Name, Vorname, Matr.Nr.: 29. Jan. 2008 1

| 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 |
| | 17 | 7 | 19 | 35 | 0 | 78 |

## Aufgabe 1: Programmanalyse

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 Werte von TGRA u TGRB, Datenbreite */

  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;          /* clear by TGRB, 5 , CLK/64 */

  TPU_TIOR2=0x9A;         /* Input capture at bei TGRB bei TGRA input cap. at */

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

  while (1)               /* Endlosschleife um den folgenden Programmblock */
  {
    TPU_TSR2 &=0xFC;      /* Flage TGFA u TGFB auf 0 setzen */

    while((TPU_TSR2 & 0x01) == 0x00); /* Warte bis TGFA nicht mehr 0 ist */

    a=TPU_TGR2A;          /* Werte von TGR2A ablesen und in a speichern */

    while((TPU_TSR2 & 0x02) == 0x00); /* warte bis TGFB nicht mehr 0 ist */

    b=TPU_TGR2B/2;        /* Werte von TGR2B durch 2 dividieren und in b speichern */

    if (a<b) P2DR=0x01; else P2DR=0x02;
    /* a < b => P2(0) = 1 und die Reste = 0 */
    /* a >= b => P2(1) = 1, die Reste = 0 */

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

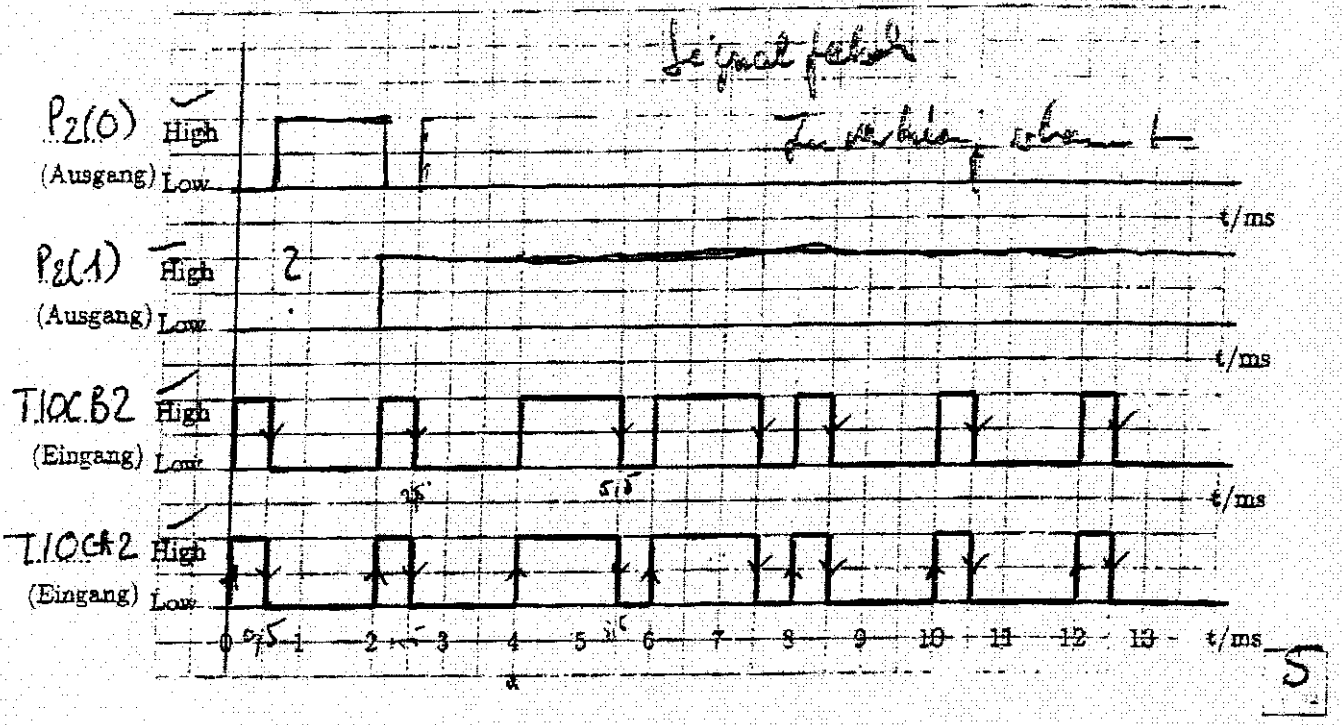
9

Name, Vorname, Matr.Nr.: ..... 29. Jan. 2008 2

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

Eingabe: T.IOCB2 T.IOC*2 Ausgabe: P2(0) P2(1)

