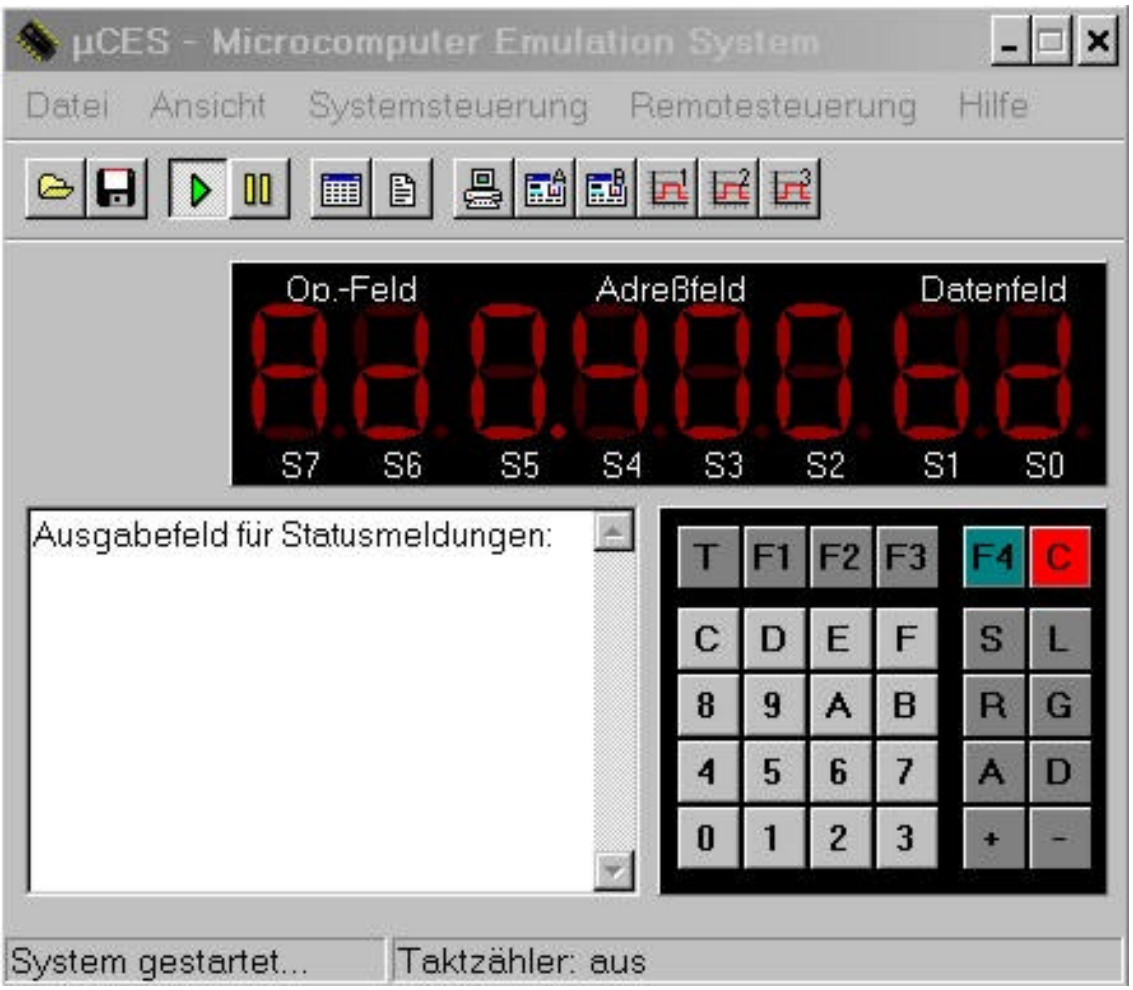
Skript zum Mikrorechner-Praktikum

Kapitel 4:

Aufbau und Funktion der Schnittstellenbausteine - Teil I

Autor: Helmut Bähring



Inhalt des Kapitels 4:

4. Aufbau und Funktion der Schnittstellenbausteine - Teil I

In den beiden folgenden Kapiteln werden die im Bild 3.1-1, Kapitel 3, dargestellten Schnittstellenbausteine (Peripheriebausteine) und Schnittstellen ausführlich beschrieben. Dies sind

- der Parallel-Schnittstellenbaustein MC6821 (Abschnitt 4.2),
- der asynchron serielle Schnittstellenbaustein R6551 und die V.24-Schnittstelle (Abschnitt 4.3),
- der Zeitgeber-/Zähler-Baustein (Timer) MC6840 (Kapitel 5).

4.1 Allgemeines zu den Schnittstellenbausteinen

Die Schnittstellenbausteine dienen zur Übertragung eines einzelnen Datums in serieller oder paralleler Form oder einzelner Signale zwischen der CPU und einem Peripheriegerät, wie z. B. einem Terminal oder einem Drucker. Sie übernehmen dazu die Steuerung dieser Peripheriegeräte und die Synchronisation des Datentransports.[1]

Sie werden häufig als "programmierbare" Interface-Bausteine bezeichnet, da sie vom Prozessor unter Programmkontrolle in verschiedene Arbeitsweisen gesetzt werden können. Dennoch darf man diese Art der Programmierung nicht mit der Programmierung des Mikroprozessors verwechseln, da hier lediglich durch die Eingabe eines Steuerwortes eines von mehreren fest vorgegebenen "Programmen" im Baustein aufgerufen wird. Im Bild 4.1-1 ist grob der allgemeine Aufbau eines Schnittstellenbausteins skizziert.

Auf der linken Seite sieht man die Anschlüsse zum Mikroprozessor. Dazu gehört vor allem der bidirektionale **Datenbus**, der bei den meisten Bausteinen 8 bit breit ist. Durch die Schreib/Lese-Leitung $R / \overline{W}$ kann die Richtung des Datentransports vom µP in den Baustein (schreiben, *write*) bzw. aus dem Baustein zum µP (lesen, *read*) bestimmt werden. Der Eingang E stellt dem Baustein den gemeinsamen **Systemtakt** zur Verfügung (vgl. die Signale des 6809 im Abschnitt 1.3, Kapitel 1). Durch ein L-Signal am *Reset*-Eingang kann der Baustein in einen definierten Grundzustand zurückgesetzt werden, der dadurch gekennzeichnet ist, daß einige oder alle Register (bis auf spezielle Bits) auf einen definierten Anfangswert (*default*), oft $0..0, gebracht werden.
Im allgemeinen bestehen zwei verschiedene Möglichkeiten, einen Schnittstellenbaustein durch die CPU anzusprechen[2]:

- Bei der ersten verfügt der Prozessor über besondere Ein-/Ausgabe-Befehle, die spezielle Signale zur Anwahl der Schnittstellenbausteine erzeugen (vgl. M/IO beim Intel 80x86). Aus den unteren 8 bzw. 16 Bits des Adreßbusses werden Aktivierungssignale zur Auswahl genau eines von maximal 256 bzw. 65536 Registern der Peripherie-Bausteine gewonnen (*Chip Enable* - CE, *Chip Select* - CS).

---

[1] Schnittstellenbausteine werden ausführlich in Kapitel II.3 beschrieben.

[2] vgl. Abschnitt I.2.6.

- Im zweiten Fall wird der Schnittstellenbaustein durch den Adreßdecoder (vgl. Abschnitt 3.1 in Kapitel 3) in einem fest vorgegebenen Bereich des Adreßraumes der CPU angesprochen (*Memory Mapped I/O*). Dazu bietet der Baustein einen oder mehrere Eingänge CS_i an. Diese Art besitzt gegenüber der ersten den Vorteil des leichteren Ansprechens des Bausteins und seiner Komponenten, die wie gewöhnliche Speicherzellen angesprochen werden. Deshalb können in der Regel alle Adressierungsarten für die Ein-/Ausgabe benutzt werden. Jedoch hat sie den Nachteil, daß Speicherraum verloren geht und eine aufwendigere Adreßdecodierung nötig ist.

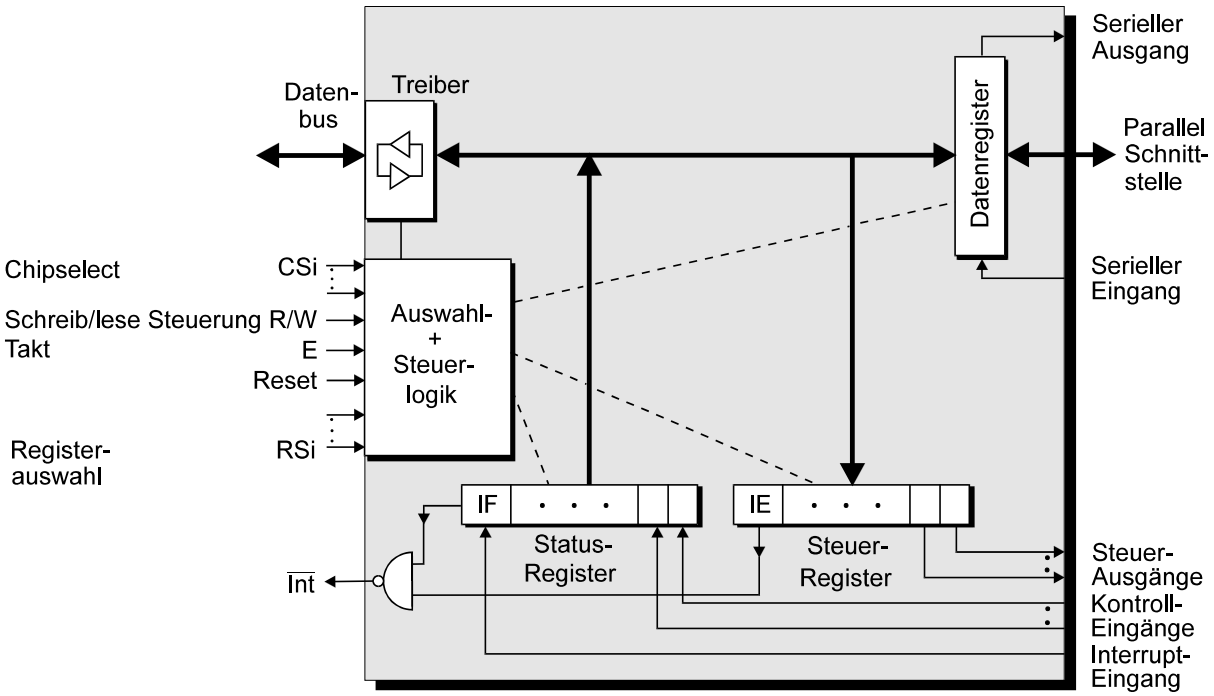


Bild 4.1-1: Prinzipieller Aufbau eines Schnittstellenbausteins

Bei beiden Arten dienen spezielle Eingänge RS_i zur Anwahl der einzelnen, unterscheidbaren Komponenten des Bausteins. Sie werden am einfachsten durch Adreßsignale niedriger Wertigkeit (z.B. A0, A1,..) ohne zusätzliche Decodierung beschaltet, wodurch der Aufwand verkleinert wird, aber auch einige Adressen „verloren“ gehen.

Der **Interruptausgang** $\overline{INT}$ dient zur Übermittlung einer im Baustein erzeugten oder von der Peripherie an einem seiner Eingänge gemeldeten Unterbrechungsanforderung an die CPU.

Alle Schnittstellenbausteine verfügen in der Regel über wenigstens 3 interne Register:

- zunächst dem **Steuerregister** (*Controlregister* - CR) zur Aufnahme des oben genannten Steuerwortes,
- einem **Statusregister** (SR) zur Speicherung und Abfrage des augenblicklichen Zustands des Bausteins

- und dem **Datenregister** (*Data Register* - DR) zur Zwischenspeicherung des aus- bzw. einzugebenden Datums.

Der wesentliche Unterschied zwischen den Bausteinen besteht in der "Breite" der Information, die an das Peripheriegerät abgegeben oder von dort empfangen wird: Bei den seriellen Bausteinen ist sie in der Regel 1 bit breit, bei den parallelen 1 byte. Im letzgenannten Fall spricht man von einem (Parallel-)*Port*. Dazu kommen bei beiden Bausteinarten noch einige Steuer- und Kontroll-Leitungen.

Die Information des Prozessors über das Vorliegen eines Datums von der Peripherie bzw. den Wunsch des Peripheriegerätes, ein Datum zu empfangen, geschieht in der Regel über ein Unterbrechungssignal an einem speziellen Eingang (Interrupteingang). Dieses Signal kann zu jedem Zeitpunkt auftreten, also asynchron zum Systemtakt (d.h. dem Signal E beim 6809). Durch ein besonderes Bit im Steuerregister (*Interrupt Enable* - IE) kann die Weitergabe dieser Unterbrechung an die CPU erlaubt oder unterbunden werden. Diese Weitergabe geschieht, wie oben beschrieben, über den Interruptausgang ($\overline{\text{INT}}$) des Bausteins, der mit einem der Interrupteingänge der CPU verbunden wird (vgl. $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$ im Abschnitt 1.3.2). Die Information, daß ein Unterbrechungssignal von der Peripherie eingetroffen ist oder im Baustein erzeugt wurde, wird in einem Bit des Statusregisters (*Interrupt Flag* - IF) gespeichert. Dieses Flag wird in der Regel durch das Lesen des vorliegenden Datums bzw. durch das Einschreiben des geforderten Datums in das Datenregister durch die CPU zurückgesetzt. Für zeitunkritische Anwendungen kann die asynchrone Unterbrechung des Prozessors durch einen Programmteil ersetzt werden, in dem der Prozessor zyklisch dieses Bit abfragt. Da der Prozessor in diesem Fall ausschließlich damit beschäftigt ist, auf das Eintreffen einer Unterbrechungsanforderung zu warten, spricht man vom **aktiven Warten** (*Busy-Waiting*).

Meistens sind im Statusregister noch weitere Bits vorhanden, die den aktuellen Zustand verschiedener Eingangsleitungen widerspiegeln und zur Abfrage durch den Prozessor dienen (**Kontroll-Eingänge**). Entsprechend existieren im Steuerregister häufig Bits, deren Zustand direkt auf spezielle Ausgangsleitungen des Bausteins gegeben wird, und durch deren Veränderung der Prozessor das Peripheriegerät steuern kann (**Steuer-Ausgänge**).

In der einfachsten Form ermöglichen je eine Eingangs- bzw. Ausgangsleitung mit ihren zugehörigen Bits im Status- bzw. Befehlsregister des Bausteins der CPU und dem Peripheriegerät, den Datenaustausch im **Quittungsbetrieb** (*Handshake*) durchzuführen[3]. Dazu werden beide Leitungen zwischen Prozessor und Peripheriegerät kreuzweise miteinander verbunden. Der Sender eines Datums teilt über seine Ausgangsleitung dem Empfänger das Vorliegen eines gültigen Datums mit, der Empfänger quittiert die Übernahme dieses Datums über seine Ausgangsleitung. Die Rücknahme der Handshake-Signale kann auf drei verschiedene Weisen geschehen:

[3] s. Abschnitt I.2.6 und II.3.1.

- programmgesteuert durch die CPUs von Sender und Empfänger der Daten, indem das zugehörige Bit im Steuerregister zurückgesetzt wird,
- automatisch gesteuert durch die Hardware des Peripheriebausteins, und zwar entweder
  - nach Ablauf einer festen Zeit, z.B. eines Maschinenzyklus (*Strobe*-Signal); diese Variante bietet sich immer dann an, wenn der Empfänger über einen hinreichend großen Pufferspeicher verfügt, in dem die Daten, durch das Strobe-Signal getaktet, eingeschrieben werden,
  - oder durch das Handshake-Signal des Partners veranlaßt; diese Variante muß dann eingesetzt werden, wenn der Empfänger sehr langsam arbeitet und keinen Pufferspeicher besitzt.

Beide letztgenannten Steuerungsvarianten werden vom Parallel-Schnittstellenbaustein MC6821 unterstützt und im folgenden Abschnitt 4.2 beschrieben.

4.2 Die Parallel-Schnittstelle

4.2.1 Aufbau des Portbausteins MC6821

Der Baustein MC6821 stellt eine Schnittstelle zum parallelen Anschluß von Peripheriegeräten zur Verfügung. Er wird von der Herstellerfirma Motorola mit dem Namen

Peripheral Interface Adapter (PIA)

bezeichnet. Andere Hersteller nennen ihre vergleichbaren Produkte[1] beispielsweise

Parallel I/O Circuit (PIO).

Die genaue Struktur des Bausteins kann man dem Bild 4.2-1 entnehmen.

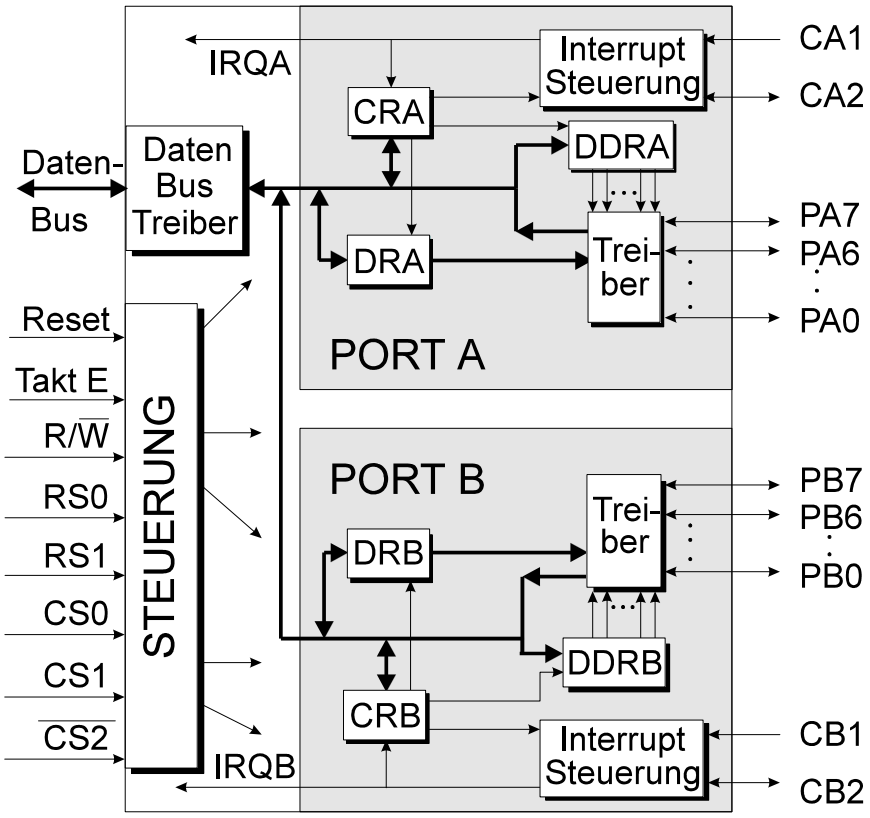


Bild 4.2-1: Blockschaltbild des Bausteins MC6821

Dieser Baustein besitzt zwei 8 bit breite Ein-/Ausgabe-*Ports* (Port PA, Port PB). Wie die Leitungen des Datenbusses können sie Signale in beiden Richtungen, also bidirektional, übertragen. Jedoch kann hier für jede Leitung getrennt festgelegt werden, ob sie als Eingangs- oder Ausgangsleitung dienen soll. Diese Festlegung geschieht für jeden Port in einem speziellen Register, dem Datenrichtungsregister (*Data Direction Register* - DDR), auf das weiter unten genauer eingegangen wird. Zu jedem Port

[1] s. Abschnitt II.3.5.

gehören außerdem je ein Interrupteingang (CA1, CB1) und eine weitere "programmierbare" Steuerleitung (CA2, CB2). Diese kann einerseits als zusätzlicher Interrupteingang oder aber als Strobe-Ausgang benutzt werden. Ihre Funktion wird in einem kombinierten **Status-** und **Steuerregister** (*Control Register* - CR) festgelegt. In ihm sind auch die Bits enthalten, durch die die Interrupteingänge aktiviert bzw. deaktiviert werden können (*Interrupt Enable*), bzw. die das Eintreffen einer Unterbrechungsanforderung anzeigen (*Interrupt Flag*).

Die Schnittstelle zum Prozessor wurde bereits im einleitenden Abschnitt 4.1 im wesentlichen erklärt. Die einzige Abweichung besteht darin, daß von jedem der beiden Parallel-Ports ein eigener Interruptausgang ($\overline{IRQA}$, $\overline{IRQB}$) herausgeführt wird. Diese Ausgänge können entweder getrennt auf zwei verschiedene Interrupteingänge des Prozessors oder aber miteinander verbunden auf den gleichen Eingang gegeben werden (*Open Collector*-Eigenschaft[*)]). In diesem Fall muß der Prozessor anhand der Interrupt-Flags in den Steuerregistern entscheiden, von welchem Port eine Unterbrechung angefordert wurde. Im Praktikumsrechner wird das Auswahlsignal der PIA ($\overline{CS2}$)vom Adreßdecoder erzeugt (s. Kapitel 3). Die Selbsttestaufgabe S3.1-2 zeigt, daß dafür das Signal $\overline{CS8}$ benutzt und dadurch der Adreßbereich $F000 – $F003 selektiert wird (CS0=CS1=+5V).

Das Bild 4.2-2 zeigt die Anschlußbelegung des Bausteins MC6821.

