Fachbereich Informatik

Kurs 1708 „Technische Informatik II“

und

Kurs 1705 „Mikrorechner-Technik“

Musterlösungen zur Nachklausur am 17.03.2001

Aufgabe 1: (10 Punkte)

Geben Sie für die folgenden Aussagen an, ob sie richtig oder falsch sind.

	richtig	falsch
a) In einem *N-Way Set Associative Cache* wird der Eintrag zur Aufnahme eines neuen Datums eindeutig durch das **Indexfeld** in seiner Speicheradresse bestimmt. *Für jeden Eintrag stehen n Plätze zur Verfügung, die assoziativ verwaltet werden.*	☐	X
b) Je größer ein (zusammenhängender) Speicherbereich ist, desto mehr Adreßsignale braucht ein **Adreßdecoder** für die Erzeugung eines Auswahlsignals für diesen Bereich (z.B. *Chip Select-Signal*). *Je größer der Bereich, desto weniger Signale werden benötigt. Die Hälfte des Adreßbereichs kann z.B. durch ein einziges Adreßsignal selektiert werden.*	☐	X
c) Das Ausgangssignal eines **Digital/Analog-Wandlers** kann unendlich viele verschiedene (analoge) Spannungswerte annehmen. *Das Ausgangssignal eines n-bit-Wandlers kann nur 2n verschiedene Werte annehmen, ist also eine Treppenfunktion.*	☐	X

Ergänzen Sie die folgenden Sätze:

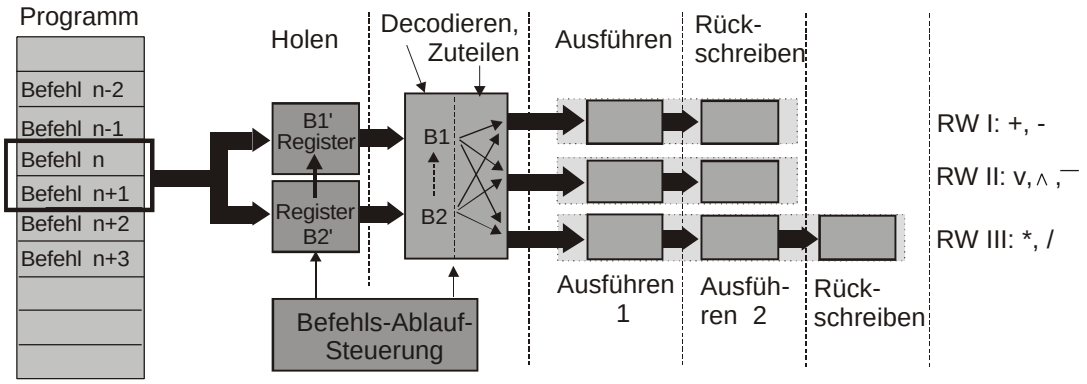
d) Der wesentliche Nachteil des **Rückschreibverfahrens** gegenüber dem **Durchschreibverfahren** besteht darin, daß *der Erhalt der Cache-Kohärenz schwieriger zu gewährleisten ist und mehr Aufwand für die Cache-Steuerung erfordert.*

d) Unter dem Verfahren des **verteilten Auffrischens** versteht man *die gleichmäßige Verteilung der Auffrischzyklen über ein Auffrischintervall, so daß nach jeweils einer festen Zeit (15,6 ms) eine neue Speicherzeile aufgefrischt wird.*

e) **Digitale Signalprozessoren** sind Mikroprozessoren, die *für die schnelle Berechnung von Algorithmen der digitalen Signalverarbeitung optimiert sind* und dazu insbesondere
über eine Harvard-Architektur und eine Multiplizier/Akkumulier-Einheit verfügen.

Aufgabe 2: (30 Punkte)

Ein Mikroprozessor besitze drei einfache Rechenwerke (RW): RW I für die Addition und Subtraktion, RW II für die logischen Operationen ‚Und', ‚Oder' und ‚Nicht' sowie RW III für die Multiplikation und Division. Jedes Rechenwerk verarbeite seine Befehle in separaten Pipelines, die jedoch die Befehls-Holphase und –Decodierphase gemeinsam haben. RW I, II führen ihre Befehle in einer Pipelinephase aus, RW III benötigt dafür zwei Phasen. Daran schließt sich in jeder Pipeline eine Rückschreib-Phase an.



