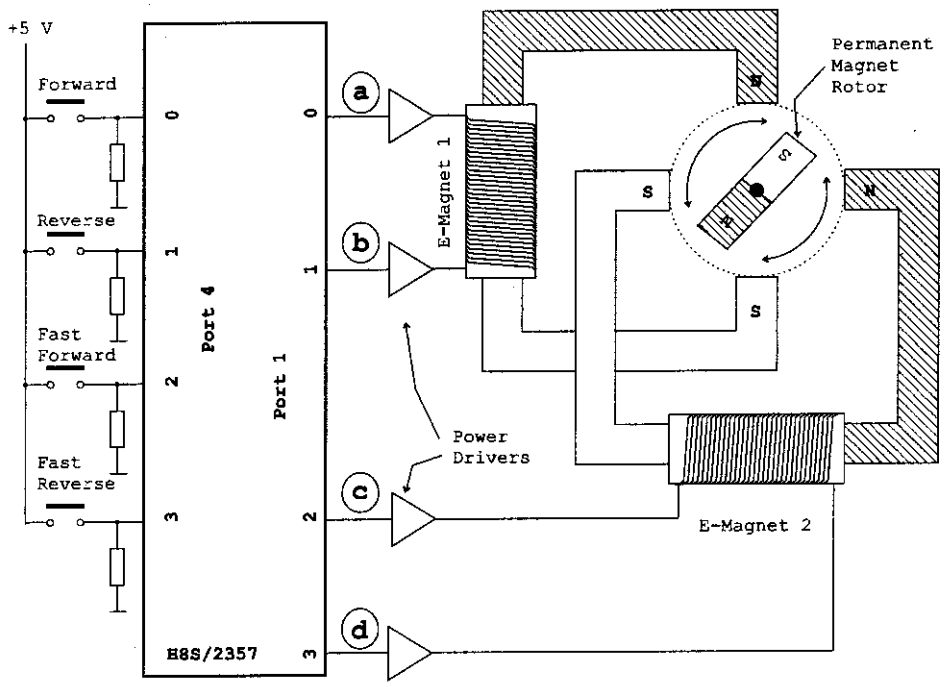


Bild 2: Schaltung der Schrittmotorsteuerung (Zustand Schritt 1)

	Signal	d	c	b	a
Drehrichtung vorwärts	Folge für eine Umdrehung vorwärts				
	0. Schritt	1	0	0	1
	1. Schritt	0	1	0	1
	2. Schritt	0	1	1	0
	3. Schritt	1	0	1	0
	für nächste Umdrehung weiter mit 0.				
Drehrichtung rückwärts	Folge für eine Umdrehung rückwärts				
	3. Schritt	1	0	1	0
	2. Schritt	0	1	1	0
	1. Schritt	0	1	0	1
	0. Schritt	1	0	0	1
	für nächste Umdrehung weiter mit 3.				
	Motor STOP	0	0	0	0

Tabelle 1: Signalfolge der Schrittmotorsteuerung

- Wenn keine Taste gedrückt ist, dann ist der Motor gestoppt.
- Solange die Taste „forward" gedrückt ist, dreht sich der Motor mit 300 Umdrehungen pro Minute vorwärts.
- Solange die Taste „reverse" gedrückt ist, dreht er genauso schnell rückwärts.
- Solange die Taste „fast forward" bzw. „fast reverse" gedrückt ist, dreht der Motor mit der doppelten Umdrehungszahl vorwärts bzw. rückwärts.
- Beim Ende oder Wechsel der Tastenbetätigung wird stets der aktuelle Schritt beendet. Eine Umdrehung muss jedoch nicht vollendet werden. Der passende nächste Schritt wird mit der nächsten Tastenbetätigung gewählt.
- Werden versehentlich mehrere Tasten zugleich gedrückt, dann bleibt der Motor gestoppt.

[3] H8S/2357 mit 18.432 MHz

a) Nennen Sie die jeweilige Dauer der Schritte.