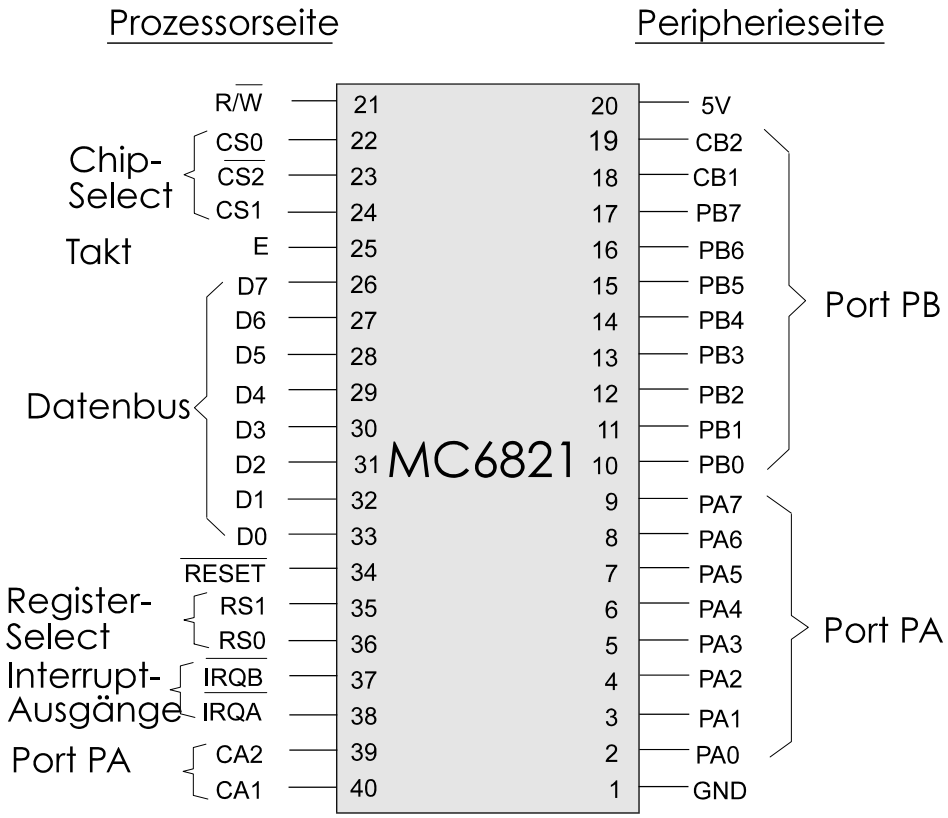


Bild 4.2-2: Anschlußbelegung des Bausteins MC6821

*) vgl. Abschnitt I.2.1.

Der Baustein ist in einem 40-poligen Gehäuse untergebracht. Auffällig ist, daß der Hersteller sich bemüht hat, entsprechend dem oben gezeigten Blockschaltbild 4.2-1, die Signal-Ein-/Ausgänge nach Prozessor- und Peripherieseite getrennt anzuordnen. Auf der "rechten Seite" (Pins 1-20) finden Sie die beiden Ports PA, PB mit ihren jeweils 8 Leitungen PA0,..,PA7 bzw. PB0,..,PB7 und den Steuerleitungen CB1, CB2. Aus "Platzgründen" mußten die Steuerleitungen des Ports PA (CA1, CA2) auf die linke "Prozessorseite" (Pins 21-40) verlegt werden. Dort finden Sie den bidirektionalen Datenbus DB, die Steuerleitungen ($R/\overline{W}$, $\overline{\mathrm{RESET}}$, E) sowie die drei Chip-Select-Eingänge $\mathrm{CS0}$, $\mathrm{CS1}$, $\overline{\mathrm{CS2}}$. (Mehrere Auswahleingänge mit unterschiedlichen aktiven Pegeln erleichtern den Aufbau des Adreßdecoders.) Zur Auswahl der einzelnen Komponenten des Bausteins stehen die zwei Eingänge RS0 und RS1 zur Verfügung.

Jeder **Port** (Port PA, Port PB) besteht aus

- 8 Ein-/Ausgangsleitungen mit der zugehörigen Leitungsanpassung (Treiber),
- einem Datenrichtungsregister DDR,
- einem Datenregister DR,
- einem Steuerregister CR,
- einer Interrupt-Steuerlogik.

4.2.2 Der Registersatz

Das **Datenrichtungsregister (***Data Direction Register)* DDR ist 8 bit breit. Der Wert des i.ten Bits bestimmt, ob die entsprechende Portleitung PX_i als Eingang oder Ausgang benutzt werden kann, und zwar gilt genauer für beide Ports PA, PB:

$$\mathrm{DDRX_i := Bit\ i\ von\ DDRX} = \begin{cases} 0\,, \text{ Leitung } \mathrm{PX_i} \text{ von Port PX ist Eingang} \\ \\ 1\,, \text{ Leitung } \mathrm{PX_i} \text{ von Port PX ist Ausgang.} \end{cases} \quad \text{(X=A,B)}$$

Das **Datenregister** (*Data Register)* DR bezeichnet die logische Zusammenfassung zweier Komponenten:

- Ein Datum, das zum Peripheriegerät übertragen werden soll, wird in einem **Ausgaberegister** zwischengespeichert. Daten, die vom Peripheriegerät gelesen werden, werden über die Portleitungen ohne Zwischenspeicherung direkt auf den Datenbus DB geschaltet. Hier wird die im Befehl angegebene Adresse des "Datenregisters" lediglich zur Aktivierung der Leitungstreiber ausgewertet. Dies wird nun genauer beschrieben.
- Die **Leitungsanpassung** besteht aus Tristate-Treibern (vgl. Kapitel 3) und einer Schaltlogik zur Richtungsumschaltung. Diese Logik ist für beide Ports unterschiedlich ausgebildet und wird im folgenden beispielhaft für den Port PB erklärt, da dieser als einziger über 2-mm-Buchsen auf die Rückwand des Gehäuses herausge-

führt wurde und dem Benutzer dort zur Kopplung des Mikrorechners und eines Experimentierboardes zur Verfügung steht.

Das Bild 4.2-3 zeigt den inneren Aufbau des Ports PB, dargestellt für eine einzelne Leitung PB_i. (Bzgl. der Schaltsymbole vgl. Selbsttestaufgabe S3.1-1 in Kapitel 3.)

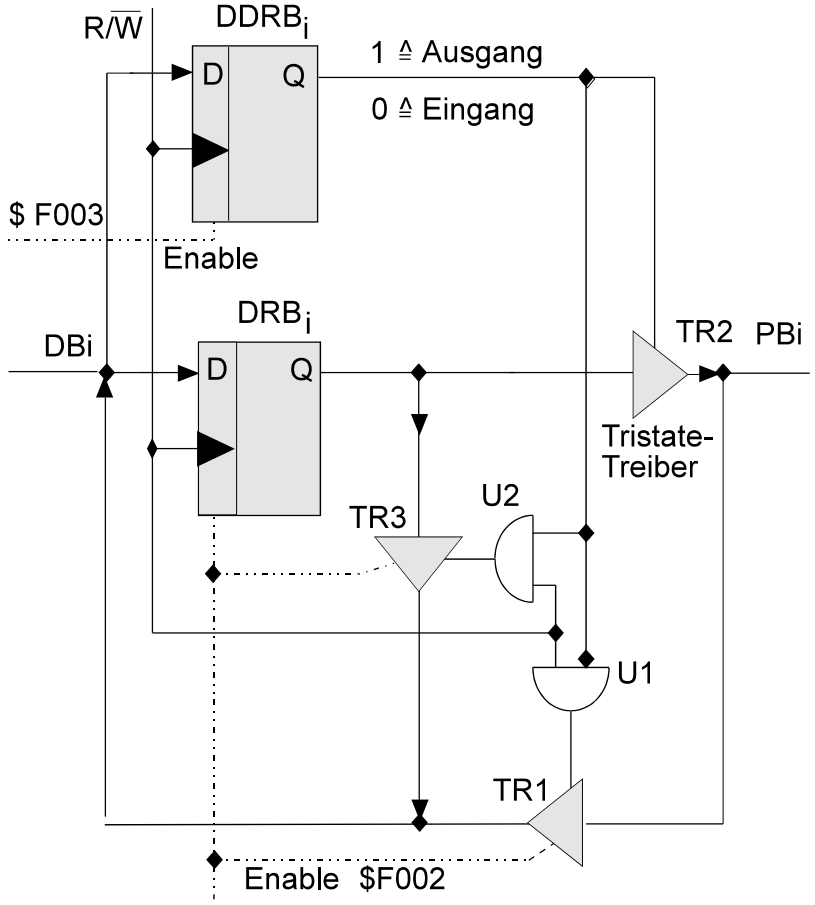


Bild 4.2-3: Realisierung der Ansteuerung einer einzelnen Leitung im Port PB

Funktionsweise

Durch einen Schreibbefehl mit der Adresse des Datenrichtungsregisters DDRB wird die Information des Datenbits DB_i in das Bit $DDRB_i$ übernommen.

- **$DDRB_i$=0**

  Die Leitung PB_i ist als Eingang geschaltet. Dann wird durch einen Lesebefehl mit der Adresse des Datenregisters DRB über das UND-Gatter U1 der Tristate-Treiber TR1 aktiviert. Dadurch wird die Portleitung PB_i auf die Datenleitung DB_i geschaltet und der Prozessor kann ihren Zustand lesen.
  Durch einen Schreibbefehl wird die Information auf der Datenbusleitung DB_i in das Bit DRB_i des Datenregisters übernommen, ohne daß diese Information gleichzeitig

auf die Portleitung PB_i durchgeschaltet wird. (Die Treiber TR2 und TR1 sind nicht aktiviert.) Die eingeschriebene Information erscheint erst nach der Richtungsumschaltung als Ausgang auf der Portleitung Pb_i.

- **$DDRB_i$=1**

  Die Leitung PB_i ist als Ausgang geschaltet. Der Treiber TR2 ist über den Ausgang Q des oberen Flipflops dauernd aktiviert, und die Information im Datenregisterbit DRB_i wird auf die Portleitung PB_i gelegt.
  Durch einen Lesebefehl wird über das UND-Gatter U2 der Treiber TR3 angesprochen und somit der Ausgang des Datenregisterbits DRB_i auf das Bit DB_i des Datenbusses gelegt. Der Prozessor liest daher das letzte ins Datenregisterbit DRB_i eingeschriebene Datum.
  (Im Unterschied dazu wird beim Lesen einer als Ausgang geschalteten Leitung des Ports PA der Zustand dieser Leitung eingelesen, wie er durch eine externe Beschaltung vorgegeben wird. Dieser kann vom Zustand des Datenregisterbits abweichen.)

Vor der Erklärung des Steuerregisters soll nun kurz auf die Adressierung der Register eingegangen werden. Wie im Bild 4.2-2 gezeigt, besitzt der Baustein dazu nur die beiden Auswahleingänge RS0, RS1. Diese sind im Praktikumsrechner durch die niederwertigen Adreßleitungen A0 bzw. A1 beschaltet. Natürlich ist Ihnen klar, daß für die Adressierung der 6 internen Register eigentlich 3 Adreßleitungen benötigt würden. Das 3. Adreßsignal wird im MC6821 durch das Bit CRX2 (X=A,B) des Steuerregisters zur Verfügung gestellt: Vom Prozessor aus werden für jeden Port unter der gleichen Adresse sowohl das Datenrichtungsregister DDRX wie das Datenregister DRX angesprochen. Die Vorauswahl unter ihnen nimmt das Bit CRX2 vor. Ist

$$CRX2 = \begin{cases} 0, & \text{so wird das Datenrichtungsregister DDRX,} \\ & \quad X = A,B \\ 1, & \text{so wird das Datenregister DRX vorselektiert} \end{cases}$$

Aus der Beschaltung des MC6821 im Praktikumsrechner ergibt sich mit dem bisher Gesagten die in der Tabelle 4.2-1 gezeigte Adreßzuteilung.

Tabelle 4.2-1: Adreßzuteilung der Register des MC6821

Adresse	CRX2=0	CRX2=1	Register
$F000	DDRA		Datenrichtungsregister Port PA
$F000		DRA	"Datenregister" Port PA
$F001	CRA	CRA	Steuerregister Port PA
$F002	DDRB		Datenrichtungsregister Port PB
$F002		DRB	"Datenregister" Port PB
$F003	CRB	CRB	Steuerregister Port PB

Das folgende Bild 4.2-4 zeigt die Aufteilung des Status- bzw. Steuerregisters CR in seine unterscheidbaren Bitfelder. Darin bezeichnet X=A,B wieder einen der beiden Ports PA, PB.

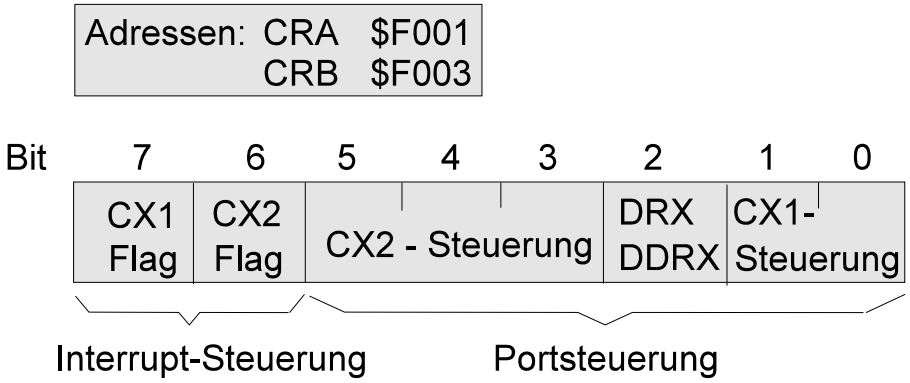


Bild 4.2-4: Das Steuerregister CRX, X= A, B, und seine Aufteilung

Es folgt nun die Beschreibung der Funktion der einzelnen Bits (X=A,B).

- **Bit 7 (Interrupt Flag für die Leitung CX1):**
  - CRX7=0: Am Interrupteingang CX1 wurde keine Unterbrechungsanforderung festgestellt.
  - CRX7=1: Am Interrupteingang CX1 wurde eine Unterbrechungsanforderung festgestellt. **Wichtig:** CRX7 wird durch das Lesen des Datenregisters DRX oder ein Reset-Signal zurückgesetzt !

- **Bit 6 (Interrupt Flag für die Leitung CX2):**
  - Falls CX2 als Ausgang geschaltet ist (s. CRX5), hat Bit 6 keine Funktion.
  - Falls CX2 als Eingang geschaltet ist, entspricht seine Funktion der von CRX7, jedoch für den Interrupteingang CX2.

■ **Bit 5:**

- CRX5=0: Die Leitung CX2 ist ein Interrupteingang.
- CRX5=1: Die Leitung CX2 ist ein Steuerausgang (*Peripheral Control Output*).

■ **Bit 4 und Bit 3:**

- CRX5=0: CX2 ist Eingang.
  - CRX4=0: Unterbrechungsanforderung wird durch negative Flanke an CX2 erzeugt.
  - CRX4=1: Unterbrechungsanforderung wird durch positive Flanke an CX2 erzeugt.
  - CRX3=0: (*Interrupt Disabled*) Weiterleitung der Unterbrechungsanforderung an die CPU nicht aktiviert. Sie wird bis zur nächsten Aktivierung zwischengespeichert (vgl. CRX0=0).
  - CRX3=1: (*Interrupt Enabled*) Sofortige Weiterleitung der Unterbrechungsanforderung am Eingang CX2 an den Prozessor.
- CRX5=1: CX2 ist Steuerausgang.
  - CRX4=0: Ausgabe eines **Handshake-Signals** (s. Abschnitt 4.1) am Ausgang CX2:
    - Für Port PA nach dem Lesen des Datenregisters DRA,
    - für Port PB nach dem Schreiben des Datenregisters DRB.
    - CRX3=0: Permantes L-Signal bis zum nächsten Interrupt an CX1 (s. Bild 4.2-5).
    - CRX3=1: kurzer Strobe-Impuls für die Dauer eines Maschinenzyklus (1 µs, Takt E).
  - CRX4=1: CX2 ist statische Steuerleitung zur Peripherie:
    - CRX3=0: CX2=0, d.h. L-Potential an CX2,
    - CRX3=1: CX2=1, d.h. H-Potential an CX2.

■ **Bit 2:**

- CRX2=0: Das Datenrichtungsregister DDRX wird vorselektiert.
- CRX2=1: Das Datenregister DRX wird vorselektiert.

■ **Bit 1:**

- CRX1=0: Eine Unterbrechungsanforderung wird durch eine negative Flanke am Eingang CX1 erzeugt.
- CRX1=1 Eine Unterbrechungsanforderung wird durch eine positive Flanke am Eingang CX1 erzeugt.

■ **Bit 0:**

- CRX0=0: (*Interrupt Disable*d) Die Weitergabe einer am Interrupteingang CX1 eintreffenden Unterbrechungsanforderung an die CPU wird unterdrückt. Diese Anforderung wird aber gespeichert (s. Bit 7) und dann an den Prozessor weitergeleitet, wenn Bit 0 auf 1 gesetzt wird.
- CRX0=1: (*Interrupt Enable*d**)** Eine Unterbrechungsanforderung am Eingang CX1 wird augenblicklich über den Ausgang $\overline{\text{IRQX}}$ (s. Bild 4.2-1) an den Prozes-sor weitergeleitet.

Aus Abschnitt 1.3.2, Kapitel 1, wissen Sie schon, daß beim Eintreffen einer Unterbrechungsanforderung am $\overline{\text{IRQ}}$-Eingang der Prozessor eine Interruptroutine ausführt, deren Startadresse in den Speicherstellen $FFF8, $FFF9 steht. In dieser Routine des Monitor-Programms wird dann zunächst entschieden, aus welcher "Quelle" die Anforderung stammt. In der Zero-Page finden Sie im Bereich $0030-$0043 die Tabelle der Startadressen, unter denen die verschiedenen Interruptroutinen beginnen, und zwar in der Form: H-Byte, L-Byte. Diese Tabelle wird bei jedem Rücksetzen des Monitors (Taste C) initialisiert. Für die meisten Unterbrechungsmöglichkeiten zeigt die Vorbelegung auf den Befehl RTI (*Return from Interrupt*). Durch Ändern der Startadresse können Sie die Ausführung einer eigenen Interrupt-Routine erzwingen. Für die Interrupt-Quellen des MC6821 sind die in Tabelle 4.2-2 aufgeführten Adressen zuständig:

Tabelle 4.2-2: Vorbelegung der IRQ-Interruptvektoren

Adressen	Quelle	Vorbelegung
$0036,$0037	CA1	Break-Taste F4
$0038,$0039	CB1	RTI
$003A,$003B	CA2	RTI
$003C,$003D	CB2	RTI

Im Bild 4.2-5 ist die Ausgabe eines Handshake-Signals über die Ausgangsleitung CA2 bzw. CB2 dargestellt.

Dazu müssen, wie oben gezeigt, im Steuerregister das Bit5=1 und das Bit4=0 sein. Durch das Symbol ↯ ist eine Unterbrechungsanforderung des Partners am Eingang CX1 gekennzeichnet, die über den Ausgang $\overline{\text{IRQX}}$ an den Prozessor weitergeleitet wird (X=A,B). Durch sie wird der Prozessor zur Übernahme (Port PA, obere Hälfte des Bildes) bzw. zur Abgabe eines Datums (Port PB, untere Hälfte des Bildes) aufgefordert. Beim Port PA wird das Handshake-Signal durch das Lesen des Datenregisters DRA, beim Port PB durch das Schreiben des Datenregisters DRB ausgelöst. Es beginnt jeweils mit der negativen Flanke des Systemtaktes E. Man sieht, daß in Abhängigkeit vom Bit CRX3 des Steuerregisters das Strobesignal für CRX3=0 bis zum nächsten Interrupt des Partners, für CRX3=1 jedoch - im Minimalfall - nur einen Zyklus

des Systemtaktes E dauert. Diese Zykluszeit muß je nach Ausführung des MC6821 wenigstens 0.5 µs bzw. 1 µs dauern. Die Deaktivierung des Strobesignals beginnt in diesem Fall mit der 1. negativen Flanke des Taktes E nach der Deaktivierung (*deselection*) der Prozessorschnittstelle des Portbausteins (durch die Signale CSi).

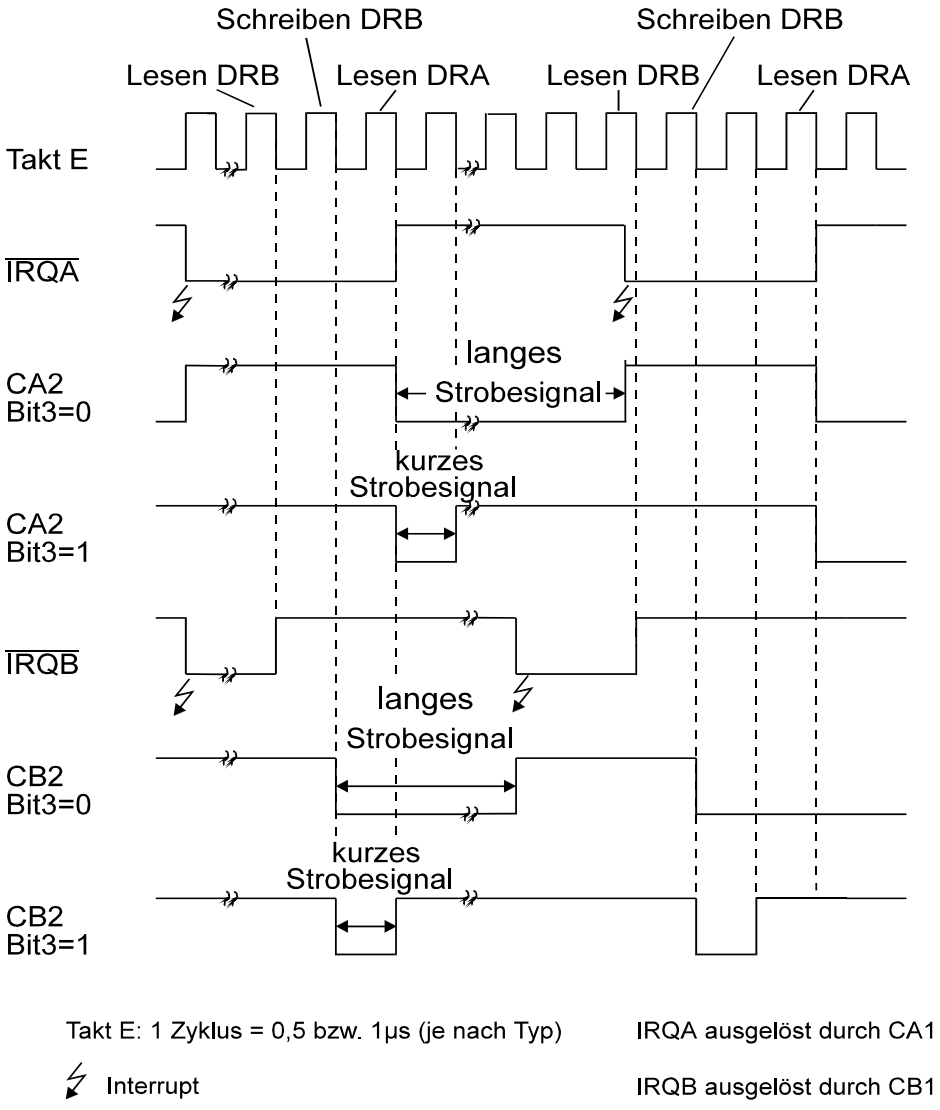


Bild 4.2-5: Ausgabe eines Handshake-Signals durch den MC6821

4.2.3 Erklärung der Schnittstellen-Erweiterungskarte

Ein wesentliches Ziel des Praktikums ist es, Ihnen das Arbeiten mit den Schnittstellen des Praktikumsrechners und ihre Programmierung nahezubringen. Natürlich können wir Sie während der Heimphase nicht mit geeigneten Peripheriegeräten zum Anschluß an diese Schnittstellen versorgen. Daher haben wir eine spezielle Schnitt-stellen-Erweiterungskarte entwickelt und jedem Praktikumsrechner beigelegt, die es erlaubt, alle wesentlichen Funktionen der parallelen Schnittstelle und des Zeitgeber/ Zählerbausteins auszutesten Im Bild 4.2-6 ist der Aufbau dieser Karte grob dargestellt. Einen detaillierten Schaltplan dazu finden Sie im Anhang.

