FSR - Klausurensammlung
08 / WS | Semester: E4 | Fach: CT | Dozent: RMS

Name, Vorname, Matr.Nr.: Aldag, Mario, [redacted] ..... 15. Juli 2009 1

HAW Hamburg - Prüfungsklausur Computertechnik - SS 2009					
Aufgabe	1	2	3	4	Summe
Punkte	??	??	??	??	??
	14	25	18	26	

A3

1 Adressbereiche

Summe P3 13 v 15 P.

An einen Controller H8S/2357 sind ein ROM-Chip und ein RAM-Chip angeschlossen. Die Adressdecoder-Schaltungen in Bild 1 und in Bild 2 sind einem Gesamtschaltbild entnommen.

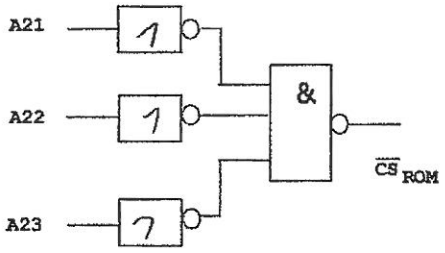


Bild 1: Adressdecoder ROM

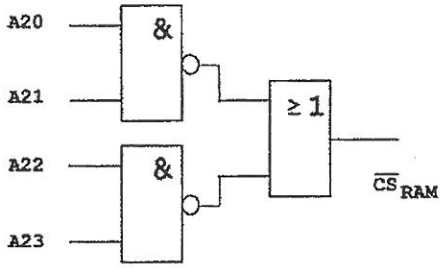


Bild 2: Adressdecoder RAM

a) Nennen Sie die Gleichung des Adressdecoders des ROM, vereinfachen Sie diese.

$$\overline{CS_{ROM}} = \overline{\overline{A_{21}} \wedge \overline{A_{22}} \wedge \overline{A_{23}}} = A_{21} \vee A_{22} \vee A_{23}$$

4

b) Nennen Sie die Gleichung des Adressdecoders des RAM, vereinfachen Sie diese.

$$\overline{CS_{RAM}} = \left(\overline{A_{20} \wedge A_{21}}\right) \vee \left(\overline{A_{22} \wedge A_{23}}\right)$$

$$= \overline{A_{20}} \vee \overline{A_{21}} \vee \overline{A_{22}} \vee \overline{A_{23}}$$

4

c) Nennen Sie die Kapazität (in MB) des ROM und des RAM-Chip und des freien Bereichs sowie die dazu passende Anfangs- und Endadresse (hexadezimal).

5

Chip / Bereich	Kapazität	Anfangsadresse	Endadresse
ROM	$2^{21} = 2MB$	0x00 0000	0x1 F FFFF
RAM	$2^{20} = 1MB$	0x20 0000 1	0x2 F FFFF 1
Frei	13MB	0x300000 1	0xFF FFFF 1

Aufgabe 2: Drehzahlmessung

SS / WS	Semester	Fach	Dozent
OS	E4	CT	RMS

FSR - Klausurensammlung ?/9

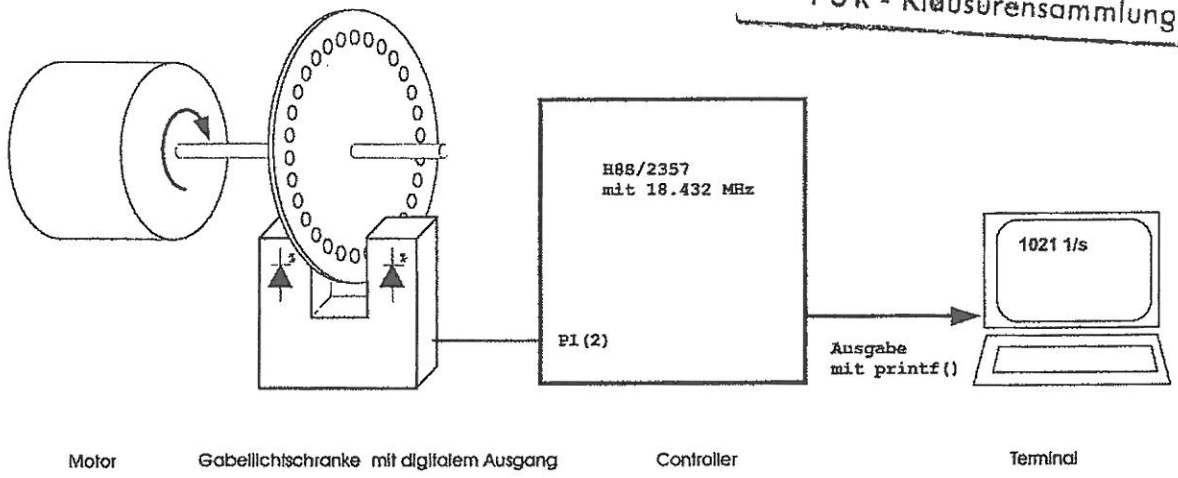


Bild 3: Drehzahlmessung (Erfassung der Encoderpulse am Port 1 Pin 2)

Die Drehzahl eines Motors soll aus Sicherheitsgründen gemessen und an ein Terminal ausgegeben werden.
Auf der Drehachse des Motors ist dazu eine Encoder-Scheibe mit 30 Löchern montiert. Eine Gabellichtschranke liefert bei Lichtdurchtritt durch ein Loch der Encoder-Scheibe ein High-Signal. Über die Zahl der Pulse pro Zeiteinheit kann die Drehgeschwindigkeit bestimmt werden.
Die Ausgabe der Umdrehungsgeschwindigkeit erfolgt mittels der Funktion `send_actual_speed( unsigned int turns_per_second )`. Die Einheit des übertragenen Wertes ist $1/s$ (Umdrehung pro Sekunde). Zu Entwicklungszwecken erfolgt einfach eine Ausgabe per `printf()` auf einem Terminal.
Die Drehzahl des Motors soll gemessen werden und im Abstand von 1 s übertragen werden. Der Messzeitraum ist entsprechend jeweils 1 s.
Schreiben Sie ein Programm, welches die Impulse der Lichtschranke zählt und den Wert im Abstand von 1 s überträgt. Verwenden Sie für die Zeitmessung den TPU Channel 3. Das Sensorsignal ist verbunden mit Port 1 Pin 2.

a) In welchem Modus wollen Sie die TPU3 betreiben?

~~Input Capture~~ Output Compare ✓
~~⇒ TPU_TCR3A = 0x10~~ ~~TPU_TCR3 = 0x~~

1

b) Auf welche Weise sind 60 Bewegungsinformationen bei 30 Löchern bei einer Encoder-Umdrehung möglich? Wieso spricht man hier von einer 60er Teilung des Encoders?

Der Encoder gibt "High" und "Low" Pulse aus. Daraus sind 60 Infos pro Sekunde ✓

1

c) Berechnen Sie den Teilungsfaktor (Prescaler) und den zugehörigen Compare-Wert (Schwellwert) für den TPU Channel 3!

Prescalar			Ticks/µs	Ticks/µs
1	18,432 MHz	54,2535 ns		184,19905
1024	18000 MHz	55,55 µs		18001,80
4096	4500 Hz	222,22 µs		4500,045
256	72000 Hz	13,88 µs		72046,109

Es wird Scalar 1024 verwendet. Ticks/µs: 18000,0

4

Name, Vorname, Matr.Nr.: Aldag, Mario, [redacted] 15. Juli 2009 3

d) Wie hoch ist die maximal messbare Drehzahl, wenn eine Zählvariable vom Typ unsigned int verwendet wird?

~~Int~~ int auf H8S/2359 entspricht 16bit.

$\Rightarrow 2^{16} = 65536$ []

Teiler? Drehzahl!

01

e) Erstellen Sie das Programm mit genauen Kommentaren. Die Funktion send_actual_speed() ist gegeben. Achten Sie darauf, dass stets eine Übertragung im Abstand von einer Sekunde stattfindet.

```
#include <mpp1.h> //Einbinden des H8S/2359 Header
#include <stdio.h>// Für printf berechtigt
                  speed
void send_actual_speed( unsigned int ~~turns_per_second~~ ){ printf("%i\n", speed); };
int status=0; // Variable um Status High und Low zu unterscheiden
void main( void )
{                                   etwas mit außen
  unsigned int speed = 0;
  while(1){ // Endlosschleife für Programm
  // Timer einrichten
  TPU_TSTR = 0x00; // Timer Start Register mit Nullen
                      initialisieren
  TPU_TCR3 = 0x25; // Control Register von Ch3: Prescaler ist
                      1024; Counts on rising edge; Counter
  TPU_TSR3 &= ~0x03; clear by TGRA compare match
                      // Flags rücksetzen
  TPU_TIOR3H = 0x00; // Output compare, keine Ausgabe
  TPU_TGR3A = 18001-1; // 18001 Ticks à 55,55µs ≙ 1s
  TPU_TCNT3 = 0x00; // Ch3 auf 0 setzen
  TPU_TSTR = 0x08; // Ch3 starten
  while((TPU_TSR3 & 0x01) == 0){
    // Der Timer zählt. Diese Schleife zählt nun die Ticks
    auf High von P1(2) in int speed
```

P1DDR = 0x00; // Port 1 nur lesen

7

3

15 / WS	Semester	Fach	Dozent
03	E4	cT	RMS

FSR - Klausurensammlung 4/9

Name, Vorname, Matr.Nr.: Aldag, Mario, [redacted] 15. Juli 2009 4

```
if((PORT1& 0x04)==1) // Pin 2 an Port 1 ist High
{
  speed += 1; // Zähler um eins erhöhen.
}
} // while schleifen

if(status==0 && ((PORT1& 0x04)==1)) // Pin 2 auf PORT 1 ist High wieder
{
  speed += 1; // Zähler um 1 erhöhen
  status = 1; // Merken, dass es nun high ist.
}
if(status==1 && ((PORT1& 0x04)==0)) // Pin 2 auf PORT 1 wieder low
{
  speed += 1; // Zähler erhöhen
  status = 0; // Merken, dass nun low ist
}
} // Timer hat fertig gezählt
// Umdrehung/s ausrechnen
speed = speed/60; // Nur ganzzahlige Umdrehungen ausrechnen.
send_actual_speed(speed);
}
}
```

8

Flags clear?

18 / 20

25 / 25

Name, Vorname, Matr.Nr.: Aldag, Mario, [redacted] ..... 15. Juli 2009 5

Aufgabe 3: Fragen

a) Mit einem ADC soll bei 10 Bit Auflösung Spannungen im Bereich von 0 V bis 4 V gemessen werden. Welcher Spannung U_{LSB} entspricht ein Bit (LSB) ?

$$U_{LSB} = \frac{4V}{2^{10}} = 3{,}90625\,mV$$ ✓

2

b) Der ADC hat eine Umsetzdauer (im Scan-Mode) von 12,5μs bei 10,24 MHz Takt. Es werden 4 Kanäle abgetastet. Welche Abtastfrequenz (Sample Rate) wird maximal pro Kanal erreicht?

50µs, da die Kanäle nacheinander abgearbeitet werden.

$\Rightarrow 1/50\mu = 20\,kHz$

2

c) Kann der ADC des H8S/2357 im Scan-Mode mehrere Messwerterfassungen zeitsynchron durchführen ? Wenn Sie ja antworten, welche Besonderheit müssen Sie berücksichtigen ? Wenn Sie nein antworten, nennen Sie Ursache.

Ja, die Umsetzzeit ist erst beim zweiten Durchlauf konstant. Bei CKS = 0 => 13,889µs ; CKS = 1 => 6,944µs
Richtig, aber hier nicht der Grund.
=> siehe Multiplexer am Eingang

d) Das 10Bit breite Ergebnis des ADC des H8S/2357 kann nur in 8Bit breiten Teilwerten ausgelesen werden. Warum muss das höherwertige Byte zuerst gelesen werden?

Die Daten aus ADDRnL könnten fehlerhaft sein, wenn diese zuerst gelesen wurden. Das ADDRnH zu lesen verursacht keine fehlerhaften Daten. => fehlerhaft ist n. Konvert. => von „falschen" d.h. nicht den zeitlichen Zuordnungen der Kanäle

1

e) In der Vorlesung haben wir die asynchrone Übertragung der seriellen Schnittstelle detailliert behandelt. Erklären Sie den Unterschied zwischen synchroner und asynchroner Übertragung.

Bei asynchroner Übertragung wird kein Takt benötigt. Dort wird nur die Bitrate, Anzahl von Stoppbits und Parität even/odd/none eingestellt. Dadurch wird ein Frame angelegt und der Empfänger kann innerhalb dieses Frames die gesendeten Daten empfangen.
Bei synchroner Übertragung wird ein Takt mitgesendet, sodass der Empfänger und Sender Takt synchron arbeite

3

f) Die serielle Schnittstelle SCI kann mit verschiedenen Protokollparametern und mit verschiedenen Bitraten arbeiten. Wir benutzen das Laborsystem mit 18.432 MHz. Welche Einstellungen für Protokoll 8N2 und der Übertragungsrate 7200 Bit/s wählen Sie? Geben Sie den gut kommentierten C-Code an.

Für Ch2 →

```
SCI2_SMR=0x00; // 8 Datenbit, kein Parity, 2 Stoppbit, normaler Clock
SCI2_BRR=79; // 18,432 MHz/32/80=7200 bit/s
             daher muss 79 eingetragen werden.
~~SCI2_BS~~
SCI2_SCR=0x10; // Empfangen anschalten.
               nicht gefragt
```

4

g) Betrachten Sie die Einstellungen von Aufgabe f) nochmals, gibt es alternative (zulässige) Einstellungen? Geben Sie diese an!

Prescalar	Takt	BRR+1	(Bit/s) = 18,432 / 32
1	18,432 MHz	80	7200
4	4,608 MHz	20	7200 } ok
16	1,152 MHz	5	7200
64	288 kHz	1,25	7200 ← geht nicht

4

f) Wie viele ASCII-Zeichen sind pro Sekunde mit dem Protokoll 8N2 und der Übertragungsrate 7200 Bit/s maximal zu empfangen oder zu senden?

ASCII entspricht 1 Byte.
7200 Bit/s enthält 8 Bit Daten + 2 Stopp bit + 1 Startbit
⇒ 720 Zeichen ~~dit~~ pro Sekunde

2

18

Weiss Musste o.k.

