SS 07/08 WS	Semester E4	Fach CT	Dozent RMS

sehr gut! 15v. 15 P. Das freut mich! Riemschneider

82

Name, Vorname, Matr.Nr.: Tang, Benjamin, 1827576 . . . . 29. Jan. 2008

HAW Hamburg - Prüfungsklausur Mikroprozessortechnik - WS 2007/08

Aufgabe	1	2	3	4	Zus.	Summe
Punkte	9+2+12+5+4=32	3+8=11	6+6+9=21	2+8+9+8+9=36	4+6=10	100+10
	29	13 (inkl. 2 Zus. pkt.)	21	33+1	Ø	97

Aufgabe 1: Programmanalyse

Re Channel 2 ✓

a) Vervollständigen Sie die Kommentare des nachfolgenden Programms:

```
#include "mpp1.h"  /* Headerfile stellt Registernamen bereit */

void main(void)   /* Hauptprogramm ohne Rückgabe und Parameter */
{
  unsigned short a,b; /* Variablen für ..2 Variable, jeweils 16 Bit -> wofür ?....... */

  P2DDR=0x03; P2DR=0x02; /* Portausgabe P2(1) und P2(0), P2(1) auf high */

  TPU_TCNT2=0x0000;       /* Timer Counter von Channel 2 auf Null setzen */

  TPU_TCR2=0x43;          /* 0100 0011 reset TGRB, ↑, Clk/64 */ ✓

  TPU_TIOR2=0x9A;         /* 1001 1010 A -> input capture ↓, B -> input capture ↓ ↑ */ (vertauscht)

  TPU_TSTR |=0x04;        /* Timer channel 2 starten */

  while (1)               /* Endlosschleife um den folgenden Programmblock */
  {
    TPU_TSR2 &=0xFC;      /* clear flags */ ✓

    while((TPU_TSR2 & 0x01) == 0x00);   /* wait for TGRA matching */ (capture event on TIOCA)

    a=TPU_TGR2A;          /* auslesen des Registers TPU_TGR2A */

    while((TPU_TSR2 & 0x02) == 0x00);   /* wait for TGRB matching */

    b=TPU_TGR2B/2;        /* auslesen Register TGR2B und die Hälfte */

    if (a<b) P2DR=0x01; else P2DR=0x02;  /* Vergleich welcher größer ist und Signal ausgeben */ ✓

  }                       /* Ende des Blocks in der Endlosschleife */
}  /* Ende des Hauptprogramms */
```

