Name, Vorname, Matr ..... 28. Januar 2016

HAW Hamburg - Prüfungsklausur MP - Wintersemester 2015/16					
Aufgabe	1	2	3	4	Summe
Punkte	13	28	42	17	100
	10	9	40	8	67

77

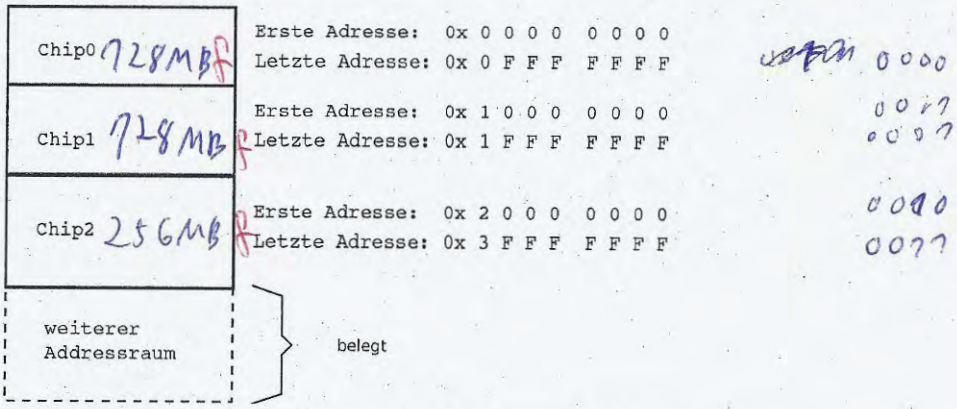


Bild 1: Speicherbelegung des Adressraums mit drei Speicherchips

1 Adressen und Adressdecoder

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

0 v. 3

b) Welche Schaltungen sind geeignete Adressdecoder für die drei Chips mit Select-Eingängen in negativer Logik ($\overline{CS_x}$), welche Schaltungen sind ungeeignet? (Jeweils ankreuzen in Bild 2)

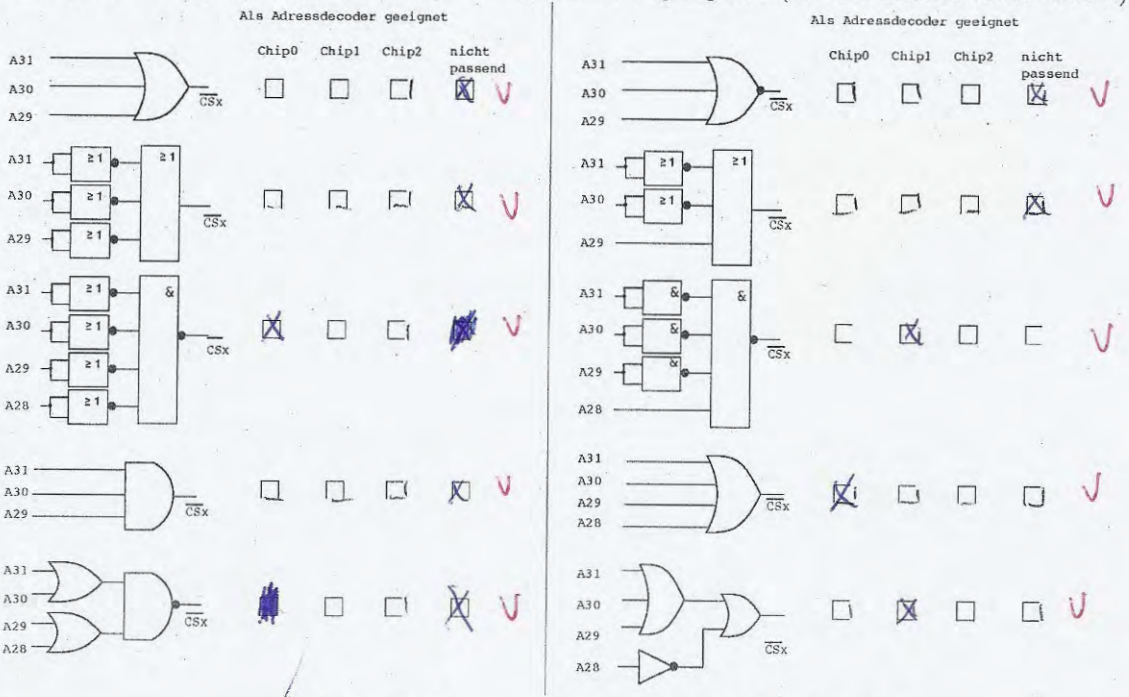


Bild 2: Schaltungsvorschläge für die Adressdecoder

10 v. 10

10 v. 13

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 dem Wert 1
  char s[]=STRINGCONST;         // Initialisierung des Characters s mit der STRINGCONST
  char h;                       // Deklarierung des Characters h
  char *p=&s[0];                // Pointer auf das erste Char im Arrays
  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;                    // Linksschieben und auffüllen mit Nullen
     c=c|b;                     // Maskieren c mit b
   }
  a=a&0x1FFF;                   // Bitweise Maskierung Ausblendung von a mit 0x1FFF
  b=b^0x1000;                   // Bitweise prüfen auf Gleichheit
  c=c|0x1E00;                   // Bitweise Maskieren und Bits 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 = 10000000
c = 1111111

b) Schreiben Sie die erste Ausgabe auf die Console mit der Funktion `printf()` auf.

