

Praktikum Automatisie- rungstechnik	Serielle Schnittstelle 1 Elektrische Signale und Programmierung der Register	Prof. R. Göhl Prof. Dr. J. Höcht
--	---	-------------------------------------

Lernziele des Versuchs:

Im vorliegenden Versuch sollen Sie das Zusammenspiel der Register der seriellen Schnittstelle untersuchen. Dabei wird in diesem ersten Versuch noch nicht die Steuerung der V24-Übertragung durch Interrupts behandelt. Dies ist erst Thema des Versuchs „Serielle Schnittstelle 2“

Nach Durchführung dieses Versuchs sollen Sie folgende Fertigkeiten aufweisen:

- Die Bedeutung der einzelnen Register für die Datenübertragung und deren Steuerung erläutern,
- den Zusammenhang zwischen den elektrischen Signalen auf den V24-Leitungen und den Zuständen einzelner Bits in den Registern erläutern,
- einfache, kleine lineare Assemblerprogramme zur Bedienung der Schnittstelle schreiben und zum Laufen bringen und
- einzelne Daten senden und empfangen können.

Die Abwicklung einer vollständigen handshake-gesteuerten Datenübertragung mit und ohne Interrupt wird Thema des Versuchs „Serielle Schnittstelle 2“ sein.

Inhalt:

1. Programmieren in Assembler unter der Umgebung von Borland-Pascal
2. Bedeutung der Register und Steuerung der Hardware

1. Programmieren in Assembler unter der Umgebung von Borland-Pascal

Sie werden wie Beim Versuch „Mikroprozessor 80x86, Architektur und Programmierung“ mit Hilfe des in der Entwicklungsumgebung integrierten Debuggers und Assemblers das Experimentierprogramm schrittweise abarbeiten und dabei die Inhalte der Prozessorregister sowie insbesondere der Register des V24-Bausteins beobachten. Diese Registerinhalte werden deshalb nicht nur durch den Einlesebefehl im AL-Register als Hexzahl sichtbar, sondern mit einem eigenen speziell dazu vorbereiteten Unterprogramm als Bitmuster am Bildschirm angezeigt.

Der Umgang mit der Entwicklungsumgebung ist Ihnen aus dem Versuch „Mikroprozessor 80x86“ bekannt und wird dort ausführlich beschrieben.

Das folgende Programm enthält die ausführliche Erläuterungen zu der Bedeutung der einzelnen Register. Um Ihren Lernerfolg zu optimieren sind die Assemblerbefehle zunächst ausgespart und durch das Sonderzeichen ♣ ersetzt. Ergänzen Sie die Zeilen durch die nötigen Befehle.

Im Pascalprogramm, mit dem Sie dann arbeiten, sind diese Befehle dann aber bereits eingefügt, um nicht durch Programmfehler den Erfolg des Versuchs zu gefährden.

2. Bedeutung der Register und Steuerung der Hardware

Im Programm dieses Versuchs arbeiten Sie mit den folgenden Registern.

Modem Control Register	MCR
Modem Status Register	MSR
Line Status Register	LSR
Receiver Data Register	RD
Transmitter Data Register	TD

Bits, die nicht im Versuch verwendet werden, sind schattiert dargestellt

Line Status Register LSR

Adresse \$3FD

0	Data ready (Empfangsreg)
1	Overrun Error
2	Parity Error
3	Framing Error
4	Break Interrupt Indicator
5	Transmitter Holding Reg. empty
6	Transmitter Shift Reg. Empty
7	nicht verwendet

Modem Control Register MCR

Adresse \$3FC

0	Ausgabe DTR	DTR Pin 20
1	Ausgabe RTS	
2		RTS Pin 4
3		
4		
5		
6		
7		

Modem Status Register MSR

Adresse \$3FE

0	Δ CTS=1 CTS hat sich geändert	CTS Pin 5
1	Δ DSR=1 DSR hat sich geändert	
2		
3		
4	CTS: Zustand von CTS	
5	DSR: Zustand von DSR	
6		DSR Pin 6
7		