Fach C7 | Semester E4 | Dozent: RMS | SS / WS 05 | FSR - Klausurensammlung 6/9

FSR - Klausurensammlung
Semester E4 | Fach CT | Dozent RMS | SS / WS 09

Aufgabe 4: Wetterstation

In einer kleinen Wetterstation werden die Innen- und Außentemperatur mittels eines H8S/2357 gemessen. Die beiden Temperaturen werden durch Temperatursensoren in Spannungen zwischen 0V und 4V umgewandelt. Die Referenzspannung V_{ref} ist 4 V.
Der Innentemperatursensor ist an AN0 (entspricht P4(0)) angeschlossen und die Außentemperatursensor an AN1 (entspricht P4(1)). Die Temperaturen sollen nach jeweils 3 s auf zweistelligen 7-Segment Anzeigen mit BCD-Code ausgegeben werden (Bild 5). Das Vorzeichen der Temperaturen wird durch P1(7) bzw. P2(7) an eine passende Anzeige ausgegeben, wobei die Ausgabe eines Wertes '1' an diesen Pins das Vorzeichen „minus" anzeigt.
Die Umrechnung von Spannung in Temperatur ist vorgegeben $\theta = -20°C + U_{in} \cdot \frac{20°C}{1V}$.

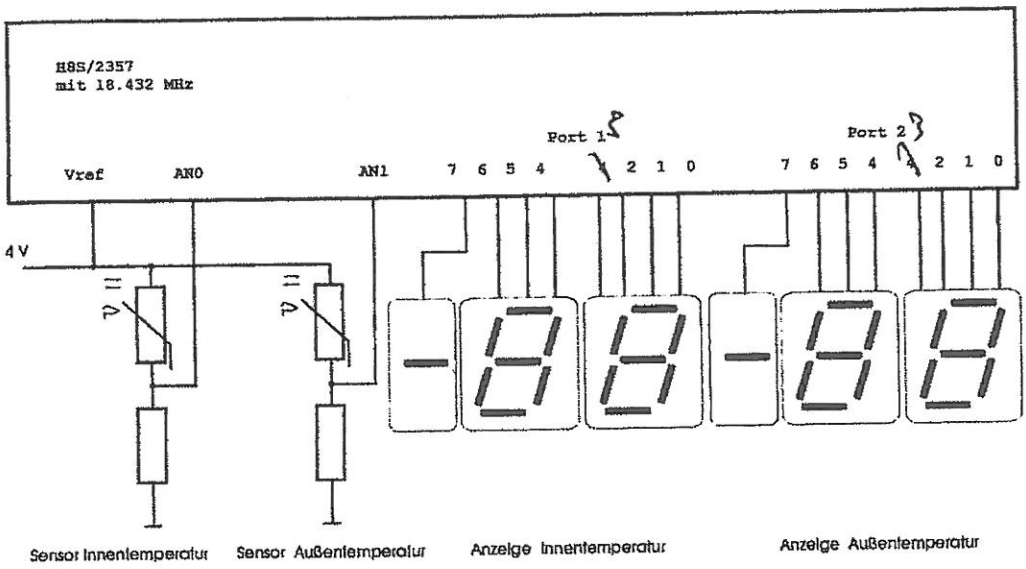


Bild 5: Schaltung der Wetterstation

a) Die Zeit von 3 s soll mit der TPU realisiert werden. Ermitteln Sie den Teilungsfaktor (Prescaler) sowie den erforderlichen Vergleichswert bei einem Systemtakt von 18,432 MHz.

Presc.	Takt	Scalar	Ticks für 3s	
4	18,432 MHz	1024	54000	← wird gewählt, da Auflösung höher
	-"-	4096	13504	

b) Schreiben Sie ein Programm, welches die TPU und den ADC initialisieren, die Temperaturen im Scan-Modus messen und zum richtigen Zeitpunkt ausgeben. Die Temperatur soll ohne Nachkommastelle in Grad Celsius ausgegeben werden.

Hinweis: Verwenden Sie möglichst nur Ganzzahlarithmetik und ausreichende Datentypen.

SS / WS	Semester	Fach	Dozent
05	E4	CT	RMS

FSR - Klausurensammlung 8/9

```
#include <mpp1.h>
#include <stdio.h>

void main( void )
{ int temp0; int temp1; //Temperaturen
  short an0; short an1; // Zahlwerte des ADU
  P2DDR = 0xFF;
  P1DDR = 0xFF;
  char bcd_low; char bcd_high; // für BCD Code
  while (1){
    //Timer initialisieren
    TPU_TSTR = 0x00; //Timer Start Register mit Nullen füllen
    TPU_TCR3 = 0x25; //TGRA clear by compare match; count on rising edge; Clock/1024
    TPU_TSR3 &= ~0x03; //Flags zurücksetzen nur TGFA nötig
    TPU_TIOR3H = 0x00; // output compare, keine Ausgabe
    TPU_TGR3A = 54000; //Ticks: 54000 · 55,55µs ≙ 3s
    TPU_TCNT3 = 0x00; //Ch3 auf 0 setzen

    //ADU einstellen
    ADCSR |= 0x 41; // Scan Mode; AN0, AN1 werden genutzt

    //ADU starten
    ADCSR |= 0x 20;
    //Timer starten
    TPU_TSTR3 = 0x08; //Ch3 Start
    while((TPU_TSR3 & 0x01) == 0)
    { Timer läuft nun 3 sek.
      while((ADCSR & 0x 80) == 0);
      { //EU Ch0 und Ch1 werden nun umgesetzt, Ergebnisse verarbeiten.
      an0 = ADDRAH << 8; // Hohes Register auslesen und linksschie
      an0 |= ADRRAL; // niedriges Register dazu lesen
      an0 = an0 >> 6; // alles nach rechts schieben
```

Name, Vorname, Matr.Nr.: Ndeus, Mario, [redacted] 15. Juli 2009 9

SS / WS	Semester	Fach	Dozent
09	E4	CT	RMS

FSR - Klausurensammlung 9/9

```
an1 = ADDRBH<<8;   //S.0.
an11 = ADDRBL;     //S.0.
an1 = an1>>6;      //S.0.
                   //S.0.
```

besser beide Werte * 10 statt 39L dann genauer 39,06

~~temp0 = -20 + an0.~~

~~temp0 = -20+(an0/1000)~~

temp0 = (int)(-20+ (an0 · 39L ·20)/10000); // Umrechnung in Temperatur

// Wertebereich:
$an0 < 2^{10}$; $2^{10} \cdot 39L \cdot 20 = 798720$ // durch L hinter 39 konversion auf Long.

$\frac{4V}{2^{10}} \cdot \frac{1}{10} \cdot 10 = 39$

temp1 = (int)(-20+(an1 · 39L ·20)/1000);

} // Timer hat fertig gezählt. S.0

ADCSRA = 0x20; // ADU stoppen nicht nötig!

//Ausgabe nun
//temp0 hat Temperatur links, temp1 rechts

~~if(PORT1&~~

~~if((temp & 0x80) == 1)~~

if(temp0<0)
{
P1DR = 0x80; //Minus Zeichen ausgeben
} else { P1DR = 0x00; } //kein Minus ausgeben

~~bcd_high =~~

bcd_low = temp0 % 10; // Hintere Stelle rausnehmen

bcd_high = (temp0 - bcd_low)/10; // Hohe Stelle rausnehmen

P1DR |= bcd_low + (bcd_high << 4);

if(temp1<0){
P2DR = 0x80; //Minuszeichen ausgeben ok.
} else { P2DR = 0x00; } //kein Minus ausgeben
bcd_low = temp1 % 10;
bcd_high = (temp1 - bcd_low)/10;
P2DR |= bcd_low + (bcd_high << 4);

}

}

23

26

Prinzip OK

HAW Hamburg / Department IuE	Datum: 29.01. 2015
Fachgruppe: Prozessortechnik	Prüfer: Prof. Dr.-Ing. Kölzer
Klausur: [redacted] E4b/CT	
Name, Vorname:	Matr.Nr:

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5		Summe	Note
Punkte	10	28	20	46	6		100+	0..15
Erreichte Punkte	7	21	12	32	0		72	11

10.02.15

Beachten Sie: Beginnen Sie bitte jede Aufgabe auf einem neuen Blatt und nummerieren Sie die Blätter. Es werden nur die Lösungen anerkannt, deren Lösungswege eindeutig erkennbar und nachvollziehbar sind.

Erlaubt: alle schriftlichen Unterlagen bis auf alte Klausuren

Nicht erlaubt: PC, Laptop, Handy, rote Stifte

Jahr	Semester	Fach	Dozent
2014 WS	E4b	CT	KZR

FSR - Klausursammlung

Aufgabe 1: (Adresscodierung – 10 Punkte)

An einen ARM-Cortex-Controller (32 Bit Adressbus) werden drei Speichermodule RAM, RAM1, RAM2 angeschlossen. Der Speicher ist byteweise organisiert. Die beiden höchstwertigsten Adressbits Bit A31 und A30 werden von den Adressdekodern der Module in folgender Weise benutzt:

$$\overline{CS_{ROM}} = \overline{A31}$$

$$\overline{CS_{RAM1}} = \overline{A30 \wedge \overline{A31}}$$

$$\overline{CS_{RAM2}} = \overline{\overline{A30} \wedge \overline{A31}}$$

a) Bestimmen Sie die Anfangs- und die Endadressen der drei Module (hexadezimal).

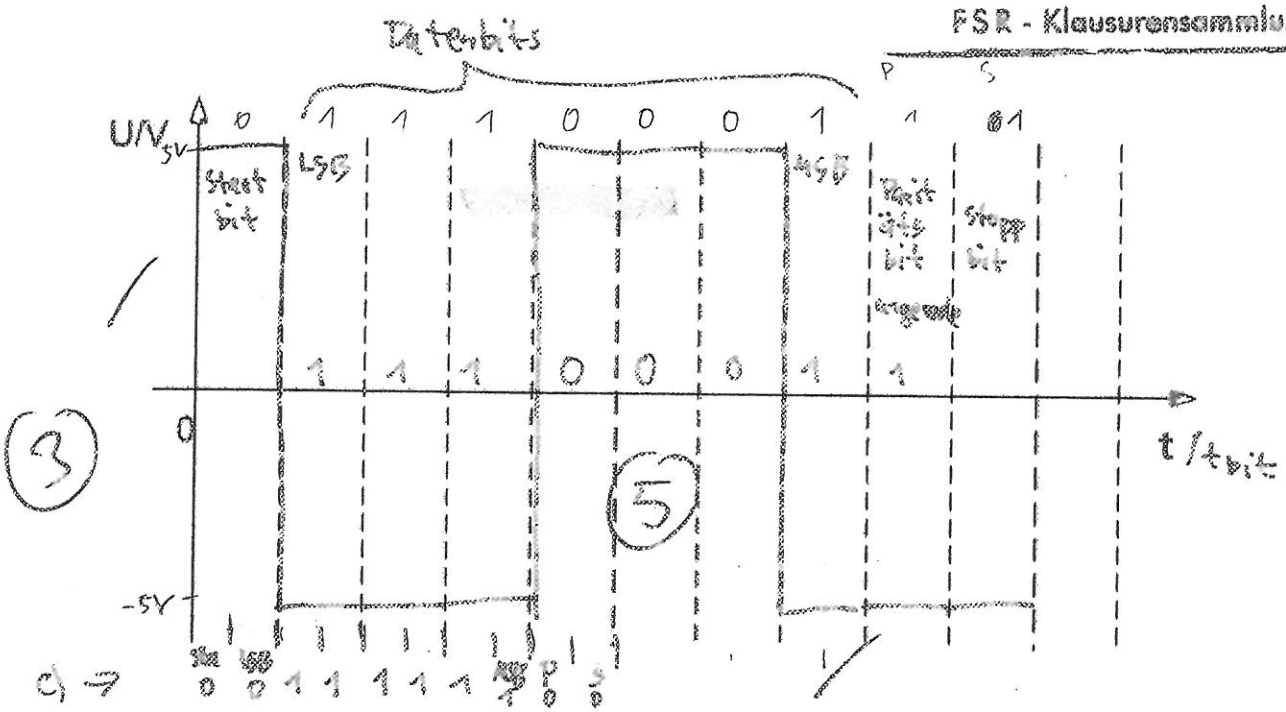
b) Wie groß ist jeweils die Speicherkapazität (in MB) der drei Module?

c) Zeichnen Sie die Memory-Map des Controllers mit den drei Modulen, inkl. Adressangaben.

Aufgabe 2: (UART / RS232 - 28 Punkte)

Das UART2-Modul des Controllers LM3S9B92 sendet periodisch das ASCII-Zeichen ‚G' (0x47). Das asynchrone Datenformat enthält 7 Datenbits, ein Paritätsbit für ungerade Parität und ein Stoppbit. Die Baudrate ist auf 19200 bits/s eingestellt.

a) Skizzieren Sie das RS232-Signal des Datenrahmens. Kennzeichnen Sie alle Bits und geben Sie eine korrekte Skalierung beider Achsen an.

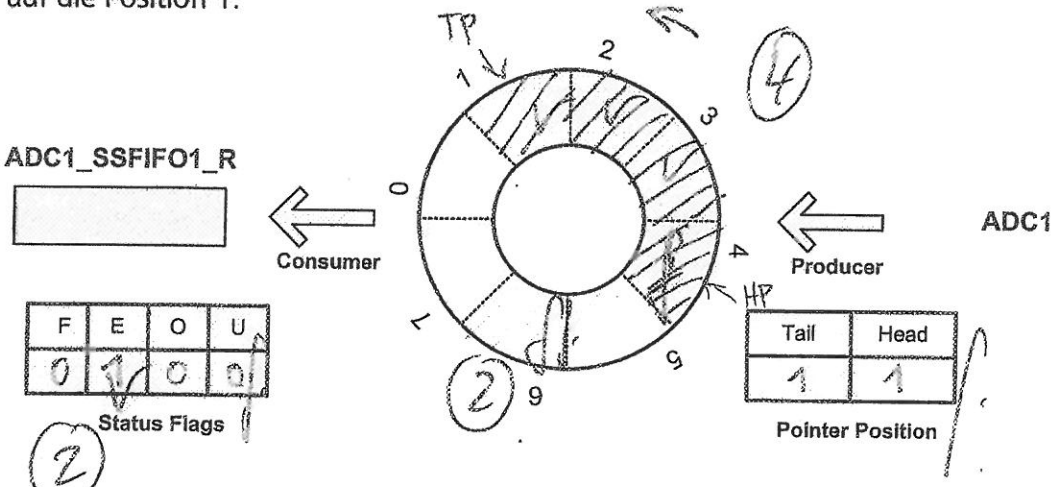


b) Wie muss das UART2-Modul konfiguriert werden, damit obiger Datenrahmen versendet werden kann? Geben Sie alle notwendigen Schritte an. Wählen Sie ein geeignetes Port für U2Tx aus und konfigurieren Sie es ebenfalls.

c) Welches Zeichen würde ein Empfänger feststellen, dessen Baudrate fälschlicherweise auf 38400 bits/s eingestellt ist. Geben Sie die Reaktion auf das empfangene Paritätsbit und Stoppbit an.