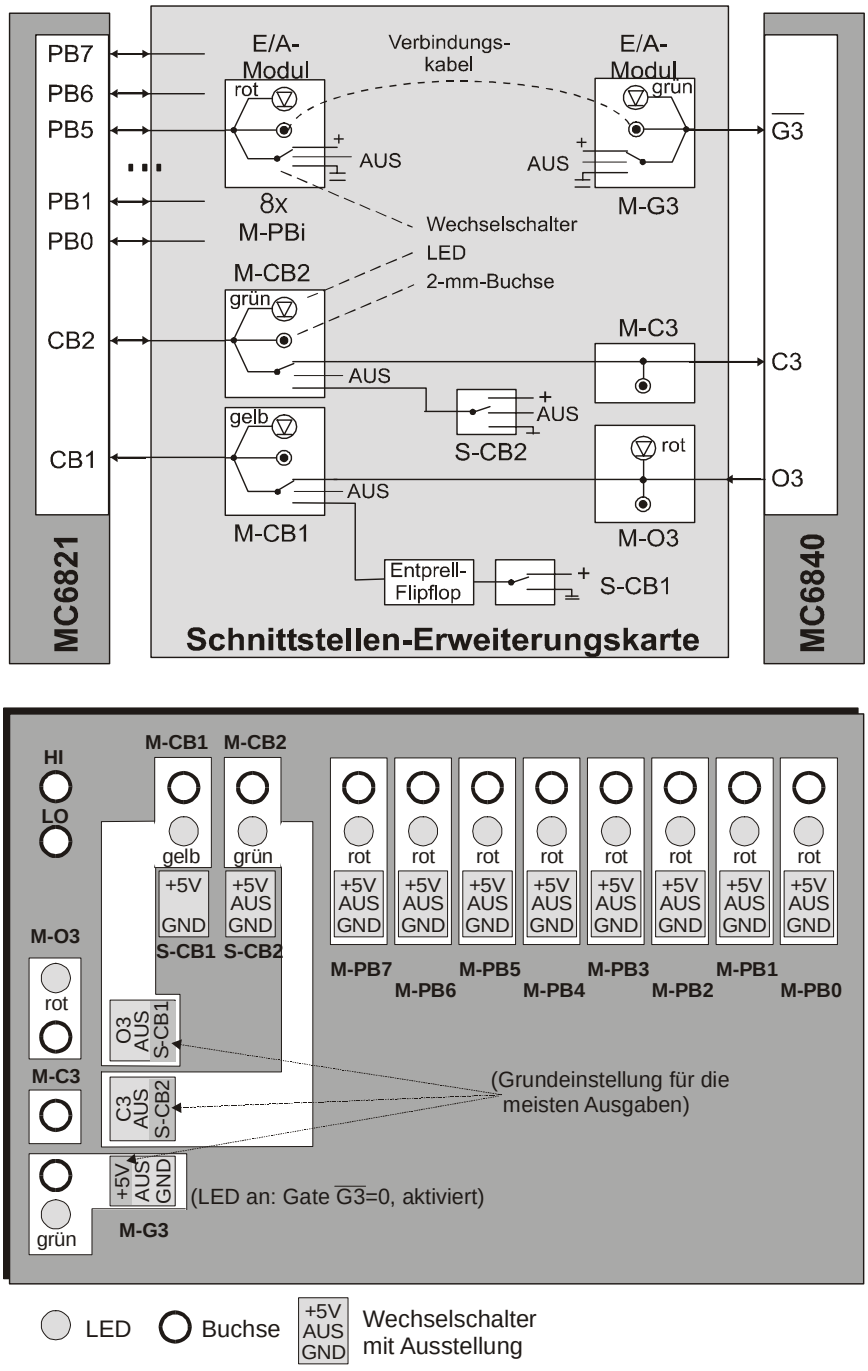


Bild 4.2-6: Blockschaltbild der Schnittstellen-Erweiterungskarte

Im Bild 4.2-6 werden bereits die drei E/A-Leitungen des Zeitgeber/Zähler-Bausteins dargestellt, der erst in Kapitel 5 ausführlich beschrieben wird. Hier soll nur kurz erwähnt werden, daß C3 ein externer Takteingang, O3 der Zeitgeberausgang und $\overline{G3}$ ein Steuereingang ist, durch den die Funktion des Bausteins von außen aktiviert bzw. deaktiviert werden kann, der aber für spezielle Funktionen auch als Signaleingang dient.

Die mit E/A-Modul bezeichneten Schaltungen bestehen aus:

- einer Leuchtdiode LED, die über einen Treiber mit Strom versorgt wird und daher die Signalleitungen nicht zu stark belastet,
- einer 2-mm-Buchse zur Verbindung mit anderen E/A-Modulen mit Hilfe der beiliegenden Kabel, [2]
- einem Wechselschalter zur Auswahl zweier Eingabesignal-Quellen mit Aus-Mittelstellung.

Die Erweiterungskarte unterstützt die folgenden Funktionen:

- Über eine als Ausgang geschaltete Portleitung PB_i kann die (rote) LED ein- oder ausgeschaltet und der Pegel an der 2-mm-Buchse entsprechend auf H- oder L-Potential gelegt werden. (Der Wechselschalter sollte dabei auf „Aus“ stehen.)
- An einer als Eingang geschalteten Portleitung PB_i kann durch den Wechselschalter ein H- oder L-Pegel erzeugt werden, der gleichzeitig durch die (rote) LED angezeigt wird und an der 2-mm-Buchse abgegriffen werden kann. (In Aus-Stellung des Schalters ist der Eingang hochohmig und unbestimmt.)
- Durch den Schalter mit nachgeschaltenen Entprell-Flipflop kann am Steuereingang CB1 ein H- oder L-Pegel eingestellt und durch die gewählte Flanke (s. Abschnitt 4.2.2) eine Interruptanforderung gestellt werden. Über den Schalter im E/A-Modul an CB1 kann aber auch der Ausgang O3 des Zeitgebers MC6840 auf CB1 gelegt werden, so daß einzelne oder periodische Interruptanforderungen durch den Timer im MC6840 erzeugt werden können. Eine dritte Interruptquelle kann an der 2-mm-Buchse angeschlossen werden. Der aktuelle Zustand an CB1 wird durch die (gelbe) LED angezeigt.
- Der Zustand des Ausgangs O3 wird durch eine (rote) LED angezeigt und kann an der 2-mm-Buchse abgegriffen werden, z.B. zur Verbindung mit einem der Porteingänge PB_i.
- Für die bidirektionale, multifunktionale Steuerleitung CB2 gilt prinzipiell das zur Port leitung PB_i Gesagte. Nur der Wechselschalter ist anders beschaltet. Durch ihn kann einerseits mit Hilfe des vorgeschalteten 2. Wechselschalters ein bestimmter Pegel oder eine Interruptanforderung an CB2 erzeugt werden, andererseits kann die als Ausgang geschaltete CB2-Leitung mit dem externen Takteingang C3 des Zeitgeber/Zähler-Bausteins MC6840 verbunden werden. So können insbesondere auch nicht periodische Ereignisse am Ausgang CB2 gezählt werden. (In diesem Fall muß der Wechselschalter an CB2 auf „Aus“, also in Mittelstellung stehen.)

[2] Die Bausteineingänge sind durch Vorwiderstände an den 2-mm-Buchsen und den Wechselschalter geschützt.

- Der Takteingang C3 kann jedoch auch von jedem Portausgang PB_i über die 2-mm Buchse angesteuert werden.
- Das E/A-Modul an G3 von MC6840 erlaubt die Aktivierung bzw. Deaktivierung dieses kombinierten Steuer-/Signaleingangs durch den Wechselschalter oder ein Signal an der 2-mm-Buchse. Der aktuelle Zustand wird durch die (grüne) LED angezeigt.

Praktische Übung P4.2-1:

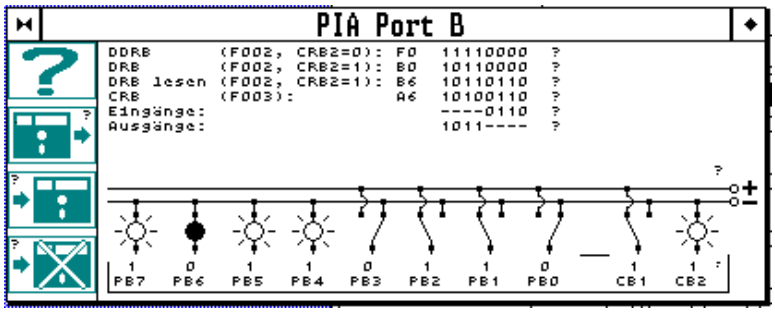
Verbinden Sie Ihren Praktikumsrechner über die 64-polige-Steckerleiste auf der Rückwand des Gehäuses mit der Schnittstellen-Erweiterungskarte. Schalten Sie danach den Praktikumsrechner ein.

I. Wählen Sie über die Tastatur (Funktionstaste A) die Register des MC6821 an. "Programmieren" Sie den Baustein so, daß im Port PA jede zweite Leitung als Ausgang, die anderen als Eingang geschaltet werden. Lesen Sie danach mehrfach das Datenregister DRA und schreiben Sie beliebige Werte ein.

II. Im Port PB sollen die vier niederwertigen Leitungen PB0-PB3 als Ausgänge, die restlichen als Eingänge initialisiert werden. Lesen Sie das Datenregister DRB für verschiedene Schalterstellungen und schreiben Sie mehrere Werte nacheinander ein. Überprüfen Sie die Ergebnisse an den LEDs. (Beachten Sie, daß Sie für jedes Lesen das Register DRB erneut anwählen müssen !)

III. Verbinden Sie mit Hilfe der beiliegenden Kabel die vier „unteren“ Portleitungen PB0-PB3 mit den „oberen“ Leitungen: $P_i \leftrightarrow P_{i+4}$ (i=0,..,3). Schreiben Sie jetzt wie unter II. mehrere Werte in DRB ein und lesen Sie jeweils danach DRB.

Zur Arbeit mit dem 6809-Emulator:
Teil I und II von Aufgabe P4.2-1 können Sie auch mit dem 6809-Simulator ausführen. Das folgende Bild zeigt das geöffnete Fenster zum Port PB (Menüpunkt: Bausteine, Eintrag: Port PB). Wie unter II. gefordert, enthält das Datenrichtungsregister DRRB den Wert $F0=1111 0000, d.h. die oberen 4 Leitungen (PB7-PB4) sind als Ausgänge programmiert, was durch die Lämpchen-Symbole angezeigt ist, und die unteren 4 Leitungen (PB3-PB0) als Eingänge, was durch die Schalter-Symbole angezeigt wird. Über die Ausgangsleitungen wird beispielhaft der Wert $B=1011 ausgegeben, über die Eingänge der Wert $6=0110 eingelesen. Im Datenregister DRB findet sich daher beim Lesen der Wert $B6. Sie können nun jeden beliebigen Hexadezimalwert $0...$F eingeben, indem Sie mit der Maus die Schalter an PB3-PB0 anklicken. Auf die gleiche Weise können Sie an CB1 eine Interruptanforderung stellen und die Auswirkung im Statusregister CRB beobachten. (Selbsttest: Wie wird das Interrupt-Flag Bit 7 in CRB gelöscht ?) Im Bild ist - im Vorgriff auf die folgende Praktische Übung P4.2-2 - die Leitung CB2 als Ausgang programmiert, über den ein (negativer) Strobe-Impuls ausgegeben wird.



Praktische Übung P4.2-2:

Schließen Sie die Schnittstellen-Erweiterungskarte an den Praktikumsrechner an.

I. "Programmieren" Sie nun den Baustein MC6821 über die Tastatur so, daß
   1. kein Interrupt zum Prozessor weitergeleitet wird,
   2. eine positive Flanke am Eingang CB1 als Unterbrechungsanforderung gewertet wird,
   3. CB2 als Handshake-Ausgang geschaltet wird,
   4. ein "langer" Handshake-Impuls an CB2 bis zum nächsten Interrupt an CB1 ausgegeben werden kann.

II. Erzeugen Sie nun ein Strobe-Signal an CB2. Lesen Sie das Steuerregister CRB vor und nach der Erzeugung des Strobe-Signals und interpretieren Sie dessen Inhalt. Geben Sie dann durch den Schalter einen kurzen L-Impuls auf den Interrupteingang CB1. Beobachten Sie dabei die LED und interpretieren Sie den neuen Wert im Steuerregister CRB. Löschen Sie vor einer erneuten Durchführung dieses Versuches das Interrupt Flag (Bit CRB7) im Steuerregister.

III. Programmieren Sie den 6809-Simulator, wie oben angegeben, mit der Änderung, daß ein kurzer Strobe-Impuls an CB2 ausgegeben wird.

Praktische Übung P4.2-3:

Schließen Sie die Schnittstellen-Erweiterungskarte an den Praktikumsrechner an. Schreiben Sie ein Programm, daß ein zyklisches Rechtecksignal am Ausgang CB2 erzeugt. Dabei sollen die Pulsdauer T_D und die Pulspause T_P als Vielfache von 1 ms vor dem Programmstart in den Registern Y und U eingegeben werden können. Erzeugen Sie nacheinander Signale unterschiedlicher Frequenz (zwischen 0,1 bis 100Hz) und unterschiedlicher Impuls/Pausen-Verhältnisse und beobachten Sie die Ausgangssignale an der LED der Leitung CB2.

Hinweise:

1. Zur Erzeugung der Zeitverzögerung von 1 ms benutzen Sie am besten die Routine DLY1MS aus Kapitel 1, Abschnitt 1.2.2.
2. Nach jedem Betätigen der Clear-Taste C werden alle Register, also auch Y und U, zurückgesetzt.
3. Diese Routine benötigen Sie noch für die Bearbeitung der Praktischen Übungen in Kapitel 5. Sie sollten sie daher im PC abspeichern.

Mit Hilfe der Lösung zur Praktischen Übung P4.2-3 können Sie nun die Ihnen vom 6809-Emulator gebotene Möglichkeit austesten, Ausgangssignale an den Parallel-Ports in einer Datei aufzuzeichnen bzw. von dort einzulesen. Bevor Sie Ihr Programm starten, können Sie durch Anklicken der Ikone

eine Datei bestimmen, in die für jeden Schreibzugriff auf das Datenregister des Ports (Adresse Port PA: $F000, Port PB: $F002) alle Ausgangszustände eingetragen werden[3]. (Solange diese Datei geöffnet ist, wird in der Statuszeile ein Diskettensymbol mit einem hinweisenden Pfeil auf der linken Seite und der Beschriftung 'PIA' angezeigt.) Durch die Ikone

können Sie die Aufzeichnung stoppen. Nachdem Sie auch Ihr Programm beendet haben, können Sie durch Anklicken der Ikone

die Datei (oder eine andere) erneut öffnen, aus der nun mit jedem Lesezugriff auf das Datenregister des Ports der Zustand der als Eingänge geschalteten Leitungen eingelesen wird. Dabei wird jeder Eingabewert so lange unverändert gehalten, wie er bei der Ausgabe in die Datei an den Ausgängen anlag. (Die Übertragung der Zeichenkette aus der Datei in den Emulator wird in der Statuszeile durch ein Diskettensymbol mit einem wegweisenden Pfeil auf der rechten Seite und der Beschriftung 'PIA' angezeigt.)

[3] Hinweis: Die erzeugte Datei (mit der Endung „.PIA“) können Sie sich mit jedem Editor anschauen. In ihr sind die ausgegebenen Ausgangssignale zeilenweise im folgenden Format abgelegt: 8 Datenbits, Cx1, CX2, Zeitdifferenz (X=A,B). Die Zeitdifferenz gibt den Abstand zum letzten aufgezeichneten Wert in Maschinenzyklen an. Ein '-' kennzeichnet eine Leitung, die als Eingang geschaltet ist.

4.3 Die V.24-Schnittstelle

4.3.1 Allgemeine Grundlagen

In Abschnitt 4.2 wurde eine parallele Schnittstelle zum Anschluß von Peripheriegeräten beschrieben. Eine weitere Möglichkeit zur Anbindung dieser Geräte an einen Mikrorechner stellt die sogenannte V.24-Schnittstelle dar[1]. Die V.24-Schnittstelle ermöglicht die asynchrone, serielle Übertragung der Daten. Durch die Normung dieser Schnittstelle ist es möglich, Peripheriegeräte unterschiedlicher Hersteller mit den verschiedensten Rechnern, aber auch diese unter einander zu verbinden. In den USA wurde diese Schnittstelle unter der Bezeichnung RS232C (US-Norm EIA[2]) und in Europa unter der Bezeichnung V.24/V.28 (CCITT-Norm[3], DIN 66020) festgelegt. Dabei sind in der V.24-Norm die funktionellen, in der V.28-Norm die elektrischen Eigenschaften beschrieben.

Bild 4.3-1 zeigt den Aufbau der V.24-Schnittstelle des Praktikumsrechners.

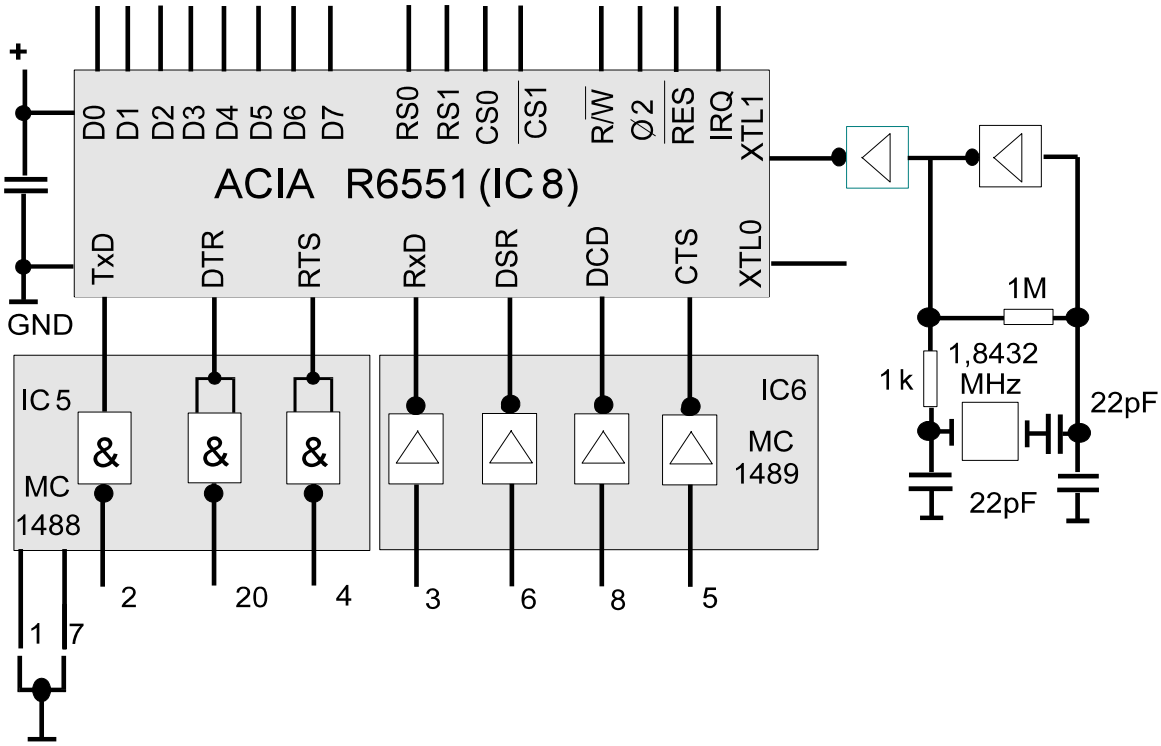


Bild 4.3-1: Die V.24-Schnittstelle des Praktikumsrechners

Da auf dem Datenbus des Rechners die Daten in paralleler Form anliegen, wird zur Realisierung einer seriellen Schnittstelle ein Baustein benötigt, der sowohl eine Parallel/Serien- wie auch eine Serien/Parallel-Umwandlung der Daten vornimmt. Ein solcher Interface-Baustein wird häufig

---

1 Sie wird in Abschnitt II.3.6 ausführlich beschrieben.
2 EIA - *Electronic Industries Associates*
3 CCITT - *Comité Consultatif International Télégrafique et Téléfonique,* heute ITU*: International Telecommunication Union*

Asynchronous Communications Interface Adapter - ACIA

genannt. Ein Beispiel dafür ist z.B. der im Praktikumsrechner eingesetzte R6551 der Firma Rockwell. Bevor die Funktion dieses ACIA-Bausteins näher erklärt wird, stellen wir kurz die Grundlagen der V.24-Schnittstelle dar.

