Name, Vorname, Matr.Nr.: [redacted] 13. Juli 2007 1

HAW Hamburg - Prüfungsklausur Mikroprozessortechnik - SS 2007					
Aufg.	1	2	3	4	Summe
Pkt.	12+13=25	3 + 3+ 4=10	4 + 4 + 3 + 3=14	6+4+5+5+4+3+13+8+3=51	100
	18+1	9	9	42	78+1

79,

Aufgabe 1: Programmanalyse

a) Kommentieren Sie das nachfolgende Programm für das Laborsystem.

```
#include <mpp1.h>          /* include headerfile w. registernames */
void main(void)            /* Main-Funktion ohne Parameter und ohne Rückgabewert
 {
  P1DDR = 0xF0;            /* Richtung der Daten im 1. Port wird Festgelegt:
                              Bits 0-3 : Eingang, Bits 4-7 : Ausgang */

  TPU_TCNT1 = 0x0000;      /* Der 2. Timer wird resettet und auf 0 gesetzt */
  TPU_TCR1  = 0x40;        /* Eigenschaften des 2. Timers: Takt: CLK, steigende
  0100 0000                   Flanke, Timer clear method: Bei Eregnis von TGRB2, periodic mod
  TPU_TGR1A = 18431;       /* TGRA: 18432, also zählt der Timer 18432 Schritte, dies
                              entspricht: 18432 · 55µs = 1,01376 sec */
  TPU_TGR1B = (18432*3)-1; /* TGRB: 55296, also zählt der Timer 55296 Schritte, dies
                              entspricht: 3,04128 sec (Rechnung: 55296 · 55µs) */
  TPU_TSR1 &= 0xFC;        /* Das TGRA und TGRB Flag werden auf 0 gesetzt */
    1111 1100
  TPU_TSTR  = 0x02;        /* Der 2. Timer (2. Kanal des Timers) wird gestartet */

  while(1)                 /* Endlosschleife */
  {

   P1DR = 0x10;            /* Setzen des 1. Ports auf 00010000 */

   while((TPU_TSR1 & 0x01) == 0x00); /* Warten bis TGRA gesetzt ist */
   P1DR = 0x20;            /* Setzen des 1. Ports auf 00100000 */

   while((TPU_TSR1 & 0x02) == 0x00); /* Warten bis TGRB gesetzt ist */
   TPU_TSR1 &= 0xFC;       /* Flags werden wieder auf 0 gesetzt */
   P1DR = 0x40;            /* 1. Port auf: 01000000 */

   while((TPU_TSR1 & 0x01) == 0x00); /* Warten bis TGRA gesetzt ist */
   P1DR = 0x80;            /* 1. Port auf: 10000000 */

   while((TPU_TSR1 & 0x02) == 0x00); /* Warten bis TGRB gesetzt ist (auf eins) */
   TPU_TSR1 &= 0xFC;       /* Flags auf 0 setzen (TGRA, TGRB) */
  }
}
```