Receiver Data RD NUR LESBAR!
(Empfangsschieberegister)
Adresse \$3F8

Transmitter Data TD NUR BESCHREIBBAR!
(Sendeschieberegister)
Adresse \$3F8

RD

TD

Program V24_Regs;

```

Höcht, 29.4.2002
Experimente mit den Registern der seriellen Schnittstelle, die
für die Kommunikationsleitungen RTS/CTD DTR/DSR verantwortlich
sind.

KEINE Benutzung der BIOS-Routinen, sondern
direktes Senden mit den Assemblerbefehlen IN und OUT

uses CRT; { Bibliotheksprogramme zur Ein- und Ausgabe von Zeichen über
           Tastatur und Bildschirm

CONST  p_MSR = $3FE;    { Adresse des Modem-Status-Registers      COM1
      p_MCR = $3FC;    { Adresse des Modem-Control-Registers    COM1
      p_LSR = $3FD;    { Adresse des Line-Status-Registers      COM1
      p_TD  = $3F8;    { Adresse des Sendeschieberegisters      COM1
      p_RD  = $3F8;    { Adresse des Sendeschieberegisters      COM1

VAR     MSR      : Byte; { Inhalt des Modem-Status-Registers
      MCR      : Byte; { Inhalt des Modem-Control-Registers
      LSR      : Byte; { Inhalt des Line Status Registers
      TD       : Byte; { Inhalt des Transmitter Schieberegisterpuffers
      RD       : Byte; { Inhalt des Receiver Schieberegisterpuffers

      I        : Byte; { Laufvariable als Zeichenindex in s
      zykl     : Integer; { Anzahl von Wiederholzyklen

```

```
Procedure V24 Initialisieren(Baud: Byte);
```

```
{+
Höcht, 28.6.1999
Initialisieren der seriellen Schnittstelle mit Hilfe des Interrupts $14

Höcht, 29.4.2002
Eine direkte Initialisierung mit den Befehlen OUT und IN sowie den geeigneten Bitkombinationen wäre zwar möglich, würde aber detaillierte Kenntnisse über die Programmierung des seriellen Bausteins erfordern. Daher wird hier der bequemere Weg gegangen und die BIOS-Routine aufgerufen, die u.a. diese Aufgabe bewältigt.
Dagegen sind die anderen Funktionen, wie Zeichen empfangen, senden, Register abfragen oder Register setzen, um die Spannung von Leitungen zu ändern, sehr einfach zu bewältigen und wird hier geübt.

Der Aufruf der BIOS-Routine für die verschiedenen Aufgaben erfolgt nach folgendem Muster:

- Belegung des AH-Registers (höherwertiges Byte von AX) mit einer Zahl, die nach dem Aufruf der gesamten V24-BIOS-Funktion mit dem Befehl INT($14) die Verzweigung in die richtige Unterfunktion steuert. Näheres dazu siehe unten

- Belegung des AL-Registers mit einer geeigneten Variablen (s.u.), die für
```

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

die mit AH angewählte Funktion nötig ist.

Falls die Angewählte Funktion ein Ergebnis liefert, dann steht auch dieses Ergebnis in dem Register AL. Reicht dieses Register nicht, dann wird zusätzlich noch AH für die Ergebnisübergabe verwendet.

- In das DX-Register wird die Nummer der betreffenden Schnittstelle geschrieben. DOS unterstützt 4 serielle Schnittstellen COM1 bis COM4. Die Numerierung mit Hilfe von DX beginnt aber bei 0 und endet bei 3, also Vorsicht an dieser Stelle!
- Aufruf der BIOS-Funktion durch den Assemblerbefehl INT(\$14)

```
VAR AL_, AH_ : Byte; { lokaler Zwischenspeicher für den Inhalt von AH und AL }
    DX_      : Word; { dgl für das Register DX }
```

Begin

{ Belegung des AL-Registers mit der Variablen zur Initialisierung der V24-Schnittstelle:

Bit Nr.:

7	6	5	4	3	2	1	0
Baudrate (= Bit/s)			-- Parität --	Stopbit	- Datenbits -		

0 0 0	-->	110 Bd	x 0	-->	keine	0 -->	1	0 0	-->	5 Bit
0 0 1	-->	150 Bd	0 1	-->	ungerade	1 -->	2	0 1	-->	6 Bit
0 1 0	-->	300 Bd	1 1	-->	gerade			1 0	-->	7 Bit
0 1 1	-->	600 Bd						1 1	-->	8 Bit
1 0 0	-->	1200 Bd								
1 0 1	-->	2400 Bd								
1 1 0	-->	4800 Bd								
1 1 1	-->	9600 Bd								

```
AL_ := $07; { $03 = 0000 0111 --> 110 Bd, keine Parität,
              2 Stopbit, 8 Datenbits
              Diese geringe Übertragungsgeschwindigkeit gibt
              Gelegenheit, die Übertragung eines einzelnen
              Zeichens als Flackern der Diode bei der Sende-
              leitung Pin Nr 2 (25-poliger Stecker) zu sehen }
```

if Baud = 96 then

```
    AL_ := $E3; { Zur Erzeugung eines Framing-Errors durch ein Datenbyte der
                  Gegenstation wird die eigene erwartete Baudrate spä-
                  ter auf 9600 Baud mit 1 Stopbit geändert. }
```