Bild 4.3-2 zeigt das serielle Datenformat der V.24-Schnittstelle. Die Zeichen werden als Folge von Einzelbits übertragen, die von einem Startbit (logisch "0") angeführt und von einem, 1½ oder zwei Stopbits (logisch "1") abgeschlossen werden. Die Übertragung beginnt mit dem niederwertigen Bit 0 (LSB). (Da zwischen zwei "1"-Bits das Signal nicht zum "0"-Wert zurückkehrt, spricht man von einer "Non-Return-to-Zero"-Codierung - NRZ.)

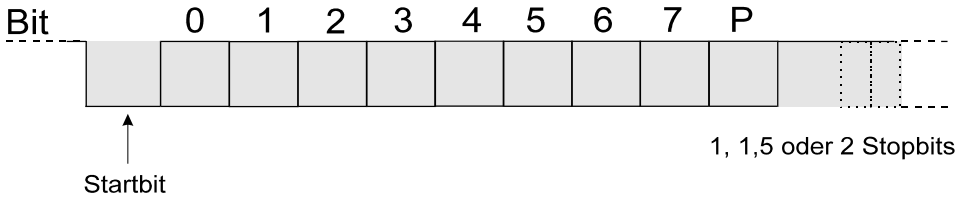


Bild 4.3-2: Datenformat der V.24-Schnittstelle
(Beispiel: 8-bit-Daten mit Paritätsbit)

Die Kommunikation zwischen den Geräten erfolgt über Daten-, Takt-, Melde- und Steuersignale. Die Pegel der V.24-Schnittstelle auf den Datenleitungen sind in "negativer“ Logik, die der Steuersignale in "positiver" Logik festgelegt:

logischer Zustand	Datenbit	Steuerinformation
0	H-Pegel: +3 V bis +15 V	L-Pegel: -15 V bis -3 V
1	L-Pegel: -15 V bis -3 V	H-Pegel: +3 V bis +15 V

Der Übergangsbereich zwischen -3V und +3V ist undefiniert. Die Übertragungsgeschwindigkeit kann bis zu 56.000 baud betragen, d.h. es können bis zu 56.000 bit/s übertragen werden. Die maximale Übertragungsentfernung ist abhängig von der Baudrate. Sie beträgt bei 19.200 baud ca. 30 Meter.

Die Schnittstellendefinition und die Pinbelegung des 25-poligen-Steckers sind der Vollständigkeit halber im Anhang B dargestellt. Bei der Funktionsbeschreibung werden wir uns hier auf die wichtigsten Signale zur Kommunikation zwischen Mikrorechnern untereinader und mit Terminals, Druckern und anderen Peripheriegeräten beschränken (s.Tabelle 4.3-1).

Bemerkung: Einige dieser Signale sind im L-Pegel aktiv und werden daher meist in negierter Form (mit Überstreichung o.ä.) geschrieben. Wir lassen diese Überstreichungen (aus Gründen der einfacheren Darstellung) dann weg, wenn der Zusammenhang eine exakte Darstellung nicht verlangt.[4] Bild 4.3-1 können Sie entnehmen, daß die Steuersignale des Schnittstellenbausteins R6551, den wir in diesem Abschnitt

[4] In Abschnitt II.3.6 kann man die exakten Signalbezeichnungen nachlesen.

beschreiben werden, invertiert werden (IC5 und IC6). Daher muß hier festgestellt werden, daß im folgenden immer die Signale am V.24-Schnittstellenstecker gemeint sind, wenn nicht explizit auf die R6551-Signale Bezug genommen wird.

Tabelle 4.3-1: Die wichtigsten Signale einer V.24-Schnittstelle

Pin	Signal	Eingang/Ausgang	Abkürzung
2	Transmit Data	Ausgang	TxD (TD)
3	Receive Data	Eingang	RxD (RD)
4	Request to Send	Ausgang	RTS
5	Clear to Send	Eingang	CTS
6	Data Set Ready	Eingang	DSR
7	Signal Ground		SG (GND)
8	Data Carrier Detect	Eingang	DCD
17	Receiver Clock	Ein-/Ausgang	RxC
20	Data Terminal Ready	Ein-/Ausgang	DTR

In der Regel ist es nicht möglich, zwei **Datenendeinrichtungen** (DEEs, z.B. zwei Rechner) mit V.24-Anschluß direkt durch ein einfaches Kabelbündel zu verbinden. Die Normen sehen zur Verbindung zweier Datenendeinrichtungen den Einsatz von **Modems** (Datenübertragungseinrichtungen, DÜEs) vor. Dies ist für Entfernungen von wenigen Metern unwirtschaftlich. Um den Einsatz von Modems zu umgehen, bedarf es eines speziellen, an die Anforderungen der zu verbindenden Einheiten angepaßten Adapters. Ein Adapter in der einfachsten Form kann z.B. wie folgt realisiert werden:

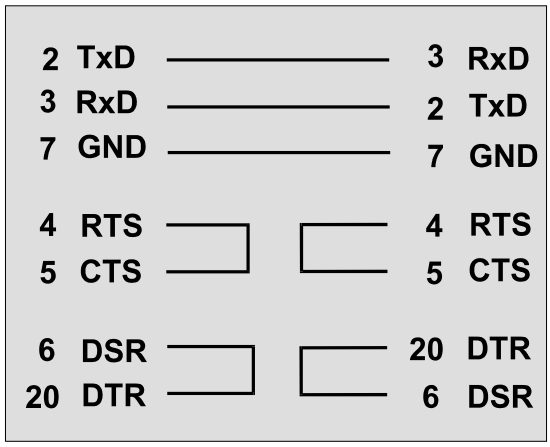


Bild 4.3-3: Die einfachste V.24-Schnittstelle

Die Datensignale werden auf den Leitungen TxD *(Transmit Data*) und RxD *(Receive Data*) übertragen. Als Bezugspotential dieser Signale dient das Signal Betriebserde (*Signal Ground*). Kurz gesagt, kann die Steuerung der Datenübertragung über die Steuerleitungen folgendermaßen vorgenommen werden:

Ist der Rechner (bzw. das Terminal) bereit, Daten auszutauschen, so signalisiert er dies durch ein H-Signal am Ausgang DTR (*Data Terminal Ready*) der V.24-Schnittstelle. Dabei ist die Richtung des Datenaustauschs nicht vorgegeben. Die Betriebsbereitschaft wird vom Kommunikationspartner über die Leitung DSR (*Data Set Ready*) durch einen H-Pegel mitgeteilt. Durch das Signal RTS kann der Prozessor in einem Peripheriegerät den Empfangsteil (*Receiver*) einschalten. Dieses Gerät kann wiederum über das Signal CTS den Prozessor über seine Empfangsbereitschaft informieren oder sogar erst den Sender (*transmitter*) in der Prozessorschnittstelle einschalten.

Benutzt man den oben dargestellten Adapter, so kontrollieren die verbundenen Einheiten jeweils die eigenen „kurzgeschlossenen“ Steuersignale. Der Anwender ist selbst für die Funktionsfähigkeit der Verbindung verantwortlich, da über die Steuerleitungen keine Informationen des Kommunikationspartners zur Verfügung stehen.

Gebräuchliche Datenübertragungsprotokolle der V.24-Schnittstelle

- **RDY/BSY-Prozedur**

  Die Ready/Busy-Prozedur (s. Bild 4.3-4) ist die einfachste und in der Praxis wohl am häufigsten verwendete Prozedur. Sie ist eine "Hardwareprozedur", d.h. der Anwender braucht keine spezielle Software für diese Prozedur, da die Abwicklung der Kommunikation von den Schnittstellenbausteinen der Geräte übernommen wird. Zur Signalisierung des Status "*Ready*" (bereit zum Datenaustausch) oder "*Busy*" (nicht bereit) wird im einfachsten Fall nur eine Steuerleitung benötigt. Soll so zum Beispiel ein Terminal an einem Rechner angeschlossen werden, ist die Signalleitung DTR des Terminals mit der Signalleitung DSR des Rechners zu verbinden. Ein positives Signal auf dieser Leitung signalisiert dem Rechner, daß das Terminal empfangsbereit ist (*Ready*), ein negatives Signal den *Busy*-Status.

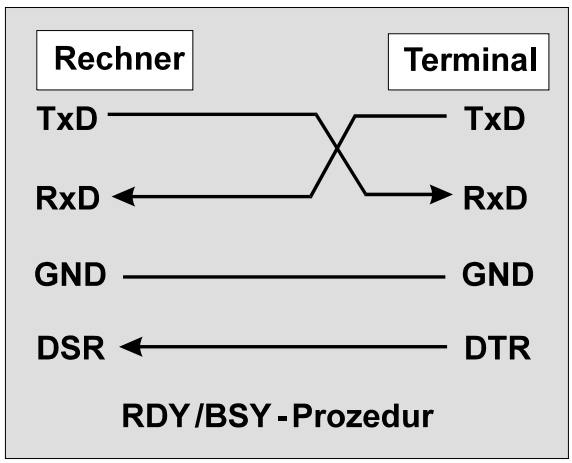


Bild 4.3-4: Die Schnittstelle der RDY/BSY-Prozedur

- **ETX/ACK-Prozedur**

Bei dieser Prozedur (s. Bild 4.3-5) werden die ASCII-Zeichen ETX ($03) und ACK ($06) zur Steuerung der Übertragung angewandt. Ist z.B. das Peripheriegerät bereit, Daten zu empfangen, so aktiviert es seine Signalleitung DTR und sendet das ACK-Zeichen (*acknowledge*) an den Rechner. Dieser sendet nun die Daten, die er mit dem ETX-Zeichen (*end of text*) abschließt. Wird vom Empfänger das Zeichen ETX unter den Daten erkannt, schickt dieser wiederum den ACK-Code zum Rechner und signalisiert damit, daß er das nächste Datenpaket übertragen kann. Wichtig ist, daß der ETX-Code im Datenfluß des Rechners entsprechend der Pufferkapazität des Peripheriegerätes eingesetzt wird, d.h. das jeweilige Datenpaket mit dem ETX-Code darf nicht größer als die Pufferkapazität des Empfängers sein.

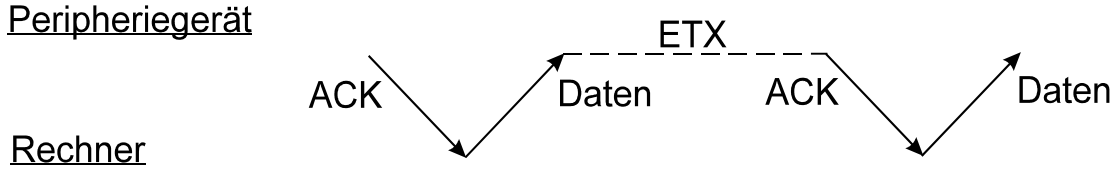


Bild 4.3-5: Die ETX/ACK-Prozedur

- **XON/XOFF-Prozedur**

Bei dieser Prozedur kommen die ASCII-Steuerzeichen XON ($11) und XOFF ($13) zur Anwendung. XON entspricht im ASCII-Code dem *Device Control DC1* und XOFF dem *Device Control DC3*. Das Peripheriegerät sendet bei Empfangsbereitschaft (*Ready*) den XON-Code, im anderen Fall (*Busy*) den XOFF-Code.

4.3.2 Aufbau des Bausteins R6551

Nachdem nun die für uns wesentlichen Teile der V.24-Schnittstelle und ihrer Handhabung beschrieben wurden, wird der Interface-Baustein R6551, kurz ACIA (*Asynchronous Communications Interface Adapter*) genannt, näher erklärt. Im Bild ist das Blockschaltbild der ACIA dargestellt[5]. Dieses läßt sich zunächst grob in 3 Teile aufteilen:

- den Empfängerteil,
- den Sendeteil und
- den Steuerungs- bzw. Überwachungsteil.

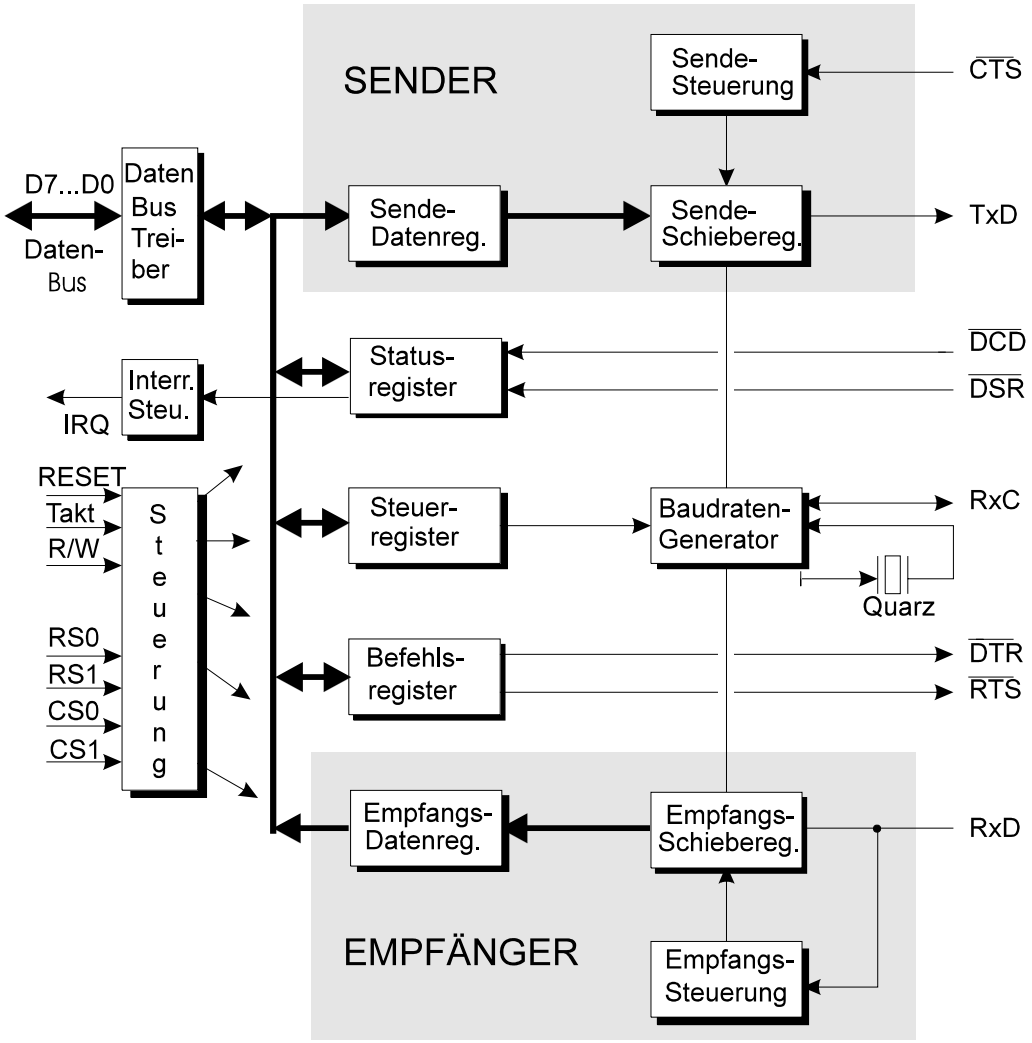


Bild 4.3-6: Blockschaltbild der ACIA R6551

Der **Empfängerteil** besteht aus dem Empfangs-Schieberegister, das die seriellen Daten über den Anschluß RxD aufnimmt und in paralleler Form an das Empfangs-Datenregister weiterleitet (Serien/Parallel-Wandlung). Dort stehen die Daten dann über den Datenbus-Treibern dem Prozessor zur weiteren Verarbeitung zur Verfügung.

[5] Der allgemeine Aufbau einer ACIA wird in Abschnitt II.3.6 ausführlich beschrieben.

Der **Sendeteil** enthält ein Sende-Datenregister, aus dem die Daten in paralleler Form in das Sende-Schieberegister übertragen werden. In diesem findet die Parallel/Serien-Wandlung statt. Die seriellen Daten werden über den Anschluß TxD ausgegeben.

Der **Steuerungs- und Überwachungsteil** enthält die weiteren Register, auf die im folgenden ausführlich eingegangen wird: Status-, Steuer- und Befehlsregister. Der "on-chip"-Baudratengenerator erzeugt aus der durch einen extern angeschlossenen Quarz stabilisierten Frequenz eine ganze Palette von unterteilten Frequenzen. Aus diesen muß die Übertragungsfrequenz selektiert werden, mit der die Schieberegister getaktet werden. (Der benutzte Grundtakt ist das 16fache des intern daraus abgeleiteten Schiebetaktes, so daß jedes Bit 16 Vollschwingungen dieses Grundtaktes lang ist.) Anstelle der intern erzeugten Frequenz können (am Eingang XTLI, s.u.) auch ein externer Takt angeschlossen werden und XTLO unbeschaltet bleiben.

Die Interruptlogik wertet die im Statusregister vorliegenden Ergebnisse einer Übertragung aus und unterbricht in bestimmten Fällen die CPU. Die letzte Komponente besteht aus der Schreib/Lese-Steuerung sowie der Auswahllogik für die einzelnen Register. Die Datenbus-Treiber sind bidirektional. Das folgende Bild 4.3-7 zeigt die Anschlußbelegung des R6551.

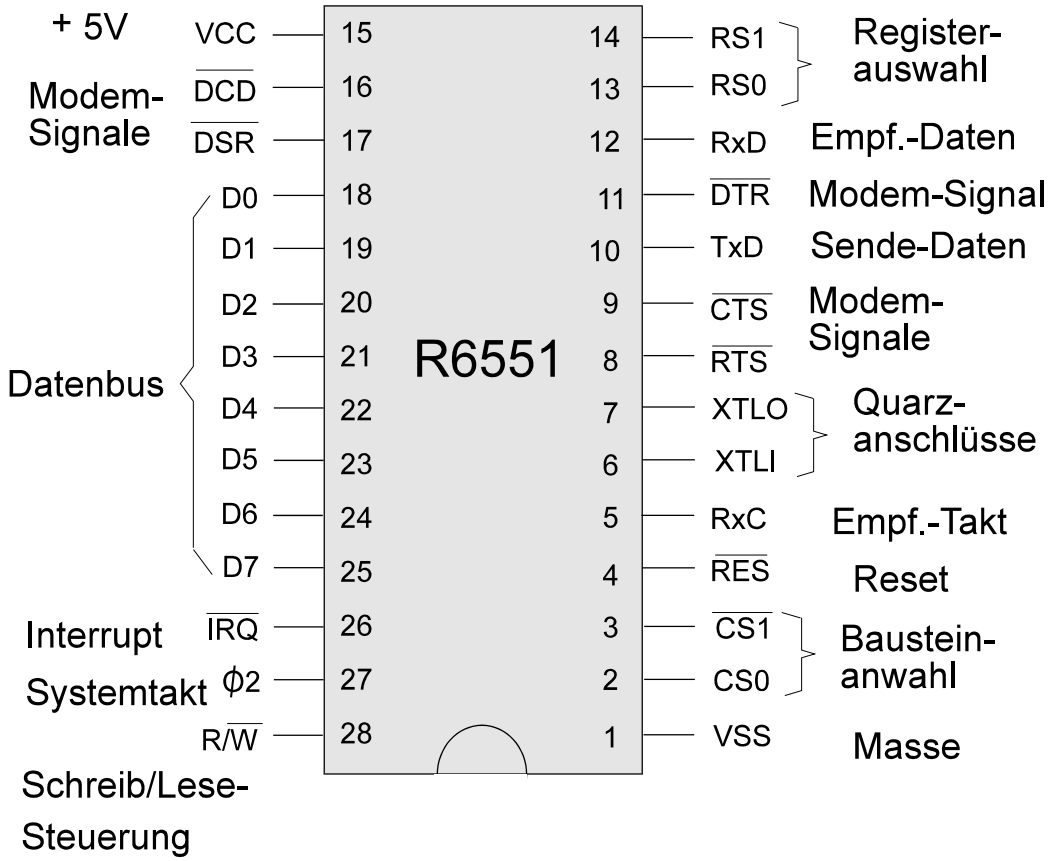


Bild 4.3-7: Anschlußbelegung des R6551

Die positive Flanke des Systemtaktes Φ2 synchronisiert alle internen Steuerungsvorgänge. Die Modem-Ausgangssignale DTR, RTS werden durch zugeordnete Bits im Befehlsregister unmittelbar beeinflußt, der Zustand der Eingangssignale DSR, DCD kann im Statusregister abgefragt werden. Die Bedeutung dieser Signale muß durch

die Übertragungssoftware zur V.24-Kommunikation berücksichtigt werden. Nur das Eingangssignal CTS wirkt sich - wie unten beschrieben - direkt auf den Sendeteil aus.

Die Selektion der ACIA geschieht über die beiden Eingänge CS0 und $\overline{CS1}$ (*Chip Select*). Dazu muß an CS0 ein H-Pegel und an $\overline{CS1}$ ein L-Pegel angelegt werden. Die Auswahl der einzelnen Register geschieht dann über Eingänge RS0 und RS1. Diese Eingänge sind im Mikrorechner mit den Adreßleitungen A0 und A1 der CPU verbunden. Dabei können einige Register nur beschrieben, andere nur gelesen werden, so daß zur Auswahl die Schreib-/Leseleitung R/$\overline{W}$ herangezogen werden kann. Ein H-Pegel (*read*) am R/$\overline{W}$-Eingang schaltet die Daten auf dem internen Datenbus der ACIA über die bidirektionalen Datenbus-Treiber auf den System-Datenbus des Mikrorechners durch. Ein L-Pegel an diesem Eingang überträgt die Daten vom System-Datenbus auf den Datenbus der ACIA (*write*). Tabelle 4.3-2 zeigt die Zuordnung der Register zu bestimmten Adressen im Mikrorechner und die mit den Registern ausführbaren Operationen.

