67

HAW Hamburg - Prüfungsklausur Computertechnik - WS 2008/09					
Aufgabe	1	2	3	4	Summe
Punkte	12	25	47	16	100
	12	24	39	6	81

11 v. 15 P

1 Adressbereiche

An einen Controller H8S/2357 sollen drei Speicherchips angeschlossen werden:

1. Der erste Speicherchip ist ein ROM-Chip mit 2 Megabyte Kapazität. Der Adressbereich des ROM-Chip soll an den unteren Adressen des Controllers ab 0x000000 liegen.
2. Der zweite Speicherchip ist ein EEPROM-Chip mit 2 Megabyte Kapazität. Der Adressbereich des EEPROM-Chip soll ab der Controller-Adresse 0x800000 beginnen.
3. Der dritte Speicherchip ist ein RAM-Chip mit 2 Megabyte Kapazität. Der Adressbereich dieses RAM-Chip umfasst die obersten Adressen des Controllers und endet mit der Adresse 0xFFFFFF.

a) Welche Adressbereiche bleiben frei für spätere Erweiterungen? Nennen Sie die freie Kapazität in MB, die Anfangs- und Endadresse (hexadezimal).

Kapazität	Anfangsadresse	Endadresse
~~2 MByte 6 MByte~~	~~0x000000~~	~~0x1FFFFFF~~
6 MByte ✓	0x200000 ✓	0x7FFFFFF ✓
4 MByte ✓	0xA00000 ✓	0xDFFFFF ✓

6 v. 6

b) Geben Sie die Gleichungen der Chipselect-Signale ($\overline{CS}$) von ROM, EEPROM und RAM an, wenn diese aus den Adressleitungen des Controllers erzeugt werden.

ROM

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

EEPROM

$$\overline{CS_{EEPROM}} = \overline{A_{23}} \vee A_{22} \vee A_{21}$$ ✓

6 v. 6

RAM

$$\overline{CS_{RAM}} = \overline{A_{23}} \vee \overline{A_{22}} \vee \overline{A_{21}}$$ ✓

12

2 Periodische Signalfolgen mit der TPU erzeugen

Die folgenden Signale (Bild 1) sind mit dem Laborsystem zu erzeugen. Die Signale laufen periodisch weiter. Nutzen Sie den Channel 2 der TPU und die im Bild 1 genannten Portpins. Planen und schreiben Sie dafür ein C-Programm.