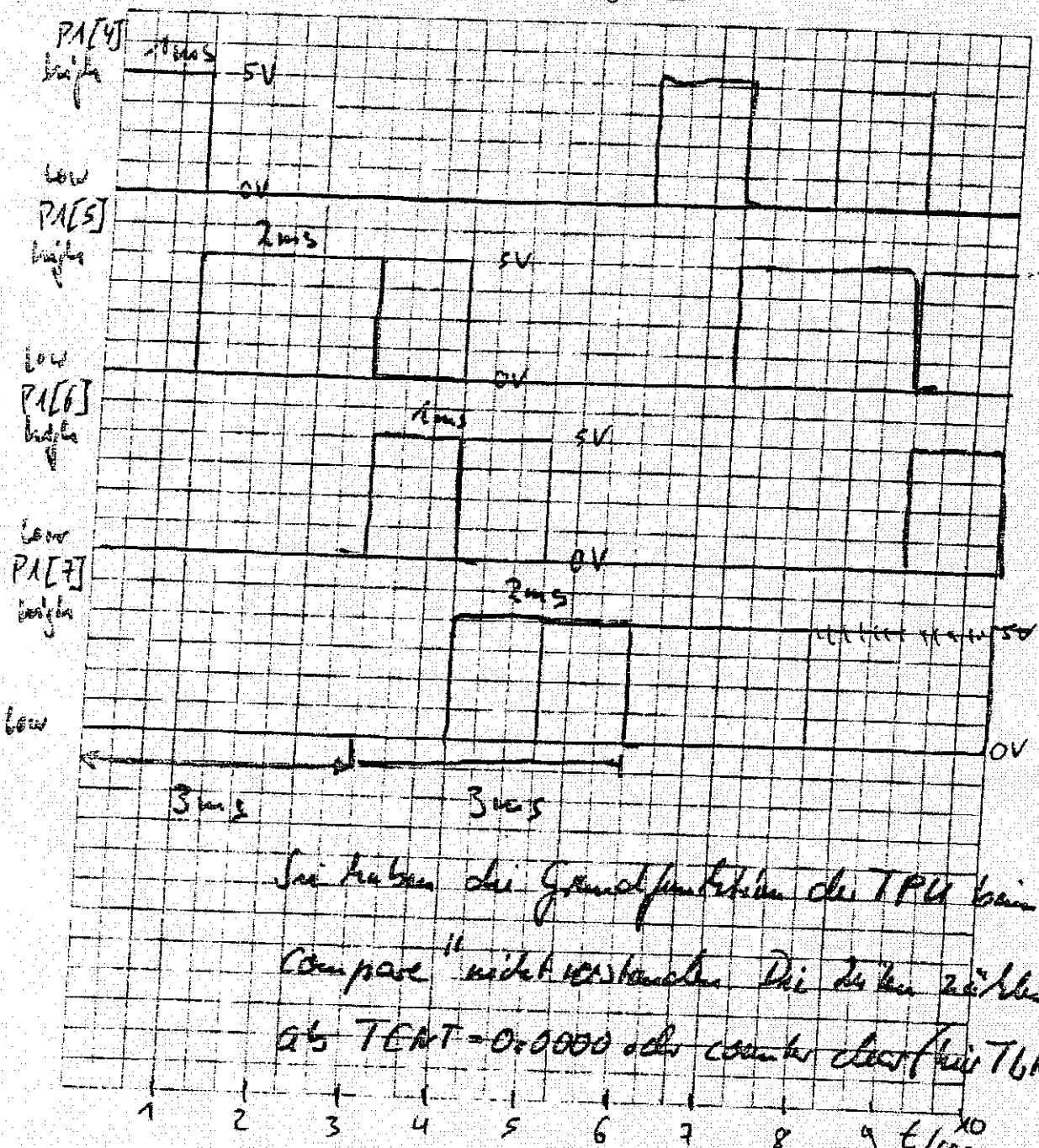
9
v. 19

① Besser: Channel 4 des Timers auf dem Board!

② Hier werden die Ausgangsbits belegt!

Name, Vorname, Matr.Nr.: [redacted] 13. Juli 2007 2

b) Skizzieren Sie die Ausgabe-Signale des Programms zuvor.
Beschriften Sie die x- und y-Achsen des Diagramms.



usw.

Sie haben die Grundfunktion der TPU beim „output Compare" nicht ausgenutzt. Die Zähler COUNT=0x0000 oder counter clear für TCRB wurde

~~Das PA[7] wird nicht auf 0 getest, da der Timer nicht gestoppt wird!~~

9 + 1

Da eine Endlosschleife im Programm ist, wird PA[4] nach 9 sec wieder auf high gesetzt! Ebenso jeweils also nach 6 (ms).

Aufgabe 2: Adressdecoder

Ein ROM-Chip und ein RAM-Chip sollen an einen Controller H8S/2357 angeschlossen werden. Beide Chips haben 1 Megabyte Kapazität (1M x8). Der ROM-Chip soll mit den untersten Adressen ab 0x00 0000 angesprochen werden, der RAM-Chip mit den nächsthöheren Adressen.