8

SS WS	Semester	Fach	Dozent

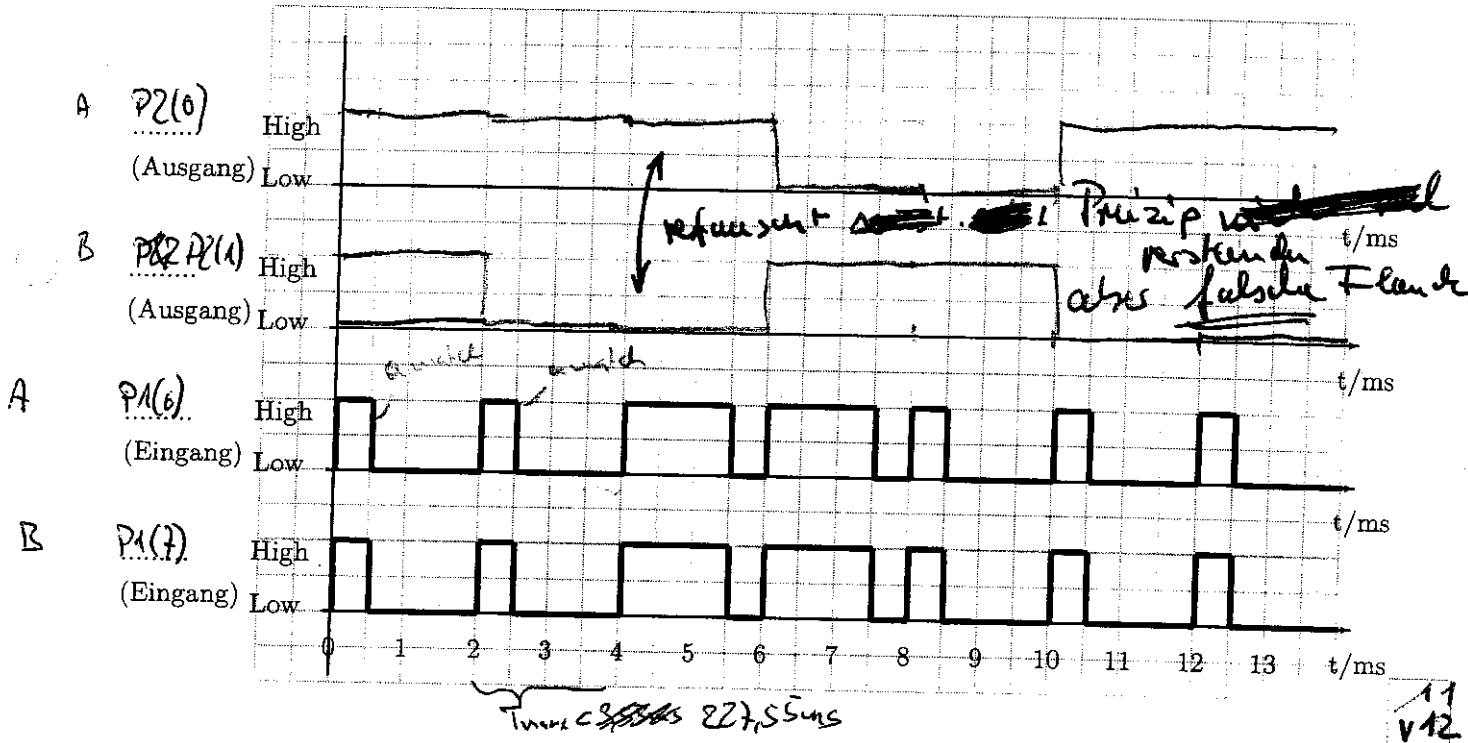
Name, Vorname, Matr.Nr.: Tang, Benjamin 182514 29. Jan. 2008 2

b) Nennen Sie die im Programm zur Ein- oder Ausgabe genutzten Pins (Namen angeben).

Eingabe: P1(6) P1(7) Ausgabe: P2(0) P2(1)

2 V2

c) Zeichnen Sie die Ausgangssignale für die Zeit, in der Eingangssignale vorgegeben sind.



11 V12

d) Beschreiben Sie die Aufgabe des Programms in der Teilaufgabe c) verallgemeinert.

Wenn an die Eingangspins P1(6) und P1(7) dasselbe periodische Signal angelegt wird, dann ...

gilt P2(0) = P2(1)

* ausgezählt wie lang das Signal '0' und '1' ist.
wenn es doppelt solang '0' ist wie '1', dann wird P2(4) 'H', sonst nicht.

b = TAR_TGR2/2 = i

4

e) Bis zu welcher maximalen Periodendauer T_{max} des/der Eingangsignals/e ist eine solche Programmfunktion möglich ? (Funktion lt. d))

T_{max} = 227,55ms möglich, bis Timer bis $2^{16}-1$ gezählt hat für TGR2

1

29 Punkte

Name, Vorname, Matr.Nr.: Tang, 1822576 29. Jan. 2008 3

Aufgabe 2: Adressdecoder

Ein ROM-Chip und ein RAM-Chip sollen an einen Controller H8S/2357 angeschlossen werden.
- Der ROM-Chip hat 4 Megabyte Kapazität.
- Der RAM-Chip hat 8 Megabyte Kapazität.
- Der ROM-Chip soll mit den untersten Adressen ab 0x00 0000 angesprochen werden.
- Der RAM-Chip beginnt mit den unmittelbar höheren Adressen.
- Es gibt keine Lücke in der Adressenbelegung zwischen ROM und RAM.