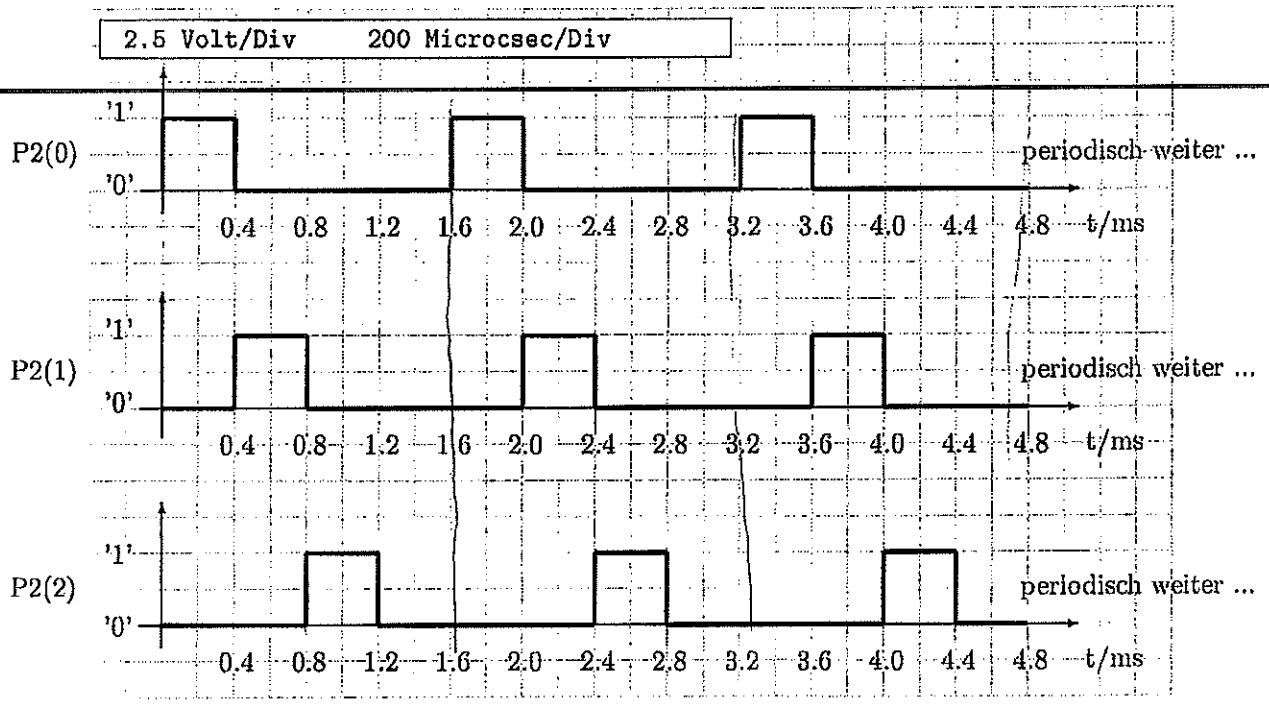

Bild 1: Mit dem Laborsystem zu erzeugende Signale

Bevor Sie programmieren, beantworten Sie bitte zunächst die Fragen:

a) Welchen Arbeitsmodus der TPU werden Sie nutzen ? Output Compare 1 v. 1

b) Welchen Pre-Scaler (Vorteiler-Faktor) wählen Sie aus? 000 ? 0

c) In welchem Register wird die benötigte Zeit eingestellt ? TPU – TGR2A 1 v. 1

d) Welchen Zeiteintrag (Zahlenwert in 'Counter-Ticks') wählen Sie dafür? ..... 2 v. 2

$$TGR2A = \frac{0,4 \cdot 10^{-3}\,s}{59,2535 \cdot 10^{-9}\,s} = 7373,8 \approx 7374$$

ganzen 7373-1

include <cpp1.h> /* Headerfile für das Setup im Labor */

e) Schreiben Sie für die Erzeugung der Signale ein C-Programm mit genauen Kommentaren.

```
/* ---- Wartefunktion für 0,4 ms --- */
void warte (void)
{
  TPU_TCNT2 = 0x00; /* Timer Reset */
  TPU_TCR2 = 0x20; /* Loeschen bei TGRA
                      ziehlen bei steigender Flanke
                      CCK = 18,432 MHz, T = 24,2535ns */   ?
  etwas ungenau!
  TPU_TGR2A = 7374; /* siehe d) */ √
  TPU_TSR2 = 0x00; /* Flags zurücksetzen (TGFA=0) */
  TPU_TSTR = 0x04; /* Timer Channel 2 starten */
  while ((TPU_TSR2 & 0x01) == 0x00); /* Warten auf TGRFA Flg;
  TPU_TSTR = 0x00; /* Timer stoppen */
  TPU_TCNT2 = 0; /* Timer 2 Reset */
}
```

möglichst dürfen!

```
void main (void)
{
  P2DDR = 0x07; /* P2(0), P2(1) und P2(2) Ausgänge */
  while (1) /* Endlosschleife */
  {
    P2DR = 0x01; /* P2(0) high, Rest low */
    warte (); /* warten */
    P2DR = 0x02; /* P2(1) high, Rest low */
    warte (); /* warten */
    P2DR = 0x03; /* P2(2) high, Rest low */
    warte(); /* warten */
    P2DR = 0x00; /* P2 alle (low) */
    warte(); /* warten */
  }
}
```

20 v. 20

24 v. 24

3 Verschlüsseln und Entschlüsseln der seriellen Übertragung

Mit den Laborsystemen ist ein einfaches Ver- und Entschlüsselungssystem in C zu programmieren. Die Daten werden seriell mit dem SCI Channel 1 empfangen, und mit dem SCI Channel 2 weitergeleitet. Es werden dabei die Bitraten 9600 bit/s mit den Parametern '8O1' eingestellt.