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

$\overline{CS}_{ROM}$ =

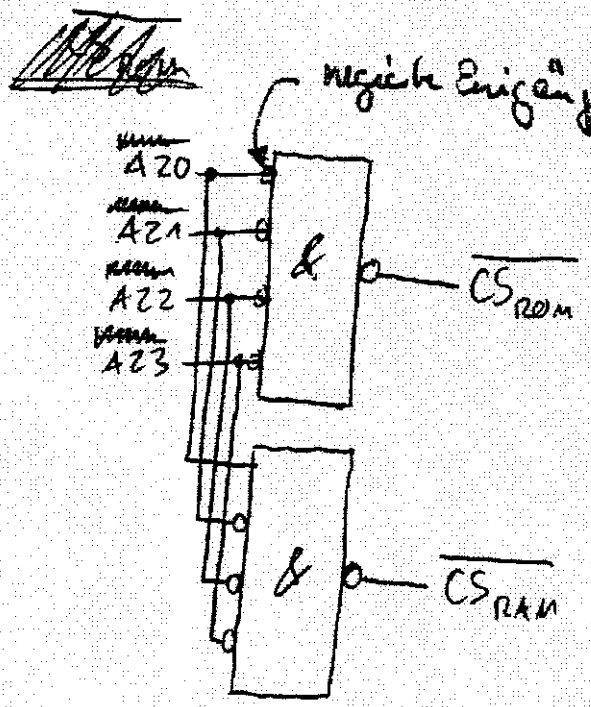
$A23 \vee A22 \vee A21 \vee A20$

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

$\overline{CS}_{RAM}$ =

$A23 \vee A22 \vee A21 \vee \overline{A20}$

c) Skizzieren Sie die Schaltung für die beiden Chip-Select-Signale. Sie haben entweder nur NAND-Gatter oder nur NOR-Gatter zur Verfügung.



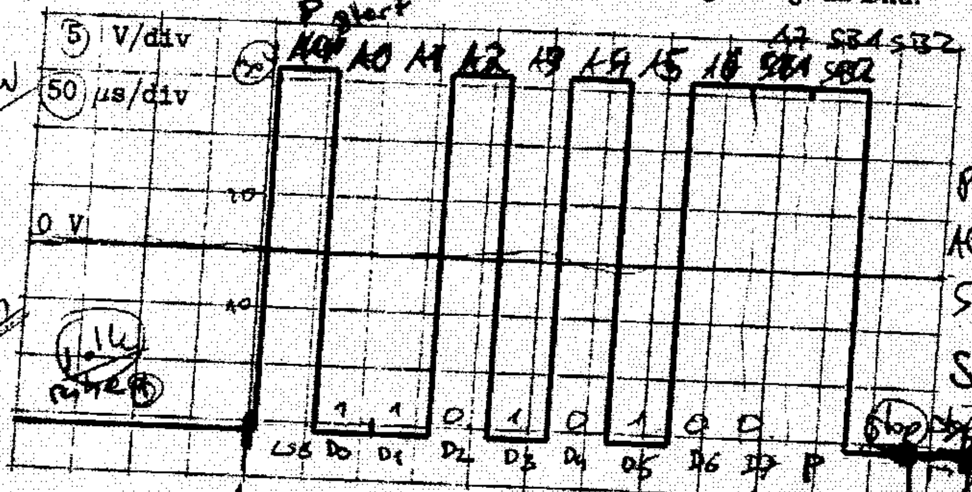
Ich leuchte hier und Gatter, da um das Zeichen für NOR nicht umfällt...!
Sie wollen aber Dipl.-Ing. ET werden?

FSR - Klausurensammlung | SS 07 | Semester E4 | Fach CT | Dozent RMS

Name, Vorname, Matr.Nr.: [redacted] ... *13. Juli 2007*

Aufgabe 3: Serial Interface

Das folgende Signal ist mit einem Speicheroszilloskop auf der Datenleitung eines RS232-Kabels gemessen worden. Hinweis: Beachten Sie die Zeit- und Spannungsanzeige im Bild.