Die Befehlsablaufsteuerung kann in jedem Takt (maximal) zwei unmittelbar hintereinander folgende Befehle aus dem Programm entnehmen und in einem Befehlsregister-Paar zwischenspeichern, die in der nächsten Pipelinephase decodiert werden können. Der Decoder teilt die Befehle den ihrem Typ entsprechenden Pipelines der Rechenwerke zu – sofern diese im augenblicklichen Takt dafür zur Verfügung stehen[1]. Alle benötigten Operanden befinden sich in Registern und stehen am Ende der Decodierphase den Rechenwerken zur Verfügung. Das gilt insbesondere auch für Ergebnisse, die im selben Takt in ihr Register zurückgeschrieben werden. Bei der Zuteilung wird die folgende **Strategie** realisiert:

1. Befehlspaar (B1‘, B2‘) in die Befehlsregister holen;
2. Befehlspaar (B1‘, B2‘) zum Decoder bringen [(B1‘, B2‘) ⇒ (B1, B2)] und decodieren; gleichzeitig nächstes Paar (B1‘, B2‘) in die Befehlsregister holen;
3. Zuteilung:
   a) B1 kann nicht zugeteilt werden ⇒
      (B1, B2) wartet einen Takt - und damit auch (B1‘, B2‘); weiter mit 3.
   b) (B1, B2) kann gemeinsam zugeteilt werden ⇒
      beide Befehle zuteilen, weiter mit 2.
   c) B1 kann, B2 kann nicht zugeteilt werden ⇒
      B1 zuteilen, B2 einen Takt warten lassen;
      B1‘ aus Register nach Decoder und neues Paar bilden: aus (B2, B1‘) wird (B1, B2);
      B2‘ im Befehlsregister [aus B2 ‘ wird B1‘] verschieben und neues B2‘ holen;
      weiter mit 3.

[1] In einem Takt können z.B. nicht zwei Addierbefehle zugewiesen werden.

Wird in einem Takt einer Pipeline kein neuer Befehl zugewiesen, so speist die Ablaufsteuerung automatisch einen NOP-Befehl in sie ein.

a) Können zwei Befehle in einer Reihenfolge zur Ausführung an die Rechenwerke <u>übergeben</u> werden, die von der durch den Programmablauf vorgegebenen Reihenfolge abweicht (*Out-of-Order Execution*) ?

Nein. Von zwei aus dem Speicher geladenen Befehlen nimmt immer der ‚vorausgehende' die Position von B1 ein. Die Zuteilungsstrategie stellt danach sicher, daß von einem Befehls-Paar (B1,B2) zwar B1, aber nie B2 allein zugeteilt werden kann.

b) Kann die Bearbeitung zweier Befehle in einer Reihenfolge <u>beendet</u> werden, die von der durch den Programmablauf vorgegebenen Reihenfolge abweicht (*Out-of-Order Completion*) ?

Ja. Von einem Befehlspaar (B1, B2), das in einem Takt zugeteilt wird und in dem z.B. B1 ein Multiplizier- oder Dividierbefehl und B2 ein Addier- oder Subtrahierbefehl ist, wird der Befehl B2 einen Takt früher beendet als B1.

c) Es sei vorausgesetzt, daß beim Auftreten einer Datenabhängigkeit die Pipeline des Rechenwerks, das den im vorgegebenen Programmablauf folgenden Befehl enthält, solange mit NOP-Befehlen gefüllt wird, bis die Abhängigkeit aufgelöst wird.
Geben Sie in der unten stehenden Tabelle den Verlauf der Befehlsbearbeitung an, wenn die folgenden Befehle eingelesen werden. (Es reicht, die Nummer der Befehle einzutragen bzw. ‚-' für den NOP-Befehl.) Vereinfachend sei angenommen, daß zum Takt 0 alle Pipelinestufen ‚leer' sind, d.h. NOP-Befehle ausführen. Geben Sie auch an, nach welcher der Regeln 3a) – 3c) die Zuteilungen jeweils geschehen.

