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

Summe P3 13 v 15 P.

1 Adressbereiche

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.

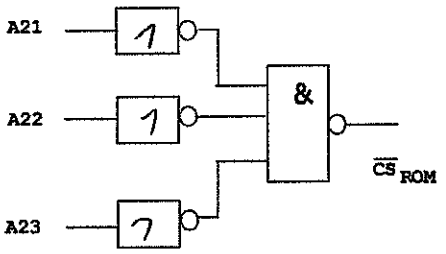

Bild 1: Adressdecoder ROM

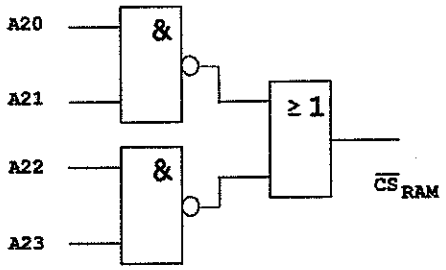

Bild 2: Adressdecoder RAM

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

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

4 v. 1

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

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

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

4 v. 1

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 v. 0

4 v. 17

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

1 1 1

Aufgabe 2: Drehzahlmessung

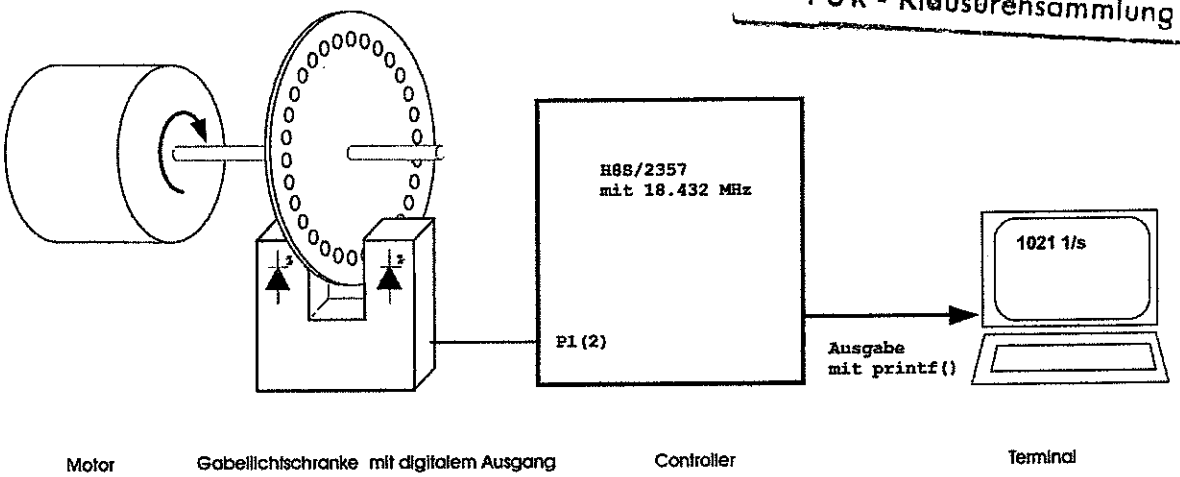

Bild 3: Drehzahlmessung (Erfassung der Encoderpulse am Port 1 Pin 2)

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_TCR3 = 0x10~~ TPU_TCR3 = 0x

1,1

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. Dies sind 60 Datos pro Sekunde ✓

1,1

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

Prescaler			Ticks/µs	
1	18,432 MHz	54,2535 ns	184,31950	Es wird Scaler 1024 verwendet, Ticks/µs: 18000
1024	18000 MHz	55,55 µs	18001,80 ←	
4096	4500 Hz	222,22 µs	4500,045	
256	72000 Hz	13,88 µs	72046,109	

4,1

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

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

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

Teiler? Drehzahl!

Ø1 v. 2

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 benötigt

                                          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 weit außen
  unsigned int speed = 0;
  while (1){ // Endlosschleife für Programm
  P1DDR=0x00; // PORT 1 nur lesen
  // 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
                   clear by TGRA compare match
  TPU_TSR3 &= ~0x03; // Flags rücksetzen

  TPU_TIOR3H = 0x00; // output compare, keine Ausgabe
  TPU_TGR3A = 18001; // 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
```

7

3

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

if (status == 0 && ((PORT1&0x04)==4)) // Pin2 auf PORT1 wieder ist High
{
  speed += 1; // Zähler um 1 erhöhen
  status = 1; // Merken, dass er nun high ist.
}
if (status == 1 && ((PORT1&0x04)==0)) // Pin2 auf PORT1 wieder low
{
  speed += 1; // Zähler erhöhen
  status = 0; // Merken, dass nun low ist
}
} // Timer hat fertig gezählt
// Umdrehung/5 ausrechnen
speed = speed / 60; // Nur ganzzahlige Umdrehungen ausrechnen.
send_actual_speed(speed);
}
}
```

Flags clear?

✓ ✓ 8

18
25

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) ?

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

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 Kanale nacheinander abgearbeitet werden.