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




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

Wenn an die Eingangspins T.IOCA2 T.IOCB2 dasselbe periodische Signal angelegt wird, dann kann man muss man die aktive Flanke unterscheiden.

unvollständig

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

$T_{max}$ =

## 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.

$\overline{CS}_{ROM} = A_{23} \vee A_{22}$

3

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

$\overline{CS}_{RAM} = (A_{23} \vee \overline{A_{22}}) \wedge (\overline{A_{23}} \vee A_{22})$

4

7

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

## Aufgabe 3: Serial Interface

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

```
SCI2_SMR = 0x74;   /* 7 0 1 * / / * P = 1 * /
SCI2_BRR = 0x3B;   /* BRR = 59 → 9600 bits/s * / (Fehler)
SCI2_SCR = 0x20;
```

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)
{
  unsigned int Buchstaben[3]={0x48,0x41,0x57};
  int i=0;
  for(i=0; i < 2; i++)
  {
    while((SCI2_SSR &0x80)==0x00);
    /** warte bis TDR frei ist, d.h TDRE = 1 * /
    SCI2_TDR = Buchstaben[i];
    /* werte werden in TDR geschrieben, zum Senden * /
    SCI2_SSR &= ~0x08;
    /* TDRE wird wieder zu 0 gesetzt * /
  } /* End for * /
} /* End Hauptprog. * /
```

6.

[1] 18.432 MHz Systemtakt

Name, Vorname, Matr.Nr.: ..... 29. Jan. 2008 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 | 100 0001 | 1 |
| 'W' | 87 | 0x57 | 1010111 | 0 |

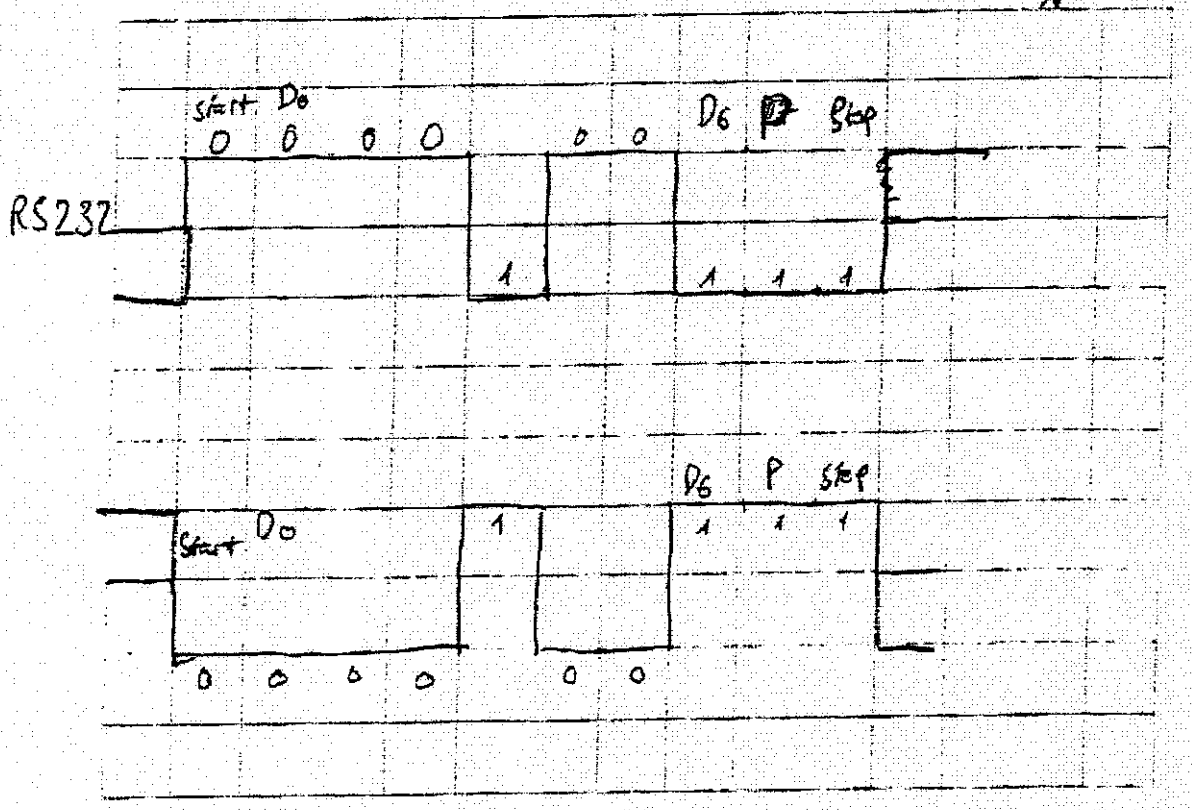
5

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:
- 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.

fehlt bei Bes




1

1,5

[2] Grob darstellbare Genauigkeit ist ausreichend.

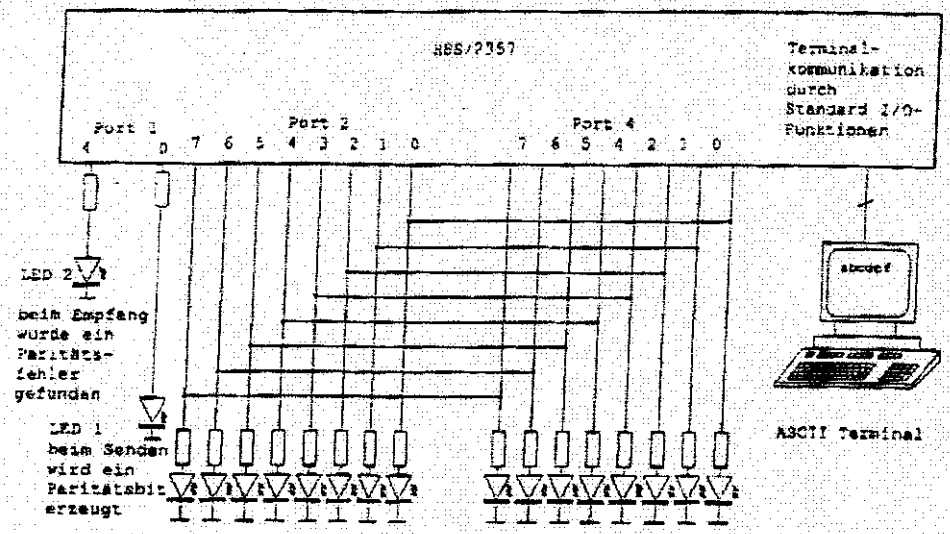



**Abbildung 4.1:** Aufbau des Paritätsversuchs im Laborsystem

## 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.

- 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:
  - LED 1: um das erzeugte Paritätsbit beim Senden anzuzeigen.
  - 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 warte(int time){
  int i=0;
  for(i=0;i<time;i++);
}  /* gemäß Laborversuche haben wir ermittelt */
   /* time = 500 entspricht 5ms */
```