Aufgabe 3: (ADC / FIFO - 20 Punkte)

Nehmen Sie an, der Head-Pointer des ADC1-FIFO zeigt auf die Position 4, der Tail-Pointer zeigt auf die Position 1.



a) Markieren Sie in der Abbildung oben die Bereiche im FIFO, die neue Werte des ADCs enthalten, aber durch die SW noch nicht gelesen wurden.

b) Der ADC schreibt nun 2 neue Abtastwerte in den FIFO. Die SW liest daraufhin sechsmal das Register ADC1_SSFIFO1_R aus. Tragen Sie bitte in der Abbildung oben die Positionen für die Pointer sowie die Bits der Flags in die vorbereiteten Tabellen ein.

c) Was steht nun im Register ADC1_SSFIFO1_R?

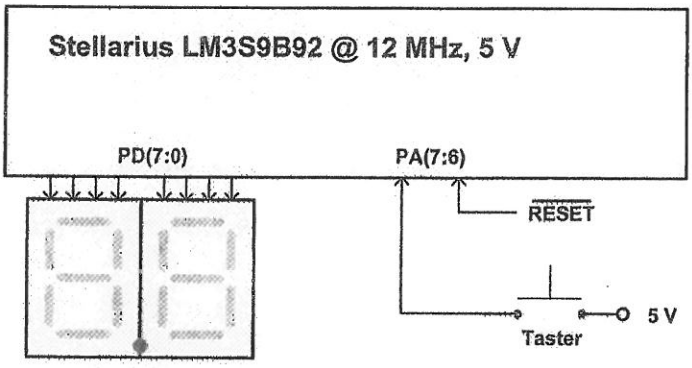
d) Wie kann man programmtechnisch sicher stellen, dass das FIFO nicht in Unterlauf gerät. Formulieren Sie die entsprechende Bedingung in C.

Jahr / WS	Semester	Fach	Dozent
2014	E4b	CT	KZR

FSR - Klausurensammlung

Aufgabe 4: (C-Programmierung – 46 Punkte)

Mit dem Stellarius-Controller LM3S9B92 (CPU-Takt: 12 MHz) soll eine Stoppuhr für kurze Zeiten von 0 bis maximal 9,9s realisiert werden. Die Zeitausgabe soll zweistellig mit einer Auflösung von 0,1s durch eine Siebensegmentanzeige erfolgen. Die Anzeigenbausteine enthalten einen BCD zu Siebensegment Dekoder. Am Port PA(7) ist ein high-aktiver, entprellter Taster angeschlossen. Bei jedem Tastendruck beginnt eine neue Zeitmessung mit dem Wert 0. Die Zeitmessung endet mit dem Loslassen des Tasters. Nach dem Loslassen des Tasters bleibt die Anzeige unverändert, bis der Taster erneut gedrückt wird und die nächste Zeitmessung wieder bei Null startet. Über einen zweiten low-aktiven Taster an PA(6) soll sich die Anzeige auf Null zurücksetzen lassen.



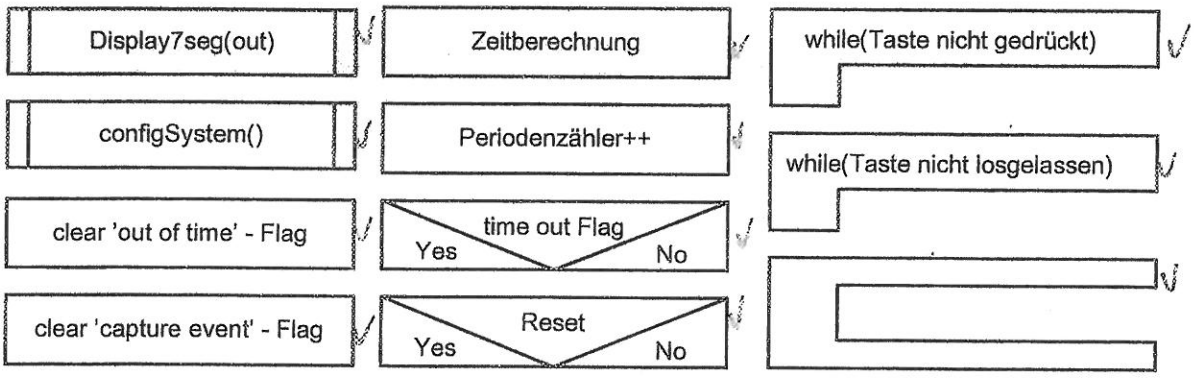
Zeit zwischen 2 Flanken

Die Zeitmessung soll mit Hilfe des Timer1B im Capture Mode (edge time) durchgeführt werden. Der Ablauf soll ohne Interrupts erfolgen. Gehe Sie davon aus, dass die Taste an PA(7) erstmalig erst dann gedrückt wird, wenn das Programm gestartet ist.

Die folgenden Makros und Funktionen sind in der main.c definiert:

```
// Konfiguriere die Ports A,D und den TIMER 1B
void configSystem(void);
// Ausgabe auf 7-Segmentanzeige
void display7seg(unsigned int);
```

a) Zeichen Sie das **Struktogramm (Nassi-Shneiderman)** des main-Programms. Nutzen Sie dazu die Elemente unten. Falls erforderlich, können Sie Elemente mehrfach verwenden oder eigene, zusätzliche Elemente hinzufügen. Hinweis: Skizzieren Sie das Struktogramm zunächst mit Bleistift und zeichnen Sie es später mit Kugelschreiben etc. nach.



b) Konfigurieren Sie **Timer1B** in configSystem() so, dass dieser nach 5ms abläuft.

```
void configSystem(void)   {
unsigned int waitCycle=0;
// configure timer1B

...
```

c) Konfigurieren Sie die **Ports A,D** in der Funktion configSystem().

```
// configure port A + D

...
```

d) Schreiben Sie die **Funktion display7seg(**unsigned int **out)**, die auf der zweistelligen 7-Segmentanzeige das vorliegende Zeitmessung im Format X.Xs darstellt.

```
void display7seg(unsigned short out)
{
        ...
```

e) Schreiben Sie das **Hauptprogramm** entsprechend Ihrem Struktogramm.

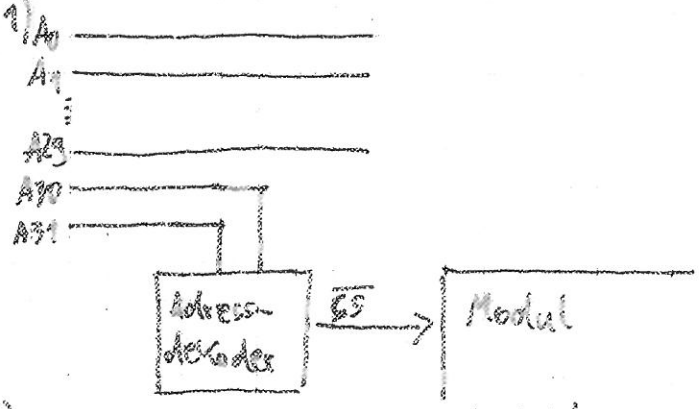
```
void main(void) {

        ...

configSystem();

while(1){
                ...

        }
}
```

Aufgabe 5: (C-Programmierung – 6 Punkte)

Ergänzen Sie das Hauptprogramm in Aufgabe 4 so, dass während der laufenden Messung die hochlaufende Zeit fortwährend angezeigt wird.

SS, WS 2014 | Semester E4b | Fach CT | KZR
FSR - Klausurensammlung

1)



a)

	A31	A30		Startadresse	Endadresse
ROM:	1	X	→	0x80000000	0xBFFFFFFF
RAM:	0	1	→	0x40000000	0x7FFFFFFF
RAM2:	0	0	→	0x00000000	0x3FFFFFFF

5

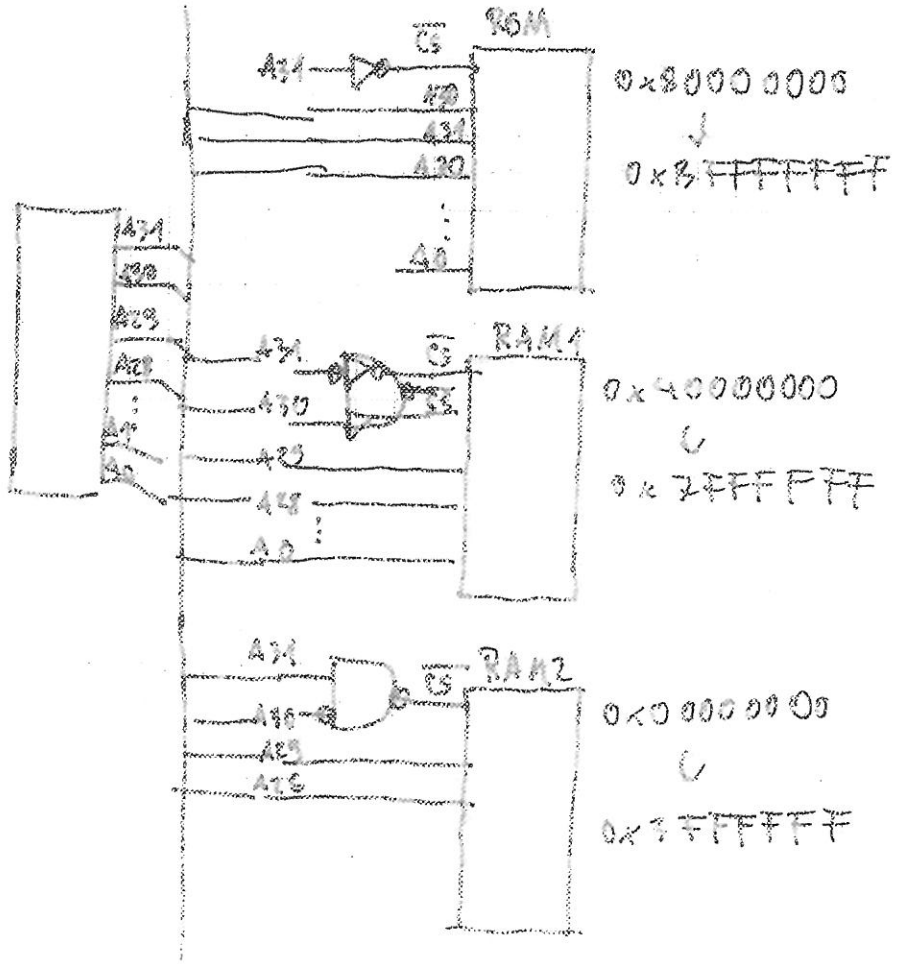
b) ROM + RAM1 + RAM2 = $2 \cdot 2^{30}$ Byte = $(3 \cdot 2^{30} / 2^{20}$ MByte = 3072 MByte)

$+ 1 \cdot 2^{31} = 2 \cdot 2^{30}/2^{20} + 2^{31}/2^{20}$ MB

= 4096 MB

2

c)



Memory-Map

2)

a) 0x47 7/0/1 Baud = 19200 bits/s

0x47 = 0|1000111
MSB LSB

1

b) Konfiguration

//UART 2 16 MHz

$BRD = \frac{SysClk}{ClkDiv \cdot Baudrate} = \frac{16\,MHz}{16 \cdot 19200} = 52{,}08\overline{3}$

~~SYSCTL_RCGC1_R |= 0x~~

BRDI = 52

DIVINT = int(BRDF · 64 + 0,5)

= int(0,083 · 64 + 0,5) = int(5,08) = 5

1

//UART2

SYSCTL_RCGC1_R |= 0x00000004; //clock UART2 wtt

1

UART2_CTL_R &= ~0x0001; //disable for config

UART2_IBRD_R = 52; // BRDI

UART2_FBRD_R = 5; // DIVINT

2

UART2_LCRH_R = 0x00000040; // 7/0/1 0 1 0 6 | 0 0 0 0

UART2_CTL_R |= 0x0001; // enable UART2

//port 6 war 4

1

SYSCTL_RCGC2_R |= 0x00000040; // Clock port 6 wtt!

GPIO_PORT6_DEN_R |= 0x02; // enable pin

GPIO_PORT6_DIR_R |= 0x02; // output PG1

GPIO_PORT6_AFSEL_R |= 0x02; // alternate function PG1

4

GPIO_PORT6_PCTL_R |= 0x00000010; // mit UART2 verbinden

c)

Die Zeit pro bit halbiert sich → falsche Synchronität

→ Zeichen Hexadezimal: 0x7F ≙ DEL // 1

ABER: Stoppbit = 0 → Error da Format 7/0/1 1

Parität wäre = 0 → OK!