{Belegung des AH-Registers:

Steuerung des BIOS-Programms:

\$00 --> Initialisierung der Schnittstelle

\$01 --> Zeichen aus AL absenden

\$02 --> Zeichen empfangen, Zeichen steht dann in AL

\$03 --> Statusregister auslesen:

Übertragungszustände bis dieser letzten Funktion in AL:

Bit 7: Zeitüberschreitung

Bit 6: Sende-Schieberegister leer

-> Zeichen abgeschickt

Bit 5: Sende-Pufferregister leer

-> Zeichen wurde ins Schieberegister übertragen

Bit 4: Abbruch entdeckt

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

Bit 3: Rahmenfehler beim Empfang
Bit 2: Paritätsfehler beim Empfang
Bit 1: Zeichen im Empfangsregister wurde überschrieben, bevor abgeholt wurde
Bit 0: Zeichen im Empfangsregister angekommen
Modemzustände in AH:
Bit 7: Empfangssignal entdeckt
Bit 6: Ring Indicator (Pin der Telefonklingel) ist aktiv
Bit 5: DSR-Eingangsleitung (Pin 4) aktiv
Bit 4: CTS-Eingangsleitung (Pin 6) aktiv
Bit 3: Eine Empfangsleitung hat ihren Zustand gewechselt
Bit 2: Ansteigende Flanke des Pins für die Telefonklingel entdeckt
Bit 1: Eingangsleitung DSR hat ihren Zustand geändert
Bit 0: Eingangsleitung CTS hat ihren Zustand geändert
}

```
AH_ := $00;    { Also nun wird die Schnittstelle initialisiert:}
```

```
{ Auswahl der Schnittstelle mit dem DX-Register:
```

```
DX = $00  --> COM1
```

```
DX = $01  --> COM2
```

```
DX = $02  --> COM3
```

```
DX = $03  --> COM4    }
```

```
DX_ := $00;
```

```
ASM          { Schreiben Sie nun an diese Stelle das Assemblerprogramm }  
♣           { für die Initialisierung von COM 1                          }
```

```
♣
```

```
♣
```

```
INT($14)
```

```
END;
```

```
End;
```

```
{  
{  
{ Höcht, 29.4.2002  
{ Einige hilfreiche Prozeduren zur Anzeige von Registerinhalten und  
{ Byte-Variablen als Dualzahl  
{  
{  
{  
}
```

```
{  
{  
{ Höcht, 3.7.2000  
{ Darstellen eines Byte als Dualzahl  
{  
{  
{  
}
```

```

Function Bitmuster(t:byte):string;
{+-----+
| Höcht, 3.7.2000
| Wandelt ein Byte um in einen String aus 0 und 1 in zwei Vierergruppen
|-----+}
Var s : string;
    c : char;
    i : byte;
Begin
    s := '';
    for i := 1 to 8 do
        Begin
            if t Mod 2 = 1 then
                s := '1'+ s
            else
                s := '0' + s;
            t := t div 2;
            if i mod 4 = 0 then
                s := ' ' + s;           { Leerzeichen zwischen Vierergruppen }
        End;
    Bitmuster := s;
End;

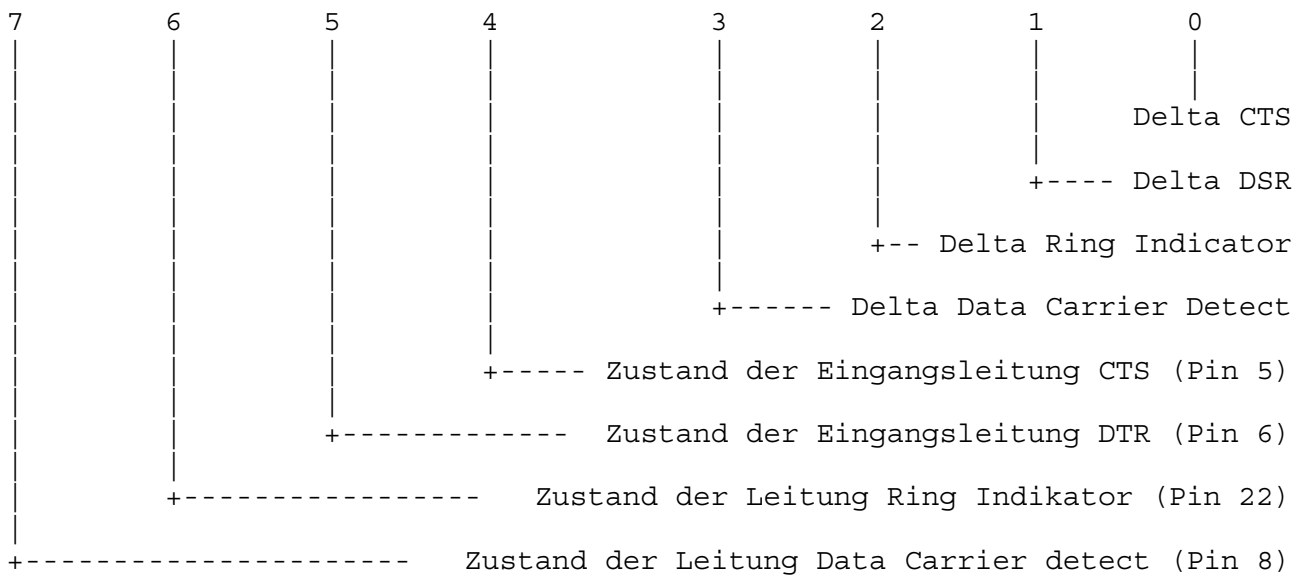
Procedure Bit_aus(t : string; b : Byte;ze,sp: Word);
{+-----+
| Höcht, 3.7.2000
| gibt b als Bitmuster auf dem Schirm an der aktuellen Stelle des Cursors
| aus und geht in die nächste Zeile
|-----+}
VAR x,y : integer;
Begin
    t := t + ' ';
    t := copy(t,1,50);
    x := wherex; y := wherey;
    Gotoxy(sp,ze);
    Writeln(t+ ': ' + bitmuster(b));
    Gotoxy(x,y);
End;