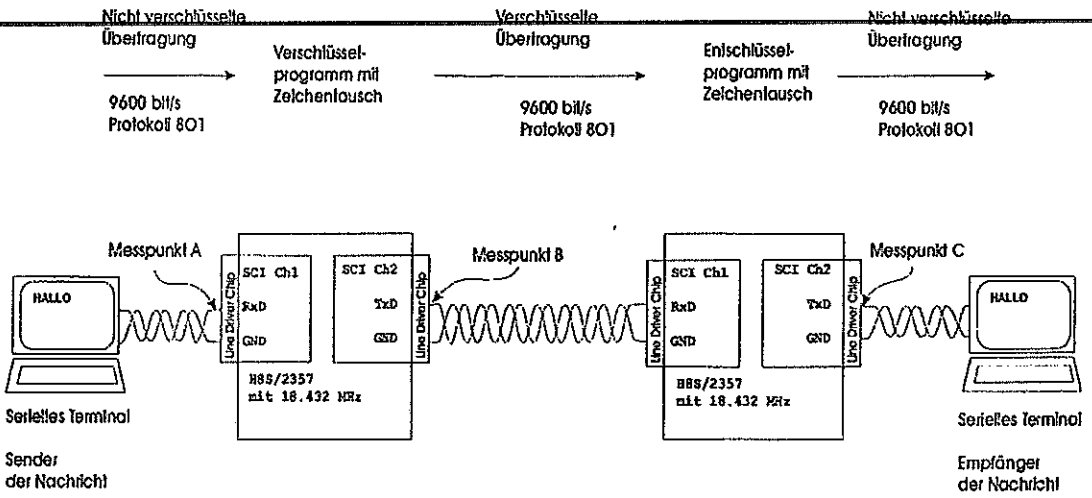

Bild 2: Übersicht zur verschlüsselten Übertragung

Für die Ver- und Entschlüsselung wird eine sehr einfache Codiervorschrift benutzt. Es sind nur Großbuchstaben ohne Umlaute erlaubt (ASCII Codes 'A' bis 'Z'), alle anderen Zeichen werden in das Zeichen '*' umgewandelt.
In der Codiervorschrift wird jedes erlaubte Zeichen durch genau ein anderes Zeichen ersetzt. Dabei werden stets Tauschpaare der erlaubten Zeichen gebildet.

Für die Aufgabe sind folgende Tauschpaare vorgegeben:

('A' ⇔ 'B')[1]

('C' ⇔ 'H')

('D' ⇔ 'X') sowie

('E' ⇔ 'Z')

Die restlichen Tauschpaare sollen Sie frei wählen und zu einer individuellen Codiervorschrift ergänzen. Die Tauschpaarung erlaubt das Ver- und Entschlüsseln mit der gleichen Vorschrift. [2]

[1] Ein Tauschpaar 'A' ⇔ 'B' bedeutet: Ein 'A' wird durch ein 'B' ersetzt und ein 'B' wird durch ein 'A' ersetzt.

[2] Beispiel 'A' ⇔ 'B': Ein 'A' wird durch ein 'B' beim Verschlüsseln ersetzt. Ebenso wird ein 'B' durch ein 'A' beim Entschlüsseln ersetzt. Das Entschlüsselgerät besitzt also das gleiche Programm wie das Verschlüsselgerät.

a) Schreiben Sie die Funktion `void main (void)`. Diese Funktion soll zunächst nur die Initialisierungen des SCI durchführen und dann zyklisch die Funktionen `readsci1`, `writesci2` und `change` benutzen. Diese drei Funktionen werden später programmiert. Nur die erlaubten Zeichen ('A' bis 'Z') sind zu tauschen. Für alle anderen ankommenden Zeichen hat die Funktion `main()` das Zeichen '*' zu senden.

```
#include <mppi.h>          /* Include Headerfile w. Registernames    */
```

```
char readsci1(void);       /* Prototype of a serial read function    */
void writesci2(char);      /* Prototype of a serial write function   */
char change(char);         /* Prototype of character change function */

void main(void)            /* main function without parameters and return value*/
{
```

int int i;
unsigned int zeit;
char zeichen;

SCI2_SCR = 0x00; /* Empfangen und Senden sperren */
SCI1_SMR = 0x30; /* 8 Datenbits, ungerade Parität, 1 Stoppbit, CCK = CCK */
SCI1_BRR = 59; /* 9600 Bits/sec. */