c)

```
//Port A
SYSCTL_RCGC2_R |= 0x9;   // Clock A and D   (1)
wtt;
GPIO_PORTA_DEN_R |= 0xC0;   // in 6+7 enable   (2)
GPIO_PORTA_DIR_R |= 0xC0;   // Input 6+7
GPIO_PORTA_AFSEL_R |= 0x20;   // Port 7 alternate F.
GPIO_PORTA_PCTL_R |= 0x7000000;   // aus Tabelle mit CCP3 = CB verbinden   (2)

//Port D
GPIO_PORTD_DEN_R |= 0xFF;   // Pin 0-7 enable
GPIO_PORTD_DIR_R |= 0xFF;   // Output   (2)
```

d)

```
void display7seg (unsigned short out)
{
    out
```

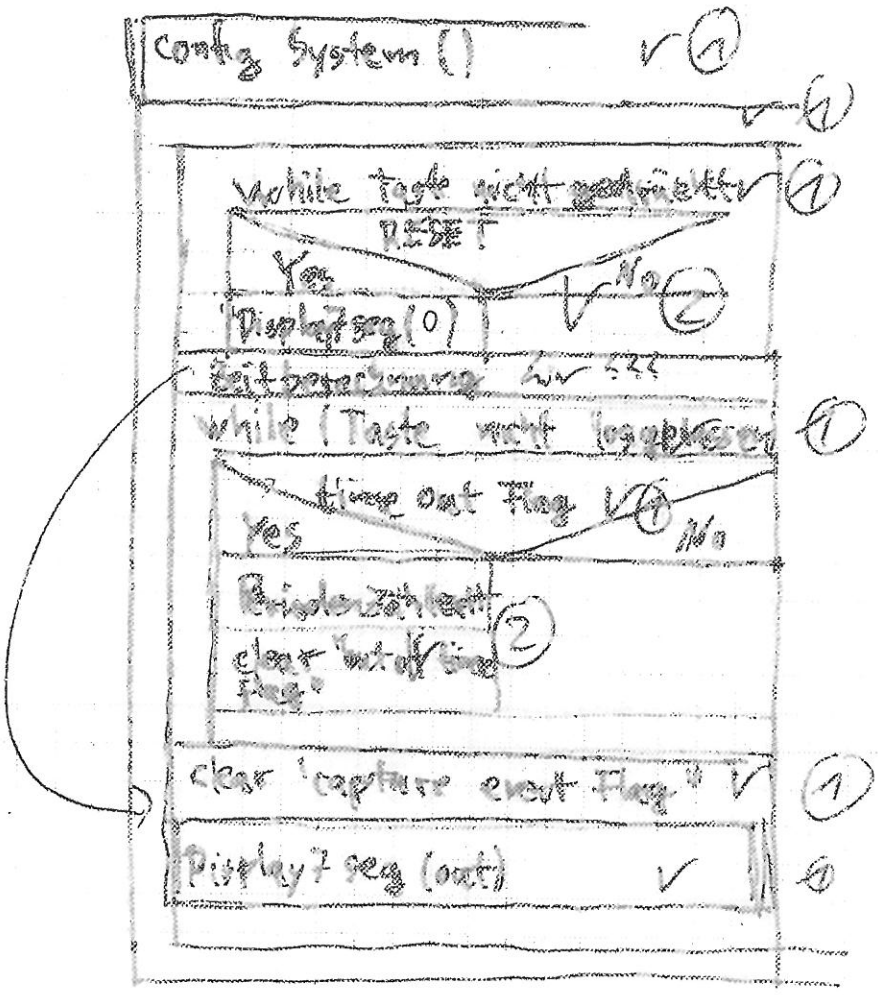
hier müsste noch nachgebessert werden

e)

```
void main(void) {                    short bis 65535
  int Periodenzähler = 0; unsigned short out = 0; unsigned long out_L = 0

  config System();
  while(1) {
      while ((GPIO_PORTA_DATA_R & (1<<7)) == 0) {
          if(GPIO_PORTA_DATA_R & (1<<7)) == 1) {
              display(0);   (2)
          }
      }
      // Timer startet bei Flanke
      while ((GPIO_PORTA_DATA_R & (1<<7)) == 1) {
          if(TIMER1_RIS_R&(1<<B) == 1) {
              Periodenzähler ++;
          } TIMER1_ICR_R |= (1<<B); // clear t-out Flag   (3)
      }
  }
  TIMER1_ICR_R |= (1<<4); // clear capture flag
  out = Periodenzähler · 60000 + TIMER1_TBMATCH;
  display7seg(out);
  out = 0; out_L = 0;
}
}
```

Fehler 2 (1)

3) a) $f = 12\,MHz$ 0 – 3,9 s



$\text{Schritte} = \frac{T \cdot f_{clk}}{Pre} = 60000$

$Pre = \frac{T \cdot f_{clk}}{\text{Schritte}} < \frac{5\,ms \cdot 12\,MHz}{2^{16}} = 0{,}915 \approx \underline{1-1}$

b) Timer 1B capture edge time

```
SYSCTL_RCGC1_R |= 0x00020000; // clock Timer 1
wtitti;
TIMER1_CTL_R &= ~0x100; // Timer 1B stop
TIMER1_CFG_R = 0x00000004; // Timer 1 16 bit mode
TIMER1_TBMR_R = 0x27; // match interrupt, capture mode, edge time
TIMER1_CTL_R |= 0x0C00; // Both edges
~~TIMER1_CTL_R |~~
TIMER1_TBPR_R = 1-1; // Prescaler auf 1 stellen (Optional)
TIMER1_TBILR_R = 60000; // Schritte, load value
TIMER1_CTL_R |= 0x100; // Timer 1B enable
```

5,16 s

3)

c) der zuletzt eingelesene neue Wert! 22

d) das Fifo kann geprüft werden ob es nicht leer ist, bevor ein Wert ausgelesen wird!

```
if(ADC1_SSFSTAT3_R & (1<<0) != 0) {
output = ADC0_SSFIFO1_R
}
```

4

Jahr, WS	Semester	Fach	Dozent
2014	E46	CT	KER

FSR - Klausurensammlung

Name, Vorname, Matr. LEISTUNGSPUNKTE

HAW Hamburg - Prüfungsklausur MP - Wintersemester 2015/16					
Aufgabe	1	2	3	4	Summe
Punkte	13	28	42	17	100
	10	0		7	

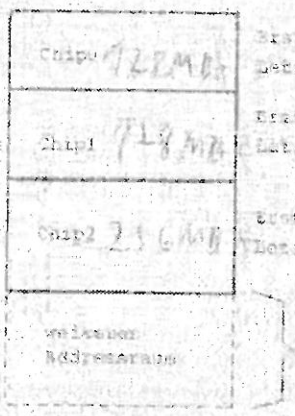


Bild 1: Speicherbelegung des Adressraums mit drei Speicherchips

1 Adressen und Adressdecoder

a) Die Speicherbelegung eines ARM-Cortex-M3-Controllers ist in Bild 1 dargestellt. Wie groß sind die drei Speicherregionen der drei Chips?

b) Welche Schaltungen sind geeignete Adressdecoder für die drei Chips mit Select-Eingängen in negativer Logik ($\overline{CS}$), wenn Schaltungen nicht angegeben? (Jeweils abbrechen in Bild 2)

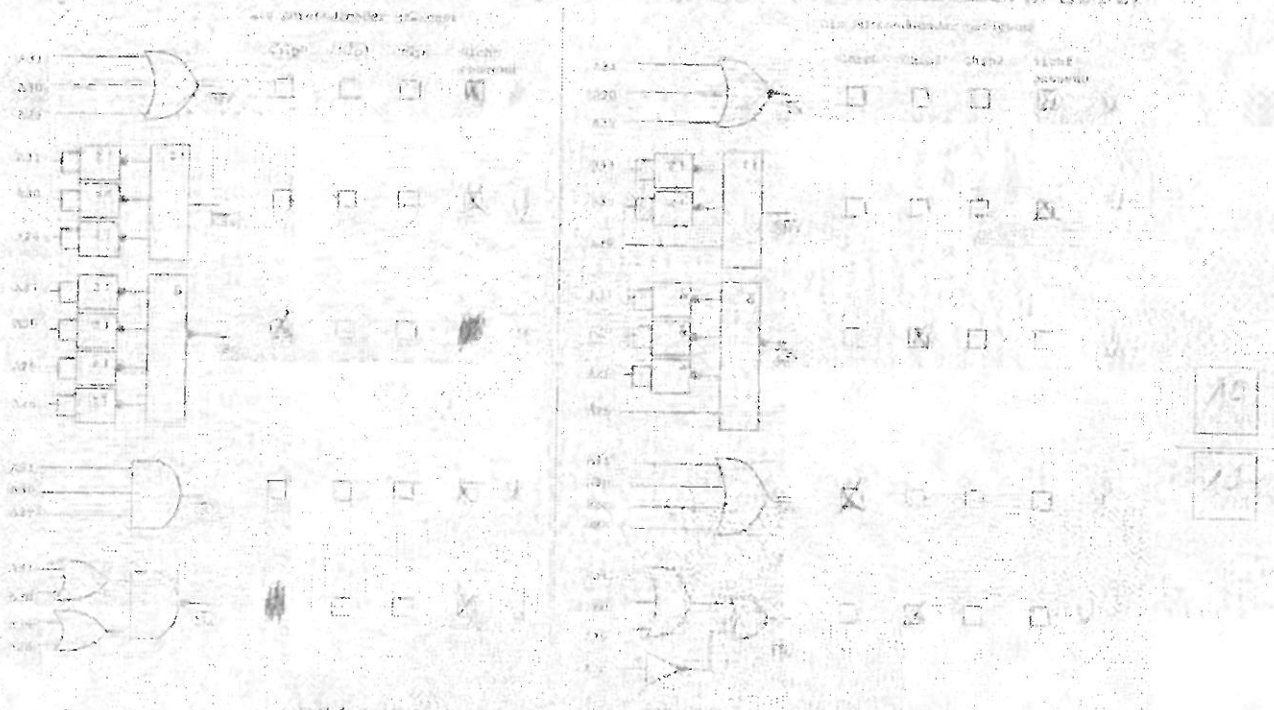


Bild 2: Schaltungssymbole für mögliche Adressdecoder

2 Bitoperationen und Pointer

a) Ergänzen Sie die Kommentare.

```
#include "lm3s9b92.h"          // Headerfile mit Informationen fuer den Prozessor
#include "stdio.h"             //
#define STRINGCONST "abcdefghi" // Präprozessor ersetzt STRINGCONST durch "abcdefghi"

void main(void)
{
  int a=1, b=1, c=1;           // Initialisierung der Variablen a, b, c mit 1
  char s[]=STRINGCONST;        // Initialisierung des Char-Arrays mit STRINGCONST
  char h;                      // Deklarierung des Characters h
  char *p=&s[0];               // Pointer auf das erste Char im Array
  char *q=p+sizeof(s)-2;       // Pointer auf das vorletzte Char des Arrays
  int i;                       // Deklarierung int i

  for(i=8;i>0;i--)             //
  {
    a=a+1;                     // Hochzählen von a
    b=b<<1;                    // Linksverschieben und auffüllen mit Nullen
    c=c|b;                     // Maskieren c mit b
  }
  a=a&0x1FFF;                  // Bitweise Maskierung. Auslöschung von a mit 0x1FFF
  b=b^0x1000;                  // Bitweise Invertierung auf Gleichheit
  c=c|0x1E00;                  // Bitweise Maskieren und Setzen

  // Ausgabe von

  printf("\n a: 0x%x\n b: 0x%x\n c: 0x%x\n",a,b,c);

  while(p<q)                   //
  {
    h=*q;                      //
    *q=*p;                     //
    *p=h;                      //
    p++;                       //
    q--;                       //
  }

  // Ausgabe von

  printf("\n s: %s \n *p: %c\n *q: %c\n ",s,(*p),(*q));
```

a = 9
b = 16200 oder
c = 7177

b) Nennen Sie die erste Ausgabe auf der Konsole mit der Funktion printf() auf.

0x9
0x4000FFFE
0x1EFF

c) Schreiben Sie die zweite Ausgabe auf die Konsole mit der Funktion printf() auf.

3 Hexdezimal-Anzeige

Es soll eine Hexadezimal-Anzeige-Einheit mit dem Controller LM4F232 erstellt werden. Mit vier Schaltern S3,S2,S1 und S0 werden vier binäre Stellen manuell eingeben. Das wird als eine hexadezimale Zahl dargestellt. Dazu wird ein Sieben-Segmente-Display benutzt.

Das Display besteht aus 7 LED mit der Kennzeichnung a bis g. Die Anordnung ist in Bild 3 dargestellt. Die LED haben eine gemeinsame Kathode, die an Vss angeschlossen ist. Die Anoden der LED a bis g sind an Port D angeschlossen, wie in Bild 3 gezeigt.
Die Schaltung der vier Schalter soll von Ihrem C-Programm laufend im Zyklus von einer halben Sekunde geprüft werden. Die Displayausgaben sollen die Ziffern 0 bis F passend darstellen.

Hinweis:
1) Nutzen Sie ein initialisiertes Array.
2) Die Zeichen b und d werden als Kleinbuchstaben ausgegeben.

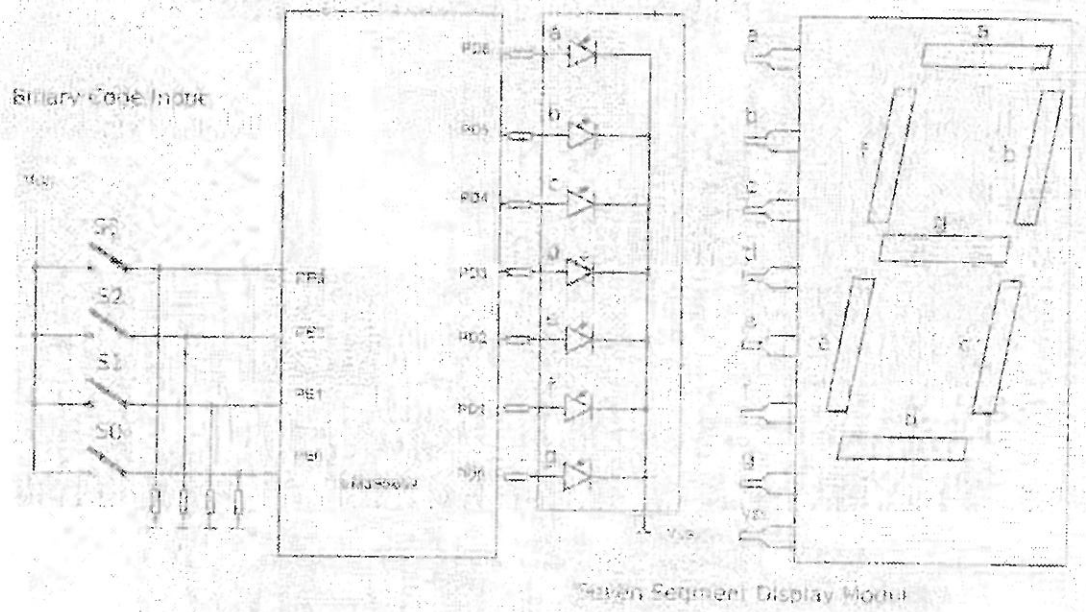


Bild 3: Übersicht über Schaltung der Hexadezimalanzeige

a) Vervollständigen Sie auf der folgenden Seite die folgende Tabelle der Anzeige-Funktionen.

Geben Sie den dezimalen Wert der Eingabe an.
Kennzeichnen Sie die angeschalteten LED jeweils durch „Anzeigen".
Schreiben Sie den binären Wert der Portausgaben für die LED a bis g auf.
Benennen Sie den Hexadezimalwert für die Portausgabe.