Tabelle 4.3-2: Adressierung der ACIA-Register

			Register-Operation	
Adresse	RS1	RS0	Schreiben (R/$\overline{W}$=0)	Lesen (R/$\overline{W}$=1)
F008	0	0	Sende-Datenregister	Empfangs-Datenr.
F009	0	1	"Programmed Reset"	Statusregister
F00A	1	0	Befehlsregister	Befehlsregister
F00B	1	1	Steuerregister	Steuerregister

■ **Sendevorgang**

Zu Beginn eines Sendevorgangs wird das Statusregister ausgelesen, um zu prüfen, ob das Sende-Datenregister frei ist (TDRE-Bit, s.u.). Ist dies der Fall, so überträgt die CPU Daten zum Sende-Datenregister (*write F008*). Von dort gelangen sie ins Sende-Schieberegister und werden als serieller Datenstrom ausgegeben. Hierzu ist eine Formatierung notwendig, die gemäß des eingestellten Steuerwortes durchgeführt wird (Anzahl der Datenbits, Stopbits und evtl. Paritätsbit, s.u.).
Der Sendevorgang wird abhängig vom Zustand des Sende-Datenregisters fortgesetzt. D.h., sobald dieses frei ist, kann ein neues Datenwort zur Übertragung in die ACIA geschrieben werden. Da das Sende-Datenregister und das Sende-Schieberegister hintereinander geschaltet sind, kann ein neues Datenwort schon während der Ausgabe ins Sende-Datenregister geladen werden. Die folgende Skizze (Bild 4.3-8) zeigt die Übertragung mehrerer Zeichen unter Interrupt-Steuerung.

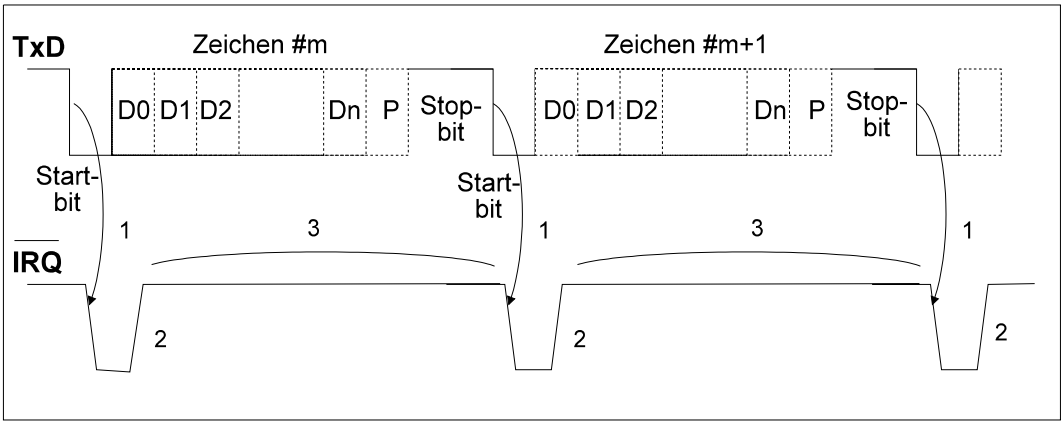


Bild 4.3-8: Zeichensenden unter Interrupt-Steuerung

Erklärung:

1. Mit dem Beginn des Startbits wird vom Sendeteil ein Interrupt IRQ zur CPU generiert.
2. Die IRQ-Anforderung wird dadurch zurückgenommen, daß die CPU in der Interrupt-Behandlungsroutine das Statusregister der ACIA liest.
3. Während der mit 3. bezeichneten Zeit muß die CPU (gewöhnlich ebenfalls in der Interrupt-Behandlungsroutine) das nächste Zeichen in das Sende-Datenregister schreiben. Ansonsten wird ein "*High Marking Signal*" ausgegeben.

Die nächste Skizze (Bild 4.3-9) zeigt den Fall, daß die CPU das nächste Zeichen nicht rechtzeitig liefert, also ein sog. Sende-Unterlauf (*Underrun*) auftritt.

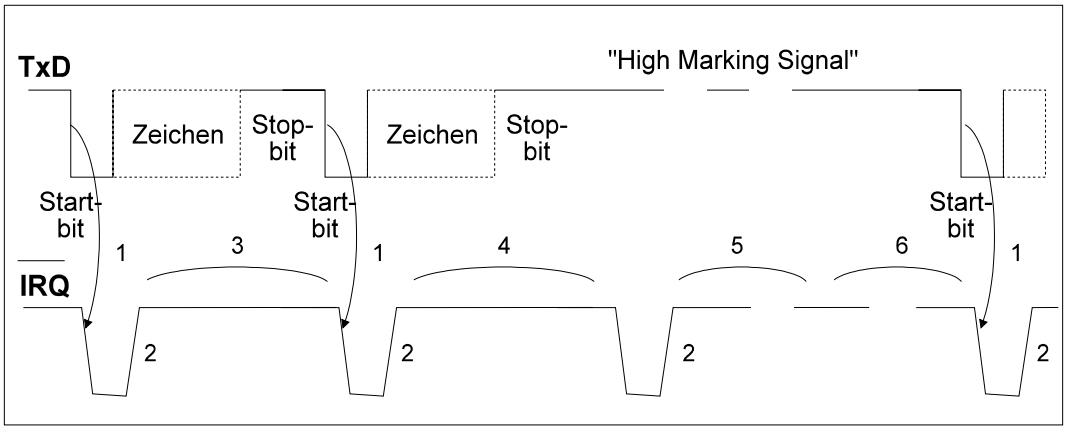


Bild 4.3-9: Auftreten eines Sende-Unterlaufs

Erklärung:

1. Mit dem Beginn des Startbits wird vom Sendeteil ein Interrupt IRQ zur CPU generiert.
2. Die (bzw. jede) IRQ-Anforderung wird dadurch zurückgenommen, daß die CPU in der Interrupt-Behandlungsroutine das Statusregister der ACIA liest.
3. Während der mit 3 bezeichneten Zeit muß die CPU (gewöhnlich ebenfalls in der Interrupt-Behandlungsroutine) das nächste Zeichen in das Sende-Datenregister schreiben.

4. Während der mit 4 bezeichneten Zeit schreibt die CPU kein neues Zeichen in das Sende-Datenregister. Daher tritt ein Underrun auf, was durch ein konstantes "*High Marking Signal*" angezeigt wird.
5. Obwohl keine weiteren Zeichen ausgegeben werden, wird in regelmäßigen Abständen, wie sie durch die Zeit 3 vorgegeben ist, eine Unterbrechungsanforderung über IRQ an die CPU gestellt.
6. Der Unterlauf wird beendet, wenn die CPU "rechtzeitig" ein neues Datum in das Ausgaberegister schreibt.

Zur Aktivierung des Sendeteils muß das Eingangssignal CTS auf L-Pegel liegen. Ein Übergang zum H-Pegel deaktiviert unmittelbar den Sendeteil und bricht dadurch eine laufende Übertragung ab. Dies wird durch einen konstanten H-Pegel (*high marking signal*) auf der Ausgangsleitung TxD dem angeschlossenen Gerät gemeldet. Der Zustand des CTS-Eingangs wird dem Prozessor durch kein Statusbit angezeigt. Auf ihn kann beim Vorliegen einer Interruptanforderung nur durch Ausschluß aller anderen Interrupt-Bedingungen geschlossen werden.

■ **Empfangsvorgang**

Über den seriellen Dateneingang gelangen die Daten zur ACIA. Sie werden ins Empfangs-Schieberegister geladen und dort auf das eingestellte Format hin überprüft. Formatfehler werden im Statusregister angezeigt. Vom Empfangs-Schieberegister gelangen die Daten ins Empfangs-Datenregister. Das Statusregister wird von der CPU vor dem Lesen der Daten aus dem Empfangs-Datenregister abgefragt. Wurde ein Datenwort empfangen und lag kein Fehler vor, kann die CPU die Daten zur weiteren Verarbeitung lesen. Die folgende Skizze (Bild 4.3-10) zeigt den Empfangsvorgang im einzelnen:

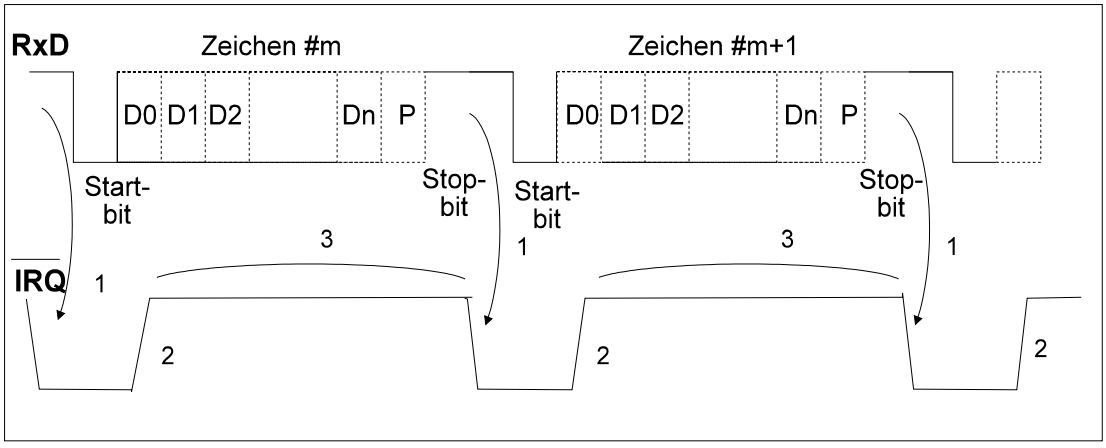


Bild 4.3-10: Zeichenempfang unter Interruptsteuerung

Erklärung:

1. Mit der 9. (von 16) Schwingungen des internen Grundtaktes vom Stopbit des empfangenen Zeichens wird vom Empfangsteil der ACIA ein Interrupt IRQ zur CPU generiert und diese dadurch zur Übernahme des Zeichens aufgefordert.
2. Die (bzw. jede) IRQ-Anforderung wird dadurch zurückgenommen, daß die CPU in der Interrupt-Behandlungsroutine das Statusregister der ACIA liest.
3. Während der mit 3 bezeichneten Zeit muß die CPU (gewöhnlich ebenfalls in der Interrupt-Behandlungsroutine) das Zeichen aus dem Empfangs-Datenregister lesen.

Die folgende Skizze (Bild 4.3-11) zeigt den Fall, daß die CPU das letzte empfangene Zeichen nicht rechtzeitig aus dem Empfangs-Datenregister liest, bevor das nächste Zeichen in der ACIA eingetroffen ist, also ein Überlauf (*Overrun*) stattfindet.

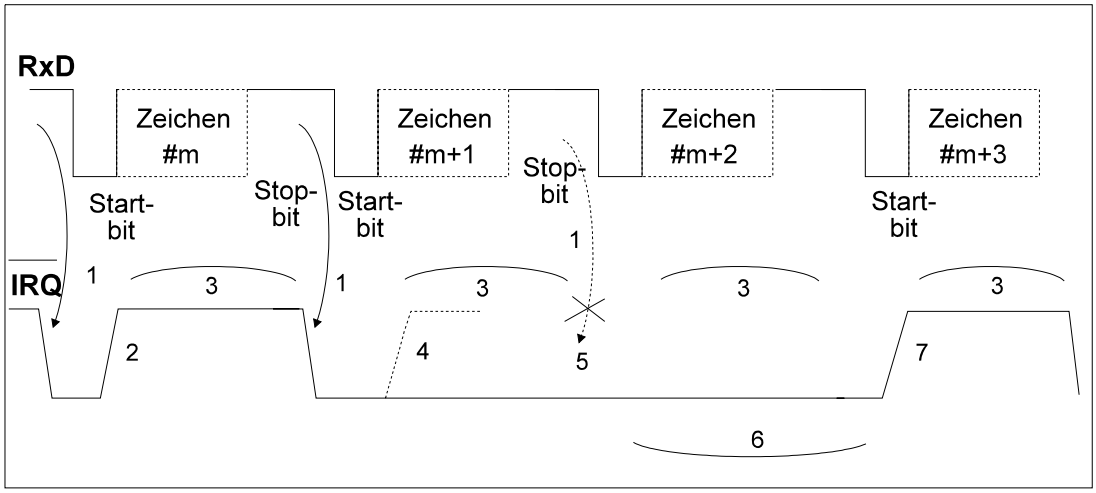


Bild 4.3-11: Auftreten eines Empfangsüberlaufs

Erklärung:

1. Mit der 9. (von 16) Schwingungen des internen Grundtaktes vom Stopbit jedes empfangenen Zeichens wird vom Empfangsteil der ACIA ein Interrupt IRQ zur CPU generiert und diese dadurch zur Übernahme des Zeichens aufgefordert.
2. Die (bzw. jede) IRQ-Anforderung wird dadurch zurückgenommen, daß die CPU in der Interrupt-Behandlungsroutine das Statusregister der ACIA liest.
3. Während der mit 3 bezeichneten Zeit muß die CPU (gewöhnlich ebenfalls in der Interrupt-Behandlungsroutine) das Zeichen aus dem Empfangs-Datenregister lesen.
4. Zu der mit 4 bezeichneten Zeit liest die CPU das neue Zeichen #m nicht aus dem Empfangs-Datenregister. Dadurch wird die Interruptanforderung nicht zurückgenommen.
5. Nach Ablauf einer weiteren Zeit 3 wird im Statusregister (s.u.) das Overrun-Flag gesetzt. Das dabei eingelesene Zeichen #m+1 wird nicht vom Empfangs-Schieberegister ins Empfangs-Datenregister übertragen.

6. Das nächste empfangene Zeichen #m+2 überschreibt das Zeichen #m+1 im Empfangs-Schieberegister. (Hier können noch weitere Zeichen folgen und das jeweils zuletzt empfangene überschreiben.)
7. Der Überlauf wird dadurch beendet, daß die CPU das Datum aus dem Empfangs-Datenregister, also das Zeichen #m, und das Statusregister liest. Die Zeichen #m+1 und #m+2 sind bei der Übertragung (im obigen Beispiel) verloren gegangen.

Der Empfangsvorgang kann wahlweise durch die intern erzeugte Frequenz des Baudraten-Generators oder aber durch eine externe Taktfrequenz am Eingang RxC (*Receive Clock*) gesteuert werden. Im Fall der internen Takterzeugung wird am Anschlußpin RxC, der nun als Ausgang geschaltet ist, die 16fache Frequenz der programmierten Baudrate, also der oben beschriebene interne Grundtakt, ausgegeben und kann so zur (synchronisierten) Steuerung weiterer ACIAs dienen.

4.3.3 Der Registersatz des R6551

Das Statusregister

Das Statusregister (SR) ist 8 bit breit und kann vom Prozessor nur gelesen werden. Ein Schreibbefehl mit der Adresse des Statusregisters hat zur Folge, daß das Bit 2 im Statusregister*) und die Bits 0,2,3,4 im Befehlsregister gelöscht (Bit 1 gesetzt) werden (*programmed reset*) und dadurch der Baustein in einen definierten Grundzustand versetzt wird.

Die einzelnen Bits des Statusregisters nach Bild 4.3-12 geben Auskunft über den Zustand der Datenregister und der externen Kontroll-Signale (wie DSR und DCD), sowie über das Vorliegen von Übertragungsfehlern (Format-, Paritäts- und Überlauffehler) und dem daraus abgeleiteten Vorliegen einer Unterbrechungsbedingung.

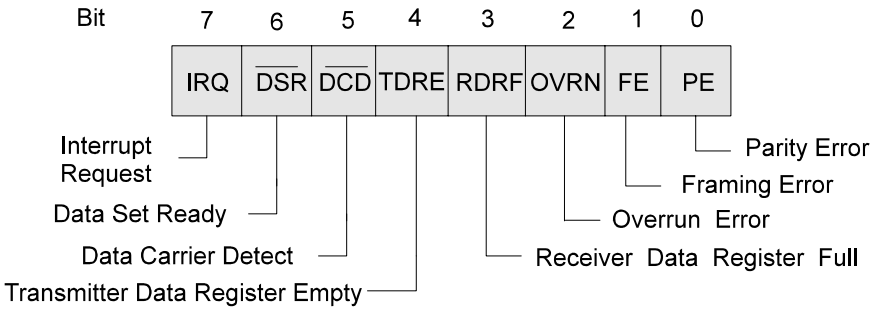


Bild 4.3-12: Der Aufbau des Statusregisters (Adresse: $F009)

- **Bit 7: Interrupt Flag**

  Dieses Bit wird gesetzt, wenn wenigstens eines der Bits 3 oder 4 gesetzt wurde bzw. sich der Zustand eines der Eingangssignale DSR oder DCD geändert hat. (Es wird jedoch nicht gesetzt, wenn eine der durch die Bits 2-0 beschriebenen Fehlerbedingungen (s.u.) auftritt. Diese müssen daher bei jeder Übertragung durch die Software "abgefragt" werden.) Es wird gelöscht beim Lesen des Statusregisters. Das Bit 7 hat keinen direkten Einfluß auf den Interruptausgang IRQ zum Prozessor. Es dient dem Prozessor lediglich zur Feststellung der Interruptbedingung. Das Durchschalten des Bits 7 zum Interruptausgang IRQ der ACIA wird durch Setzen der entsprechenden Bits im Befehlsregister aktiviert (*enabled*) oder deaktiviert (*disabled*).

- **Bit 6: Data Set Ready**

  Dieses Bit gibt den Zustand des entsprechenden Eingangs $\overline{\mathrm{DSR}}$ der ACIA wieder, das vor allem beim Modem-Betrieb benutzt wird.

  - $\overline{DSR} = 0$ bedeutet, daß das Modem bzw. der Kommunikationspartner bereit ist, Daten auszutauschen (*ready*),
  - $\overline{\mathrm{DSR}} = 1$ daß es/er dazu nicht bereit ist (*not ready*).

*) Die Bits 0 und 1 werden automatisch nach einer erfolgreichen Übertragung des nächsten Zeichens gelöscht.

- **Bit 5: Data Carrier Detect**

  Dieses Bit gibt den Zustand des entsprechenden Eingangs $\overline{DCD}$ der ACIA wieder, das vor allem beim Modem-Betrieb benutzt wird.

  - $\overline{DCD} = 0$ zeigt an, daß das Modem ein Trägersignal auf der Übertragungsleileitung festgestellt hat (*Carrier detected*) und daß es Daten zum Empfangsteil der ACIA überträgt; bei unvorhergesehenem Abbruch der Übertragung legt es dieses Signal auf H-Pegel und überträgt einen konstanten H-Pegel zum Eingang RxD,
  - $\overline{DCD} = 1$ zeigt, daß das Modem dieses Signal nicht feststellt (*Carrier not detected*).

- **Bit 4: Transmitter Data Register Empty**

  Durch dieses Bit kann der Prozessor feststellen, ob das letzte Datum, das er übertragen hat, den Baustein bereits "verlassen" hat: Es wird dadurch gelöscht, daß der Prozessor ein Datum in das Sende-Datenregister schreibt. Es wird von der ACIA gesetzt, wenn sie ein Datum vom Sende-Datenregister ins Sende-Schieberegister überträgt. Das ist aber bei kontinuierlicher Übertragung von Daten erst dann der Fall, wenn das (letzte) Stopbit des vorhergehenden Zeichens bereits gesendet wurde. Vor der Übertragung eines Zeichens in das Sende-Datenregister sollte daher die CPU stets prüfen, ob dieses Bit den Wert 1 hat. Wird das nächste Zeichen erst mit einer Verzögerung von der CPU geschrieben, gibt der Baustein den H-Pegel (des Stopbits) als "*High Marking Signal*" aus.

- **Bit 3: Receiver Data Register Full**

  Der ACIA-Baustein setzt dieses Bit beim Transfer der Daten vom Empfangs-Schieberegister zum Empfangs-Datenregister. Es wird gelöscht, wenn der Prozessor die Daten vom Empfangs-Datenregister liest. Durch dieses Bit kann also der Prozessor feststellen, ob ein neues Datum empfangen wurde und im Empfangs-Datenregister bereitsteht. Liest der Prozessor dieses Datum nicht rechtzeitig aus, kann es zu einem "Überlauf" (*overrun*, s.u.) des Empfangs-Schieberegisters kommen, wenn bereits das nächste Zeichen empfangen wird.

- **Bit 2: Overrun**

  Das Overrun-Bit wird gesetzt, wenn ein neues Datum ins Empfangs-Schieberegister übertragen wurde, bevor das vorhergehende vom Prozessor aus dem Empfangs-Datenregister gelesen wurde. Der Inhalt des Empfangs-Schieberegisters wird dabei überschrieben und geht verloren. Anhand dieses Bits kann der Prozessor also feststellen, ob er alle Daten bekommen hat. Von dieser Möglichkeit sollte bei jeder Übertragung Gebrauch gemacht werden, um eine Konsistenz der empfangenen Daten zu sichern. Es ist zu beachten, daß das Auftreten eines Overrun-Fehlers nicht zum Setzen des Interrupt-Flags (Bit 7) führt.

- **Bit 1: Framing Error**

  Ein Formatfehler wird angezeigt, wenn das empfangene Datenformat zwischen Start- und Stopbit(s) nicht mit dem im Steuerregister eingestellten übereinstimmt. Dies ist z.B. dann der Fall, wenn im Steuerregister zwei Stopbits eingestellt wurden, jedoch nur ein Stopbit empfangen wird. Ist das nächste übertragene Datenwort "in Ordnung", wird Bit 1 wieder gelöscht. Es wird aber auch automatisch durch das Lesen des Datenregisters gelöscht. Ein Formatfehler führt zu keiner Interruptanforderung.

- **Bit 0: Parity Error**

  Ist das Bit 5 im Befehlsregister gesetzt, wird beim Senden ein Paritätsbit generiert und beim Empfang eine Paritätsprüfung durchgeführt. Ergibt diese einen Fehler, wird das Bit 0 gesetzt. Ist im nächsten Datenwort das Paritätsbit "in Ordnung", wird Bit 0 wieder gelöscht. Es wird aber auch automatisch durch das Lesen des Datenregisters gelöscht. . Ein Paritätsfehler führt zu keiner Interruptanforderung.