1.	NOT	R5,R5	; R5 := NOT R5
2.	AND	R4,R4,R2	; R4 := R4 $\wedge$ R2
3.	DIV	R9,R2,R1	; R9 := R2 / R1
4.	MUL	R3,R1,R2	; R3 := R1 * R2
5.	ADD	R0,R3,R0	; R0 := R3 + R0
6.	AND	R8,R3,R8	; R8 := R3 $\wedge$ R8
7.	OR	R1,R5,R6	; R1 := R5 $\vee$ R6
8.	SUB	R4,R9,R1	; R4 := R9 – R1
9.	ADD	R7,R6,R7	; R7 := R6 + R7
10.	DIV	R6,R6,R2	;
11. ...	beliebige Befehle		

d) Wie viele andere Befehle müssen zwischen zwei Befehlen, zwischen denen eine Datenabhängigkeit besteht, wenigstens liegen, um ein Pipelinehemmnis zu vermeiden[2] ?

Besteht die Abhängigkeit zwischen einem Multiplizier-/Dividierbefehl und einem (beliebigen) nachfolgenden Befehl, so müssen zwischen beiden Befehlen mindestens zwei Takte (für die Ausführungsphase des ersten Befehls), also wenigstens vier Befehle liegen. Ist der erste Befehl kein Multiplizier-/Dividierbefehl, so reichen zwischen beiden Befehlen ein Takt bzw. zwei Befehle.

[2] Auch das Einfügen von NOP-Befehlen ist ein Pipeline-Hemmnis.

Tabelle zu Aufgabe 2c)

Takt	Holen		Decodieren		RW I		RW II		RW III			Zuteilung
	B2‘	B1‘	B2	B1	A.	R.	A.	R.	A. 1	A. 2	R	a), b), c)
0	-	-	-	-	-	-	-		-	-	-	-
1	2	1	-	-	-	-	-	-	-	-	-	-
2	4	3	2	1	-	-	-	-	-	-	-	-
3	5	4	3	2	-	-	1	-	-	-	-	c)
4	7	6	5	4	-	-	2	1	3	-	-	b)
5	8	7	6	5	-	-	-	2	4	3	-	a)
6	8	7	6	5	-	-	-	-	-	4	3	a)
7	8	7	6	5	-	-	-	-	-	-	4	a)
8	10	9	8	7	5	-	6	-	-	-	-	b)
9	11	10	9	8	-	5	7	6	-	-	-	c)
10	11	10	9	8	-	-	-	7	-	-	-	a)
11	12	11	10	9	8	-	-	-	-	-	-	c)
12	12	11	10	9	-	8	-	-	-	-	-	a)
13	14	13	12	11	9	-	-	-	10	-	-	b)

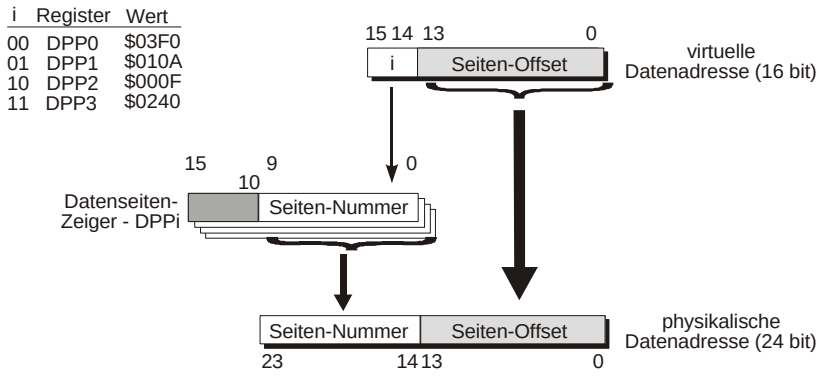
RW I,II,III: Rechenwerk I,II,III, A.: Ausführungsphase, R.: Rückschreibphase

e) Stellen Sie die unter c) angegebenen Befehle so um, daß die Funktion des Programmstücks unverändert bleibt und bei seiner Bearbeitung keine Pipeline-Hemmnisse mehr auftreten, insbesondere also in jedem Takt zwei Befehle zugeteilt werden. (Geben Sie auch die Nummern der Befehle wie unter c) an.)