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

$\overline{CS}_{ROM} = A22 \vee A23$

Ram 4MByte $\Rightarrow 2^{22} \Rightarrow A0 \ldots A21$ Not used A22, A23

3

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

$\overline{CS}_{RAM} = (A23 \wedge \overline{A22}) \vee (\overline{A23} \wedge A22)$

Ram 8MByte $\Rightarrow 2^{23} \Rightarrow A0 \ldots A22$ NOT USED A23 ✓

~~$(\overline{A23} \vee A22)$~~.

Da belegte von $0x400000$ bis $0xBFFFFF$ +2 P.

gut!

8 +2

13

Aufgabe 3: Serial Interface

Für unser Laborsystem[1] wurden folgende Register am Anfang eines Programms gesetzt:

```
SCI2_SMR = 0x74;   => 7Bit, Odd, 1Stopp, Clk
SCI2_BRR = 0x3B;   -55
SCI2_SCR = 0x20;   => just transmit
```

a) Schreiben Sie eine C-Funktion `void test_sci(void)`:
- Die Funktion sendet drei Zeichen 'H','A','W' mit dem SCI Channel 2.
- Sie brauchen die SCI-Register **nicht mehr** zu initialisieren.

```
/* Test der Sendefunktion der SCI, keine Rückgabewerte und keine Parameter */
void test_sci(void)
{
  while ((SCI2_SSR & 0x80) == 0x00); /* TDRE muss auf '1' sein */
  SCI2_TDR = 'H';                    /* write letter in Register */
  SCI2_SSR &= 0x7F;                  /* send data */
  while ((SCI2_SSR & 0x80) == 0x00); /* wait until transmitted finish */
  SCI2_TDR = 'A';                    /* 2. Letter */
  SCI2_SSR &= 0x7B;                  /* send Letter */
  while ((SCI2_SSR & 0x80) == 0x00); /* wait until transmit finish */
  SCI2_TDR = 'W';
  SCI2_SSR &= 0x7B;
  while ((SCI2_SSR & 0x80) == 0x00);

/* Mit einer for-Schleife und einem Array wäre den Program kürzer? */ Ja, das hätte ich Ihnen auch zugetraut..... trotzdem Teilpunkte gegeben => volle Punkte.
}
```