Number hexadecimal	Input binary S3 S2 S1 S0	Display	Output binary a b c d e f g	Output hexadecimal abcdefg (hex)
0x0	0 0 0 0		1 1 1 1 1 1 0	0x5E 0x7E
0x1	0 0 0 1		0 1 1 0 0 0 0	0x30 ✓
0x2	0 0 1 0		1 0 1 1 0 1 1	0x6D ✓
0x3	0 0 1 1		1 1 1 1 0 0 1	0x79 ✓
0x4	0 1 0 0		0 1 1 0 0 1 1	0x33 ✓
0x5	0 1 0 1		1 0 1 1 0 1 1	0x5B ✓
0x6	0 1 1 0		1 0 1 1 1 1 1	0x5F ✓
0x7	0 1 1 1		1 1 1 0 0 1 0	0x72 ✓
0x8	1 0 0 0		1 1 1 1 1 1 1	0x7F ✓
0x9	1 0 0 1		1 1 1 0 0 1 1	0x7B ✓
0xA	1 0 1 0		1 1 1 0 1 1 1	0x77 ✓
0xB	1 0 1 1		0 0 1 1 1 1 1	0x1F ✓
0xC	1 1 0 0		1 0 0 1 1 1 0	0x4E ✓
0xD	1 1 0 1		0 1 1 1 1 0 1	0x3D ✓
0xE	1 1 1 0		1 0 0 1 1 1 1	0x4F ✓
0xF	1 1 1 1		1 0 0 0 1 1 1	0x47 ✓

8

b) Programmieren Sie das Programm der Hexadezimalanzeige, zunächst die Funktion main(). Konfigurieren Sie dazu die Ports E und D und einen General Purpose Timer. Die Wertetabelle für die Sieben-Segment-Darstellung soll global angelegt werden (initialisiertes Feld). Eine Funktion bin2hex-out(char x, char *y) soll die LED-Ausgabe bewirken, können Sie in der Funktion main() schon benutzen.

Diese Funktion bekommt den Wert Schalteingaben als Ganzzahlparameter x (call by value) und einen Zeiger auf das Feld der Wertetabelle *y (Call by reference) übergeben bekommen. Eine Wartefunktion wait_halfsec() genügt werden.

Beide Funktionen werden erst in separaten Teilaufgaben nachfolgend programmiert. Kommentieren Sie ausführlich.

```
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"
#define GPIO_PORTD_DATA_R  PD
void wait_halfsecond(void);          // Prototype of the wait function for 0.5s
void bin2hex-out(char x, char *y);   // Prototype of output function

// global array for seven segment transformation
char ous [] = { 0x7E, 0x30, 0x6D, 0x79, 0x33, 0x
                0x5F, 0x70, 0x7F, 0x73, 0x1F,
                0x4E, 0x3D, 0x4F, 0x47 }

void main(void)
{       char *p = &s[0]
        int wt = 0;
        char s;
// Aktiviere 0,25MHz auf 16 MHz
SYSCTL_RCC_R = (SYSCTL_RCC_R & 0x540038 | 0x2B);
wt++; // Warte auf Takt
SYSCTL_RCC_R &= ~0x02; // Deaktiviere den RC-OSC

SYSCTL_RCGC2_R |= 0x18; // Takt an PE, PD
wt++;
GPIO_PORTE_DEN_R |= 0x F; // Aktiviere PE0 PE1
GPIO_PORTD_DEN_R |= 0x1F; // Aktiviere PD0 PD6
GPIO_PORTE_DIR_R &= ~0x F; // PE0 PE1 Eingang
GPIO_PORTD_DIR_R |= 0x3F; // PD0 PD6 Ausgang
```

```
SYSCTL_RCGCT_R |= 0x10000; // Takt an Timer 0
wt 1+1; // warte auf Takt
TIMER0_CTL_R &= ~0x1; // stop Timer 0
TIMER0_CFG_R = 0x0; // setze Timer 0 in 32 Bit Modus
TIMER0_TAMR_R |= 0x2; // setze Timer 0 in Periodic Modus
TIMER0_TAMR_R &= 0xFEFFFFFF; // Timer 0 zählt runter
TIMER0_TAILR_R = 8000000-1; // 8·10^6 / 16·10^6 Hz = 0,5 sek
TIMER0_ICR_R |= 0x01; // Flag TO zurücksetzen


while(1)
{
  x = (GPIO_PORTE_DATA_R & 0xF); // Einlesen PE0 PE3 in x
  bin2hex_out(x,P); // Übergabe des Ergebnis und des Pointers
                    an Ausgabefunktion
  wait_halfsec(); // warte halbe Sekunde

}

}
```

Name, Vorname: 28. Januar 2016 6

c) Programmieren Sie eine Funktion `wait_halfsecond()`, die den Programmablauf um eine halbe Sekunde verzögert. Nutzen Sie einen General Purpose Timer, den Sie am Beginn der Funktion starten und am Ende der Funktion stoppen. Der Timer ist in `main()` bereits konfiguriert. Kommentieren Sie ausführlich.

```
void wait_halfsecond(void)
{
  TIMER0_CTL_R |= 0x01; // Starte Timer 0
  while (!(TIMER0_RIS_R & 0x01)); // Polling auf Flag  warte bis Ta abgelaufen
  TIMER0_CTL_R &= ~0x01; // Stop Timer 0
  TIMER0_ICR_R |= 0x01; // Rücksetzen Flag Ta
}
```

d) Programmieren Sie die Funktion bin2hex_out, welche die LED-Ausgaben vornimmt. Kommentieren Sie ausführlich. **Beachten Sie auf Seite 5 den zweiten Absatz.**

```
void bin2hex_out(char x, char y)
{
  PD &= ~0x3F; // zurücksetzen der alten Ausgangs
                 maximal bis größte Eingabe
                 16
  for (k=0; k<=16; k++) // Zählschleife
  {
    if (k==x) // Abfrage Eingabezahl gleich Zählschleife
    {
      PD &= ~0x3F; // rücksetzen der alten Ausgabe
      PD |= dy; // Ausgabe des Inhalts der Liste an der
      break; // Beenden der Zählschleife  letzten Stelle
    }
    y++; // Zählschleife entlang Eingang, also
         nächstes Listenelement
  }
}
```

Name, Vorname, Matr.-Nr. ..... 28. Januar 2016 7

4 Fragen

a) Nennen sind die grundsätzlichen Abarbeitungsschritte eines Maschinenbefehls.

0

b) Erklären Sie Funktionsweise eines Flags in der hardwareorientierten Programmierung.

Ein Flag gibt den Status an
Z.B. Status Register der ALU setzt ein Flag
mit dem Status der ALU Carry-Flag bedeutet
~~[illegible]~~ Überlauf beim
Ergebnis nach einer Addition

1

c) Was ist der Unterschied zwischen einer Interrupt service-Routine und einer Subroutine?

0

trennt

d) Wodurch unterscheiden sich v. Neumann- und Harvard-Architekturen?

v. Neumann ~~[illegible]~~ nicht ~~[illegible]~~ Code
Memory und Data Memory
In der Harvard Architektur sind zwei Speicher
vorhanden, Data Memory und Code Memory
Rückfrage!

4

e) Mit welcher Methode wird ein Makro im Quellkod verarbeitet? Welcher Teil der Entwicklungsumgebung bewirkt dies?

~~[illegible]~~ wurden Makros gekennzeichnet. Der
Präprozessor ersetzt im Quellkext MAX durch
57.

unit

#define MAX 57

f) Zu welchem Zweck nutzt man ein Paritätsbit? Mit welcher Methode kann man es einfach erzeugen?

Zur ~~[illegible]~~ Überprüfung der korrekten
Übertragung.
Es werden alle ~~[illegible]~~ Einen der zu Übertragenden
Daten gezählt und je nach Einstellung Gerade oder Ungerade.

1

d) v. Neumann

1x Daten Bus

1x Controll Bus

1x Adress Bus

Harvard

2x Controll Bus

1x Daten-Bus
1x Adress-Bus } Code Memory

1x Daten-Bus
1x Adress-Bus } Data Memory

24

13 VON 15 LEISTUNGSPUNKTEN

HAW Hamburg - Prüfungsklausur Computertechnik - SS 2014					
Aufgabe	1	2	3	4	Summe
Punkte	18	50	23	9	100
	15	45	20	8	88

d_n: Datenbit (d_0 = LSB)

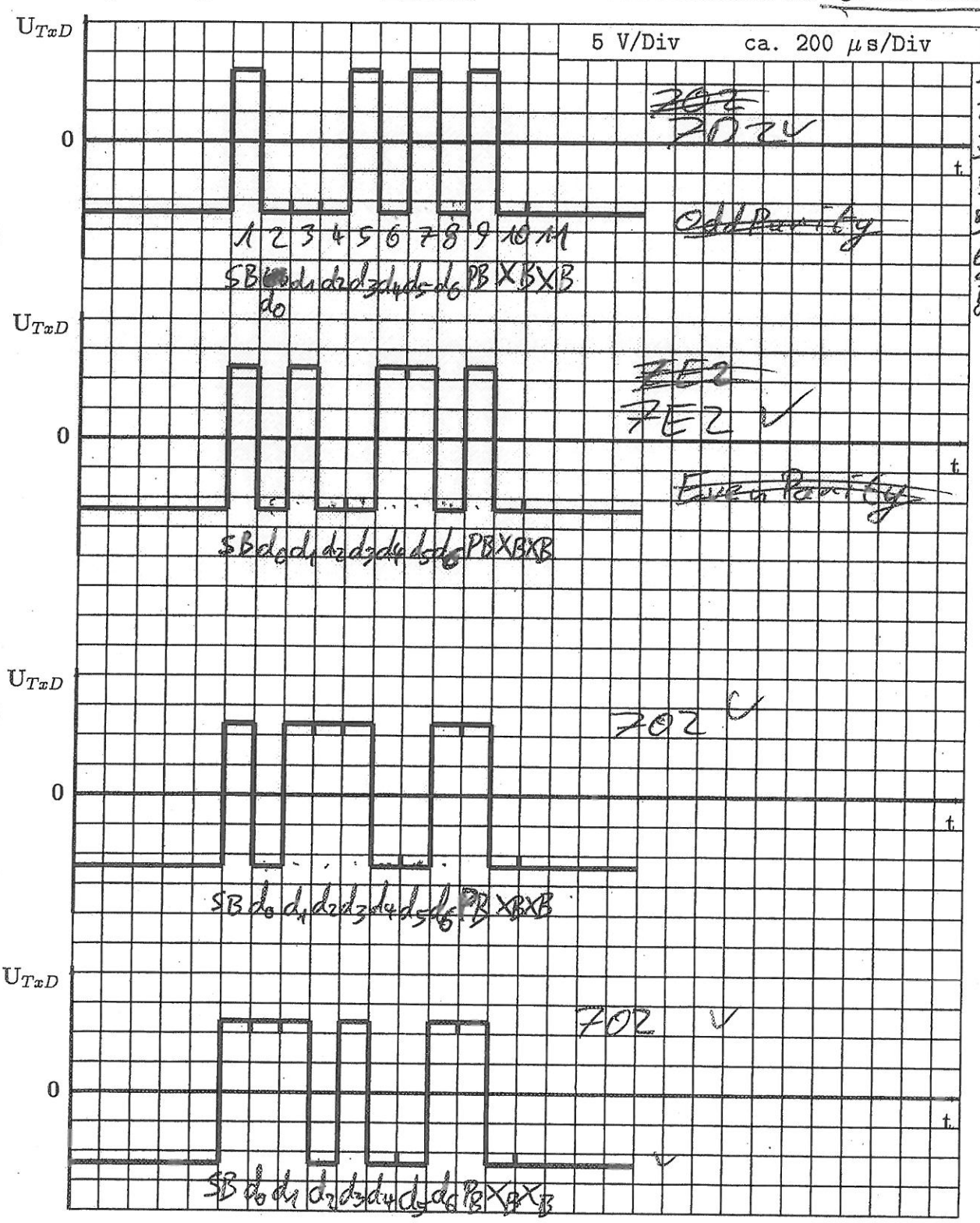
SB: Startbit
PB: Paritätsbit

XB: Stoppbit

1 Serielle Übertragung

Vier ASCII Zeichen (7 Bit) werden asynchron im RS232-Standard (Frames mit 7 Bit Nutzdaten) übertragen. Benennen Sie sämtliche Bits in Bild 1 mit Kurznamen und logischen Werten.

fehlen -1P.



1. Startbit
2. LSB (Data)
3. Datenbit
4. Datenbit
5. Datenbit
6
7

Bild 1. Signale der Zeichen am Anschluss TxD der seriellen Leitungen.

Semester 2014 | Fach ELA | CT | Dozent RMS
ESR - Klausurensammlung
v. 4

a) Welches der drei Protokolle 7N1, 701 oder 7E1 wird benutzt ?

702

Allerdings im 2. Frame 7E2

1 v. 1

b) Welche vier Zeichen werden übertragen? Geben Sie die Daten binär, hexadezimal und als ASCII-Zeichen an.

1. Frame (binär/hex/ASCII): ~~0101~~ 0 1010 111 , 0x57 , "W"
2. Frame (binär/hex/ASCII): 0100 1101 , 0x4D , M
3. Frame (binär/hex/ASCII): 0011 0001 , 0x31 , 1
4. Frame (binär/hex/ASCII): 0011 0100 , 0x34 , 4

=> WM14

8 v. 8

c) Welche der folgenden Datenraten wird genutzt ? (Bitte 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

1 v. 1

d) Welche Spannungsbereiche werden für die Wertedarstellung vor Line-Driver-Chip (RS232-Transceiver-Chip) beim RS232-Standard benutzt, welche nach diesem Chip ? Geben Sie auch die Namen der Bereiche an.

Vor: Logic Level (CMOS/TTL-Pegel)
- High (3,3V)
- Low (0V)

} Bereiche - 1P

Nach: RS-232-Pegel
- Mark (-15V)
- Space (+15V)

} Bereiche - 1P.

2

15 v. 18

	...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

Tabelle 1: ASCII-Symbole (niedrigwertige Bits in den Spalten, höherwertige Bits in den Zeilen)

Dozent RMS
Fach CT
Semester E4a
Jahr 2014
FS2 - Klausurensammlung

2 Gatterprüfmaschine

Es soll eine universelle Gatterprüfmaschine mit dem Controller LM3S9B92 erstellt werden. Sie kann Gatter der Typen AND, OR, NAND, NOR, XOR, XNOR prüfen. Die Schaltung mit Eingangs- und Ausgangspins ist in Bild 2 dargestellt.

Das Controllerprogramm erzeugt eine Kombination der Eingangswerte für das Testgatter für jeweils 10ms. Danach erfolgt Prüfung des Ausgangswerts. Dann wird die nächste Kombination der Eingangswerte angelegt, usw. ...
Welches der 6 möglichen Gatter aktuell angeschlossen ist, wird vom Programm erkannt.