P: Parität
A0 - A7: Daten
SB1-2: Stopbits (Stopbit(s))

a) Schreiben Sie die Namen für jedes Bit in das obige Bild. ✓

2

b) Welche der folgenden Standard-Bitraten wird genutzt ? (Bitte ankreuzen)

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

Anz. bit: 11
Periode 11 bit $T = 550\,\mu s$ $B_r = \frac{11\,bit}{550 \cdot 10^{-6}\,s} = 20000 \frac{bit}{s} \approx 19200 \frac{bit}{s}$

4

c) Welches serielle Protokoll wird benutzt ?

Protokoll (Kurzbezeichnung): 8 E ~~2~~ (Data / Parity / Stop)
even (gerade) 4x "1" => gerade => E

0

d) Welche Daten werden übertragen ? (Lassen Sie die zum Protokoll gehörenden Bits weg.)

Binär: ~~0001 0101~~ 0010 1011
Antw.: 0010 1011 ✓

Hexadezimal: ~~8U5~~ ~~0x35~~ 0x2B // 0x2B

ASCII Zeichen: +

3

9

ASCII Table, Hexadecimal Numbering (lower bits in Columns, upper in Rows)

	...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

Aufgabe 4: Programmieraufgabe

Ein elektronischer Würfel soll programmiert werden.

Schaltungsvorgaben (siehe Abb.4.1):

- Es soll der Controller H8S/2357 benutzt werden.
- Sieben rote LED sind an Port 1 angeschlossen.
- Eine grüne LED sind wie in Abb. 4. 1 dargestellt an den Pin TIOC5B angeschlossen.
- Zwei Tasten (Start, Stop)an Port 4 angeschlossen.

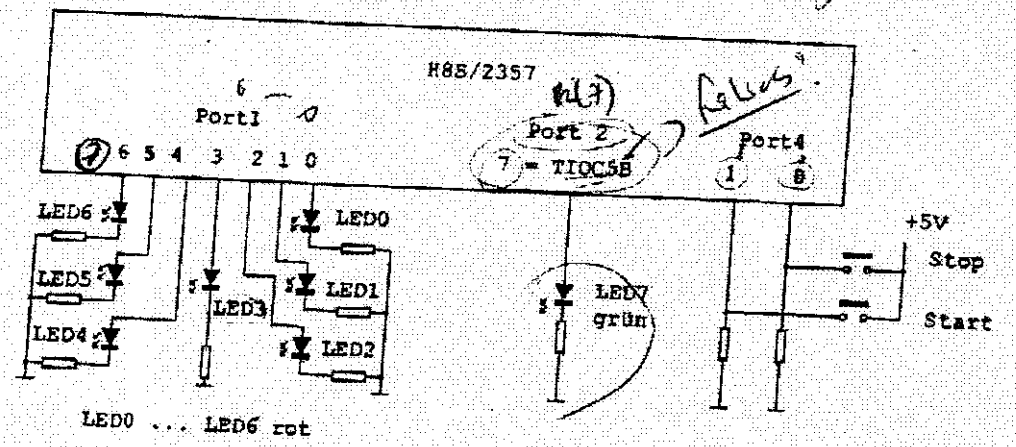


Abb 4.1: Schaltung des elektronischen Würfels

Funktionsweise:

- Nach dem Einschalten ist nur die grüne LED eingeschaltet, die roten LED sind ausgeschaltet.
- Mit der Taste Start wird der elektronische Würfel gestartet. Die grüne LED beginnt schnell zu blinken (250ms an, 250ms aus, ...).
- Es läuft dann im Programm eine zyklische Zustandsfolge 1,2,3,4,5,6, ...sehr schnell ab. Jeder Zustand dauert 10ms an. Nach dem Zustand 6 wird wieder der Zustand 1 erreicht.
- In jedem Zustand werden die roten LED mit den Augenzahlen 1 bis 6 ausgegeben (s. Abb.4.2)
- Mit der Taste Stop wird der elektronische Würfel angehalten. Die roten LED verbleiben im letzten Zustand stehen und die grüne LED wird dauernd eingeschaltet (Blinken beendet).
- Solange die Schaltung mit Spannung versorgt wird, läuft das Programm.