[1] 18.432 MHz Systemtakt

5 / 5

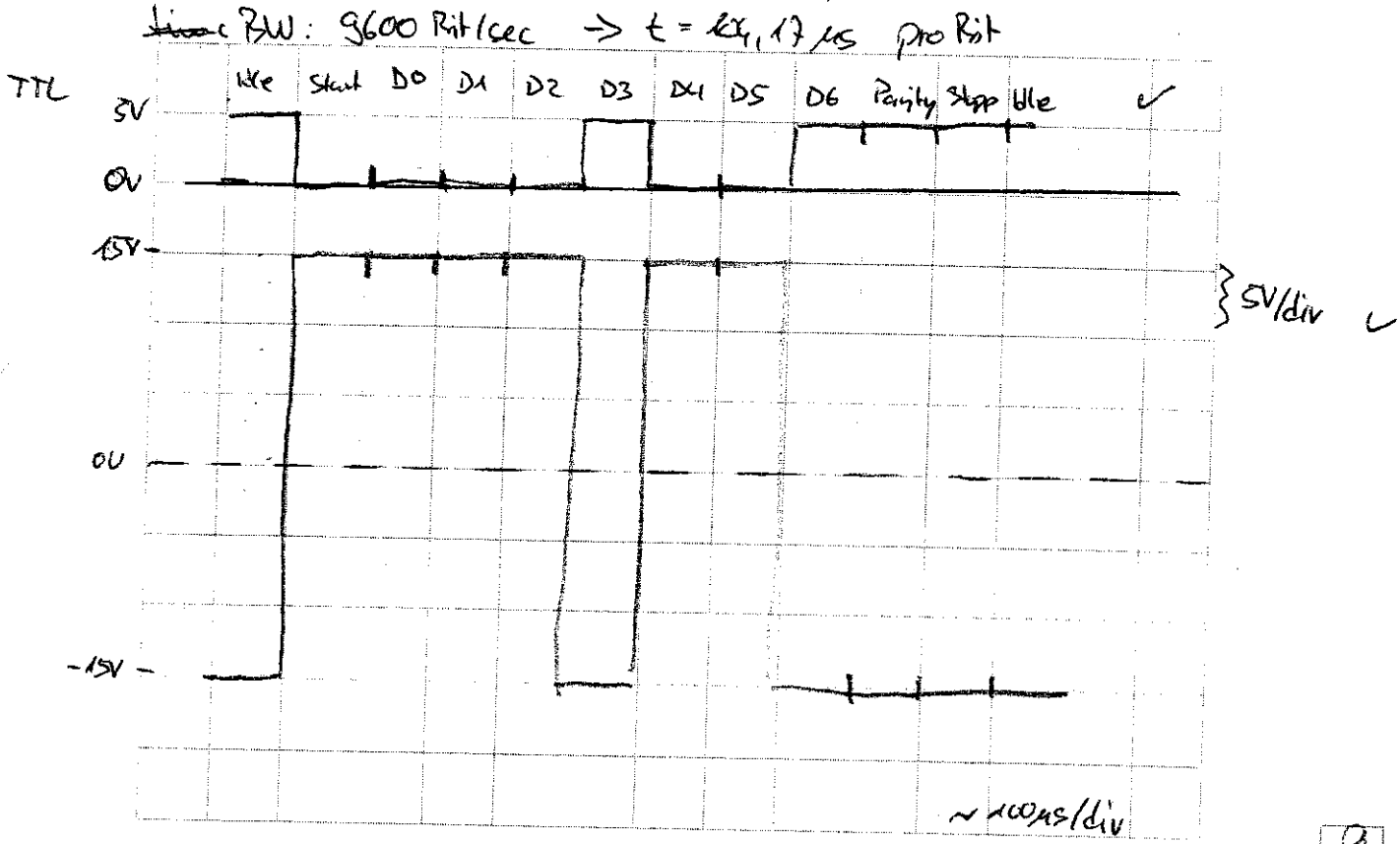
b) Vervollständigen Sie die Tabelle nach dem zuvor initialisierten Protokoll.

ASCII-Zeichen	Dezimal	Hexadezimal	binär	ggf. Paritätsbit
'H'	72	0x48	0100 1000	1
'A'	65	0x41	0100 0001	1
'W'	87	0x57	0101 0111	0

ASCII Tabelle, Hexadezimal (niedrige Bits in den Zeilen, höherwertige in den Spalten)

	...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

c) Zeichnen Sie das ausgehende Signal TxD für das **erste** Zeichen: (H)
- Als Signal vor dem Line-Driver-Chip und als Signal gemäß RS232 Standard.
- Benennen Sie jedes Bit und beschriften Sie die Auflösung [2] der x-Achse in Mikrosekunden/div und die Auflösung der y-Achse in Volt/div.



[2] Grob darstellbare Genauigkeit ist ausreichend.