0x9
0x1FFF
0x1EFF

c) Schreiben Sie die zweite Aufgabe auf die Console mit der Funktion `printf()` auf.

3 Hexadezimal-Anzeige

Es soll eine Hexadezimal-Anzeige-Einheit mit dem Controller LM3S9B92 erstellt werden. Mit vier Schaltern S3,S2,S1 und S0 werden vier binäre Stellen manuell eingeben. Das wird als eine hexadezimale Ziffer dargestellt. Dazu wird ein Sieben-Segment-Display benutzt.

Das Display besteht aus 7 LED mit der Kennzeichung 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 Stellung der vier Schalter soll von Ihrem C-Program laufend im Zyklus von einer halben Sekunde geprüft werden. Die Displayausgaben sollen die Ziffern 0 bis F passend darstellen.

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

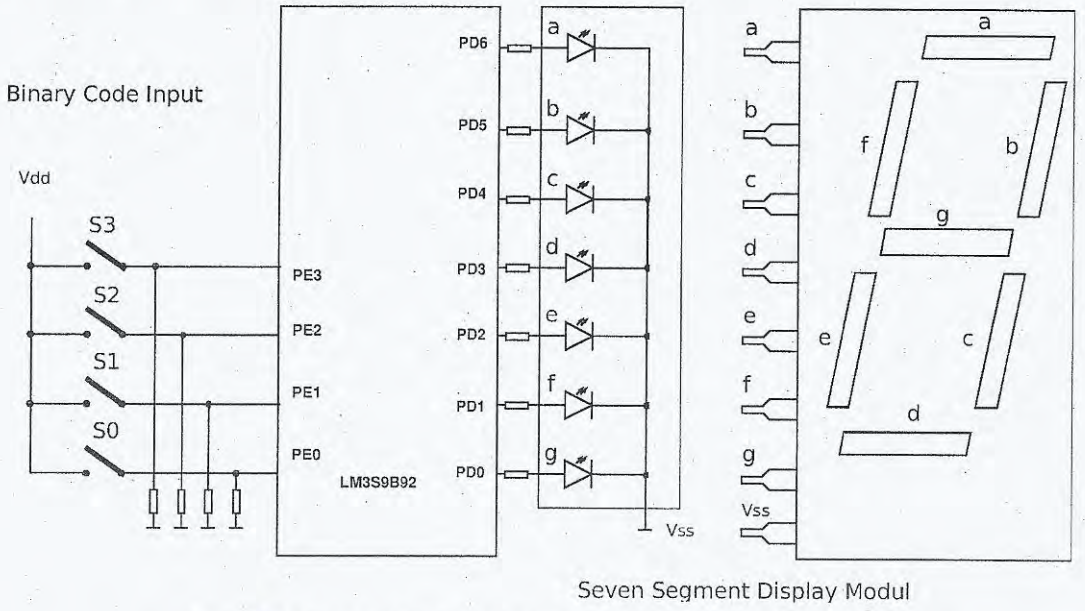
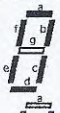
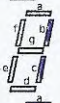
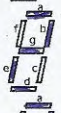
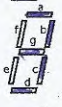
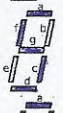
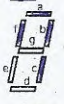
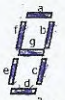
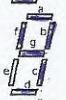
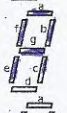
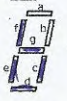
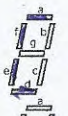
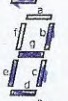
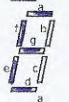
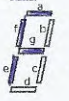


Bild 3: Übersicht über Schaltung der Hexadezimalanzeige

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

- Geben Sie den binären Wert der Eingabe an.
- Kennzeichnen Sie die angeschalteten LED jeweils farbig ('Ausmalen').
- Schreiben Sie den binären Wert der Portausgaben für die LED a bis g auf.
- Errechnen Sie den Hexadezimalwert für die Portausgabe.