2

Name, Vorname, Matr.Nr. ..... 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.
- 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)}
{
   unsigned int einsen = 0; modulo = 0;
   unsigned char temp_c = 0;

temp_c = c;
while ( temp_c > 1 )
{     modulo = temp_c % 2 /* die Einsen ermittle */
   einsen += modulo; /* Summe von "1" */
      temp_c = (temp_c - modulo)/2; /* nächste Schritt */
} /* End while */
   einsen = einsen + 1; /* 1 % 2 = 1 => muss zusätzlich */
                        /* 1 addieren */
if (einsen % 2 ) /* abfragen, ob die Summe von 1 odd oder even */
   {
      c = c & (~0x80); /* Wenn ungerade, MSB zu 0 */
      P1DR = P1DR | 0x00; /* LED1 auf 0 setzen */
   }
   else .....!
      P2DR = c;

} /* End Funktion */
```

7

Name, Vorname, Matr.Nr.: ..... 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).
- 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)
{ int ~~eisen~~ einsen = 0; modulo = 0;
  unsigned char temp = 0;
  temp = PORT 4 ;
  while (temp > 1)      /* siehe vorherige ~~letzte~~ Seite */
  {  modulo = temp % 2;
     einsen += modulo;
     temp = (temp - modulo) / 2;
  }
  einsen = einsen + 1;
  if (einsen % 2) { P1DR 1 = 0x02; /* LED 2 aus */
     return 'x'; /* Fehler meldung */
  }
  else temp = (PORT 4 & (~0x80)); /* 7 Bit vom Byte und
                                     eine 0 mit zurück
                                     gegeben */
  return temp;
}
```