Das folgende Flußdiagramm (Bild 4.3-13) zeigt die Abfrage der verschiedenen Statusbits in einer Unterbrechungsbehandlungs-Routine zur Einleitung der geeigneten Maßnahmen - Datenübertragung oder Fehlerbehandlung.

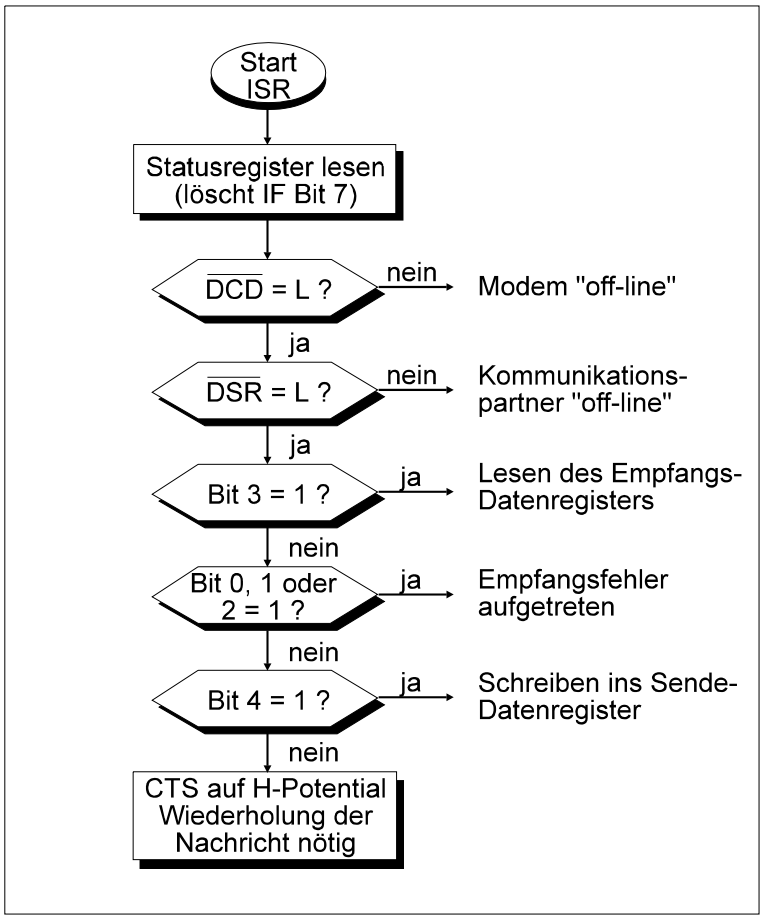


Bild 4.3-13: Flußdiagramm zur Abfrage der Statusbits

Das Befehlsregister

Das Befehlsregister (*Instruction Register* - IR, *Command Register*) hat den im Bild 4.3-14 gezeigten Aufbau. Es bestimmt die im folgenden beschriebenen Funktionen und Modi.

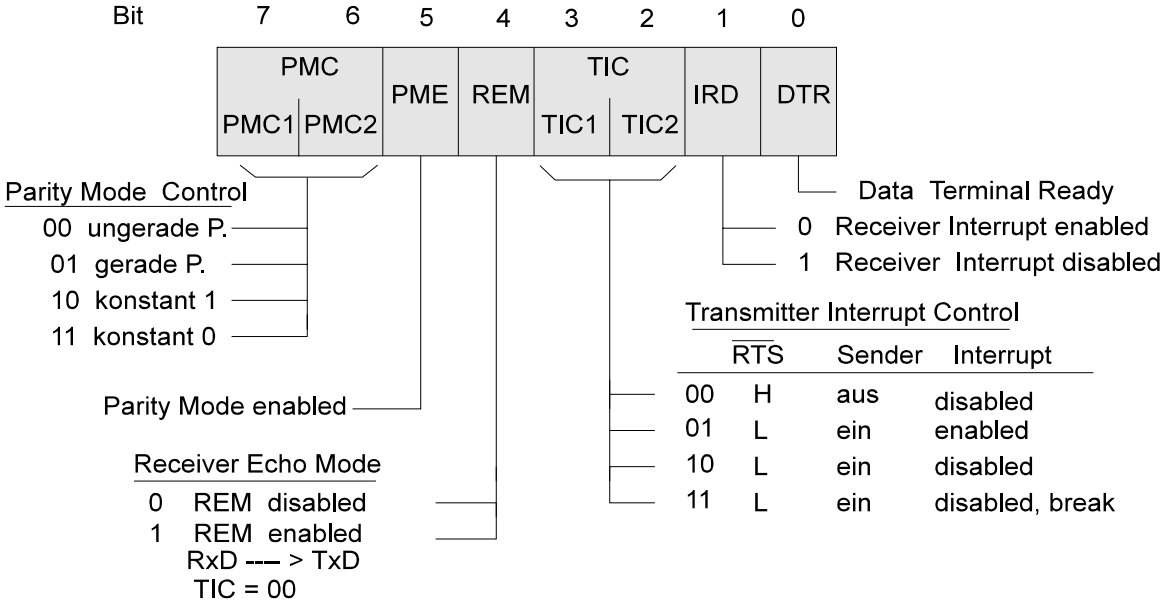


Bild 4.3-14: Aufbau des Befehlsregisters (Adresse: $F00A)

- **Bit 6 / Bit 7: PMC** (*Parity Mode Control*)

  Ist die Betriebsart mit Paritätsprüfung gewählt (s. Bit 5), so können mit Hilfe dieser beiden Bits folgende Steuermöglichkeiten für die Paritätsprüfung selektiert werden. Dabei muß in den Fällen, in denen ein Paritätsbit erzeugt und geprüft wird, im Sender und Empfänger natürlich dieselbe Parität (gerade oder ungerade) selektiert werden, da sonst bei jedem Zeichen ein Paritätsfehler festgestellt wird. Wird vom Empfänger **kein** Paritätsbit ausgewertet, aber wegen eines vorgegebenen Formats nach den Datenbits ein weiteres Bit erwartet, so besteht die Möglichkeit, dieses zusätzliche Bit hinter jedem übertragenen Zeichen konstant auf logisch "0" oder logisch "1" zu setzen. Nach Bild 4.3-2 entspricht der 1. Fall dem Zustand des Startbits, das zur Markierung des Anfangs eines Zeichens (*mark bit*) dient, der 2. dem Zustand des Stopbits, das auch als Leerzeichen oder Zwischenraum (*space*) zwischen zwei Zeichen betrachtet werden kann.

Tabelle 4.3-3: Paritätsprüfung

Bit 7	Bit 6	Steuerung
0	0	ungerade Parität (*odd parity*)
0	1	gerade Parität (*even parity*)
1	0	konstant 0-Bit (*mark parity*)
1	1	konstant 1-Bit (*space parity*)

- **Bit 5: PME** (*Parity Mode Enabled*)

  Bit 5=1: Hierdurch wird die Generierung und Überprüfung eines Paritätsbits aktiviert. Bit 6 und Bit 7 geben dazu die Art der Paritätsbildung an.

  Bit 5=0: Es wird kein Paritätsbit gebildet und dementsprechend auch keine Paritätsprüfung durchgeführt. Bit 6 und Bit 7 haben deshalb keine Bedeutung.

- **Bit 4: REM** (*Receiver Echo Mode*)

  Ist Bit 4 gesetzt, so wird jedes über den Eingang RxD empfangene Zeichen - am Sendeteil vorbei - über den Ausgang TxD (mit einer Verzögerung von einer halben Bitzeit) zurückgesendet. Für diese Betriebsart müssen Bit 3 und Bit 2 (s.u.) auf 0 gesetzt, d.h. der Sendeteil deaktiviert sein. Dieser Modus ist für Testzwecke und für den Anschluß eines Terminals vorgesehen, bei dem jedes zum Rechner gesendete Zeichen erst nach dem Zurücksenden durch die ACIA auf dem Bildschirm erscheint. (Dadurch bekommt der Benutzer auf einfache Weise eine Kontrolle, ob das gewünschte Zeichen eingegeben wurde.)

- **Bit 3 / Bit 2: TIC** (*Transmitter Interrupt Control*)

  Diese beiden Bits steuern den Zustand des Sendeteils in der ACIA und der zugeordneten Signalleitung $\overline{\text{RTS}}$. Ist der Sendeteil "eingeschaltet", so liegt der Ausgang $\overline{\text{RTS}}$ auf L-Pegel (*Transmitter enabled*). Ist der Sendeteil ausgeschaltet, so ist $\overline{\text{RTS}}$ auf H-Pegel (*Transmitter disabled*). In diesem Fall kann der Empfangsteil weiter arbeiten. Außerdem bestimmen Bit 3 und Bit 2, ob durch das Setzen des Bit 4 im Statusregister (TDRE - *Transmitter Data Register Empty*) eine Unterbrechungsanforderung zur CPU durchgeschaltet wird (*Transmitter Interrupt enabled*) oder nicht (*Transmitter Interrupt disabled*). Der Fall Bit3=0, Bit2=1 legt den "normalen" Betriebsmodus fest, bei dem die Zeichenübertragung durch Interruptsteuerung vorgenommen wird und das Signal $\overline{\text{RTS}}$ durch einen L-Pegel seine Übertragungsbereitschaft anzeigt. Im letzten Fall der folgenden Tabelle 4.3-4 sendet die ACIA solange über den TxD-Ausgang als spezielles Unterbrechungs-Zeichen (*Break*) einen L-Pegel, bis die Bits 2 und 3 von der CPU "umprogrammiert" werden. Dieses Zeichen informiert den Empfänger ohne zusätzliche Steuerleitung darüber, daß der Sender nicht bereit ist.

Tabelle 4.3-4: Steuerung des Senders

Bit3	Bit2	$\overline{\text{RTS}}$	Sendeteil	Transmitter Interrupt
0	0	H	aus	nicht aktiviert
0	1	L	ein	aktiviert
1	0	L	ein	nicht aktiviert
1	1	L	ein	nicht aktiviert, Break

- **Bit 1: IRD** (*Receiver Interrupt Request Disabled*)

  Dieses Bit steuert die De-/Aktivierung des IRQ-Ausgangs zur CPU, nicht aber die interne Interruptlogik, d.h. Interruptanforderungen werden bei Vorliegen der entsprechenden Bedingungen zwar intern generiert, aber nicht zur CPU weitergereicht.

  Bit 1=1: Der Empfängerteil kann **keine** Unterbrechungsanforderung an die CPU stellen (*Receiver Interrupt disabled*).

  Bit 1=0: Unterbrechungsanforderung durch den Empfängerteil werden zugelassen (*Receiver Interrupt enabled*), wenn gleichzeitig Bit 0=1 ist.

- **Bit 0: DTR** (*Data Terminal Ready*)

  Dieses Bit steuert den gesamten Baustein (m*aster enable/disable*) und speziell den Zustand der Ausgangsleitung DTR, die stets den inversen Zustand des Bits 0 hat.

  Bit 0=0: Die CPU ist nicht bereit, Daten auszutauschen (*disabled*). Das Signal DTR liegt auf H-Potential (*Data Terminal not Ready*). Durch das Rücksetzen dieses Bits während eines laufenden Sendevorgangs wird der Sendeteil augenblicklich abgeschaltet. Während eines Empfangsvorgangs wird vor dem Abschalten des Empfangsteils zunächst die Übertragung des Zeichens beendet.

  Bit 0=1: Die CPU ist bereit zur Datenübertragung (*enabled*). Das Signal DTR liegt auf L-Potential (*Data Terminal Ready*).

Hinweis:
Die Startadresse der Interruptroutine, die eine Unterbrechungsanforderung der ACIA bedient, ist im Praktikumsrechner unter den Adressen

$0040, $0041

abgelegt und kann dort beliebig geändert werden. Nach dem Rücksetzen des Rechners zeigt die Startadresse auf den Befehl RTI (*Return from Interrupt*).

Das Steuerregister

Das Steuerregister (*Control Register* - CR) ist ebenfalls 8 bit breit. Es enthält Informationen über die Übertragungsgeschwindigkeit, die Quelle des Empfangs-Taktes sowie über das Format des **Übertragungsrahmens**, d.h. über die Wortlänge und die Zahl der Stopbits (s. Bild 4.3-15).

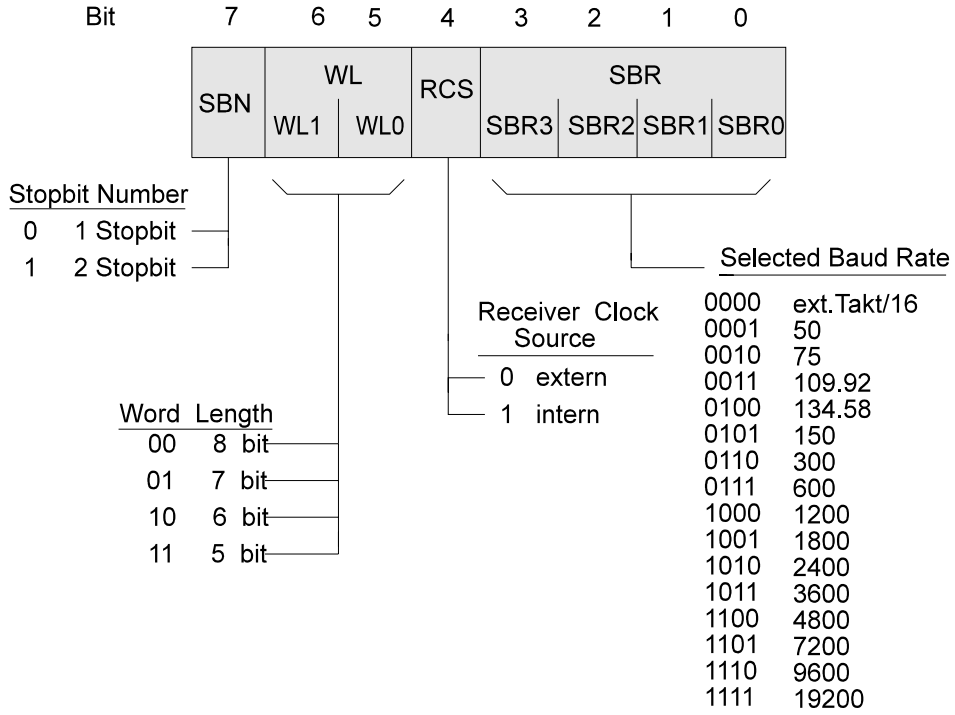


Bild 4.3-15: Der Aufbau des Steuerregisters (Adresse: $F00B)

- **Bit 7: SBN** (*Stop Bit Number*)

  Bit 7=0: Die Daten werden mit einem Stopbit abgeschlossen.
  Bit 7=1: im allgemeinen 2 Stopbits nach jedem Zeichen; zur Unterstützung verschiedener Protokolle generiert die Hardware jedoch 1½ Stopbits bei einer Wortlänge von 5 bit ohne Paritätsbit bzw. 1 Stopbit bei einer Wortlänge von 8 bit mit Paritätsbit.

- **Bit 6-5: WL** (*Word Length*)

  Diese Bits bestimmen die Wortlänge der Übertragungsdaten nach Tabelle 4.3-4.

Tabelle 4.3-4: Wortlänge der Datenübertragung

Bit 6	Bit 5	Wortlänge
0	0	8 bit
0	1	7 bit
1	0	6 bit
1	1	5 bit

- **Bit 4: RCS** (*Receiver Clock Source*)

  Bit 4=1: Sende- und Empfangstakt werden intern erzeugt und sind identisch, damit also auch die Übertragungsgeschwindigkeiten in beiden Richtungen (RxC = TxC, C=Clock). Am Ausgang RxC wird ein Signal mit der 16fachen Frequenz der programmierten Baudrate (s. Bits 3-0) zur Ansteuerung weiterer ACIAs ausgegeben.

  Bit 4=0: Der Datenempfang wird durch einen externen Takt gesteuert. Der externe Takt ist an RxC (Pin5) der ACIA anzuschließen. Beim Praktikumsrechner ist leider der Anschluß eines externen Taktes nicht möglich. D.h. für den Betrieb der V.24-Schnittstelle muß dieses Bit gesetzt sein, und die Übertragungsgeschwindigkeiten der Kommunikationspartner müssen gleich sein.

- **Bit 3-0: SBR** (*Selected Baud Rate*)

  Die Tabelle 4.3-5 zeigt die durch die Bits 3-0 selektierbaren Übertragungsgeschwindigkeiten, wie sie für den Praktikumsrechner gelten. (Diese Werte sind wählbar, wenn zwischen den Bausteinanschlüssen XTALI, XTALO ein 1.8432-MHz-Quarz angeschlossen ist. Wird ein Quarz mit einer anderen Frequenz f (in MHz) benutzt, so sind die angegebenen Baudraten mit dem Faktor f/1.8432 zu multiplizieren.)

Tabelle 4.3-5: wählbare Übertragungsraten

Bit	3	2	1	0	Baud-Rate
	0	0	0	0	(externer Takt)/16
	0	0	0	1	50
	0	0	1	0	75
	0	0	1	1	109.92
	0	1	0	0	134.58
	0	1	0	1	150
	0	1	1	0	300
	0	1	1	1	600
	1	0	0	0	1200
	1	0	0	1	1800
	1	0	1	0	2400
	1	0	1	1	3600
	1	1	0	0	4800
	1	1	0	1	7200
	1	1	1	0	9600
	1	1	1	1	19200

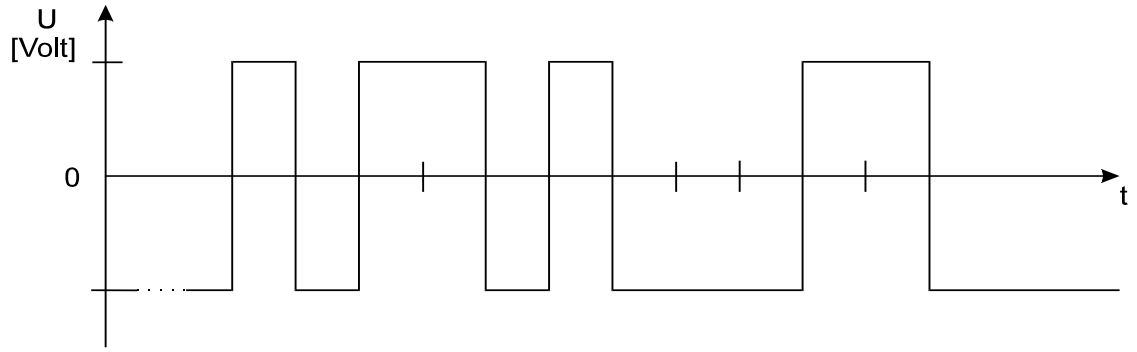
Selbsttestaufgabe S4.3-1:

a) Mit welchen Werten in binärer und hexadezimaler Form müssen das Steuer- und Befehlsregister/Instruktionsregister (CR, IR) initialisiert werden, wenn die Schnittstelle in der folgenden Betriebsart eingesetzt werden soll:

- 7-bit-Zeichen,
- gerade Parität,
- 1 Stopbit,
- 2400 baud,
- interner Empfängertakt,
- *Receiver Echo Mode disabled*
- Sender- und Empfänger-Interrupt zugelassen (*enabled*)
- DTR=1

b) Wie sieht für die unter a) bestimmte Übertragungsart das Timing-Diagramm auf der V.24-Übertragungsleitung TxD für das ASCII-Zeichen 'K' ($4B) aus ? Tragen Sie auf ihrem Lösungsblatt auch geeignete Spannungswerte für H- und L-Pegel und die Bitdauer b ein. Skizzieren Sie auch das „Pausensignal“ bis zur Übertragung des nächsten Zeichens.

c) In der Betriebsweise, für die die Schnittstelle unter a) initialisiert wurde, wird auf der V.24-Empfangsleitung RxD folgendes Timing-Diagramm beobachtet:



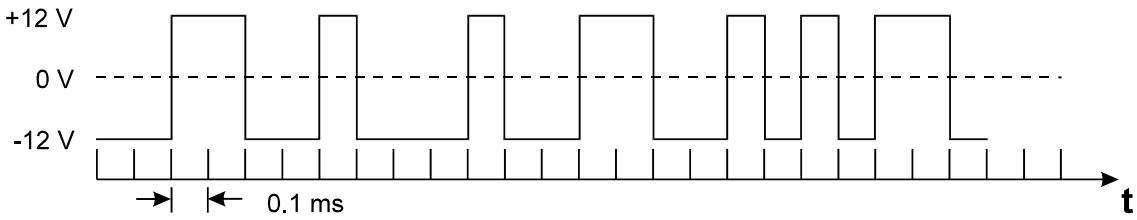
Bestimmen Sie den Inhalt des Statusregisters SR in binärer und hexadezimaler Form vor dem Lesen des Empfangsdatenregisters RDR durch den µP. Setzen Sie dabei voraus, daß die Schnittstelle als Null-Modem betrieben wird, d.h. daß an der Schnittstelle RTS und CTS sowie DSR und DTR miteinander verbunden sind. Das Sendedatenregister TDR ist leer und DCD=1, d.h. $\overline{DCD} = 0$.

Praktische Übung P4.3-1:

Programmieren Sie die V.24-Schnittstelle im 6809-Simulator, wie in Selbsttestaufgabe S4.3-1a vorgegeben. Geben Sie nun das Zeichen 'K' aus und vergleichen Sie das Zeitdiagramm mit dem in der Selbsttestaufgabe S4.3-1b ermitteltem Bild. (Hinweis: Verkleinern Sie das ACIA-Fenster durch Anklicken der Raute in der oberen, rechten Ecke.)

Selbsttestaufgabe S4.3-2:

Das folgende Bild stellt die serielle Übertragung **zweier** Zeichen auf einer V.24-Leitung dar, wie sie mit dem Oszilloskop zu beobachten ist. Vorausgesetzt ist, daß das niederwertige Bit jedes Zeichens zuerst übertragen wird und bei der Übertragung kein Fehler auftritt.