Takt	Nr.	Befehl
1	4	MUL R3,R1,R2
	1	NOT R5,R5
2	3	DIV R9,R2,R1
	2	AND R4,R4,R2
3	7	OR R1,R5,R6
	9	ADD R7,R6,R7
4	5	ADD R0,R3,R0
	6	AND R8,R3,R8
5	8	SUB R4,R9,R1
	10	DIV R6,R6,R2

Aufgabe 3: **(20 Punkte)**

Ein Mikrocontroller besitze einen 16-bit-Datenbus sowie einen 24-bit-Adreßbus. Er verfüge über eine rudimentäre Form der „virtuellen Speicherverwaltung“, die im Bild unten skizziert ist. Der Adreßbereich wird dabei in 16 kbyte große ‚Seiten‘ eingeteilt. Für die Auswahl von bis zu vier aktuellen Seiten werden vier 16-bit-Register DPPi (Datenseiten-Zeiger, *Data Page Pointer*) benutzt, in denen jedoch nur die unteren 10 Bits ausgewertet werden. Die im Befehl angegebene ‚virtuelle‘ Adresse ist 16 bit lang. Ihre höchstwertigen beiden Bits selektieren das verwendete DPP-Register. Die Speicheradresse (physikalische Adresse) ergibt sich aus der ‚Verkettung‘ (Konkatenation) des DPP-Registerinhalts und der unteren 14 Bits der virtuellen Adresse, dem Seiten-Offset.



a) Wie viele Seiten kann der Datenspeicher maximal umfassen ?
Können Sich zwei verschiedene Seiten ‚überlappen‘, also wenigstens eine Speicherzelle gemeinsam enthalten ? (Begründung !)

Aus der Länge der Seitennummer von 10 bit $\Rightarrow 2^{10}$ = 1024 Seiten.

Die Seiten können sich nicht überlappen, da die physikalische Adresse jeder Speicherzelle auf eindeutige Weise aus Seitennummer und Offset gebildet wird.

b) Welche physikalische Adresse wird durch die virtuelle Adresse $B400 angesprochen, wenn die DPP-Register die im oben stehenden Bild angegebene Wert besitzen ? (Herleitung angeben!)

Durch die beiden oberen Bits der virtuellen Adresse (‚10‘ = 2) wird DPP2 selektiert. Die Konkatenation aus den unteren 10 Bits von DPP2 und dem Offset $3400 ergibt binär:

$\underbrace{\underline{00\ 0000\ 1111}}_{\text{DPP2}}\ \underbrace{\underline{11\ 0100\ 0000\ 0000}}_{\text{Offset}}$ = 0000 0011 1111 0100 0000 0000 = $03F400

c) Welcher Wert muß ins DPP3-Register geschrieben werden, wenn damit die Speicherseite selektiert werden soll, die die Speicherzelle $A7CE38 enthält ? Wie lautet die virtuelle Adresse der Speicherzelle ? (Herleitung angeben!)

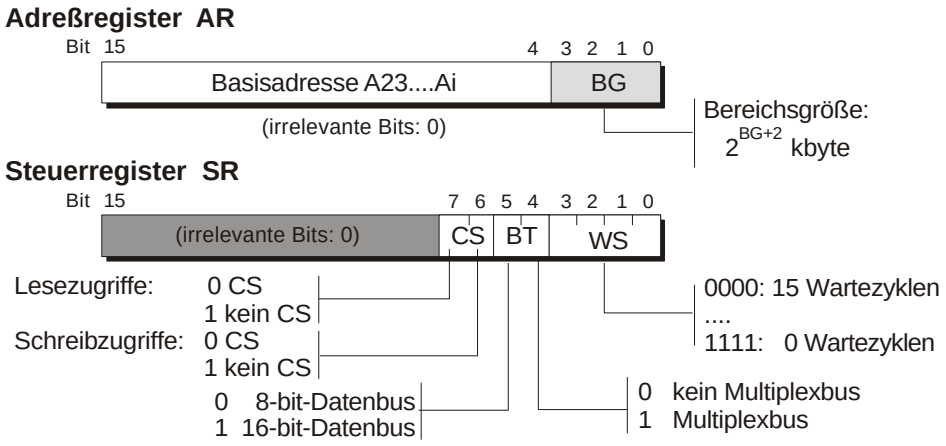
Zerlegung der binären Adresse ergibt:

$A7CE38 = $\underbrace{\underline{1010\ 0111\ 11}}_{\text{DPP3}}$ / $\underbrace{\underline{00\ 1110\ 0011\ 1000}}_{\text{Offset}}$

Damit ergibt sich: DPP3 = 0000 00$\underline{10\ 1001\ 1111}$ = $029F

Virtuelle Adresse: 11 $\underline{00\ 1110\ 0011\ 1000}$ = $CE38

d) Die Systembus-Schnittstelle des Mikrocontrollers verfüge über eine Schaltung, die die Festlegung mehrerer externer Adreßbereiche erlaubt. Jedem Adreßbereich ist ein Adreßregister AR (s. Bild unten) zugeordnet, in dem die Größe des Bereichs definiert und die oberen Bits seiner Basisadresse bestimmt werden. In einem Steuerregister SR kann die Anzahl der von der Schnittstelle bei Zugriffen auf diesen Bereich automatisch eingefügten Wartezyklen angegeben werden. Weiterhin kann dort festgelegt werden, ob die Schnittstelle bei Lese- oder Schreibzugriffen (oder beiden) auf diesen Adreßbereich ein *Chip-Select*-Signal CS erzeugen soll oder nicht.