9

d) Schreiben Sie nun eine C-Funktion `void main (void)`.
- Es werden die Ports geeignet initialisiert.
- Dannach laufen zyklisch wiederholend folgende Teilschritte [3] ab:
  1) Einlesen eines Zeichens vom Terminal, wenn dort eine Tasteneingabe erfolgt. Nutzen Sie dazu eine Standard I/O-Funktion der C-Entwicklungsumgebung.
  2) Absenden des Zeichens mit passender Parität von einem Port .
  3) Warten für eine sehr kurze Signallaufzeit zwischen den Ports.
  4) Empfang des Zeichens am anderen Port und Prüfung der Parität.
  5) Ausgeben des Zeichens an das Terminal, ebenfalls mit einer der Standard I/O-Funktionen.

```
void main(void)
{
  P2DDR = 0xFF; P2DR=0x00; /*Ports initialisieren */
  P1DDR = 0xFF; P1DR=0x00;
  char zeichen=0; ergebnis = 0;
  scanf(&zeichen); /* Einlesen vom Terminal */
  send_generate_parity(zeichen);
    /* Parität erzeugen */ /* Korrektur kann ausgeführt werden */
  warte(500); /* warten */
ergebnis=receive_check_parity(); /* Checken und Ergebnis erzeugen */
  printf("%c", ergebnis); /* Ausgabe in Terminal */
}
```

[3] Nutzen Sie möglichst schon programmierte Funktionen.