Hinweis: Arbeiten Sie die folgenden Aufgaben möglichst nacheinander ab. Bitte kommentieren Sie ausführlich und aussagefähig. Das Headerfile mpp1.h ist verfügbar.

Name, Vorname, Matr.Nr.: [redacted] 13. Juli 2007 6

a) Tragen Sie Portausgaben der Zustände 1 bis 6 in die Tabelle (Abb.4.2) ein.

Zustand (Augenzahl)	Anzeige x = LED an, z = LED aus	Portausgabe binär	Portausgabe hexadezimal	binär	hex
1	z6 z0 / z5 x3 z1 / z4 z2	0 0 0 0 1 0 0 0	0x08	0 0 0 0 1 0 0 0	0x08
2	z6 z0 / z5 x3 z1 / z4 x2	0 0 0 0 0 1 0 0	0x04	0 1 0 0 0 1 0 0	0x44
3	z z / z x z / z z	0 0 0 0 1 0 0 0	0x08	0 0 0 1 1 1 0 0	0x4C
4	x6 x0 / z x z / z4 z2	0 0 0 1 0 0 0 0	0x10	0 1 0 1 0 1 0 1	0x55
5	z z / z x z / x x	0 0 1 0 0 0 0 0	0x20	0 1 0 1 1 1 0 1	0x5D
6	z z / z x z / z z	0 1 0 0 0 0 0 0	0x40	0 1 1 1 0 1 1 1	0x77

Abb 4.2: Zustandstabelle des elektronischen Würfels

6 V.6

b) Programmieren Sie eine Funktion: `void state_out(int state_number)`
Sie soll nur die Portansteuerung der roten LED gemäß der Tabelle zuvor bewirken.

```
#include <mpp1.h>          /* include headerfile w. registernames */
void state_out(int state_number)
{
  switch (state_number)
  {
    case 1:
        P1DR = 0x08;
        break;
    case 2:
        P1DR = 0x44; break;
    case 3:      0x4C
        P1DR = 0x08; break;
    case 4:      0x55
        P1DR = 0x10; break;
    case 5:      0x5D
        P1DR = 0x20; break;
    case 6:      0x77
        P1DR = 0x40; break;
  }
```

/* Es wird geprüft welche Nummer */
/* ausgegeben werden soll und dann */
/* der PORT1 mit der jeweiligen */
/* Hex-Zahl eine aus belegt */

4 V.4

c) Programmieren Sie mit TPU Channel 2 eine Wartefunktion `void wait_ms(int time)`.
- Der Parameter time soll die Wartezeit in Millisekunden festlegen.
- Die Funktion soll minimal 2 und maximal 1000 Millisekunden warten.

```
void wait_ms(int time)
{
TPU_TCNT1 = 0x00;
TPU_TCR1 = 0x20; /* Es soll auf Ereignis von TGRA1 gewartet werden */
TPU_TGR1A = (time/10,055)-1; /* Berechnung der Dauer. Da time in msec übergeben wird und die Periodendauer bei 17,432MHz 55µsec beträgt, kürzt sich einmal 10^-3 raus und es bleibt der oben genannte Ausdruck. Die Datenverluste (wegen Konvertierung zu int) betragen nur wenige Prozent, sind also zu vertreten */
TPU_TSR1 &= 0xFE; /* Flag auf 0 setzen (TGFA) */
while ((TPU_TSR1 & 0x01) == 0x00);
/* Warteschleife, bis Timer hochgezählt hat */
TPU_TSTR = 0x00; /* zurücksetzen bzw. stoppen */
}
```

Randnotizen (Korrektur): Dies wäre clk/1! Dies wäre clk/1024! Edge?, prescaler; mehr; TPU_TSTR=0x02 /* 2. Timerchannel starten */; 2 (v. 8)

d) Programmieren Sie eine Funktion: `void start_green_blink (void)`
- Sie soll die grüne LED mit dem Signal TIOC5B des TPU channel 5 ansteuern.
- Die Funktion soll nicht warten, sondern nach dem Start des Blinkens zurückkehren.