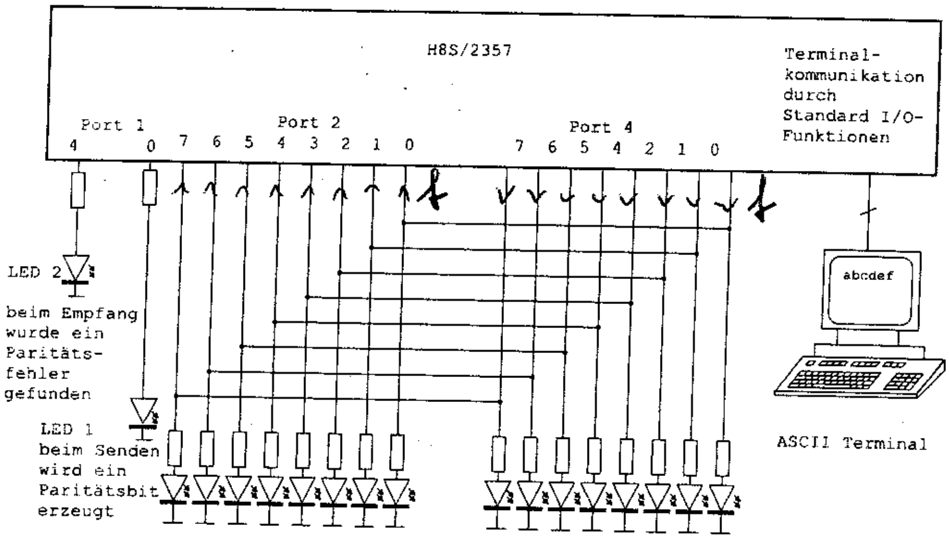


Abbildung 4.1: Aufbau des Paritätsversuchs im Laborsystem

Aufgabe 4: Programmieraufgabe

Die Studenten der HAW wollen die Nutzung von Paritätsbits besser üben und dazu eine eigene Software schreiben. Mit unserem Laborsystem ist ein Versuch aufgebaut (Abbildung 4.1):
- Es wird Port 2 mit Port 4 an allen Pins verbunden.
- Es werden dadurch Bytes von einem Port zum anderen Port parallel übertragen.
- Dabei dient das höchste Bit als Paritätsbit für eine gerade Parität.

PORT2 PORT4 Even Parity Px(7)

- Die Daten werden am ASCII-Terminal mit der Tastatur zeichenweise eingegeben.
- Nach der korrekten Übertragung werden sie erneut am Terminal angezeigt.
- Zwei Leuchtdioden werden zusätzlich benutzt:

P1(0) - LED 1: um das erzeugte Paritätsbit beim Senden anzuzeigen.
P1(4) - LED 2: um das Auftreten eines Paritätsfehlers beim Empfang anzuzeigen.

Schreiben Sie nun gemäß der Teilaufgaben das Programm. Kommentieren Sie ausführlich!

a) Schreiben Sie zunächst eine einfache Wartefunktion für einige ms.

```
#include "mpp1.h"
#include "stdio.h"
void wait (void) {
TPU_CNT2 = 0x0000;        /* Nullen */
TPU_TCR2 = 0x20;          /* TGRA, ⌐, Clk/ , 0010 0000 */
TPU_TSR2 &= 0xFE;         /* reset flag */
TPU_TGR2A = 65536-1;      /* wait for 3,55 ms */
TPU_TSTR = 0xFF;          /* start all counter */
while ((TPU_TSR2 & 0x01) == 0x00);   /* warten */
TPU_TSTR = 0x00;          /* stopp timer */
}
```

Vollkommen O.K. etwas zu viel Schreibarbeit, eine einfache Schleife würde reichen. ✓

2

+12 Zus.P.

Name, Vorname, Matr.Nr.: Tams 1227576 29. Jan. 2008 7

b) Schreiben Sie eine C-Funktion `void send_generate_parity(unsigned char c)`.
- Zu den sieben unteren Bit des Parameters c wird auf dem höchsten Bit zur geraden Parität ergänzt. Eben bei P(2)
- Das Ergebnis wird an den passenden Port ausgegeben (Abbildung 4.1).
- Zusätzlich wird der Wert des Paritätsbits durch die LED 1 angezeigt.

```
void send_generate_parity(unsigned char c){
{
  int i=0;
  int x;
  if( (c&0x01)==0x01) i+=1;
  if( (c&0x02)==0x02) i+=1;
  if( (c&0x04)==0x04) i+=1;
  if( (c&0x08)==0x08) i+=1;
  if( (c&0x10)==0x10) i+=1;
  if( (c&0x20)==0x20) i+=1;
  if( (c&0x40)==0x40) i+=1;

  x = i%2;
  if (x==1) {
  P1DR |= 0x80;
  P1DR = 0x01;
  } else if {
  P1DR &= 0x7F;
  P1DR &= 0xFE;
  }
}
```

/* P2DDR = 0xFF; /* Ausgänge */
P1DDR = 0x81;

/* suchen ob letztes Bit 1 oder 0 */
/* ist. Bei 1, wird i ein hoch */
/* gezählt. */
/* Dies wird für alle 7 Datenbits */
/* getestet */

ist o.k. geht aber "eleganter"
⇒ XOR + Schieben ...