SCI2_SCR = 0x00; /* Empfangen und Senden sperren */
SCI2_SMR = 0x30; /* 8 0 1, CK = CCK */
SCI2_BRR = 59; /* 9600 Bits/sec. */

wrote zeit = 1+ (unsigned long) 1./(5e-6 * 9600);
for (i=0; i<zeit; i++); /* Warten auf Einschwingen des Sende Taktes */

SCI1_SCR = 0x10; /* SCI1 zum Empfangen */
SCI2_SCR = 0x20; /* SCI2 zum Senden */

b.w.

b) Schreiben Sie eine Einlesefunktion char readsci1(void), die ein Zeichen vom SCI Channel 1 seriell einliest und das Zeichen als Rückgabewert zurückgibt.

```
char readsci1(void)
{
char zeichen;
while ((SCI1_SSR & 0x40) == 0x00);
                /* Warten auf volles Empfangsregister */
zeichen = SCI1_RDR; /* Zeichen aus dem Senderegister der
                       Variable Zeichen zugewiesen */
SCI1_SSR &= ~0x40; /* Empfangsregister wieder auf leer setzen */
return zeichen; /* Zeichen zurückgeben */ Flag RDRF resetet
}
```

5

c) Schreiben Sie eine Ausgabefunktion writesci2(char out), die ein an die Funktion übergebenes Zeichen mit dem SCI Channel 2 seriell ausgibt.

```
void writesci2(char out)
{
while ((SCI2_SSR & 0x80) == 0x00);
                /* Warten auf Sendedatenregister „leer" */
SCI2_TDR = out; /* Buchstaben in das Transmit Register schreiben */
SCI2_SSR &= ~0x80; /* Senderegister als voll angeben, Daten werden gesendet
                      danach wird das TDRE-Flag automatisch wieder
                      auf 1 gesetzt */
}
```

5

d) Schreiben Sie die Funktion char change(char in), die genau ein als Parameter übergebenes Zeichen mit einem anderen Zeichen (Rückgabewert) tauscht. Die Tabelle der Tauschpaare soll **dort lokal**, aber für die gesamte Programmlaufzeit initialisiert sein.

Lösungshinweis: Vermeiden Sie damit eine einfache Fallunterscheidung (if-else /switch-case oder dergleichen), suchen Sie eine einfache Lösung, welche die initialisierte Tabelle nutzt!

≙ Stack fehlt hier

```
char change(char in)
{
  char out = '*'; /* Standardmässig mit '*' verschlüsseln */
  int i = 0; /* Laufvariable, Start wert = 0 */
  char vorher[26] = {'A','C','D','E','I','K','M','O','Q','S','U','Y',
                     'B','H','X','Z','G','J','C','N','P','R','T','V','+'};
  char nachher[26] = {'B','H','X','Z','G','J','C','N','P','R','T',
                      'V','+','Y','U','S','Q','O','M','K','I','F',
                      'E','D','C','A'};
  while (in != vorher[i]) {
    i++
  }
  return nachher[in-'A'];
  return nachher[i]; /* nach dem Verschlüsselnden Zeichen suchen */
  while((in != vorher[i]) && (i<26) {
    i++;
  }
  if (i != 26) { /* wenn Zeichen verschlüsselt, mit Zeichen aus "nachher" Liste ersetzen; sonst bleibt out = '*' s.o. */
    out = nachher[i];
  }
  return out; /* verschlüsseltes Zeichen zurückgeben */
}
```

Sehr umständliche Lösung, Tabelle nicht!

e) Mit dem Oszilloskop wird ein Signal an drei Messpunkten(Messpunkte A,B,C in Bild 2) aufgenommen. Das Zeichen 'C' wird vom Terminal gesendet. Zeichnen Sie die Signale. Ergänzen Sie die gesamte Beschriftung, kennzeichnen Sie sämtliche Bits mit logischen Werten und Abkürzungen.