$300\,U/min \Rightarrow \frac{60s}{300 \cdot 4} = 50ms$ / Schritt | $600\,U/min : \frac{60s}{600 \cdot 4} = 25ms$

b) Schreiben Sie eine kommentierte Funktion, welche die obige Dauer der Schritte mit einem Parameter auswählbar wartet. Die Werte der Parameter sind bereits als Makro vordefiniert. Nutzen Sie die TPU und das Headerfile `mpp1.h`

```
#include "mpp1.h"
#define SLOW 0  /* 300 U/min gewaehlt */
#define FAST 1  /* 600 U/min gewaehlt */
void wait_a_step(int speed)
{
TPU_TCNT0 = 0x00; // Clear Counter 0
TPU_TGR0A = (2-speed)*28800-1; // 28800 Timerticks = 25ms
TPU_TCR0 = 0x02; // CLK/16, Clear disabled
TPU_TSR0 &= 0xFE; // Clear TGFA-Flag
TPU_TSTR = 0x01; // Start Timer Ch 0
while ((TPU_TSR0 & 0x01)==0x00); // Warten
// bis Flag gesetzt ist
TPU_TSTR = 0x00; // Timer 0 stoppen
}
```

c) Zeichnen Sie das Struktogramm oder das Flussdiagramm des Steuerprogramms (Hauptprogramm), nutzen Sie die Wartefunktion. Tipp: Speichern Sie die Ausgaben der Schritte in einem Feld.

Siehe Extrablatt

Name, Vorname, Matr.Nr.: Jacobsen, Martin, 1857660 1. Juli 2008 9

d) Programmieren Sie das Steuerprogramm, kommentieren Sie aussagefähig.

```
void step (int i); // siehe Extrablatt
void main(void)    // Funktion zum schritte setzen
{
 int i=0;  //Zählvariable
// Port4 ist Input Only
P1DDR = 0x0F; //Port1 ist Ausgang
P1DR = 0;  //P1 auf 0 setzen

while (1) {
   switch ((PORT4) & 0x0F ) {
     case 0x01:  step(i);
                 wait_a_step(SLOW);
                 i++;  break;
     case 0x02:  step(i);
                 wait_a_step(SLOW);
                 i--; break;
     case 0x04:  step (i);
                 wait_a_step(FAST);
                 i++;
                 break;
     case 0x08:  step(i);
                 wait_a_step (FAST);
                 i--;
                 break;
     case default: P1DR = 0x00; //Motor Stop
   }
   if(i == (-1))
       i=3;
   if(i == 4)
       i=0;

}
```

(+) viel einfacher wenn sie dem Tipp auf Seite 8 beachtet hätten Dort umbestücken

↑ warten!

FSR - Klausurensammlung | Semester E4 | Fach CT | Prozent PMS | 08 MS | 10/13

Aufgabe 1)

0010 0010

```
TPU_TCR1 = 0x22; // Clear by TGRA1-Event
                 // Clock Edge Selection: rising Edge
                 // Clock Source: CLK/16
TPU_TGR1A = 21599; // 16/18.432MHz · 21600 timerticks
                   ≙ 18,75 ms
TPU_TGR1B = 7199;  // 16/18.432MHz · 7200 timerticks
                   ≙ 6,25 ms
```

o.k. vom µ gezählt

Martin Jacobsen HAW Hamburg Mat-Nr.: 1857660

Aufgabe 4c

Main

int i=0;

P4 (3)...(0) sind Eingang

P1 (3)...(0) sind Ausgang

while(1)

Switch Case Port 4 ==

0x01 ?	0x02 ?	0x04 ?	0x08 ?	default ?
Step(i);	Step(i);	Step(i);	Step(i);	P1OR=0 (Motor Stop)
50ms warten	50ms warten	25ms warten	25ms warten	
i++;	i--;	i++;	i--;	

if (i==(-1))

Ja

i = 3;

Nein

if (i==(4))

Ja

i = 0;

Funktion step() auf der nächsten Seite

↓ Pkt. von eingeha

Martin Jacobsen HAW Hamburg Mat-Nr.: 1857660

Aufgabe 4c)

Funktion Step(i)

case i ?			
0	1	2	3
~~P0DR~~ P1DR = 0x09	P1DR = 0x05	P1DR = 0x06	P1DR = 0x0A

Martin Jacobsen HAW Hamburg Mat-Nr.: 1857660

Aufgabe 4d)


```
void step(int i) {
  switch(i) {
    case 0: P1DR=0x09;
            break;
    case 1: P1DR=0x05;
            break;
    case 2: P1DR=0x06;
            break;
    case 3: P1DR=0x0A;
            break;
  }
}
```

✓