Weiterhin kann der Bustyp (BT) für den Adreßbereich spezifiziert werden (wobei die höchstwertigen Adreßsignale A23...A16 in allen Fällen über eigene Leitungen ausgegeben werden):

- <u>kein</u> Adreß/Daten-Multiplexbetrieb mit Adreßbus A15...A0 und (gemultiplextem) 8-bit-Datenbus D7...D0,
- <u>kein</u> Adreß/Daten-Multiplexbetrieb mit Adreßbus A15...A0 und Datenbus D15..D0,
- Adreß/Daten-Multiplexbetrieb mit 16-bit-Adreß/Datenbus AD15...AD0,
- Adreß/Daten-Multiplexbetrieb mit 8-bit-Adreßbus für die ‚mittleren' Adreßsignale A15...A8 und gemultiplextem Adreß/Datenbus AD7...AD0.

Aufgabe:
Geben Sie in binärer und hexadezimaler Form die Belegung der Register AR und SR an, wenn im Adreßbereich ab $FC8000 (nur) ein 32-kbyte-Speicherbaustein[3] mit der Datenbreite 8 bit angesprochen werden soll, der für jeden Zugriff fünf Wartezyklen benötigt und nicht beschrieben werden darf.

BG = 3 ⇒ Bereichsgröße = 2^{BG+2} kbyte = 2^5 kbyte = 32 kbyte

AR = 1111 1100 1<u>000</u> <u>0011</u>$_2$ = $FC83 (<u>0</u> irrelevant, <u>0011</u> Bereichsgröße)

Bustyp: 00 – kein Multiplexbus mit 8-bit-Datenbus

SR = <u>0000 0000</u> 0100 <u>1010</u>$_2$ = $004A (<u>0</u> irrelevant, <u>1010</u> Wartezyklen)
/ / —

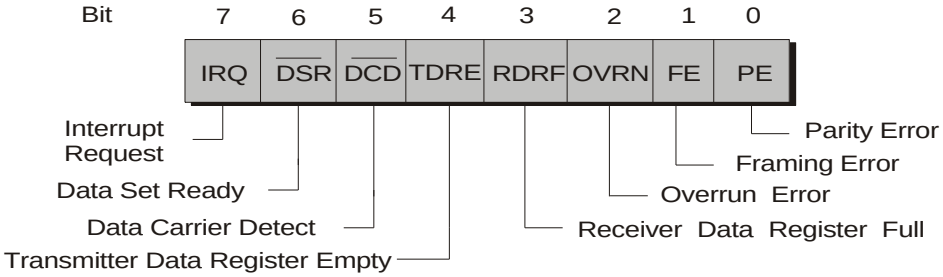
[3] Dabei müssen Sie von einem Standard-Baustein ausgehen, wie er im Kurs beschrieben wurde. Ihre Lösung soll ohne den Einsatz weiterer Bausteine, wie Latches, Gatter usw., auskommen.