Das Programm erkennt auch, wenn ein defektes (oder kein) Gatter angeschlossen ist, also wenn keine der o.g. logischen Funktionen erfüllt wird. Eine LED an Pin PJ6 zeigt diesen Fall an.

Beispiel: Die LED an PJ0 leuchtet, wenn das Testgatter bei in allen vier Eingangswert-Kombinationen als AND-Gatter korrekt funktioniert.

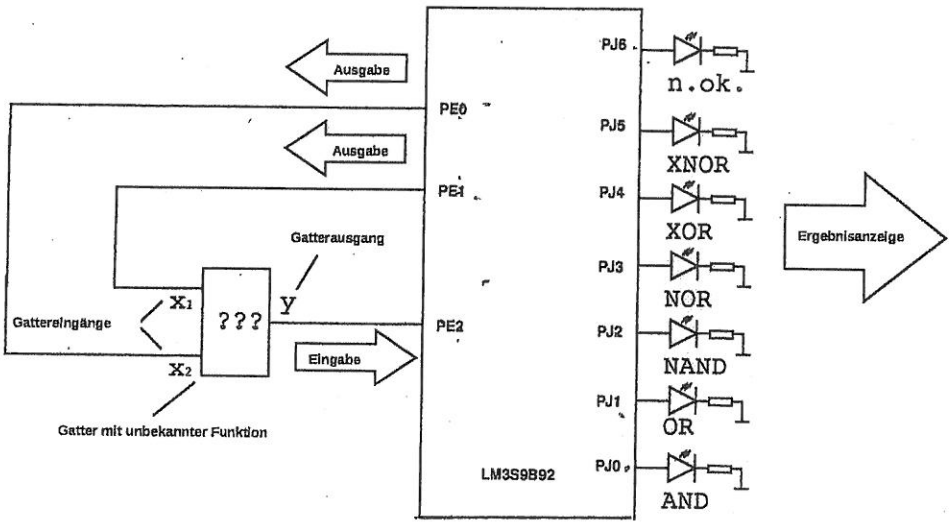


Bild 2: Übersicht über den zu erstellenden Paritätsgenerator

a) Vervollständigen Sie die folgende Tabelle der Gatterfunktionen.

	Gattereingang		Gatterausgang y						
	x_1	x_2	AND	OR	NAND	NOR	XOR	XNOR	
0	0	0	0	0	1	1	0	1	1
1	0	1	0	1	1	0	1	0	2
2	1	0	0	1	1	0	1	0	4
3	1	1	1	1	0	0	0	1	8
			8	14	7	1	6	9	

Tabelle 2: Logische Funktionen, die geprüft werden

3

FSR - Klausursammlung
Semester 2014 Fach E4a CT Dozent RMS

b) Programmieren Sie das Programm der Gatterprüfmaschine, zunächst die Funktion `main()`. Konfigurieren Sie die Ports E und J und einen General Purpose Timer für die 10ms Verzögerung. Eine Funktion `test_gate()` für den Prüfablauf und LED-Ausgabe können Sie schon benutzen.
Die Wartefunktion `warte_10ms()` soll in der Funktion `test_gate()` genutzt werden. Beide werden erst nachfolgend programmiert.
Der Ablauf der Gatterprüfung soll fortlaufend ohne Unterbrechung erfolgen. Kommentieren Sie ausführlich.

```
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"
#define TIMERTIME 160000-1 // Ticks des Timers festlegen
void warte_10ms(void); // Prototype of the wait function for 10ms
void test_gate(void);  // Prototype of test function

void main(void)
{
  int wt=0;                    // 16MHz Oszillator Aktivieren
  SYSCTL_RCC_R =((SYSCTL_RCC_R|0x0540)&~0x02B3);
  wt++;
  SYSCTL_RCGC2_R|= 0x0110; // Takt für E & J zuschalten
  wt++;
  GPIO_PORTJ_DEN_R=0xFF; // Port J 0...6 aktivieren   (0x7F, Digitale Fkt.)
  GPIO_PORTJ_DIR_R =0xFF; // Port J.0...6 = Output
  GPIO_PORTE_DEN_R=0x07; // Port E 0,1,2 dig. Fkt. aktivieren
  GPIO_PORTE_DIR_R=0x03; // Port E 0,1 = Output
  // Timer 3B initialisieren
  SYSCTL_RCGC1_R|=0x080000; // Takt zuschalten
  wt++;
  TIMER3_CTL_R&=~0x0100; // Timer stoppen
  TIMER3_CFG_R =0x04;    // 16-Bit Modus
  TIMER3_TBPR_R=160-1;   // Prescaler setzen
  (TIMER3_TBMR_R |=0x02; // periodischen Modus einschalten)
  TIMER3_TBMR_R&=~0x10;  // herunter zählen
  TIMER3_TBILR_R= TIMERTIME; // Startwert festlegen
  while(1){
    test_gate();         // Gatter prüfen
  }
```

3

c) Programmieren Sie eine Funktion `warte_10ms()`, die den Programmablauf 10ms verzögert. Nutzen Sie einen General Purpose Timer, den Sie am Beginn der Funktion starten und am Ende der Funktion stoppen. Der Timer ist in `main()` bereits konfiguriert. Kommentieren Sie ausführlich.

```
void warte_10ms(void)
{
  TIMER3_CTL_R |=0x0100; // Timer 3B starten
  while((TIMER3_RIS_R&0x0100)==0x00); // Warten bis Timer abgelaufen
  TIMER3_CTL_R &=~0x0100; // TIMER 3B stoppen
  TIMER3_ICR_R |= 0x0100; // Timeout flag löschen
```

5

Datum 2014 | Semester ELA | Pech CT | Dozent RMS
FSR - Klausurensammlung

d) Programmieren Sie die Funktion `test_gate()`, welche die Tests mit allen Eingangswertkombinationen durchführt, diese auswertet und die LED-Ausgaben vornimmt. Kommentieren Sie ausführlich.

durchaus können sonst aber auf einfachen auch mit gestalten

```
void test_gate(void)
{
  int y[4]={0;0,0,0}; // Vektor für Ergebnis anlegen
  int out=0;          // Ausgabe zugehende Kombinationen
  int Lauf=0;         // Laufvariable
  int ergebnis=0;     // Endergebnis
  for(Lauf=0; Lauf<=3; Lauf++) // nacheinander Kombinationen prüfen & Ergebnisvektor füllen
  {
    GPIO_PORTE_DATA_R=out; // 1. Lauf: out=0, 2. Lauf: out=1 ... out=3
    out++;
    warte_10ms();   // verzögerung
    if((GPIO_PORTE_DATA_R&0x04)) { // wenn (markiertes) Bit gesetzt
      y[Lauf]=1;                   // Ergebnisvektor an dieser Stelle=1
    } else
      y[Lauf]=0;                   // sonst=0
  }
  ~~out=0;~~
  ergebnis=y[0]·1+y[1]·2+y[2]·4+y[3]·8; // Endergebnis berechnen
  switch(ergebnis){
    case 8:   // AND-Gatter
      GPIO_PORTJ_DATA_R=0x01; // LED an PJ0 an
      break;
    case 14:  // OR-Gatter
      GPIO_PORTJ_DATA_R=0x02; // LED an PJ1 an
      break;
    case 7:   // NAND
      GPIO_PORTJ_DATA_R=0x04; // LED an PJ2 an
      break;
    case 1:   // NOR
      GPIO_PORTJ_DATA_R=0x08; // LED an PJ3 an
      break;
    case 6:   // XOR
      GPIO_PORTJ_DATA_R=0x10; // LED an PJ4 an
      break;
    case 9:   // XNOR
      GPIO_PORTJ_DATA_R=0x20; // LED an PJ5 an
      break;
    default:  // Fehler/ nok.
      GPIO_PORTJ_DATA_R=0x40; // LED an PJ6 an
      break;
```

34 v. 34

6 v. 6

FSR - Klausurensammlung
WS 2014 | Semester ETA | Fach CT | Dozent RMS

3 Adressen

a) Die Speicherbelegung eines ARM-Cortex-M3-Controller ist in Bild 3 dargestellt. Ergänzen Sie dort die fehlenden Adressen.

4 v. 4

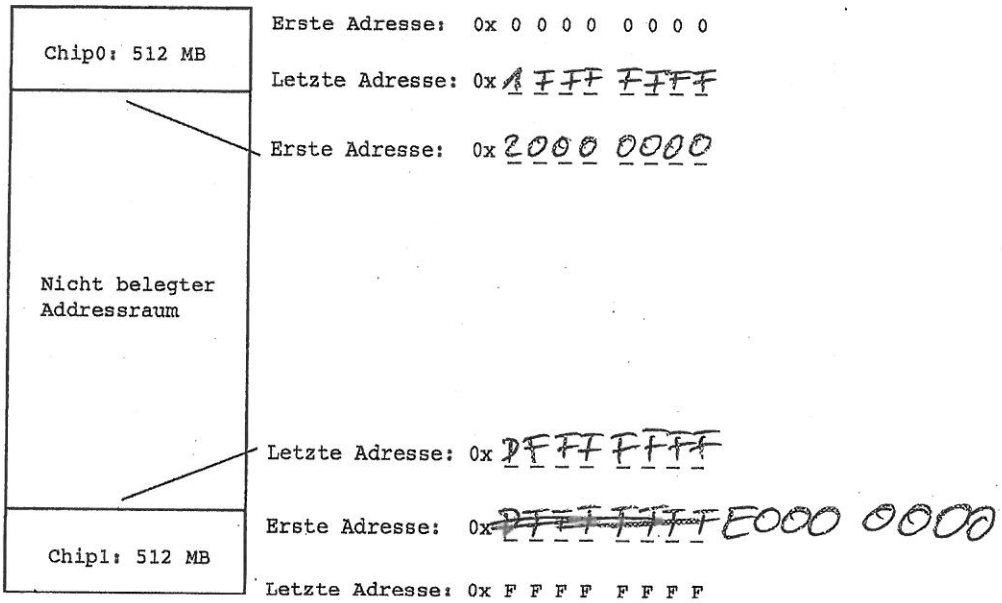


Bild 3: Speicherbelegung des Adressraums mit zwei Speicherchips

b) Wieviele Adressleitungen haben die beiden Speicherchips ? 32

0 v. 1

c) Welche der folgenden Schaltungen sind geeignete Adressdecoder für den Chip 0 ? Streichen Sie alle ungeeigneten Schaltungen in Bild 4 durch.

9 v. 9

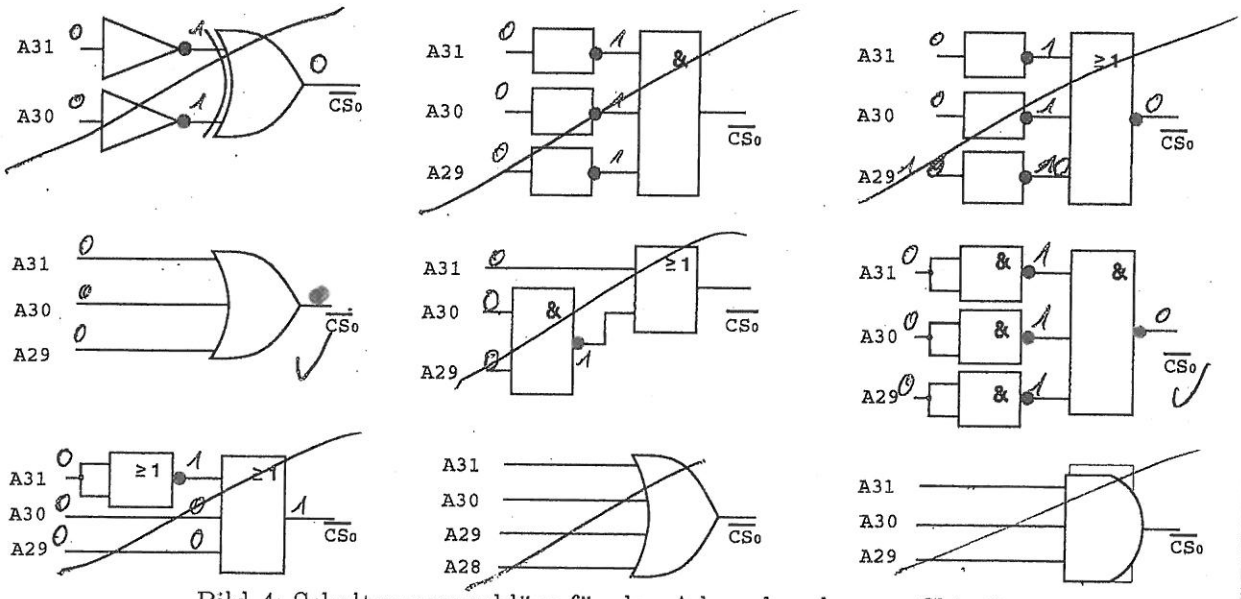


Bild 4: Schaltungsvorschläge für den Adressdecoder von Chip 0

o.k.

chip0: 1. Nibble:

0x0000 => 0b0000

0x1FFF => 0b0001

FSR - Klausurscan
Semester 2014
Fach ELA
Dozent CT
RMS

d) Welche der folgenden Schaltungen sind geeignete Adressdecoder für den Chip 1 ? Streichen Sie alle ungeeigneten Schaltungen in Bild 5 durch.