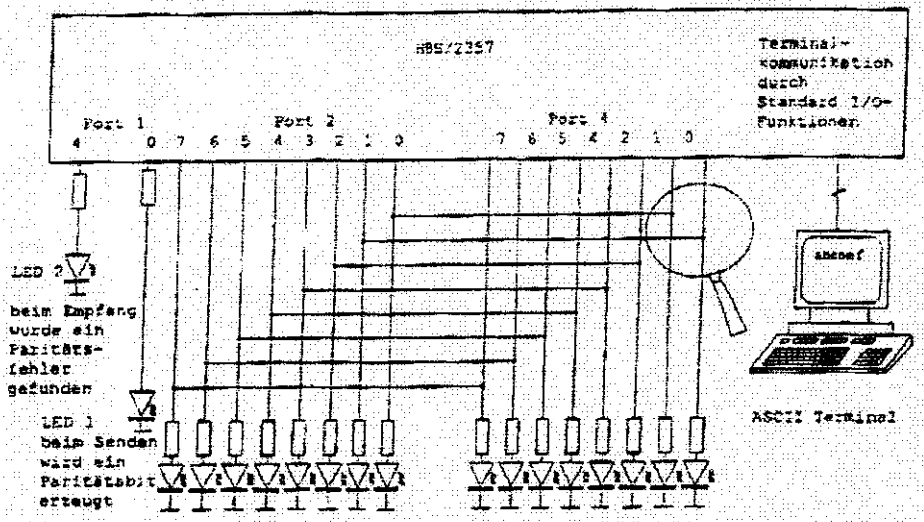



Abbildung 4.2: Aufbau des Paritätsversuchs mit Verdrahtungsfehler

e) Leider ist ein kleiner Verdrahtungsfehler aufgetreten (Abbildung 4.2). Sie möchten die Verdrahtung nicht 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)
{
  unsigned char temp=0; temp0=0; temp1=0;
  temp = P2DR;
  temp0 = temp & 0x01; /* P2(0) nehmen in temp0 */
  temp1 = temp & 0x02; /* P2(1) ____ temp1 */
  temp0 = temp0 << 1; /* Tauschen 2 Bits */
  temp1 = temp1 >> 1;
  temp0 = temp0 | temp1 /* 2 Bit zusammen in Temp0
  temp = temp & (~0x03); /* fügen */
  temp = temp | temp0; /* Löschen 2 Bit LSB von temp und fügt 2 Bit von temp0 */
  P2DR = temp; /* Korrigieren */
  return temp; }
```

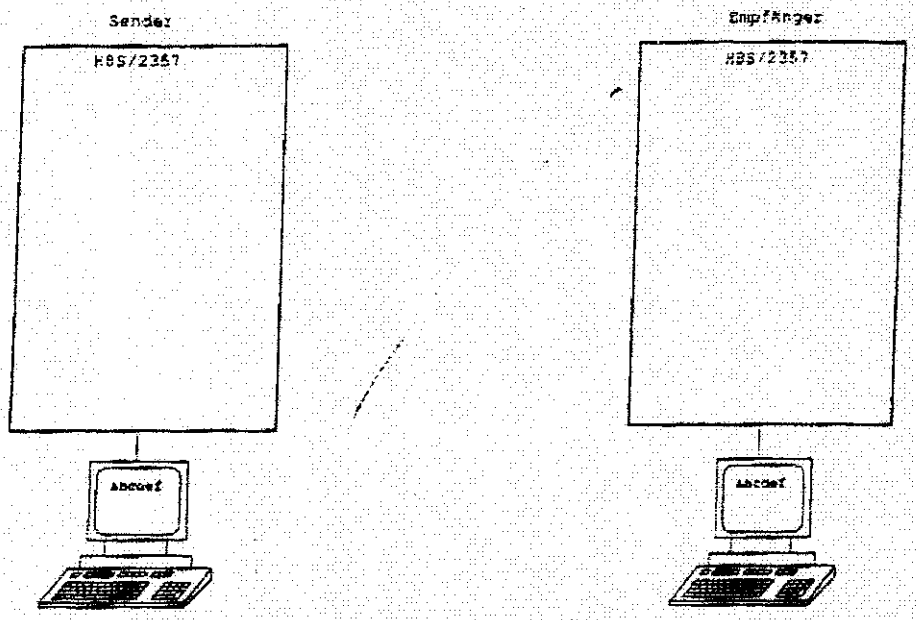



**Abbildung 4.3:** Parallele Übertragung mit Parität - Blockschaltbild bitte ergänzen !

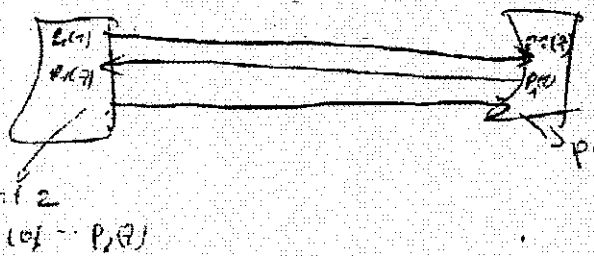
Zusatz f) Das Programm lt. Aufgabe 4 ist sehr einfach für ein Sender- und ein Empfängersystem (zwei Controllersysteme) anzupassen. LED1 und LED2 können wegfallen. Ergänzen Sie eventuell notwendige Leitungen zur Synchronisation von Sender und Empfänger, die Sie beliebig an noch freie Pin von Port1 anschließen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

Zusatz:

```
void main (void)
{
  unsigned char c;
  P1DDR = 0x72; P2DDR = 0xFF;
  while(1) {
  while ((PORT1 & 0x80) == 1);
  c = getchar(); send_generate_parity(c);
  P1DR |= 0x20; // set Data valid signal 1
  while ((PORT1 & 0x80) == 0);
  P1DR &= ~0x20;
}}
void main (void)
{ unsigned char c;
  P1DDR = 0x72; P2DDR = 0xFF;
  while (1) {
  while ((PORT1 & 0x80) == 0);
  c = receive_clock_parity(); putchar(c);
  P2DR1 = 0x20;
  while ((PORT1 & 0x80) == 1)
  P1DR &= ~0x20 // Data quit signal 0
}}
```

Zusatz



b)
```
void send_generate_parity(unsigned char c){
  unsigned char p;
  p = (c>>6) ^ (c>>5) ^ (c>>4) ^ (c>>3) ^ (c>>2) ^ (c>>1) ^ (c);
  p &= 0x01;
  if (p) P1OR1 = 0x01; else P1DR &= 0xFE;
  if (p) P2DR = c|0x80; else P2DR = c&0x7F;
  /* wenn die parität der sieben bit ungerade
     dann 1 an 8. stellen setzen */
}
```

c)
```
unsigned char receive_check_parity (void){
  unsigned char p;
  p = (PORT4>>7) ^ (PORT4>>6) ^ (PORT4>>5)
  ^ (PORT4>>4) ^ (PORT4>>3) ^ (PORT4>>2) ^
  (PORT4>>1) ^ (PORT4);
  p &= 0x01;
if (!p) P1DR &= 0x10; else P1DR |= 0x10;
  /* kein ungerade parität d.h. gerade ist ok
if (!p) return (PORT4|0x80); else return ('x');
/* wenn parität der Acht bit an PORT4 gerade
  dann Rückgabe so. ist Zeile 'x';
}
```

c1)

```
void main (void)
{
  unsigned char c;
  P1DDR=0x11; P2DDR=0xFF;
  while (1)
  {
    c=getchar(c);
    send_generate_parity(c);
    waite(1000);
    c=receive_check_parity(c);
    c=correct(c);
    putchar(c);
  }
```

c2)

```
unsigned char correct(unsigned char)
{ char ch;
  ch=c;
  if((c&0x01)==0x00) ch&=~0x02; else
    ch|=0x02;

if((c&0x02)==0x00) ch&=~0x01; ch|=0x0;
return(ch);
}
```