Input		Output		
Number hexadecimal	binary S3 S2 S1 S0	Display	binary a b c d e f g	hexadecimal abcdefg (hex)
0x0	0 0 0 0		1 1 1 1 1 1 0	~~0x3E~~ 0x7E
0x1	0 0 0 1		0 1 1 0 0 0 0	0x30 ✓
0x2	0 0 1 0		1 1 0 1 1 0 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 1 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 dort 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 Schaltereingaben als Ganzzahlparameter x (call by value) und einen Zeiger auf das Feld der Wertetabelle *y (Call by reference) übergeben bekommen. Eine Wartefunktion `wait_halfsec()` genutzt werden.

soll

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 aus [] = { 0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B,
                0x5F, 0x72, 0x7F, 0x77, 0x1F,
                0x4E, 0x3D, 0x4F, 0x47 };

void main(void)
{   char *p = &s[0];
    int wt = 0;
    char x;
// Aktiviere Oszillator auf 16 MHz
SYSCTL_RCC_R = (SYSCTL_RCC_R | 0x540) & ~0x2B7);
wt ++; // Warte auf Takt
SYSCTL_RCC_R |= 0x02; //Deaktiviere intn RC-OSC

SYSCTL_RCGC2_R |= 0x18; //Takt an PE, PD
wt ++;
GPIO_PORTE_DEN_R |= 0x F; //Aktiviere PE0... PE3
GPIO_PORTD_DEN_R |= 0x3F; //Aktiviere PD0... PD6
GPIO_PORTE_DIR_R &= ~0xF; // PE0..PE3 Eingang
GPIO_PORTD_DIR_R |= 0x3F; // PD0..PD6 Ausgang
```

```
SYSCTL_RCGCT_R |= 0x10000; // Takt an Timer 0
wt ++i // 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 &= 0xFFFFFFEF; // Timer 0 zählt runter
TIMER0_TAILR_R = 8000000-1; // 8·10^6 / 16·10^6 1/sek = 0,5 sek
TIMER0_ICR_R |= 0x01; // Flag 0 zurücksetzen


while(1)
{
  x = (GPIO_PORTE_DATA-R &0xF); // Einlesen PE0..PE3 in x
  bin2hex_out(x,p); // übergabe des Eingangs und des Pointers
                    an Ausgabefunktion
  wait_halfsec() ; // warte halbe Sekunde

}

}
```

S. vorn ...

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 |= 0x1; // Starte Timer 0
while(!(TIMER0_RIS_R & 0x1)); // polling auf Flag, warte bis TO abgelaufen
TIMER0_CTL_R &= ~0x1; // Stop Timer 0
TIMER0_ICR_R |= 0x1; // Rücksetzen Flag TO
}
```

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; // rücksetzen der alten Ausgangs
                maximal bis größte Eingabezahl
                16
for(int K=0 ; K<=16 ; K++) // Zählschleife
{ if(K==x) // Abfrage Eingabezahl gleich Zählschleife
  {
    PD &= ~0x3F; // rücksetzen der alten Ausgabe
    PD |= y; // Ausgabe des Inhalts der Liste an der
    break; // Beenden der Zählschleife
  }
  y++; // Zählschleife ungleich Eingang, also
       nächstes Listenelement.
}
}
```

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

4 Fragen

a) Nennen sie die grundsätzlichen Abarbeitungschritte eines Maschinenbefehls.
...

0 v.3

f

b) Erklären Sie Funktionsweise eines Flags in der hardwareorientieren Programmierung.
...Ein FLag gibt den Status an.
Z.B. Status Register der ALU setzt ein Flag mit dem status der ALU Carry-Flag bedeutet ~~Übertrag~~ Übertrag bedeutet Ergebnis nach eine Addition.

1 v.2

c) Was ist der Unterschied zwischen einer Interruptservice-Routine und einer Subroutine ?
...

0 v.4

f

trennt

d) Wodurch unterscheiden sich v. Neumann- und Harvard-Architekturen ? ...
V. Neumann ~~unterscheidet~~ nicht ~~zwischen~~ Code Memory und Data Memory. V
In der Harvard Architecture sind zwei Speicher vorhanden, Data Memory und Code Memory. V
Rückseite!

4 v.4

e) Mit welcher Methode wird ein Makro im Quelltext verarbeitet ? Welcher Teil der Entwicklungsumgebung bewirkt dies ? ...
~~#define~~ werden Makros gekennzeichnet. Der Präprozessor ersetzt im Quelltext MAX ~~=~~ durch 57.

mit →

2 v.2

#define MAX 57

f) Zu welchem Zweck nutzt man ein Paritätsbit ? Mit welcher Methode kann man es einfach erzeugen ? ...
Zur ~~Korrekten~~ Überprüfung der Korrekten übertragung.
Es werden alle ~~eigen~~ Einser der zu übertragenden Daten gezählt und je nach Erstellung Grade oder Ungerade

1 v.2

8 v.17

d) v. Neumann

1 X Daten Bus
1 X Controll Bus
1 X Adress Bus

Harvard

2 x Controll Bus

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

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