Randnotizen (Korrektur): 1) Welcher Prescaler? clk/1024 ist nicht möglich bei Ch. 1; f channel 4 statt 5!

```
void start_green_blink(void)
{
TPU_TCNT4 = 0x00;
TPU_TCR4 = 0x40; /* warten auf Ereignis von TGRB4 */
TPU_TGR4B = 4544; /* 250ms / 55µs = 4545,45... */
TPU_TSR4 &= 0xFD; /* Flag auf 0 setzen TGRB4 */
TPU_TSTR = 0x40; /* Start vom 5 Timer-chanel */
while ((TPU_TSR4 & 0x01) == 0x00);
/* Warten bis hochgezählt */
}
```

Randnotizen (Korrektur): f Dies wäre clk/1. Dies clk/1024? falsch; Dies startet gar keinen TPU channel! (Es sind nur ch0 ... ch5 = 6 ch. vorhanden); 1 (v. 5)

Name, Vorname, Matr.Nr.: [redacted] 13. Juli 2007 8

e) Programmieren Sie eine Funktion: `void stop_green_blink (void)`
- Sie soll das kontinuierliche Leuchten der grünen LED bewirken. (Ende des Blinkens)

```
void stop_green_blink(void)
{
  TRU_TSTR=0x00;
  P2DR=0x80;
}
```

Stop alle Timer
Bsp: 8 = ~0x20
/* hier grün stellen */

① TPU_TSTR = TPU_TSTR & 0x1F;
② P2DR = 0x80;
P2(7) high → LED grün ✓

4 v.4

f) Programmieren Sie eine Funktion: `int test_key(void)`
- Diese Funktion soll '1' zurückgeben, wenn die Taste Start gedrückt wurde.
- Diese Funktion soll '2' zurückgeben, wenn die Taste Stop gedrückt wurde.
- Diese Funktion soll '0' zurückgeben, wenn keine der beiden Tasten oder beide Tasten gedrückt wurden.

```
int test_key(void)
{
  while((PORT4&0x3)==0x3);
  switch (PORT4)
  {
    case 0x01:
      return 1;
    case 0x02:
      return 2;
    default:
      return 0;
  }
```

5 v.5

Name, Vorname, Matr.Nr.: [redacted] 13. Juli 2007 9

g) Programmieren Sie die Funktion: `void main(void)`
- `Main()` soll die zuvor programmierten Funktionen benutzen.
- Bitte benutzen Sie möglichst Schleifen und keine `switch / case` Anweisung.

```
void main (void)
{
  R1DDR = 0xFF;
  P7DDR = 0x80;
  /* PORT 4 ist auf „read-only" eingestellt */
  int key, i;
  stop_green_blink();  /* Kontrolliertes, grünes Leuchten */
  while (1)
  {
    if( test_key() == 1)   /* start */
    {
      do {
        for(i=0; i<8; i++)
          state_out(i);
      } while(test_key() != 2);
    }
    run_out(i);
}
```

siehe beiliegender Zettel (22)

h) Damit es noch ein wenig spannender wird, soll der Würfel langsam 'auslaufen'.
- Ergänzen[1] Sie die Funktion `main()` durch einen weiteren Funktionsaufruf dafür.
- Nach der Taste `Stop` werden von einer Funktion `void run_out(int last_state)` noch 7 folgende Zustände ausgegeben.
- Bei diesen Zuständen wird die Wartezeit jeweils verdoppelt (also 20ms, 40ms, 80ms, 160ms, 320ms, 640ms);
- Programmieren Sie diese Funktion, bitte wieder mit Schleifen.