Es werde vorausgesetzt, daß die übertragenen Zeichen aus dem erweiterten ASCII-Zeichensatz stammen, wie er in der Tabelle im Anhang D dieses Kapitels angegeben ist.

a) Berechnen Sie die Übertragungsrate in bd (Baud). (Als Ergebnis soll einer der Werte aus Tabelle 4.3-5 angegeben werden.)

b) Wieviele Stopbits werden nach jedem Zeichen übertragen ?
Kennzeichnen Sie die übertragenen Bits durch die logischen Werte ‘0’ und ‘1’ bzw. durch SB (Startbit) oder StB (Stopbit).

c) Wird die Übertragung der Zeichen durch ein Paritätsbit gesichert ?
Wenn nein, begründen Sie Ihre Antwort.
Wenn ja, geben Sie die Art dieses Paritätsbits an (*even, odd, mark, space*).

d) Werden die Zeichen im 7-bit- oder 8-bit-Code übertragen ?
Wie lauten die Zeichen in binärer und hexadezimaler Form ?

e) Welche Buchstabenkombination wird übertragen ?

Praktische Übung P4.3-2:

Programmieren Sie die V.24-Schnittstelle im 6809-Simulator, wie in Selbsttestaufgabe S4.3-2 vorgegeben. Geben Sie nun die dort unter e) ermittelte Buchstabenkombinationen ein und vergleichen Sie das Zeitdiagramm mit dem in der Selbsttestaufgabe S4.3-2 gezeigten Bild.

4.3.4 Kommunikation mit dem PC

In Kapitel 2 haben Sie bereits gelernt, wie Sie Programme und Datenbereiche zwischen dem PC und dem Praktikumsrechner austauschen können und damit z.B. auf einer Festplatte abspeichern bzw. von dort lesen können. Dies geschah auf einer hohen Anwendungsebene, ohne daß Sie wissen mußten, wie die V.24-Schnittstelle im Praktikumsrechner funktioniert und programmiert werden kann. In Kapitel 3 wurde dann bei der Beschreibung der Monitorroutinen – im Vorgriff auf die Beschreibung der V.24-Schnittstelle in diesem Kapitel – das Programm LOAD zum Laden von Programmen und Daten in den Praktikumsrechner analysiert (vgl. Praktische Übung P3.2-6). Nachdem Sie in den vorangehenden Unterabschnitten die V.24-Schnittstelle ausführlich kennengelernt und in den Übungen bereits erste praktische Anwendungen dazu durchgeführt haben, sollen Sie sich in diesem Unterabschnitten nun ausführlich mit der Kommunikation zwischen zwei Rechnern über die V.24-Schnittstelle beschäftigen, und zwar auf einer sehr niedrigen, hardwarenahen Stufe. Die benutzten beiden Rechner sind Ihr PC sowie der Praktikumsrechner. Zur Vorbereitung verbinden Sie bitte beide mit dem zur Verfügung gestellten Kabel und stellen Sie die V.24-Schnittstelle des PCs, wie in Kapitel 2 beschrieben, (mit Hilfe des MODE-Befehls) ein.

Praktische Übung P4.3-3:

Schreiben Sie ein Programm, daß einen Datenblock vom PC in den Praktikumsrechner überträgt und ab einer eingebbaren Adresse im Speicher ablegt.

Im einzelnen soll das Programm:

1. die Anzeige löschen,
2. die Kennung "AA" (Anfangs-Adresse) im Operationsfeld der Anzeige ausgeben und im Adreßfeld die 4-stellige Startadresse eines Pufferbereiches einlesen, (Beenden der Eingabe durch die Taste '+')
3. die Anzeige löschen,
4. das Statusregister der ACIA solange abfragen, bis es das Vorliegen eines übertragenen Zeichens anzeigt,
5. das empfangene Zeichen im Datenfeld anzeigen und im Puffer ablegen,
6. sich selbst mit einem Sprung ins Monitorprogramm abbrechen, wenn das Zeichen '.' (Punkt) empfangen wurde, sonst
7. danach die Pufferadresse erhöhen, diese im Adreßfeld anzeigen und zyklisch mit 4. fortfahren.

Zur Arbeit mit dem Emulator:

Nachdem Sie dieses Programm mit dem Editor erstellt und assembliert haben, können Sie es nun mit dem 6809-Emulator austesten. Starten Sie dazu den Emulator und laden Sie das Maschinenprogramm (Endung .HEX) in den Arbeitsspeicher des emulierten 6809-Prozessores (Menüpunkt: Datei, Eintrag: laden)[1]. Nun können Sie

lierten 6809-Prozessores (Menüpunkt: Datei, Eintrag: laden)[1]. Nun können Sie durch Anklicken der Ikone

eine Datei bestimmen, in die alle ins Sende-Datenregister (Adresse $F008) geschriebenen ASCII-Zeichen als Zeichenkette eingetragen werden[2]. (Die Übertragung der Zeichenkette vom Emulator in die Datei wird in der Statuszeile durch ein Diskettensymbol mit der Beschriftung ACIA auf der linken Seite angezeigt.) Die Zeichenkette müssen Sie - wie oben verlangt - durch einen Punkt abschließen. Durch die Ikone

können Sie die Aufzeichnung stoppen. Danach starten Sie die Empfangsroutine (zweifaches Betätigen der Taste 'G') und öffnen durch die Ikone

die eben erzeugte Datei zur Eingabe der Zeichenkette. (Die Übertragung der Zeichenkette aus der Datei in den Emulator wird in der Statuszeile durch ein Diskettensymbol mit der Beschriftung ACIA auf der rechten Seite angezeigt.) Wenn Ihr Programm fehlerfrei arbeitet, können Sie dann im Pufferbereich die eingegebene Zeichenkette wiederfinden.

Wenn Ihr Programm im Emulator fehlerfrei läuft, können Sie es im Praktikumsrechner ablaufen lassen. Dazu müssen Sie es, wie in Kapitel 2 beschrieben, (mit dem mitgelieferten Programm TERM9.EXE) in den Praktikumsrechner übertragen. Nun erstellen Sie sich mit Hilfe des Editors eine beliebige ASCII-Datei und beenden die Eingabe mit einem Punkt. Starten Sie danach das erstellte Programm auf dem Praktikumsrechner und übertragen Sie die ASCII-Datei mit dem MS-DOS-Befehl "COPY<Dateiname> COMn:" vom PC in den Praktikumsrechner. Überprüfen Sie wiederum die Übertragung durch Vergleich der in der Datei und im Puffer abgelegten ASCII-Zeichen.

Nach dem V.24-Empfangsprogramm sollen Sie nun auch das Gegenstück, also ein Programm zum Senden von ASCII-Zeichen entwickeln. Natürlich könnten Sie das in einer höheren Programmiersprache oder aber in 80x86-Assembler schreiben und auf dem PC ablaufen lassen. Beide Programmiersprachen sind jedoch nicht Gegenstand dieses Praktikums.

Der 6809-Emulator liefert aber die Möglichkeit, den PC als vollständigen Ersatz für einen 2. 6809-Rechner zu benutzen. Dazu können Sie die emulierte V.24-Schnittstelle auf die (in Hardware realisierte) V.24-Schnittstelle Ihres PCs abbilden (Menü: Datei, Eintrag: Schnittstellen). Danach sind Sie in der Lage, durch ein vom Emulator abgearbeitetes 6809-Programm auf die PCSchnittstellen COM1: oder COM2: zuzugreifen.

---

[1] Zur Übung: Vergleichen Sie das disassemblierte Programm im Speicher mit dem Assemblerprogramm aus der Praktischen Übung P4.3-3 (Menü: Speicher, Eintrag: Disassembliert)

[2] Hinweis: Die erzeugte Datei (mit der Endung .V24) können Sie sich mit jedem Editor anschauen. In ihr sind die eingeschriebenen ASCII-Zeichen zeilenweise im folgenden Format abgelegt: Anzahl der Bits pro Zeichen, Zeichen im ASCII-Format (hexadezimal), Paritätsbit verwendet: 1=ja/0=nein, das Paritätsbit, Anzahl der Stopbits*2.

Praktische Übung P4.3-4:

Schreiben Sie ein Programm, daß einen Datenbereich aus dem Arbeitsspeicher des Praktikumsrechners liest und über die V.24-Schnittstelle überträgt.

Im einzelnen soll das Programm:

1. die Anzeige löschen,
2. die Kennung "AA" (Anfangs-Adresse) im Operationsfeld der Anzeige ausgeben und im Adreßfeld die 4-stellige Startadresse des gewünschten Datenbereiches einlesen, (Beenden der Eingabe durch die Taste '+')
3. die Anzeige löschen und - mit der Startadresse beginnend -
4. ein Zeichen des Datenbereichs lesen und im Datenfeld der Anzeige ausgeben, dabei die Speicheradresse des Zeichens im Adreßfeld anzeigen,
5. das gelesene Zeichen zur V.24-Schnittstelle ausgeben und dabei (zur Sende-Synchronisation) das Statusregister der ACIA auswerten,
6. sich selbst mit einem Sprung ins Monitorprogramm abbrechen, wenn das Zeichen '.' (Punkt) ausgegeben wurde, sonst
7. mit der um 1 erhöhten Speicheradresse mit 4. fortfahren.

Zur Arbeit mit dem 6809-Emulator:

Nachdem Sie auch dieses Programm mit dem Editor erstellt und assembliert haben, können Sie es ebenfalls mit dem 6809-Emulator austesten. Starten Sie dazu den Emulator und laden Sie das Maschinenprogramm (Endung .HEX) in den Arbeitsspeicher des emulierten 6809-Prozessors (Menüpunkt: Datei, Eintrag: laden)[3]. Nun können Sie das ACIA-Fenster (Menüpunkt: Bausteine, Eintrag: ACIA) öffnen.

Dieses Fenster ist im folgenden Bild dargestellt. In der Mitte des Fensters sehen Sie die Register der ACIA mit ihren aktuellen Belegungen. Die Bedeutungen der einzelnen Bits von Steuer- und Befehlsregister - die im Fenster abweichend zum Text mit CON (Control) und COM (Command) bezeichnet sind - ist quasi "im Klartext" rechts aufgeschrieben. Sie können nun eine Folge von zweistelligen Hexadezimalzahlen eingeben und durch die Eingabetaste abschließen. Daraufhin werden diese Zeichen sukzessiv vom Feld 'zu empfangen' ins Feld 'empfangen' übertragen und gleichzeitig das Zeitdiagramm des jeweils übertragenen Zeichens im oberen Teil des Bildes dargestellt. Durch die Tastatur wird auf diese Weise die Empfangsleitung RxD der ACIA "simuliert". Nach der Übertragung können Sie sich mit Hilfe des waagerechten Verschiebebalkens noch einmal die gesamte zeitliche Darstellung der V.24-Übertragung anschauen. Die Aussendung von Zeichen geschieht durch das Einschreiben ihres V.24-Codes in das Sende-Datenregister (Adresse: $F008) der ACIA, z.B. über die Tastatur des emulierten Praktikumsrechners. Sie können sich auch dazu das Zeitdiagramm anschauen, wenn Sie das ACIA-Fenster durch Anklicken der Raute in der oberen, rechten Ecke verkleinern.

[3] Zur Übung: Vergleichen Sie wieder das disassemblierte Programm im Speicher mit dem Assembler programm aus der Praktischen Übung P4.3-4 (Menü: Speicher, Eintrag: Disassembliert)

Durch Anklicken der Ikone

können Sie eine Datei bestimmen, in die alle ins Sende-Datenregister (Adresse $F008) geschriebenen Zeichen eingetragen werden. Legen Sie eine ASCII-Zeichenkette, die mit einem Punkt ('.'=$2E) endet, in einem Speicherbereich des emulierten Rechners ab. Starten Sie das Programm und geben Sie die Startadresse des Speicherbereiches ein. Nachdem das Programm mit einem Rücksprung ins Monitorprogramm endet, müssen Sie durch die Ikone

die Aufzeichnung stoppen. Danach können Sie sich durch die Ikone

das Zeitdiagramm der in der eben erzeugten Datei aufgezeichneten Zeichen anschauen. (Dabei müssen Sie den waagerechten Verschiebebalken benutzen. Wenn Ihr Programm fehlerfrei arbeitet, können Sie als Beschriftung des Zeitdiagramms die im Speicherbereich abgelegte Zeichenkette wiederfinden.

Wenn Ihr Programm fehlerfrei läuft, können Sie zur Kommunikation mit dem Praktikumsrechner einsetzen. Dazu müssen Sie das Empfangsprogramm aus P4.3-3, wie in Kapitel 2 beschrieben, (mit dem mitgelieferten Programm IDE9.EXE) in den Praktikumsrechner übertragen. Nun bilden Sie - wie oben beschrieben - die emulierte V.24-Schnittstelle auf die PC-Schnittstelle ab (Menüpunkt: Datei, Eintrag: Schnittstellen). Starten Sie danach das Empfangsprogramm im Praktikumsrechner und das Sendeprogramm im Emulator. Überprüfen Sie wiederum die Übertragung durch Vergleich der in den Speicherbereichen des Emulators und des Praktikumsrechners vorliegenden ASCII-Zeichenketten.

Lösungsvorschläge zu den Praktischen Übungen

Zu P4.2-1:

I. „Programmieren“ des Ports PA:

Selektion des Datenrichtungsregisters DDRA (Adresse $F000) durch Bit2=0 im Steuerregister CRA (Adresse: $F001):

Taste A:	$F001	($XX)	
Taste D:		$00	
Taste -:	$F000	$AA	PB7...PB0 = OIOI OIOI (I=Input/O=Output)
Taste +:	$F001	$04	Selektion des Datenregisters DRA
Taste -:	$F000	xyxyxyxy	x: ausgegebenes, y: eingelesenes Bit
		$FF	Eingabe von $FF
Tasten +,-:		$FF	Ausgabebits O auf 1, offene Eingänge I auf 1
		$00	Eingabe von $00
Tasten +,-:		$55	Ausgabebits O auf 0, offene Eingänge I auf 1

II. Programmieren des Ports PB:

Taste A:	$F003	($XX)	
Taste D:		$00	
Taste -:	$F002	$0F	PB7...PB0 = I I I I OOOO (I=Input/O=Output)
Taste +:	$F003	$04	Selektion des Datenregisters DRB
Taste -:	$F002	$0X	X: ausgegebene Tetrade
Taste +:	$F003	$04	(dient nur zur Neuanwahl von DRB)
Taste -:	$F002	$YX	Y: eingelesene, X: ausgegebene Tetrade

(Die letzten beiden Schritte können nun beliebig wiederholt werden.)

III. Verbindung der Portleitungen von PB:

Taste A:	$F003	($XX)	
Taste D:		$00	
Taste -:	$F002	$0F	PB7...PB0 = I I I I OOOO (I=Input/O=Output)
Taste +:	$F003	$04	Selektion des Datenregisters DRB
Taste -:	$F002	$0X	X: ausgegebene Tetrade
Taste +:	$F003	$04	(dient nur zur Neuanwahl von DRB)
Taste -:	$F002	$YX	X=Y: eingelesene = ausgegebene Tetrade

(Die letzten beiden Schritte können nun beliebig wiederholt werden.)

Zu P4.2-2:

I. Belegung des Steuerregisters CRB (Adresse $F003):

Bit	7	6	5	4	3	2	1	0
	-	-	1	0	0	1	1	0
	CB1-Flag	CB2-Flag	CB2-Steuerung			DRB DDRB	CB1-Steuerung	

Also wird das Register CRB mit dem Wert $26 geladen.

Taste A: $F003 ($xx)
Taste D: $26 Betriebsart gewählt und DRB selektiert (Bit 2=1)

II. Erzeugung des „langen“ Strobe-Impulses:

durch Schreiben eines Wertes ins Datenregister DRB

		$F003	$26	CB1-Flag gelöscht, kein Interrupt
*	Taste -:	$F002	$XX	LED an CB2 erlischt
	Taste +:	$F003	$26	CB1-Flag gelöscht, kein Interrupt
	Impuls über Taste an CB1:			LED an CB2 leuchtet
	Taste A:	$F003	$A6	CB1-Flag gesetzt, Interruptanforderung
	Taste -:	$F002	($XX)	löscht CB1-Flag
	Taste +:	$F003	$26	CB1-Flag tatsächlich gelöscht

weiter mit Aktion *.

Hinweise:

1. Wird nach der Interrupterzeugung über CB1 das CB1-Flag nicht gelöscht, wird durch ein erneutes Einschreiben eines Wertes ins Register DRB kein weiteres Strobe-Signal erzeugt. Dies verhindert ein „Überlaufen“ (*overrun*) des Empfängers.
2. Das CB1-Flag wird jedoch auch gelöscht, wenn Sie nach dem Impuls an CB1 zweimal hintereinander einen Wert in das Datenregister DRB schreiben. Das liegt daran, daß das Monitorprogramm nach dem ersten Einschreiben das Register DRB liest, um die Anzeige zu aktualisieren, und dadurch das Flag löscht.

III. Erzeugung des „kurzen“ Strobe-Impulses:

Der Wert im Steuerregister CRB muß geändert werden zu: $2E.
Nach Einschreiben eines Wertes ins Datenregister leuchtet die LED an CB2 kurz auf.[1] Dieser Impuls dauert beim Praktikumsrechner nur 1 µs und ist deshalb nicht mit dem bloßen Auge zu erkennen.[2]

---

[1] Hier liegt ein Programmierfehler im Simulator vor. Der reale Baustein legt den Ausgang kurzeitig auf L-Potential.

[2] Das er im 6809-Simulator zu beobachten ist, liegt darin, daß dieser langsamer arbeitet als der Praktikumsrechner und daher alle Operationen verlängert werden.

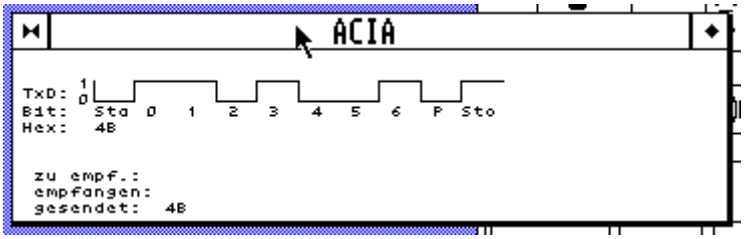
Zu P4.2-3:

Programm zur Erzeugung einer Rechteckschwingung an CB2:

```
 1        F003      CRB       EQU    $F003
 2        F160      DLY1MS    EQU    $F160
 3
 4  0400                              ORG    $0400
 5  0400  86 34       LOOP            LDA    #$34
 6  0402  B7 F0 03                    STA    CRB     ; LED an CB2 aus
 7  0405  BD F1 60                    JSR    DLY1MS  ; Verzögerung
 8  0408  86 3C                       LDA    #$3C
 9  040A  B7 F0 03                    STA    CRB     ; LED an CB2 an
10  040D  1E 23                       EXG    Y,U
11  040F  BD F1 60                    JSR    DLY1MS  ; Verzögerung
12  0412  1E 23                       EXG    Y,U
13  0414  20 EA                       BRA    LOOP
```

Zu P4.3-1:

Die Ausgabe des ASCII-Zeichens 'K' geschieht durch Einschreiben des Wertes $4B in das Sende-Datenregister der ACIA unter der Adresse $F008. Dies führt zur folgenden Darstellung im Emulatorfenster:



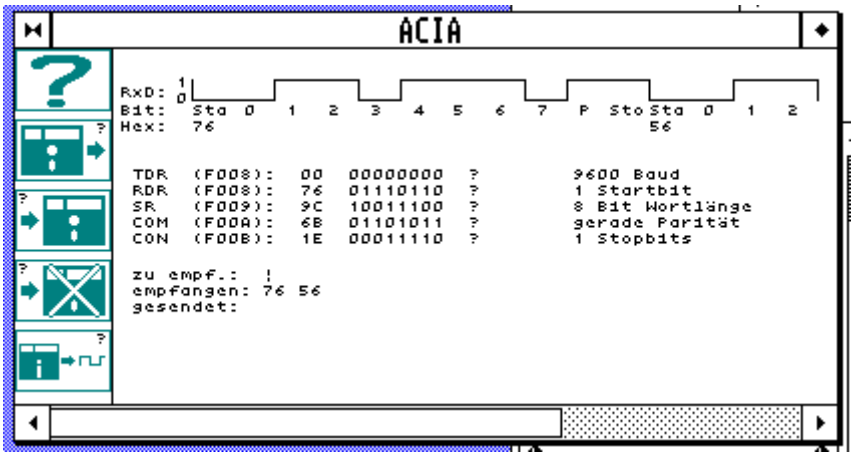
Ein (bitweiser) Vergleich mit dem Zeitdiagramm der Musterlösung zu Selbsttestaufgabe S4.3-1 zeigt, daß beide Ausgaben übereinstimmen. (Dabei muß man nur berücksichtigen, daß Start-, Daten-, Paritäts- und Stopbits in negativer Logik übertragen werden.)

Zu P4.3-2:

Die Ausgabe der ASCII-Zeichen 'vV' geschieht durch Einschreiben der Werte $76 (v), $56 (V) in das Sende-Datenregister der ACIA unter der Adresse $F008. Dies führt zu der im folgenden Bild dargestellten Emulatorfenster.

Durch Betägigen des waagerechten Verschiebebalkens wird auch das 2. Zeichen vollständig dargestellt. Ein (bitweiser) Vergleich mit dem Zeitdiagramm der Musterlösung zu Selbsttestaufgabe S4.3-2 zeigt, daß beide Ausgaben übereinstimmen.

(Dabei muß man nur berücksichtigen, daß Start-, Daten-, Paritäts- und Stopbits in negativer Logik übertragen werden.)