/* Bits sind ungerade, es muss ein */
/* Bit hinzu gesetzt werden */

/* P1(7) wird "1" */
/* LED wird angemacht */

/* P1(7) = 0 */
/* LED aus */

SS (WS)	Semester	Fach	Dozent
07/08	E4	CT	RMS

FSR - Klausurensammlung 8/11

Name, Vorname, Matr.Nr.: Tims 1127576 .... 29. Jan. 2008 8

c) Schreiben Sie eine C-Funktion `unsigned char receive_check_parity(void)`.
- Es wird von dem passenden Port ein Byte parallel eingelesen (Abbildung 4.1). PS(1) SCI2 ? ?
- Es wird die gerade Parität des Bytes geprüft.
- Liegt ein Paritätsfehler vor, so wird das ASCII Zeichen 'X' zurückgegeben.
- Zusätzlich wird ein Paritätsfehler durch die LED 2 angezeigt.
- Liegt kein Paritätsfehler vor, so werden die unteren 7 Bit vom gelesenen Byte und eine '0' dem höchsten Bit zurückgegeben.

```
unsigned char receive_check_parity(void)
{
/* Um die Daten vom Terminal einzulesen wird PS(1) genutzt
   und der SCI2 *  Es wird auch voraus gesetzt, dass
   die Einstellungen vom Terminal (Bitrate usw.)
   richtig eingestellt ist und dass der PS(1) auf reciev gestellt ist */
SCI2_SMR = 0x...;   /* 0110 xxxx , 7Bit, Even, Clk?; Stopp? */
SCI2_BRR = ;
SCI2_SCR = 0x10;    /* reciev enable */

char c;  int i=0;
c = PORT2 ;          /* einlesen */
if((c&0x01)==0x01) i+=1;     /* check Parity */
if((c&0x02)==0x02) i+=1;
if((c&0x04)==0x04) i+=1;
if((c&0x08)==0x08) i+=1;
if((c&0x10)==0x10) i+=1;
if((c&0x20)==0x20) i+=1;
if(c&0x40)==0x40) i+=1;
if(c&0x80)==0x80) i+=1;

i = i%2 ;
if(i==1) {           /* Fehler ist aufgetreten */
  P4DR=('X' & 0x7F);   /* Ausgabe des Zeichens 'X' */
  P1DR = 0x10;         /* LED 4 wird angemacht */
} elsif {
P4DR = (c & 0x7F);   /* Ausgabe des Zeichens und
                        0 vorne weg */
}
}
```

! Port 4 ist input only!
→ Sie hätten den return value nutzen sollen!

7/9

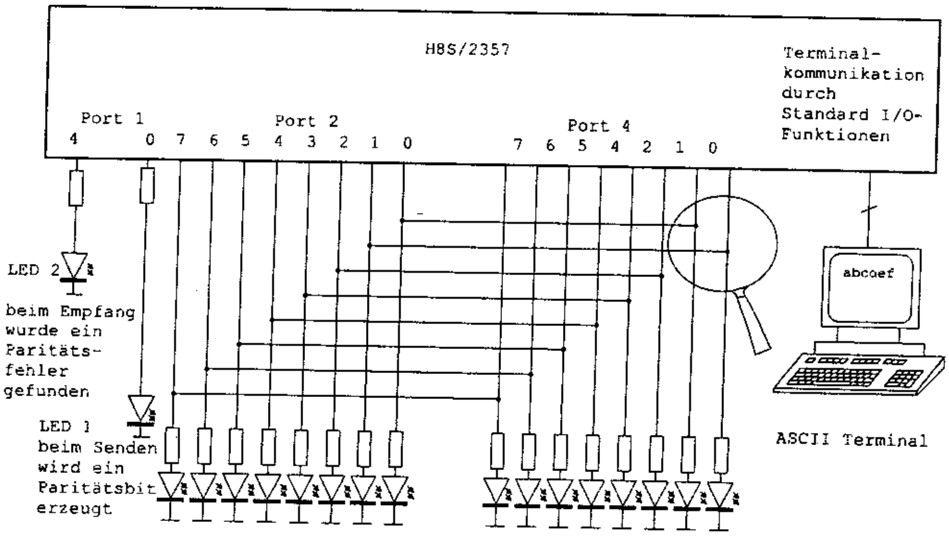


Abbildung 4.2: Aufbau des Paritätsversuchs mit Verdrahtungsfehler

e) Leider ist ein kleiner Verdrahtungsfehler aufgetreten (Abbildung 4.2). Sie möchten die Verdrahtung <u>nicht</u> ändern. Daher schreiben Sie eine Korrekturfunktion `unsigned char correct(unsigned char)`. Der Parameter ist das zu korrigierende Byte, die Funktionsrückgabe das korrigierte Byte.