```

Höcht, 29.4.2002
Bedeutung einzelner Registerinhalte im V24-Baustein:

Überwachung der Eingangsleitungen CTS, DSR, RI, DCD, RD

Bedeutung der Bits des Modem Status Registers MSR:



Die ersten 4 Bits (Nr. 0 bis 3) dienen dazu, ÄNDERUNGEN auf Eingangsleitungen der Schnittstelle anzuzeigen.

- Bit 0 Delta CTS (Delta Clear To Send):
Dieses Bit geht auf 1, wenn sich der Zustand der an diesem Pin 5 (CTS) angeschlossenen Leitung des Modems sich geändert hat, ob von 0 auf 1 oder umgekehrt, seit das Modem Status Register MSR das letzte Mal ausgelesen wurde. Das Modem antwortet mit CTS auf eine Sende Anfrage RTS der Daten-Endeinrichtung, also z.B. des PC's
- Bit 1 Delta DSR (Delta Data Set Ready)
Für dieses Bit gelten die gleichen Aussagen, wie für Bit 0, allerdings für Eingangsleitung DSR.
Das Modem (Daten-Übertragungseinrichtung = DÜE bzw. Data Communication Equipment DCE) reagiert mit dieser Leitung auf ein aktiviertes Signal DTR (Data Terminal Ready) des PC's, also der Daten-Endeinrichtung (DEE, bzw. Data Terminal Equipment DTR)
- Bit 2 Delta Ring Indicator
Das Modem muß dem Rechner weitermelden, wenn ein Anruf ein-

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

geht, also wenn das Telefon klingelt! Das Modem legt in diesem Fall die sonst auf 1 liegende Leitung Ring-Indicator RI (Pin 22) auf 0. Diese fallende Flanke verursacht eine 1 im Bit 2 des Modem Status Registers, das beim Auslesen des Registers durch den Prozessor wieder auf 0 gesetzt wird.

Bit 3 Delta Data Carrier Detect

Hat das Modem nach dem Abheben auf der Telefonleitung ein Trägersignal identifiziert (beim Fax hört man es immer als Gezwitscher am Anfang. Das ist die Trägerfrequenz für die Modulation durch die nachfolgenden Daten), so geht diese Bit auf 1.

Die Bits 0 bis 3 können auch einen Interrupt auslösen, wenn der V24-Baustein entsprechend programmiert wurde. Dazu aber später in einem eigenen Versuch mehr ...

Die weiteren 4 Bits zeigen die ZUSTÄNDE der entsprechenden Leitungen an:

Bit 4 Zustand der Eingangsleitung CTS

Ist dieses Bit gesetzt, dann ist die Leitung Clear to Send bei +10V (= logische 1). Damit signalisiert das Modem seine eigene Sendebereitschaft gegenüber der DEE (PC). Wenn diese Information eingetroffen ist beim PC, dann kann im Programm nun ein Byte in das Sendeschieberegister geschrieben werden.