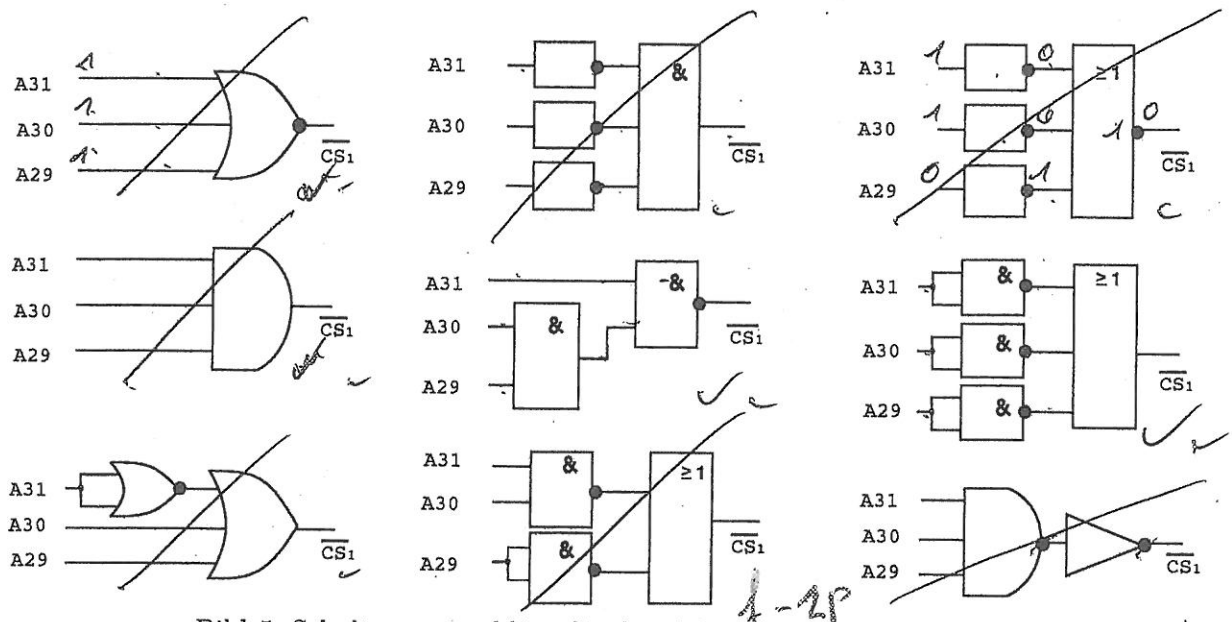


Bild 5: Schaltungsvorschläge für den Adressdecoder von Chip 1

Chip 1:
0xE000... 0b111X
0xFFFF... 0b111X

4 Fragen:

a) Erklären Sie den Unterschied zwischen einer Interruptquelle und einem Interrupthandler.

(Interrupt) Quelle ruft Interrupthandler auf, Quelle kann z. B. ein Pin des Controllers oder eine abgeschlossene AD-Wandlung des (internen) ADC sein. Der Interrupthandler "behandelt" die Interruptursache und ist (entfernt) vergleichbar mit einer Funktion in C.

b) Warum kann ein Interrupthandler - der in C wie eine Funktion geschrieben wird - grundsätzlich keine Parameter haben oder Rückgabewerte zurückgeben ?

Weil unklar ist, wann der Handler aufgerufen wird. Bei Aufruf des Interrupthandlers wird der ursprüngliche Programmablauf unterbrochen. Außerdem kann auch der Handler von einer nachrangigeren Quelle unterbrochen werden. (Race-Conditions)

*und wohin er zurückkehrt

c) Nennen Sie mindestens 5 Funktionseinheiten der CPU. Ordnen Sie diese den beiden übergeordneten Hauptteilen der CPU zu.

Control Unit:	Data Processor:
- Instruction Decoder	- ALU
- Instruction Control Sequencer	- General Purpose Register set
- Program Counter	- status Register
	- Temporary Buffer

FSR - Klausurensammlung

Name, Vorname, Matr.-Nr.: ..... 20. Januar 2014

HAW Hamburg - Prüfungsklausur Computertechnik - WS 2013/14						
Punkte	16	14	20	20	30	100
	16	11	20	15	25	85

+82P.

1 Serielle Übertragung von zwei Zeichen

Ein UART-Modul des Controllers LM3S9B92 sendet zunächst ein Frame mit den ASCII-Symbol 'g' im RS232-Standard mit dem Protokoll 8E1. Die Bitrate beträgt 4800 Bit/s.

Dann wird die UART umkonfiguriert. Jetzt wird das ASCII-Symbol '9' im Protokoll 7O2 mit 9600 Bit/s gesendet.

LEISTUNGSPUNKTE

	...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

Tabelle 1: ASCII-Symbole (niedere Bits in Spalten, höhere Bits in Zeilen)

a) Zeichnen Sie in Bild 1 und 2 die entsprechenden Signale am Ausgangs-Pin des Controllers und auf der Leitung der RS232-Verbindung.

Bezeichnen Sie dort jedes Bit vollständig mit **Kurznamen** und **logischem Wert**.

Bezeichnen Sie **beide Achsen** so, als wenn ein Oszilloskop benutzt würde (siehe Kästen über den Bildern).

g ≙ 0x67 = 0110 0111 $\overset{8E1}{\Rightarrow}$ 0 1100 1110 11

D E Stop

9 ≙ 0x39 = 0011 1001 $\overset{7O2}{\Rightarrow}$ 0 1001 1101 1

Stop

... nächste Seite beachten

WS 2013	Semester E4b	Fach CT	Dozent RMS

FSR - Klausuronsammlung

WS 2013 | Semester E4b | Fach CT | Dozent RMS
FSR - Klausurensammlung

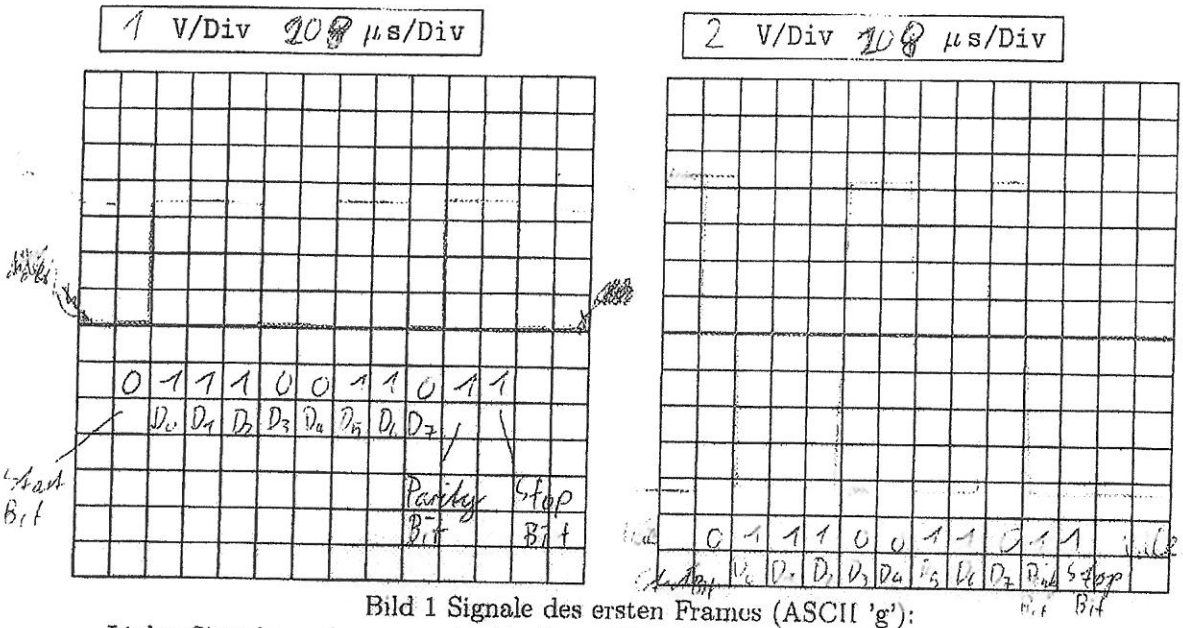


Bild 1 Signale des ersten Frames (ASCII 'g'):
Links: Signal am Ausgangspin des Controllers, Rechts: Signal auf der seriellen Leitung.

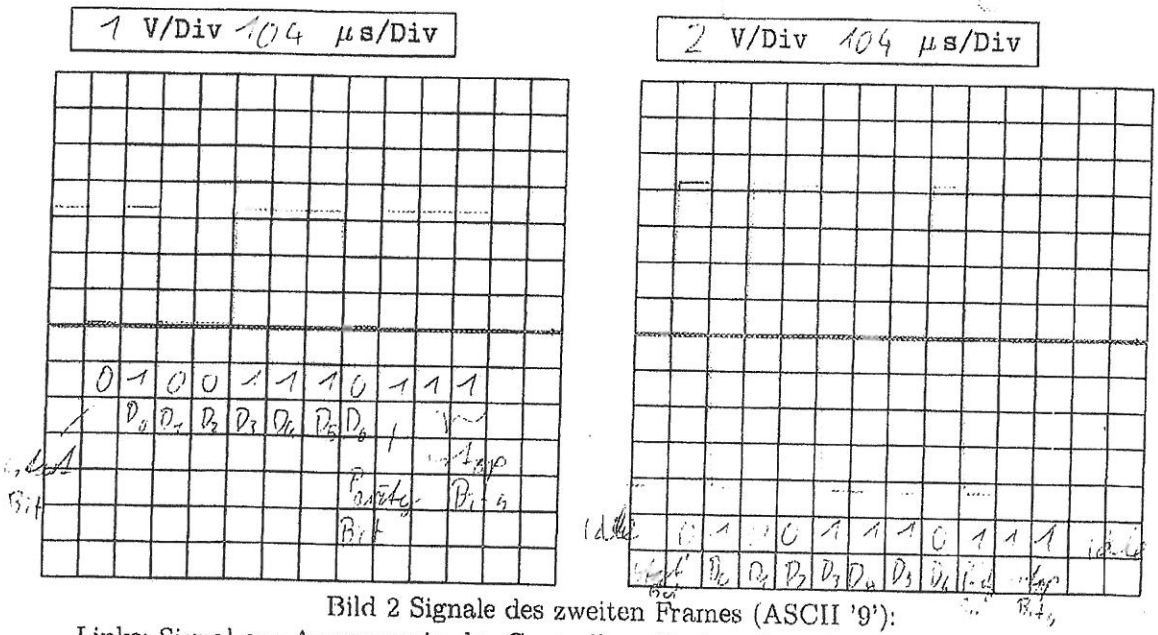


Bild 2 Signale des zweiten Frames (ASCII '9'):
Links: Signal am Ausgangspin des Controllers, Rechts: Signal auf der seriellen Leitung.

Name, Vorname, Matr.-Nr.: ..F ..... Y .....

WS 2013 | Semester 20. Januar 2014 | E46 | CT | 3 | RMS
FSR - Klausurensammlung

2 Architekturen

a) Nennen Sie die Bussysteme sowie die Funktionen in einem Computer mit einer von-Neumann-Architektur. Füllen Sie die Tabelle aus, geben Sie auch an, in welcher Richtung das Bussystem Werte transportiert. Nennen Sie außerdem die Busbreiten und die Kurznamen für mindestens 2 Bussysteme beim ARM-Controller LM3S9B92.

Bussysteme	Funktion	Richtung	Busbreite/Kurznamen
Control Bus	Regelt alle Dinge die nichts mit Daten oder Adressen zu tun haben?	CPU ↔ Memory ↔ I/O bidirektional	32 Bit
Address Bus	Adressen werden per Broadcast von der CPU gesendet um einem Speicherbereich zu adressieren	CPU → Memory, → I/O unidirektional	32 Bit A31 A0
Data Bus	Leitet Daten von der CPU zu den Speichermodulen und I/O Ports oder von den Speichermodulen und I/O Ports zur CPU	CPU ↔ Memory, ↔ I/O bidirektional	32 Bit D31 D0

6

b) Nennen Sie die Unterschiede bei den Bussystemen in Bezug auf die Tabelle in Aufgabe a), wenn eine Harvard-Architektur betrachtet wird.

....

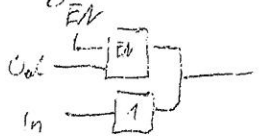
Bei der Harvard Architektur gibt es für den Code Memory ein eigenes Bussystem zusätzlich.
Hier liegen nur die Programme.

2

c) Erklären Sie, wie ein Bussystem bidirektional genutzt werden kann. Welches Problem ist zu überwinden, weil digitale Logik vorliegt ? Welche Lösung wird benutzt ?

....

Das Problem ist, dass wenn zwei Ausgänge auf die gleiche Busleitung gleichzeitig senden können die Signale nicht mehr erkannt werden.
Daher setzt man Tri-State Ein bzw. Ausgänge ein.



3

11

Name, Vorname, Matr.-Nr.: .....

WS 2013	Semester EGb	Fach E-T	Dozent RMS

20. Januar 2014

FSR - Klausurensammlung

3 Paritätsgenerator

Es soll ein Paritätsgenerator mit dem Controller LM3S9B92 erstellt werden. Am Port E Pin PE1 ist ein Schalter zum Umschalten zwischen der Parität für 8 oder 7 Bit vorhanden. Am Port E Pin PE0 wird zwischen der geraden und ungeraden Parität umgeschaltet. Es werden acht Schalter an Port D für die Eingabe von 8 bzw. 7 Bit angeschlossen, ggf. wird der Schalter an PD7 nicht beachtet. Am Port J sind 8 LED angeschlossen, welche die Schalterstellung von Port D anzeigen. Als Ergebnis wird Parität wird an Port E Pin PE3 mit einer LED angezeigt. Siehe Bild 3.

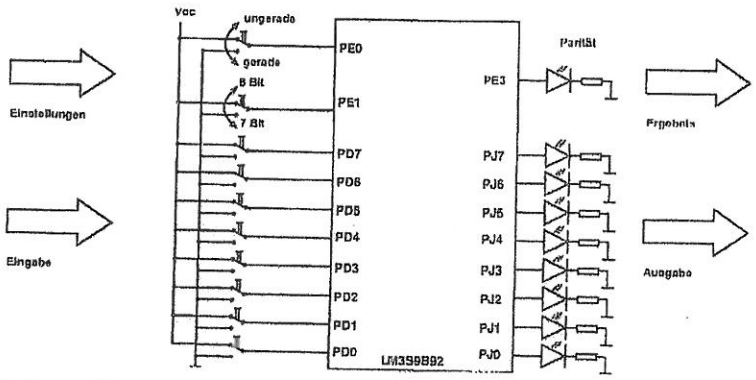


Bild 3: Übersicht über den zu erstellenden Paritätsgenerator

a) Programmieren Sie ein gut kommentiertes Programm, das fortlaufend die Schalter auswertet und die LED-Ausgaben vornimmt.

```
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"
void main(void)
{
int cw=0;
// Quarzoszillator auf 16 MHz
SYSCTL_RCC_R = (SYSCTL_RCC_R (0x00000540) & ~0x00000230);
cw++; // auf stabilen Lauf warten
SYSCTL_RCC_R |= 0x00000002; // internen RC-Osc deaktivieren
// Aktivierung der Clocks
SYSCTL_RCGC2_R |= 0x118 // Clock für Ports D,E und J
cw++;
// Initialisieren der GPIO-Ports
GPIO_PORTJ_DEN_R = 0xFF; // Port J als Ports
GPIO_PORTJ_DIR_R = 0xFF; // alles outputs
GPIO_PORTD_DEN_R = 0xFF // Port D als Ports
GPIO_PORTD_DIR_R = 0x00; // alles inputs
GPIO_PORTE_DEN_R = 0x0A; // Port E Pins 0,1 & 3
GPIO_PORTE_DIR_R = 0x08; // 0,1 inputs 3 output
```

```
#define Port_D GPIO_PORTD_DATA_R
#define Port_J GPIO_PORTD_DATA_R
#define Port_E GPIO_PORTE_DATA_R
// um Lesen zu vereinfachen
```

// Dauerschleife folgt auf der Rückseite

```
// Dauerschleife

while (1)
{
  char c = 0, temp = 0;                  // Variablen Deklarationen
  int parity = 0, odd = 0, eight = 0;

  odd = (Port_E & 0x01;)                 // Paritätsschalter abfragen
  eight = (Port_E & 0x02;)               // 8 Bit Schalter abfragen

  if (eight) c = Port_D;                 // Wenn 8 Bit dann alle einlesen
  else       c = (Port_D & 0x7F);        // Wenn 7 Bit, nur diese einlesen

  temp = c;                              // Kopieren zum gucken
  if(odd) parity = 1;                    // Wenn ungerade, ist parity auf 1, da bei 0x00 eine angefangen werden müsste
(X) while (temp)                         // Schleife um die Parität festzustellen
  {
    if(parity) parity = 0                // parity boolsch gesehen
    else parity = 1                      // invertieren
    temp &= (temp - 1);                  // temp mit (temp - 1) und verknüpfen
  }

  Port_J = c;                            // LEDs an Port J setzen
  if(parity) Port_E |= 0x08;             // wenn ein Paritätsbit setzen sollte die LED an PE3 anschalten
  else Port_E & ~(0x08);                 // sonst ausschalten / zu lassen

}
```

doch so!! kompliziert aber möglich

(X) Das geht ~~[illegible]~~: (einfacher: Bitschieben & xor!)