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

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 die Antwort.
=> siehe Multiplexer am Eingang

0

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 werden. Das ADDRnH zu lesen verursacht keine fehlerhaften Daten. ~> fehlerhaft ist n. korrekt => "von falschen" d.h. nicht den "selben Zeitpunkten" [illegible]

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 Sendetakt 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_~~
SCI2_SCR=0x10; // Empfangen anschalten.
```

nicht gefordert

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) entspricht /32
1	18,432 MHz	80	7200
4	4,608 MHz	20	7200
16	1,152 MHz	5	7200
64	288 kHz	1,25	7200 ← geht nicht

} ok

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 Stoppbit + 1 Startbit
⇒ 720 Zeichen ~~die~~ pro Sekunde

n = 55 (?)

Muster s. 4.

2

18

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 $\theta = -20°C + U_{in} \cdot \frac{20°C}{1V}$.

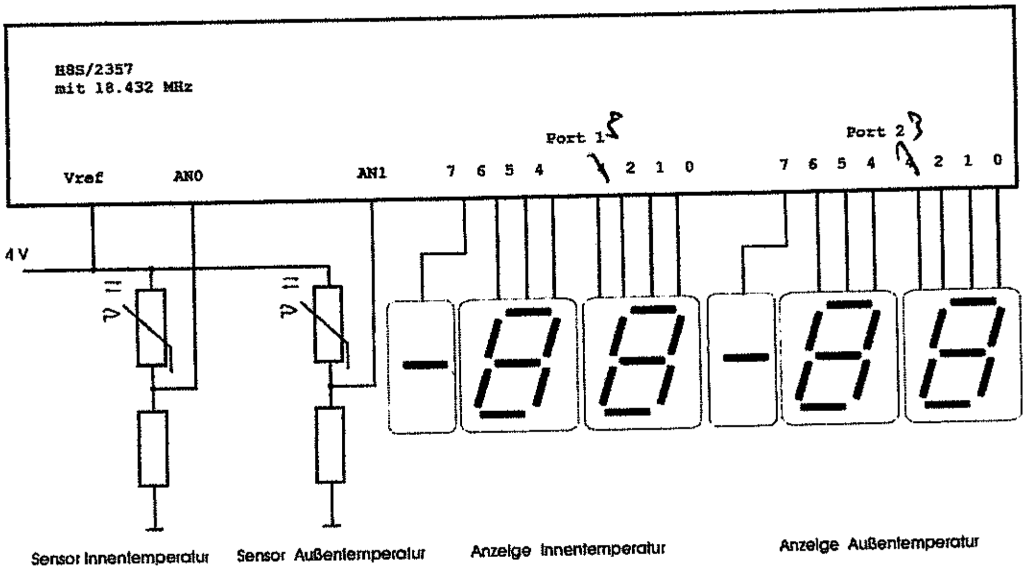

Bild 5: Schaltung der Wetterstation

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 3 s
4	18,432 MHz	1024	54000 ← wird gewählt, da Auflösung höher
	-"-	4096	13504

5

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 tempo; 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; //TCRA clear by compare match;
                    count on rising edge; Clock/1024
   TPU_TSR3 &= ~0x03; //Flagzurücksetzen 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 |= 0x11; // Scan Mode; AN0, AN1 werden gemittelt

   //ADU starten
   ADCSR |= 0x20;
   //Timer starten
   TPU_TSTR3 = 0x08; //Ch3 Start
   while((TPU_TSR3 & 0x01) == 0)
   { Timer läuft nun 3 sek.
     while((ADCSR & 0x80) == 0);
     { //EU Ch0 und Ch1 wurden nun umgesetzt, Ergebnisse
     verarbeiten.
     an0 = ADDRAH <<8; // Hohes Register auslesen und linksschi
     an0 |= ADRRAL; // niedriges Register dazu lesen
     an0 = an0 >> 6; // alles nach rechts schieben
```

```
an1 = ADDRBH<<8;   // s.o.
an11 = ADDRBL;     // s.o.
an1 = an1>>6;      // s.o.
                   // s.o.

~~temp0 = 20 + an0.~~

~~temp0 = -20+(an0/1000)~~
temp0 =(int)(-20+ (an0 · 39L · 20)/10000); // Umrechnung in Temperatur
// Wertebereich:
// an0 < 2^10 ; 2^10 · 39L · 20 = 798720 / durch L hinter 39 konversion auf Long.
       4V/2^10 · 1/10 · 10 = 39

temp1 = (int) (-20+ (an1 · 39L · 20)/1000);
} // Timer hat fertig gezählt.
ADCSR ^= 0x20 ; // ADU stoppen   nicht nötig!
// Ausgabe nun
// temp0 hat Temperatur links , temp1 rechts

~~if(PORT 1&~~
~~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 raussuchen
bcd_high = (temp0 - bcd_low) / 10 ; // Hohe Stelle raussuchen
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);
}
}
```

besser beide Werte * 10 statt 39L dann genauer 3906

s.o.

Prinzip ok.

23 / 26