C = 0x43 = 0100 0011 0

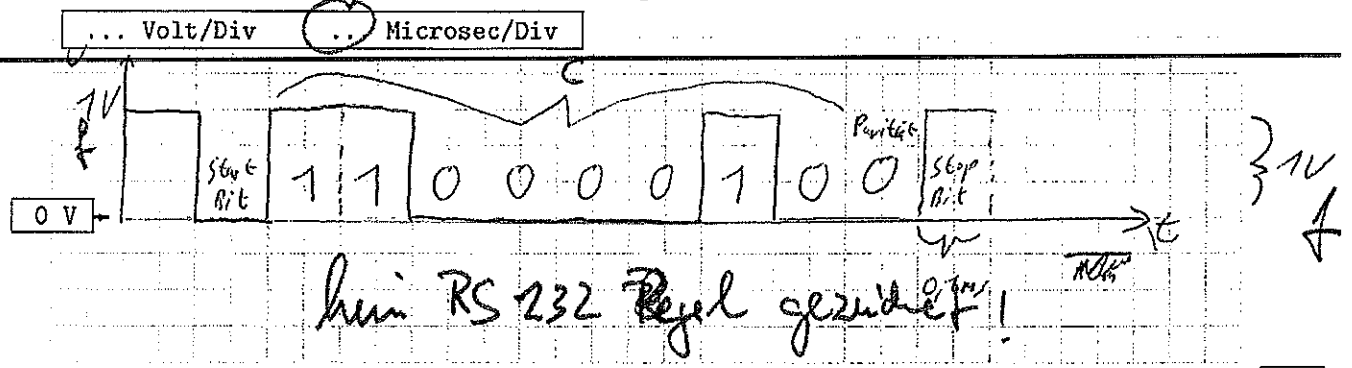

Bild 3: Das Signal für das Eingabezeichen 'C' vor der Verschlüsselung (Messpunkt A)

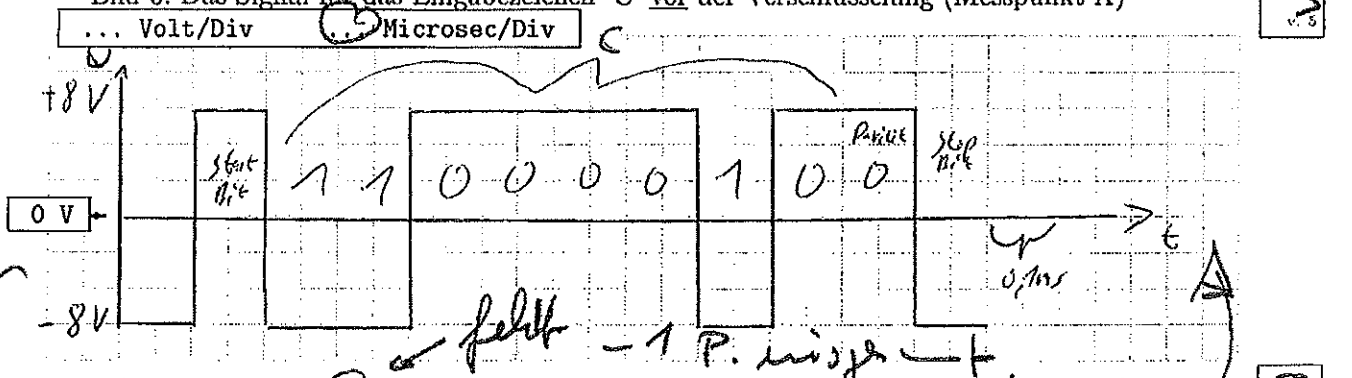

Bild 4: Das Signal für das Eingabezeichen 'C' nach der Verschlüsselung (Messpunkt B)