~~[illegible]~~

volle Punkte dafür

⇒ s. Seite zuvor

Name, Vorname, Matr.-Nr.: ....

WS 2013 | Semester | 20. Januar 2014 | E46 | CT | RMS
FSR - Klausurensammlung

4 Adressen und Fragen

An einen ARM-Cortex-Controller werden drei Speichermodule ROM, RAM1 und RAM2 mit den in Bild 4 dargestellten Adressdecodern angeschlossen.

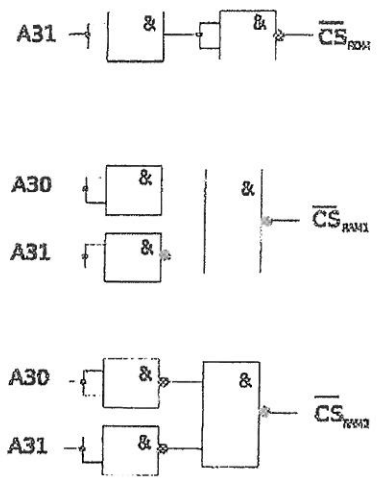


Bild 4: Schaltungen der Adressdecoder der Speichermodule ROM, RAM1 und RAM2

a) Nennen die Anfangs- und die Endadressen der drei Module.

	Anfangsadresse	Endadresse
ROM	0x00000000	0x FFFF FFFF
RAM 1	0x80000000	0x 7FFFFFFF
RAM 2	0x00000000	0x 7FFF FFFF

9 v. 9

b) Nennen Sie die Kapazität der drei Module.

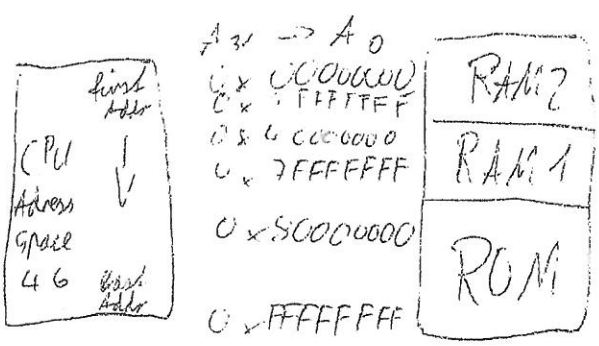
ROM = 2 GB = 2048 MB

RAM jeweils = 1 GB = 1024 MB

2 v. 3

c) Zeichnen Sie die Memory-Map des Controllers mit den drei Modulen, inkl. Adressangaben.

3 v. 3



g) Erklären Sie den Unterschied zwischen einer Interruptservice-Routine (ISR) und einer gewöhnlichen Subroutine.
Kann eine Subroutine von einer Interruptservice-Routine aufgerufen werden ?
Kann eine Service-Routine ablaufen, bevor eine Subroutine beendet ist ?
.....

0 v. 3

g) Nennen Sie die grundsätzlichen Phasen des Ablaufs eines Maschinenbefehls beim ARM-Cortex-Controller.
Worin unterscheiden sich hierbei die Registerbefehle von Speicherbefehlen (Load/Store)?
.....

0 v. 2

16 v. 20

WS 2013 | Semester E4b | Fach CT | Dozent RMS
FSR - Klausursammlung

Name, Vorname, Matr.-Nr.: .... WS 2013 | Semester E46 | 20. Januar 2014 | CT | Dozent RMS

FSR - Klausurensammlung

3 Fernsteuerung über UART

a) Ein Controller LM3S9B92 in einer Maschinenfernsteuerung nutzt eine RS232-Verbindung. Sie wird mit Tastern an den Portpins PD0 (EIN-Taster) und PD1 (AUS-Taster) bedient. Die Taster legen bei Betätigung einen LOW-Pegel an den Controller-Pin an. Ein hochohmiger Pull-Up-Widerstand sorgt ansonsten für einen HIGH-Pegel. Bei Tastenbetätigung werden die Kontroll-LEDs an den Pins PD4 (AUS) und PD5 (EIN) entsprechend geschaltet.

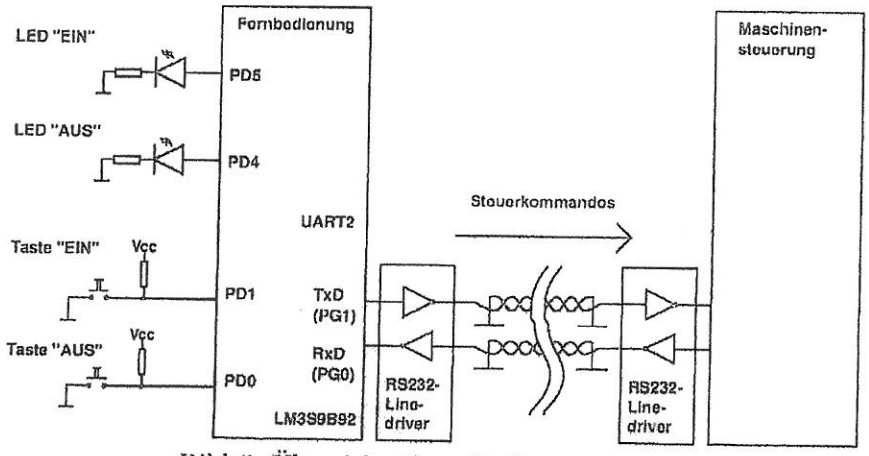


Bild 5: Übersicht über die Fernsteuerung

Die Tasten werden zyklisch von Ihrem Programm nach einer kurzen Wartepause (frei wählbar) abgefragt. Die LED-Ausgabe erfolgt danach. Bei jeder Tasterabfrage wird außerdem die Funktion `serial_transmit (int p)` aufgerufen. Der Parameter p hat den Wert -1, wenn die Maschine ausgeschaltet wird. Er hat den Wert 1, wenn die Maschine eingeschaltet wird. Er hat den Wert 0, wenn die Maschine unverändert weiterlaufen oder stillstehen soll. Dieser Parameter wird immer angewandt, wenn keine Taste oder beide Tasten bei der Abfrage betätigt sind.

Schreiben Sie dafür das Programm mit Kommentaren.

Hinweis: Diese Funktion `serial_transmit (int p)` wird in der Teilaufgabe b) programmiert. Dort wird auch die Funktion `serial_init()` für die Einstellung der UART programmiert. Für die Teilaufgabe a) sind beide Funktionsprototypen vordeklariert, damit Sie diese bereits benutzen können.

```
#define Port_D GPIO_PORTD_DATA_R
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"

void serial_init();
void serial_transmit(int p);

void main(void)
{
  int cw = 0
  // Quarz-Oszillator auf 16MHz
  SYSCTL_RCC_R = ((SYSCTL_RCC_R | 0x00000540) & ~0x00000028);
  cw++;
  SYSCTL_RCC_R |= 0x00000002;
  cw++
// Aktivierung des Clocks
  SYSCTL_RCGC2_R |= 0x008;  // Clock für Port D
  cw++
  GPIO_PORTD_DEN_R = 0x33;  // Port D: Pins 0,1,4,5
  GPIO_PORTD_DIR_R = 0x30;  // 4 5 als output
```

SS / WS 2013 | Semester E4b | Fach CT | Dozent RMS
FSR - Klausurensammlung

```
// Dauerschleife
while (1)
{
int i=0, p=0;
char c = 0x00;

for (i=0; i<1000; i++);      // Warteschleife
c = Port_D                   // Port D einlesen

if (p=0 && (c & 0x03 = 0x01)) p=1;   // wenn die Maschine eingeschaltet wird
else if (p=0 && (c & 0x03 = 0x02)) p=-1;   // wenn die Maschine ausgeschaltet wird
else p=0;                    // ansonsten bleibt sie wieder

serial_transmit(p);

if (p=1) Port D |= 0x20;     // LED "Ein"
else if (p=-1) Port D |= 0x10;   // LED "Aus"
else Port D &= ~(0x30);      // alle LEDs Aus
}
```

b) Die UART2 sendet bei einer Betätigung des Ein-Tasters das Kommando 'E' 'I' 'N' (drei Zeichen) und nach einer Betätigung des Aus-Tasters 'A' 'U' 'S' (drei Zeichen). Es wird im Format 8O1 mit 9600 Bit/s gesendet.
Schreiben Sie die bereits deklarierten Funktionen `serial_init()` und `serial_transmit(int p)`. Der Funktionsparameter p ist in der Teilaufgabe a) erklärt.

Verschlüsselung der Steuerkommandos:

Falls irgendjemand die Leitung abhört, wollen wir ihm ein kleines Rätsel aufgeben. Dafür werden die Zeichen leicht 'verschlüsselt'.

Das erste Zeichen einer Drei-Zeichen-Sendung, um den Wert einer Variablen X im ASCII-Code weitergerückt. Das zweite Zeichen um X+1 Werte, das dritte Zeichen um X+2 Werte. Die Variable X wird in der Funktion statisch vereinbart und ist anfänglich 1. Jede Drei-Zeichen-Sendung erhöht X um den Wert A. A wird zuvor als X modulo 13 errechnet. Jeder Funktionsaufruf mit dem Parameterwert p erhöht X um 1 ohne Sendung von Zeichen. Wenn X den Wert 253 oder mehr erreicht wird, dann wird X wieder auf 1 gesetzt.

leider nicht genau gelesen

```
void serial_init();
{ int w=0;
SYSCTL_RCGC1_R|= 0x00000004,   // Clock enable für UART2
w++;
SYSCTL_RCGC2_R|= 0x40;         // Clock enable für Port G
w++
UART2_CTL_R &= ~0x0301         // UART 2 deaktivieren
UART2_IBRD_R = 104;            // Baudrate von 9600
UART2_FBRD_R = 11;             // einstellen
UART2_LCRH_R = 0x64;           // 8O1 einstellen
UART2_CTL_R |= 0x0301          // UART 2 aktivieren

GPIO_PORTG_DEN_R |= 0x03;      // Port G Pins 0,1
GPIO_PORTG_DIR_R |= 0x02;      // G0 als output
GPIO_PORTG_AFSEL_R |= 0x03,    // alternative Funktion
GPIO_PORTG_PCTL_R |= 0x03;     // U2Rx
```

WS 2013 Semester E4b Fach CT Bozzai RMS
FSR - Klausursammlung

```
void serial_transmit(int p);
{
static int x=0;
serialinit();
while (!UART2_FR_R & 0x80))
if (p = 1)
{
UART2_DR_R = (0x454+x); // E senden
while (UART2_FR_R & 0x80)),
UART2_DR_R = (0x49+x+1), // I senden
while (UART2_FR_R & 0x80));
UART2_DR_R = (0x4E+x+2); // N senden
}
else if (p = -1)
{
UART2_DR_R = 0x41, +x
while (UART2_FR_R & 0x80));
UART2_DR_R = 0x55, +x+1
while (UART2_FR_R & 0x80),
UART2_DR_R = 0x53 +x+2
}}
```

// Verschlüsselung und x [illegible] ist nicht komplett
// wird bereits nur von 0, 1 und 2 benutzt werden
nicht wenn x als statische Variable
[illegible] Bereich mit FIFO leer ist verwendet wird!

10

c) **ZUSATZAUFGABE für maximal 10 Zusatzpunkte:** Programmieren Sie eine möglichst einfache C-Funktion für die Verwendung in Software der Maschinensteuerung, welche die Verschlüsselung lt. Aufgabe b) auf der Empfangsseite 'entschlüsselt'. Die Funktion hat drei Parameter x,y,z, die drei nacheinander empfangenen Zeichen entsprechen. Die Funktion gibt -1 zurück, wenn in der Zeichenfolge x,y,z das verschlüsselte Steuerkommando 'AUS' erkannt wurde. Sie gibt 1 zurück, wenn die Zeichenfolge x,y,z dem verschlüsselten Steuerkommando 'EIN' entspricht. Der Wert 0 wird zurückgegeben, wenn in x,y,z kein Steuerkommando enthalten ist.

Hinweis 1: Die Funktion soll ohne Kenntnis der aktuellen Verschiebungswerte (siehe Variable X lt. Aufgabe b) der Verschlüsselung auskommen.
Hinweis 2: Berücksichtigen Sie, dass gelegentlich einige Übertragungen nicht erfolgreich sind (Zeichenverlust), aber bei längerem Tastendruck mehrfache Sendungen direkt hintereinander erfolgen.

ok. nicht beachtet

```
int decode(char x, char y, char z)
{
int i = 0, ergebnis=0;
for (i=0; i<255, i++)
{
if (x - i = 69 && y - (i+1) = 73 && z -(i+2) = 78) ergebnis = 1;
else if (x - i = 65 && y -(i+1) = 85 && z -(i+2) = 83) ergebnis = -1;
}
return ergebnis;
}
```

Einfacher wäre es direkt x-y und y-z vergleichen als ...

8 v. 10 P zusätzlich

WS 2013	Semester F4b	Fach CT	Dozent RMS

FSR - Klausurensammlung