Ich benutze TPU-channel 3. siehe beiliegendes Blatt (Z3) warum? dort dr 2

```
void run_out(int last_state)
{  int i;
   TPU_TCNT2 = 0x00;
   ~~TPU_TCR2 = 0x20;~~   /* ... Flag TGR2A warten */
   TPU_TGR2A = 3?4;        /* entspricht 20ms / 5?µs */
   TPU_TSR2 = 0xFE;        /* Flag auf 0 setzen */
   ~~TPU_TSTR = 0x04;~~    /* channel 3 starten */
   [durchgestrichen]

   for(i=0; i<6; i++){  TPU_TSTR=0x04;  /* channel 3 starten */
      TPU_TGR2A *= 2;   /* Vielfache von 20ms warten */
      while((TPU_TSR2 & 0x01) == 0x00);
      TPU_TCNT2 = 0x00;   /* resetten */
      TPU_TSTR = 0x00;    /* und stoppen */
      ~~state_out(last_state+i);~~  /* (1) */
   }
}
```

7 / v. 8

i) Diskutieren Sie in wenigen Stichpunkten die möglichen/notwendigen Änderungen des Programmablauf, wenn die Taste `Stop` an eine IRQ Leitung angeschlossen wäre.

1. Es muss ein Interrupt initialisiert werden für eine IRQ-Leitung → Stop-gen-state(); (1) run_out(...) ✓
2. Im Interrupthandler wird das „Auslaufen" und das stoppen des grünen Blinkens gemacht. ✓

2 / v. 3

42 / v. 51

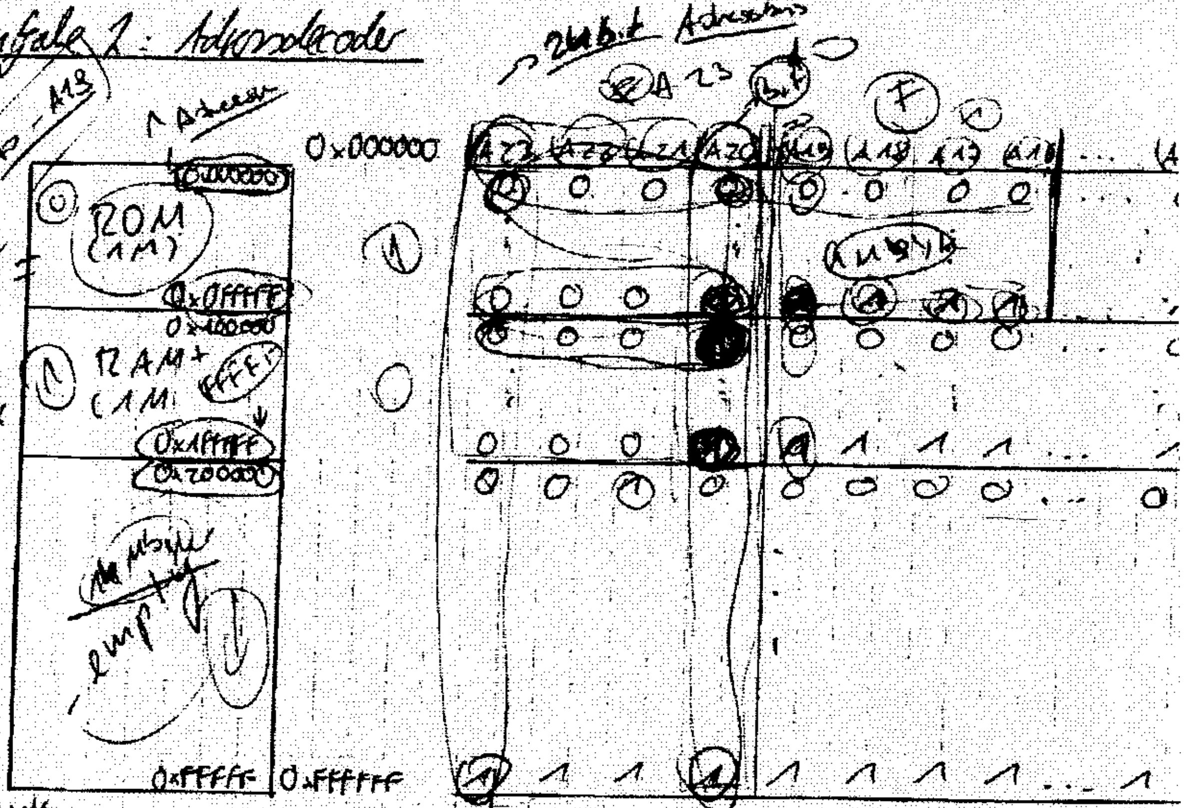
[1] Setzen sie einfach eine Einfügemarkierung.