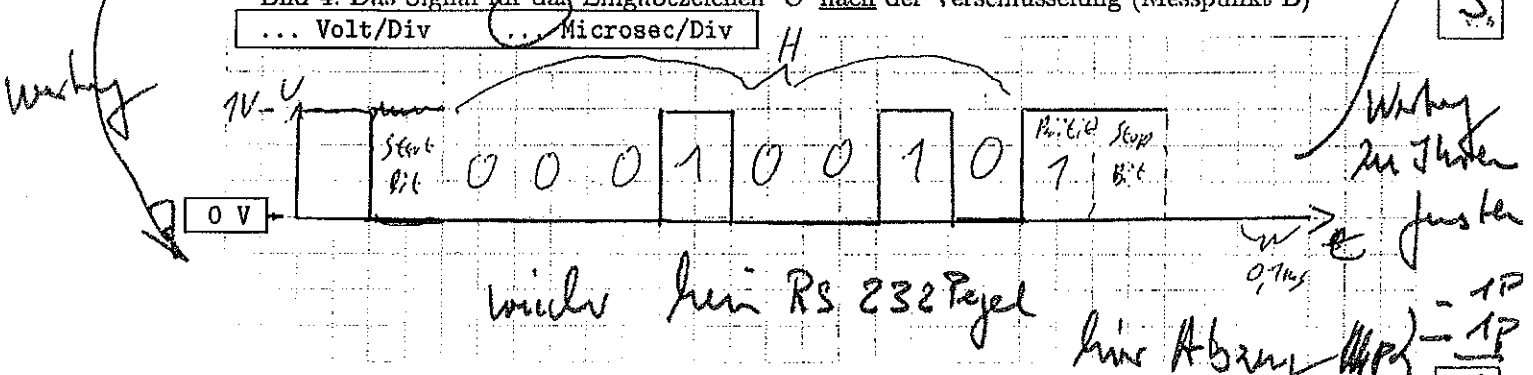

Bild 5: Das Signal für das Eingabezeichen 'C' nach der Entschlüsselung (Messpunkt C)

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

C → H = 0x48 = 0100 1000 Parität 1

4 Fragenteil

a) Markieren Sie die richtige Aussage oder die richtigen Aussagen durch ankreuzen.

In einem von-Neumann-Rechner werden die Daten im RAM und das Programm im ROM gespeichert.	[X] zwingend notwendig	[ ] möglich	[X] keinesfalls richtig	1
Ein Adressdecoder unterscheidet zwischen der Fetch- und Execute-Phase.	[X] richtig	[ ] möglich	[ ] falsch	1
Der interne ADC des H8S/2357 unterscheidet zwischen maximal wieviel Eingangswerten ?	[ ] 255	[ ] 256	[X] 1024	✓
Im Output-Compare-Mode der TPU werden die Werte welcher Komponenten verglichen ?	[ ] TIOCA u. TGFA	[X] TCNT u. TGRx	[ ] TSR u. TCR	✓
Wenn ein übersetztes C-Programm ausgeführt wird, laufen eine Reihe Fetch- und Execute-Zyklen ab.	[X] richtig	[ ] nur im Interrupthandler	[ ] falsch	✓
An dem SCI werden Daten in das Transmitter Shift Register (TSR) vom Programm unmittelbar eingetragen, wenn Daten gesendet werden sollen.	[ ] ja	[ ] Wenn die Bitrate nicht ausreicht	[X] nein	✓
Ein Rechner mit 16 Bit breiten Datenbus kann mit wieviel Hauptspeicher maximal ausgebaut werden.	[ ] 16 MByte	[X] keine Aussage möglich	[ ] 4 Gbyte	✓
Peripherie-Register des H8S/2357 werden wie normale Hauptspeicherzellen vom Programm adressiert.	[ ] ja	[X] wahlweise	[ ] nein	1
Der Data Processor ist für die Steuerung der CPU-Funktion bei der Ausführung der Maschinenbefehle zuständig.	[X] nur in der Harvard-Architektur	[X] nur in der von-Neumann-Architektur	[X] niemals kein wirk!	1
Der Controller H8S/2357 hat welche Architektur (Harvard oder v. Neumann)?	[X] Harvard-Architektur	[X] von-Neumann-Architektur	[X] abhängig vom externen Speicher	1

5 / 10

b) Was kann einen Interrupt auslösen ? Erklären Sie als kurze Stichpunkte.

... Ein zuvor definierter Interrupt „Fall" höherer Priorität ✓

...

IRQ oder Trap ✓

6 / v. 3

c) Was wird bei einem Interrupt ausgeführt? Erklären Sie als kurze Stichpunkte.

... Eine Funktion gerufen Funktion, die die aktuell

... höchste Priorität hat. ggf. deutlicher, aber nicht gemeint als typisch

Antwort: Interrupt-Handler

oder Interrupt-Service-Routine

1 / v. 3

6 / v. 16

Notizen.

Overlap

	A_{23}	A_{22}	A_{21}	A_{20}
ROM	0	0	0	0
	0	0	0	1
	0	0	1	0
	0	0	1	1
	0	1	0	0
	0	1	0	1
	0	1	1	0
	0	1	1	1

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

EEPROM	1	0	0	0
	1	0	0	1
	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1

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

RAM	1	1	1	0
	1	1	1	1

$CS_{RAM} = A_{23} \wedge A_{22} \wedge A_{21}$