Zu P4.3-3:

```
1
2                       ; Register der ACIA R6551
3       F008        DR  EQU  $F008 ; Datenregister
4       F009        SR  EQU  $F009 ; Statusregister
5       F00A        IR  EQU  $F00A ; Befehlsregister
6       F00B        CR  EQU  $F00B ; Steuerregister
7
8                       ; Monitor-Hilfsroutinen
9       F110        CLRDISP   EQU  $F110 ; Löschen der Anzeige
10      F120        SHOWB7SG  EQU  $F120 ; Darstellen eines Bytes
11      F123        SHOWD7SG  EQU  $F123 ; Darstellen zweier Bytes
12      F140        KEY       EQU  $F140 ; Tastaturabfrage ohne Halt
13      F156        SHOWADR   EQU  $F156 ; Adresse einlesen und anzeigen
14
15  0400                    ORG  $0400
16  0400  BD F1 10  KOMM    JSR  CLRDISP; Anzeige löschen
17  0403  F7 F0 09          STB  SR      ; Programmed Reset
18  0406  86 0B             LDA  #$0B    ; Programmieren des
19  0408  B7 F0 0A          STA  IR      ;  Befehlsregisters
20  040B  C6 98             LDB  #$98    ; Programmieren des
21  040D  F7 F0 0B          STB  CR      ;  Steuerregister
22  0410  8E 00 06          LDX  #$0006 ; Anzeige S6 selektieren
23  0413  C6 AA             LDB  #$AA    ; Kennung fuer Anfangsadresse
```

```
24  0415  BD F1 20            JSR   SHOWB7SG;  ausgeben
25  0418  8E 00 02            LDX   #$0002 ; Anzeige S2 selektieren
26  041B  BD F1 56            JSR   SHOWADR; Anfangsadresse einlesen
27  041E  BD F1 10            JSR   CLRDISP ; Anzeige löschen
28
29  0421  B6 F0 09      LOOP  LDA   SR      ; Abfrage des Statusregisters
30  0424  84 08               ANDA  #$08    ; RDRF-Bit maskieren
31  0426  27 F9               BEQ   LOOP    ; falls kein Zeichen empfangen
32
33  0428  F6 F0 08            LDB   DR      ; Empfangs-Datenregister lesen
34  042B  8E 00 00            LDX   #$0000 ; Anzeige S0 selektieren
35  042E  BD F1 20            JSR   SHOWB7SG; Zeichen darstellen
36  0431  E7 A0               STB   ,Y+     ; Zeichen im Speicher ablegen
37  0433  C1 2E               CMPB  #$2E    ; Abfrage auf "."
38  0435  26 01               BNE   WEITER
39  0437  3F                  SWI           ; Sprung in den Monitor
40  0438  1F 20       WEITER  TFR   Y,D     ; Adresse nach Akku D
41  043A  30 02               LEAX  2,X     ; Anzeige S2 selektieren
42  043C  BD F1 23            JSR   SHOWD7SG; neue Adresse darstellen
43  043F  20 E0               BRA   LOOP    ; Schleife
44
45  0441              END
```

Zu P4.3-4:

```
 1
 2                  ; Register der ACIA R6551
 3      F008        DR    EQU   $F008 ; Datenregister
 4      F009        SR    EQU   $F009 ; Statusregister
 5      F00A        IR    EQU   $F00A ; Befehlsregister
 6      F00B        CR    EQU   $F00B ; Steuerregister
 7
 8                  ; Monitor-Hilfsroutinen
 9      F110     CLRDISP     EQU   $F110 ; Löschen der Anzeige
10      F120   SHOWB7SG      EQU   $F120 ; Darstellen eines Bytes
11      F123   SHOWD7SG      EQU   $F123 ; Darstellen zweier Bytes
12      F140       KEY       EQU   $F140 ; Tastaturabfrage ohne Halt
13      F156    SHOWADR      EQU   $F156 ; Adresse einlesen und anzeigen
14
15  0400                ORG   $0400
16  0400  BD F1 10  KOMM  JSR   CLRDISP ; Anzeige löschen
17  0403  F7 F0 09        STB   SR      ; Programmed Reset
18  0406  86 0B           LDA   #$0B    ; Programmieren des
```

```
19  0408  B7 F0 0A          STA  IR       ;  Befehlsregisters
20  040B  C6 98             LDB  #$98     ; Programmieren des
21  040D  F7 F0 0B          STB  CR       ;  Steuerregister
22  0410  8E 00 06          LDX  #$0006 ; Anzeige S6 selektieren
23  0413  C6 AA             LDB  #$AA   ; Kennung fuer Anfangsadresse
24  0415  BD F1 20          JSR  SHOWB7SG;  ausgeben
25  0418  8E 00 02          LDX  #$0002         ; Anzeige S2 selektieren
26  041B  BD F1 56          JSR  SHOWADR    ; Anfangsadresse einlesen
27  041E  BD F1 10          JSR  CLRDISP ; Anzeige löschen
28
29  0421  8E 00 02  LOOP    LDX  #$0002 ; Anzeige S2 selektieren
30  0424  1F 20             TFR  Y,D      ; aktuelle Adresse
31  0426  BD F1 23          JSR  SHOWD7SG;  anzeigen
32  0429  E6 A0             LDB  ,Y+      ; Zeichen aus Datenbereich lesen
33  042B  30 1E             LEAX -2,X     ; Anzeige S0 selektieren
34  042D  BD F1 20          JSR  SHOWB7SG; Zeichen darstellen
35
36  0430  B6 F0 09  NFREI   LDA  SR       ; Abfrage des Statusregisters
37  0433  84 10             ANDA #$10     ; TDRE-Bit maskieren
38  0435  27 F9             BEQ  NFREI  ; Sprung, falls Sender nicht frei
39
40  0437  F7 F0 08          STB  DR       ; Zeichen ausgeben
41
42  043A  C1 2E             CMPB #$2E     ; Abfrage auf "."
43  043C  26 E3             BNE  LOOP   ; Schleife
44  043E  3F                SWI           ; Sprung in den Monitor
45
46  043F                    END
```

Lösungsvorschläge zu den Selbsttestaufgaben

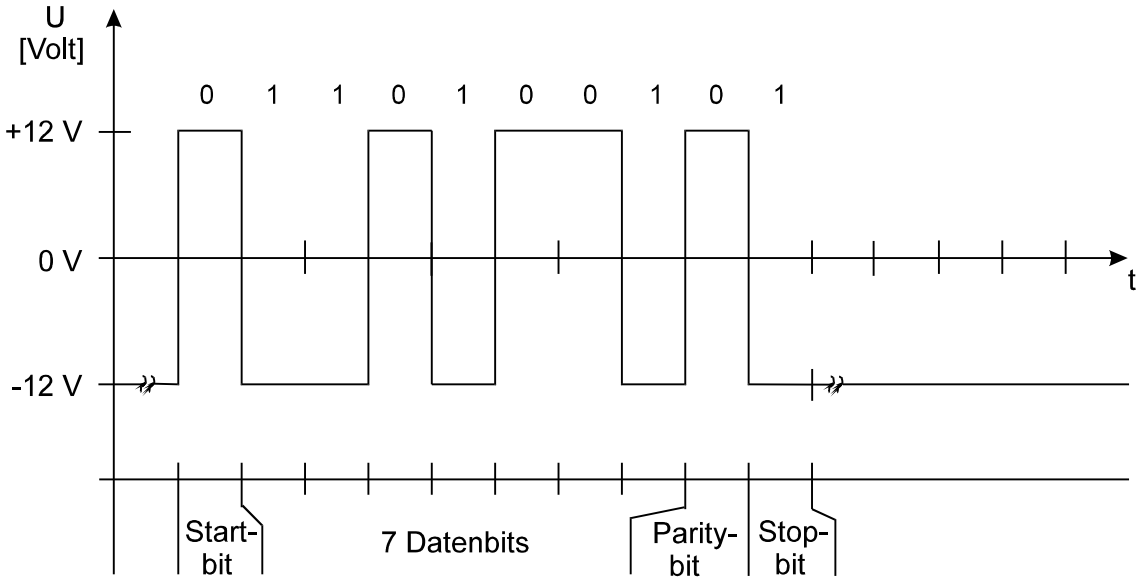
Zu S4.3-1:

a) CR = 0 0 1 1 1 0 1 0 = $3A

IR = 0 1 1 0 0 1 0 1 = $65

b) $4B = 0 1 0 0 1 0 1 1

Bitdauer b = 416,6 µs



c) SR = 1 1 0 0 1 0 1 1 = $CB

Interpretation und Begründung der Statusbits:

PE = 1 gerade Parität verletzt, da 9. Bit (Statusbit, 7 Datenbits) = L-Pegel (‘1’) statt H-Pegel (‘0’).

FE = 1 Framing error, da Stopbit (10. Bit) gleich = H-Pegel (‘0’) statt L-Pegel (‘1’).

OE = Overrun, aus Aufgabenstellung nicht zu entnehmen.
OE = 0

RDRF = 1 Zeichen empfangen, aber noch nicht von der CPU gelesen

TDRE = 1 vorgegeben

DCD = 0 vorgegeben

DSR = DTR (Nullmodem) $\Rightarrow$ DSR = DTR = 0

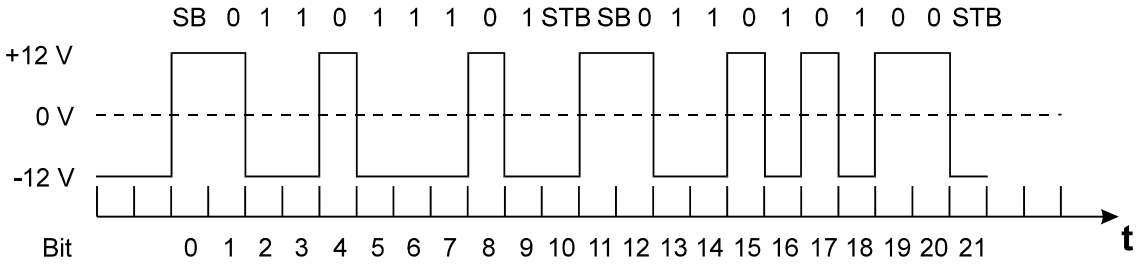
IF = 1 da Interruptanforderung des Empfängers vorliegt

Zu S4.3-2:

a) Übertragungsrate 1/0,1ms = 10kHz~ 9600 bd.
(Übertragungsrate ~ 10 kbit/s)

b) Anzahl der Stopbits pro Zeichen: 1

Begründung: Die Zeichen werden im 8-bit-Code übertragen (vgl. d)



c) Paritätsbit:

zwischen SB und StB liegen 9 Bits, hier 8 Datenbits und 1 Paritätsbit *Even Parity*, da die Anzahl der 1-Bits auf eine gerade Anzahl ergänzt werden.

d) 8-bit-Code:

Gemäß Aufgabenstellung werden die Zeichen im erweiterten ASCII-Code übertragen, d.h. 7- oder 8-bit-Code. Die Überprüfung unter Berücksichtigung möglicher Paritätsbit-Kombinationen ergibt, daß die Zeichen im 8-Bit-Code übertragen werden, damit ergibt sich die Lage der Startbits (Bit 0 und Bit 11), 2 Stopbits nicht möglich, da Bit 20 und Bit 21 alternierend sind, d.h. 1 Stopbit; damit sind Bit 10 und Bit 21 Stopbits.

Binäre Form der Zeichen: 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 0
MSB LSB, MSB LSB

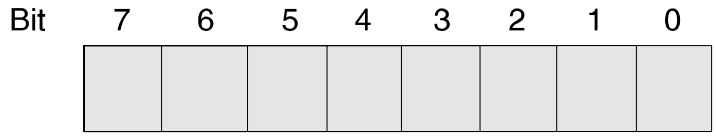
Hexadezimale Form: $76 $56

e) Buchstabenkombination: vV

Anhang A: Das Programmiermodell des Bausteins MC6821

Daten-/Datenrichtungsregister

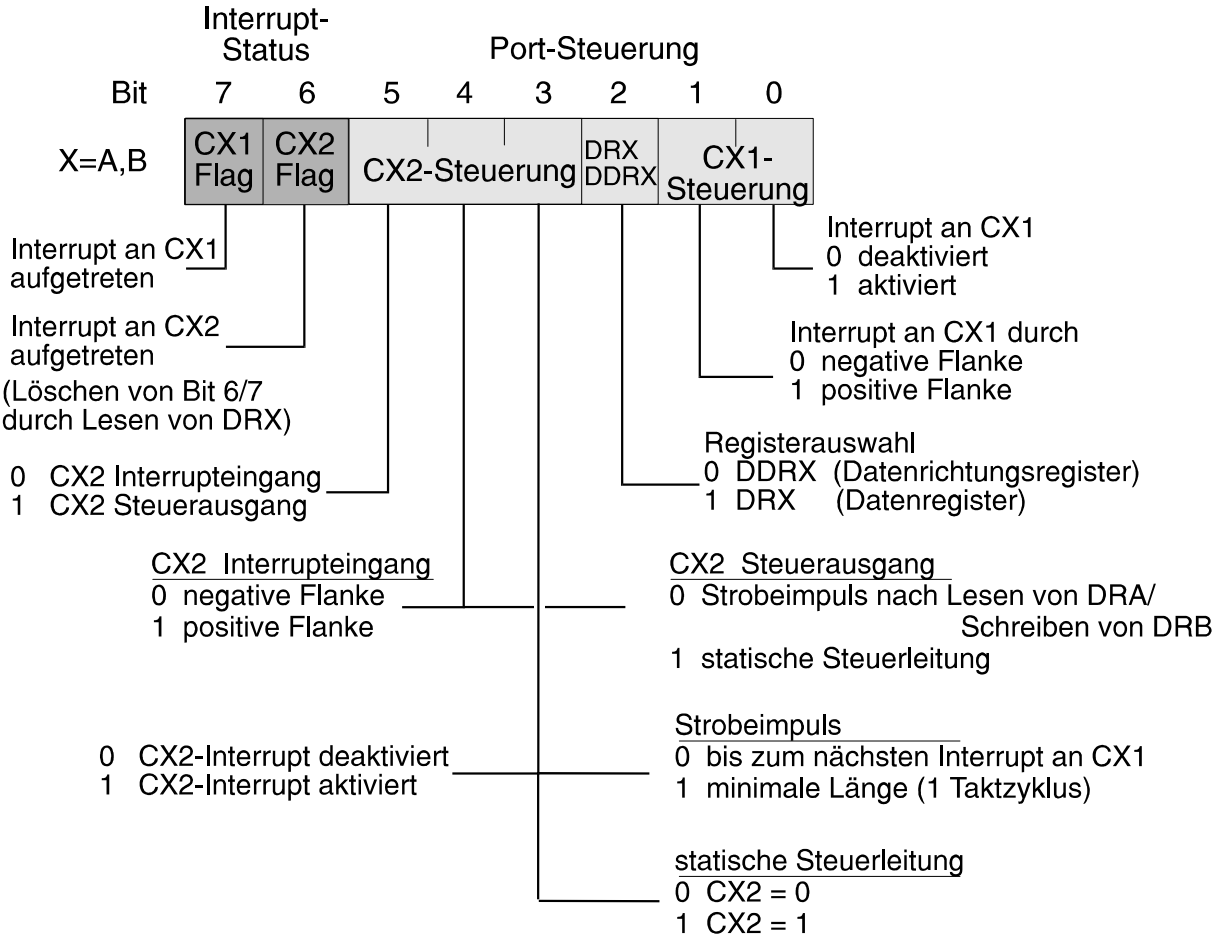
Adressen: DRA, DDRA $F000
DRB, DDRB $F002



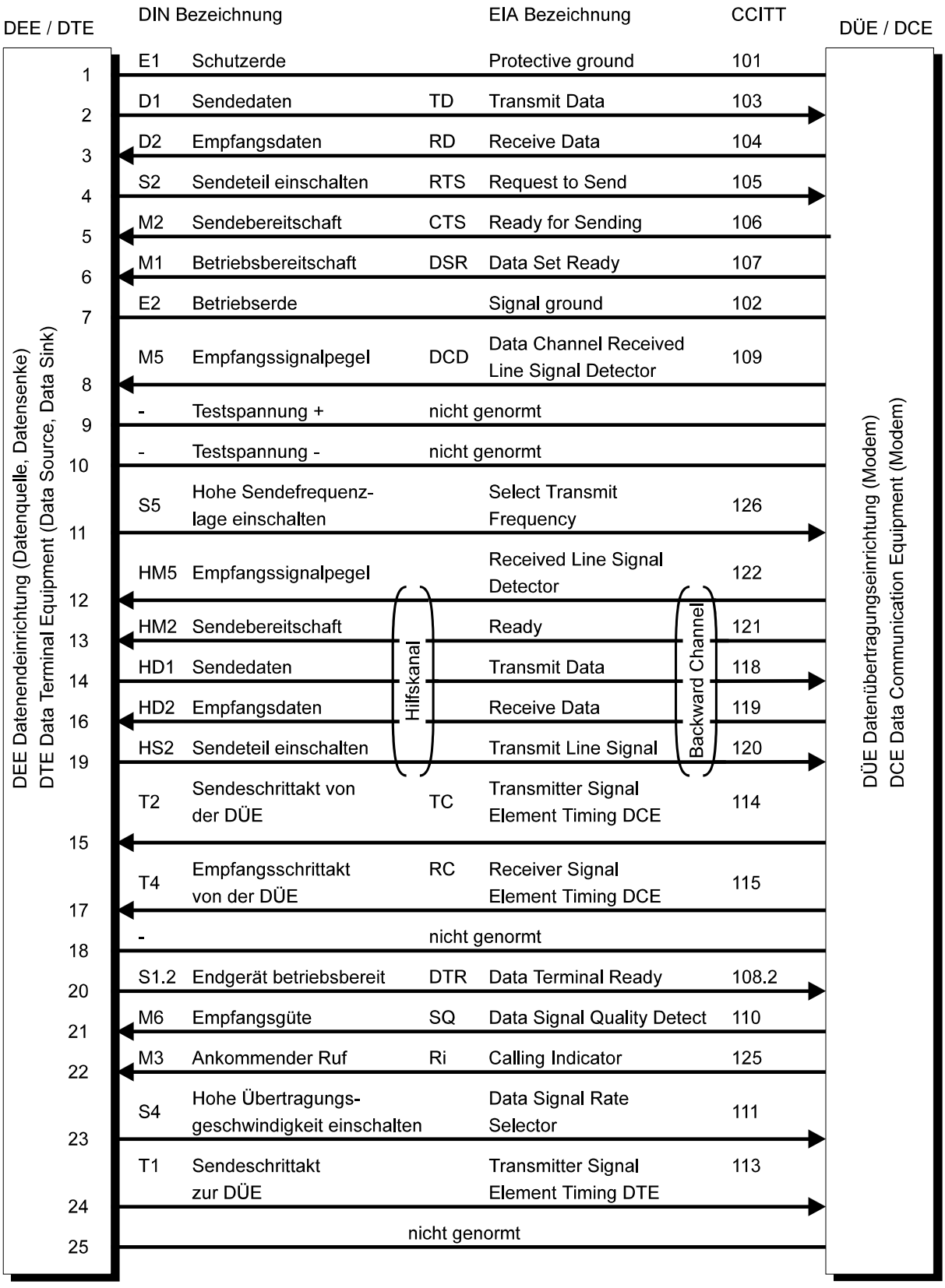
DDRXi = 0 Portleitung PXi ist Eingang
1 Portleitung PXi ist Ausgang
X=A,B

Steuerregister

Adressen: CRA $F001
CRB $F003



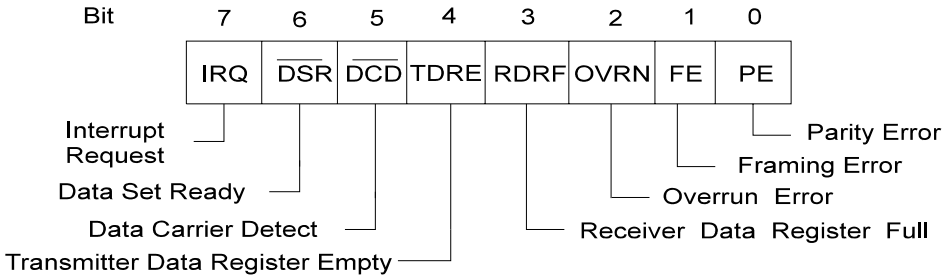
Anhang B: Die vollständige V.24-Schnittstelle



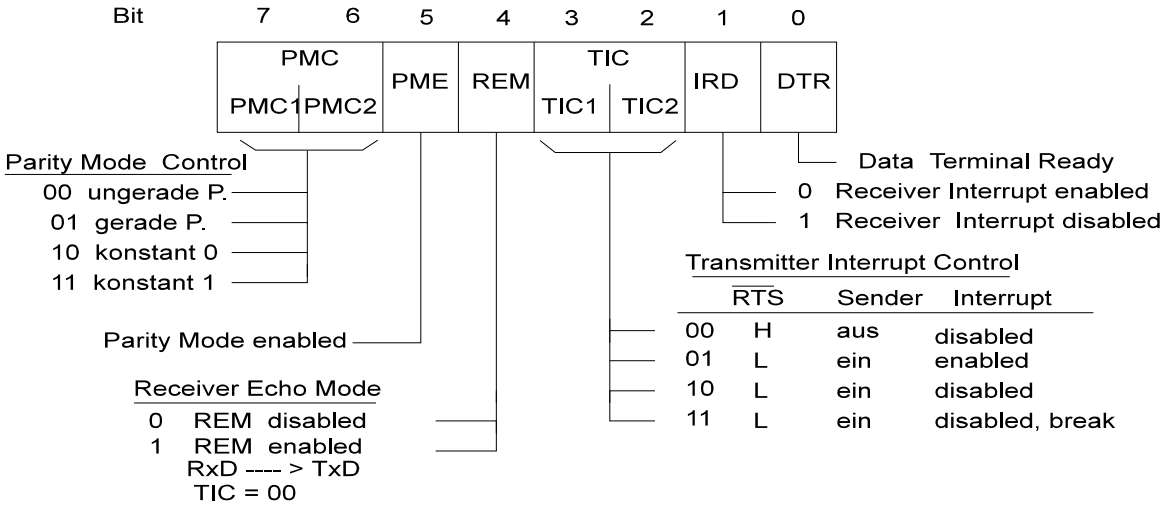
Schnittstellendefinition nach DIN 66020 (V.24 / RS 232C)

Anhang C: Das Programmiermodell des Bausteins R6551

Das Statusregister - SR — Adresse: $F009



Das Befehlsregister - IR — Adresse: $F00A



Das Steuerregister - CR — Adresse: $F00B