CS- RW BT

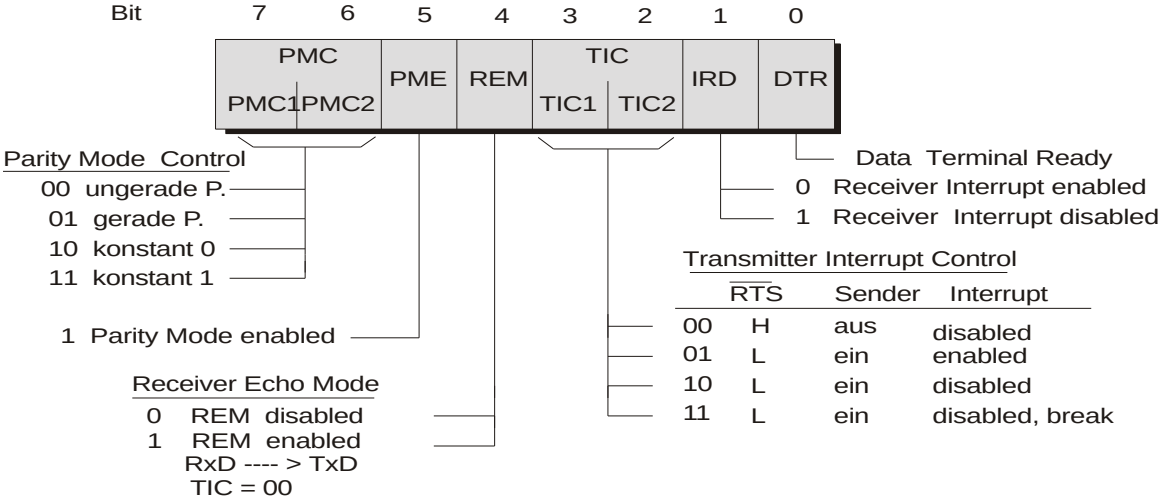
Aufgabe 4: (20 Punkte)

In dieser Aufgabe geht es um die „Programmierung“ eines asynchronen seriellen Schnittstellenbausteins. Dazu finden Sie zunächst seinen Registersatz.

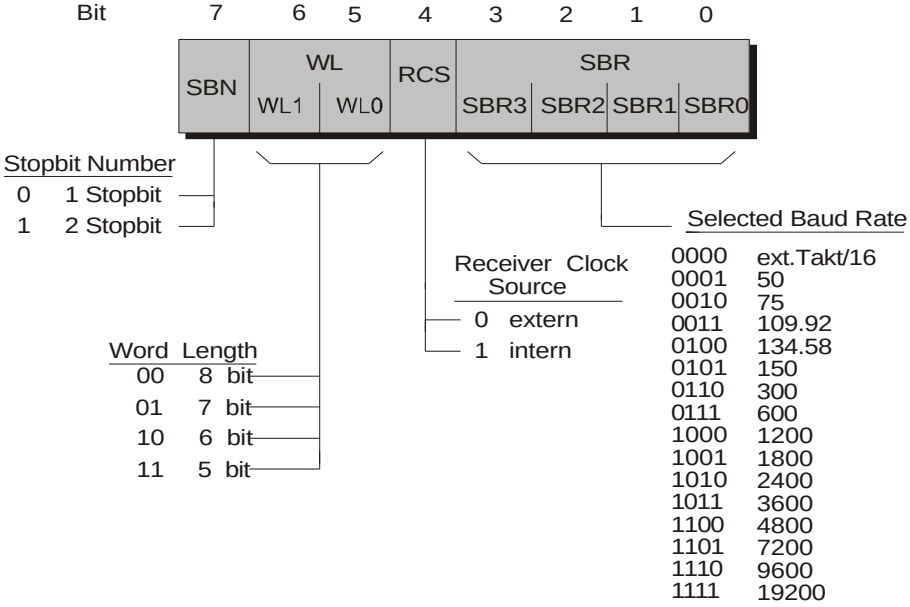
Das Statusregister SR



Das Befehlsregister IR



Das Steuerregister CR

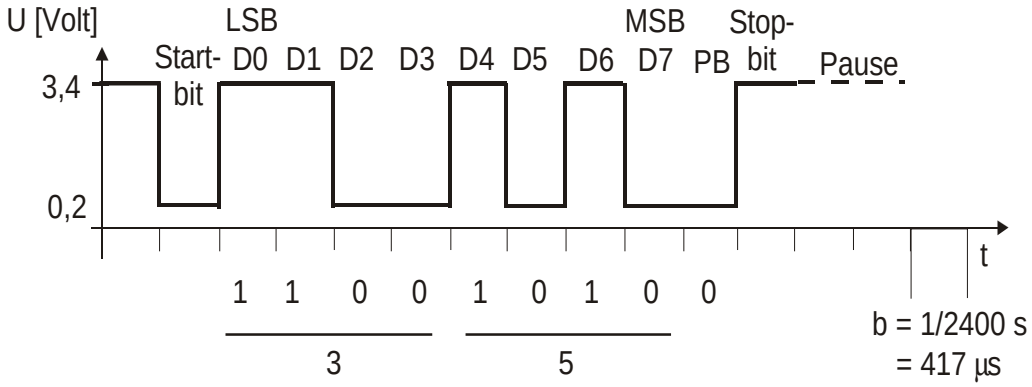


a) Mit welchem Wert in binärer und hexadezimaler Form müssen CR und IR initialisiert werden, damit die Schnittstelle in der folgenden Betriebsart eingesetzt werden kann:

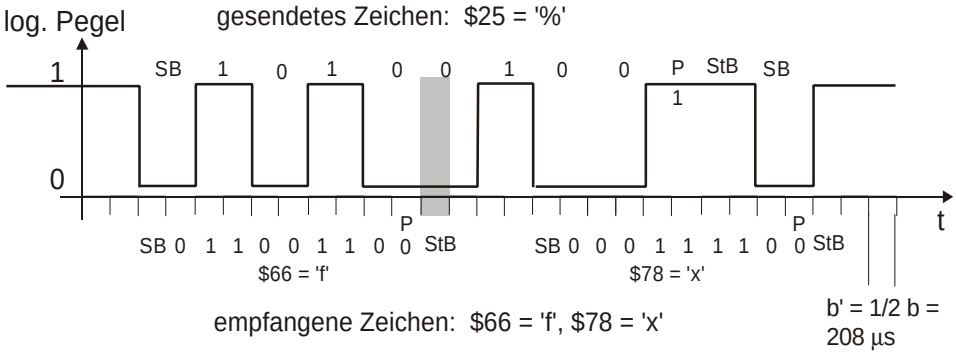
- 8 Datenbits, 1 Stopbit
- 2400 baud, interner Empfängertakt
- gerade Parität
- *Receiver Echo Mode disabled*
- Sender aktiviert, Sender-Interrupt *enabled*, Empfänger-Interrupt *enabled*
- DTR: Prozessor will Daten austauschen

CR = 0001 1010 (binär) = $1A
IR = 0110 0101 (binär) = $65

b) Tragen Sie in das folgende Diagramm für die unter a) bestimmte Übertragungsart den Signalverlauf am Ausgang des ACIA-Bausteins (TTL-Pegel **!**) für das ASCII-Zeichen ‘S’ ($53) ein ? Kennzeichnen Sie alle Bits durch ihren „Namen“ und logischen Wert. Geben Sie auch die Bitdauer b sowie das „Pausensignal“ bis zum nächsten übertragenen Zeichen an.



c) Nun seien ein Rechner und ein (reines) Ausgabegerät über eine V.24-Verbindung als Nullmodem miteinander gekoppelt. In der Betriebsart, für die die Schnittstelle des Rechners unter a) initialisiert wurde, wird am ACIA-Eingang RxD (!) des Gerätes das folgende Zeitdiagramm beobachtet:



Durch einen „Programmierfehler“ sei die serielle Schnittstelle des Peripheriegerätes auf eine Baudrate von 4800 eingestellt worden, d.h. sie liest jedes Zeichen mit doppelter Bitrate. Alle anderen Übertragungsparameter (Parität ja/nein, Bitanzahl, Stopbits) stimmen mit denen aus Teilaufgabe a) überein. Setzen Sie im folgenden voraus, daß das Peripheriegerät keine Zeichen aussendet und beim Empfang kein Überlauf (*Overrun*) auftritt.

Welches Zeichen wurde vom Sender ausgeschickt ? Welche Zeichen werden bei obenstehender Signalfolge im Peripheriegerät empfangen? (Ermitteln Sie die Zeichen mit der ASCII-Tabelle im Kurs.) Welchen Inhalt hat das Statusregisters SR des Empfängers jeweils vor dem Lesen des Empfangsdatenregisters? Welche Fehler werden festgestellt (Begründung) ?

Ausgeg. Zeichen:	Z = 0010 0101 (binär) = $25 = ‚%‘
1. Zeichen:	Z1 = 0110 0110 (binär) = $66 = ‚f‘
Statusregister:	SR = 1001 1010 (binär) = $9A
Fehler:	Es ist ein Rahmen-Fehler (*Framing Error*) aufgetreten, da das Stopbit auf ‚0‘ liegt. Kein Paritätsfehler, kein Overrun nach Voraussetzung.
2. Zeichen:	Z2 = 0111 1000 (binär) = $78 = ‚x‘
Statusregister:	SR = 1001 1000 (binär) = $98

Fehler:	kein Fehler, da Rahmen und Parität nicht verletzt, kein *Overrun* nach Voraussetzung.