Markieren Sie alle alternativen Möglichkeiten in main(), wo Sie diese Korrekturfunktion aufrufen könnten.

```
unsigned char correct(unsigned char input)
{
  unsigned char temp;
  unsigned char x=0, y=0;
  temp = input;
  x = input & 0x01;        /* ersten Bit */
  y = input & 0x02;        /* zweites Bit */
  x = x<<1;                /* Bits werden umrecht geschoben */
  y = y>>1;
  temp = temp & 0xFC;      /* letzte beiden Bits werden gelöscht */
  temp = temp|x|y;         /* das komplette Bit wird zusammen gefasst */
  return temp;
}
```

prima!

9

53 + 1

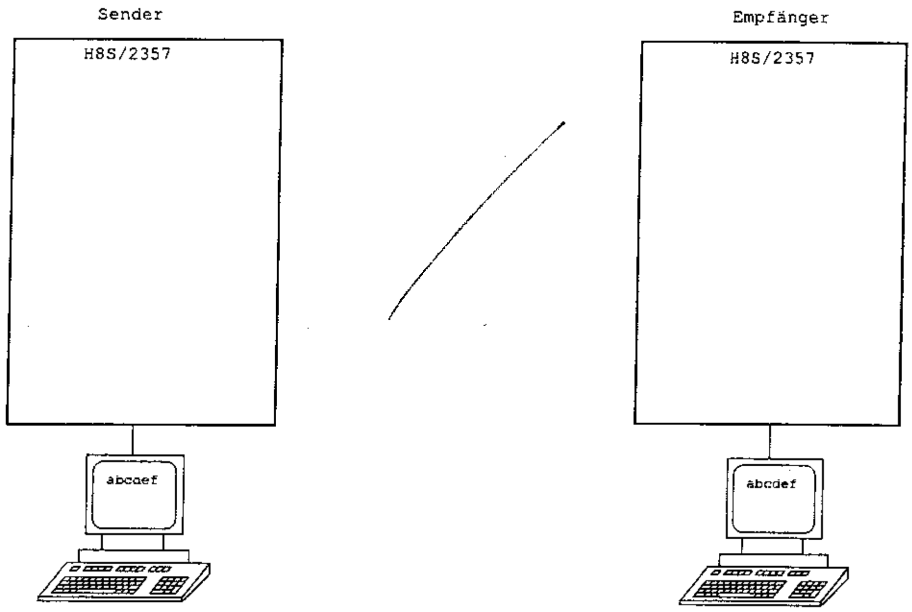


Abbildung 4.3: Parallele Übertragung mit Parität - Blockschaltbild bitte ergänzen !

Zusatz f) Das Programm lt. Aufgabe 4 ist sehr einfach für ein Sender- und ein Empfängersystem (zwei Controllersysteme) anzupassen. LED1 und LED2 können wegfallen. Ergänzen Sie eventuell notwendige Leitungen zur Synchronisation von Sender und Empfänger, die Sie beliebig an noch freie Pin von Port1 anschließen können. Skizzieren Sie die Verbindungen in Abbildung 4.3.

Zusatz g) Schreiben Sie zwei neue Hauptprogramme. Nutzen Sie ansonsten die vorhandenen Funktionen. Ein Hauptprogramm dient für die Terminal-Eingabe und parallele Sendung. Das andere Hauptprogramm übernimmt den parallelen Empfang und Terminal-Ausgabe. Nutzen Sie ggfs. die Synchronisationsleitungen lt. Vorschlag aus Teilaufgabe f).