(1) Da der Würfel ab der Zahl langsamer werden soll, wo er beim stop stand wird hier separat die Funktion aufgerufen. ✓

FSR - Klausurensammlung
Semester: E4 | Fach: CT | Dozent: RMS | 07

Klausur, Mikroprozessortechnik 13.7.0

Aufgabe 1: Adressdecoder



ROM: $1M(\times 8) = 2^{10} \cdot 2^{10} = 2^{20} = A0 - A19$

RAM: $1M(\times 8) = 2^{10} \cdot 2^{10} = 2^{20} = A0 - A19$

Wichtig sind in diesem Fall die Stellen A23 - A20, da diese sich im ROM und RAM entscheidend unterscheiden, außerdem gehen beide Bausteine bis A19.

$CS_{ROM} = \overline{A23} \wedge \overline{A22} \wedge \overline{A21} \wedge \overline{A20}$

$\overline{CS_{ROM}} = A23 \vee A22 \vee A21 \vee A20$

$CS_{RAM} = \overline{A23} \wedge \overline{A22} \wedge \overline{A21} \wedge A20$

$\overline{CS_{RAM}} = A23 \vee A22 \vee A21 \vee \overline{A20}$

o.k. vom Grundschaltbild

Z1

[redacted]

Klausur: Mikroprozessortechnik 2

13.7.07

WS / SS	Semester	Fach	Dozent
07	E4	CT	RMS

FSR - Klausurensammlung

Aufgabe 4)

```
void main (void)
{ int i;
  P1DDR = 0xFF; // P1DR = 0x00; ✓   (kurz: 0x7F;)
  P2DDR = 0x80; ✓
  /* PORT 4 ist voreingestellt auf read only */ ✓
  Stop_green_blink(); // kontinuierliches, grünes, Leuchten
  while(1) /* Solange Spannung da ist... */
  {
    if (test_key() == 1)   /* Start */
    { start_green_blink();
      do {
        for (i=1; i<7; i++)  ← Zähler
        {
          state_out(i);
          wait_ms(10); /* 10ms warten */
        }
      } while (test_key() != 2);
    }
    rem_out(i);
    stop_green_blink();   ← Klammer
  }
}
```

nicht mehr!

[redacted]

13 v. 13 P.

voll empfehlen

(22)

4h)

```
void run_out( int last_state)
{
   int i, temp;
   TPU_TCNT2 = 0x00;
   TPU_TCR2  = 0x20;   /* bei Flag TGR2A counten */
   TPU_TGR2A = 364;    /* entspricht 20ms / 55µs */
   TPU_TSR2 &= 0xFC;   /* Flag auf Null */
   for( i=0; i<6; i++)
   {
      TPU_TSTR = 0x04  /* channel 3 starten */
      TPU_TGR2A *= 2;  /* immer verdoppeln */
      while((TPU_TSR2 & 0x01) == 0x00);
      TPU_TCNT2 = 0x00; /* resetten */
      TPU_TSTR = 0x00;  /* stoppen */
      if( i + last_state > 6)      /* schauen, ob */
         temp = (i + last_state) - 6;  /* dann Zahl zurück */
      else                         /* ab 6 ist usw. */
         temp = i + last_state;
      state_out (temp);            /* separate Ausgabe */
   }                               /* und ändern des PORT1 */
}
```

7 v. 8. vom Dipl-Ing