Bit 5 Zustand der Eingangsleitung DSR

Ist dieses Bit gesetzt, dann zeigt das Modem seine Bereitschaft, eine Sendeanfrage RTS (Request to Send) zu bearbeiten. Jetzt bereits ein Byte in das Sendeschieberegister zu schreiben ist allerdings noch zu früh, da vorher noch der Sendeträger eingeschaltet werden muß. Dies hatte allerdings nur in früheren Zeiten bei mechanischen Codierern und Röhrensendern Bedeutung. Bei den modernen Modems wird dieses Signal nicht mehr verwendet. Jedoch ist diese Abfrage in der BIOS-Routine vorhanden, so daß bei einer Kommunikation über die Bios-Routine die Gegenstation entweder diese Leitung bedienen muß oder durch geeignete externe Schaltung im Stecker (siehe später ...) die BIOS-Routine "überlistet" wird (siehe später ...).

Bit 6 Zustand der Leitung Ring Indicator (RI)

Solange das Modem noch nicht abgehoben hat und ein Anrufsignal vorliegt, liegt die Leitung an Pin 22 auf -10 V. Nicht jedes Modem reicht dieses Signal weiter an die DEE.

Bit 7 Zustand der Leitung Data Carrier Detect

Solange das Modem einen Träger empfängt, ist die Leitung an Pin 8 auf + 10 V und verursacht eine 1 in diesem Bit des Registers.

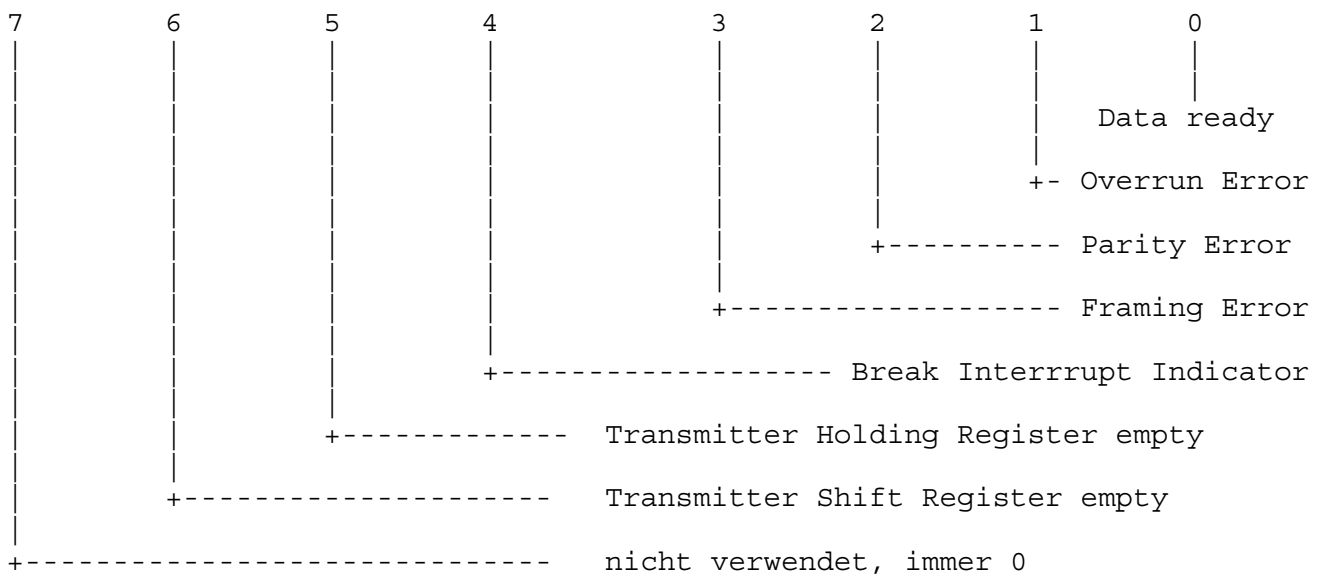
Achtung! Nochmals zur Erinnerung:

Bei den Steuerleitungen RTS, CTS, DTR, SR, RI und DCD ist die logische Eins anderes definiert als bei den beiden Datenleitungen RD (Empfangdaten) und TD (Sendedaten)

Steuerleitung: **POSITIVE** Spannung $> +3V$ (also ca. $+10V$) = LOGISCH EINS

Datenleitungen: **NEGATIVE** Spannung $< -3V$ (also ca. $-10V$) = LOGISCHE EINS

{ Bedeutung der Bits des Line Status Registers LSR:



Bit 0 Data ready: Dieses Bit wird 1 gesetzt, wenn das Empfangsschieberegister ein Zeichen vollständig empfangen hat. Dieses Bit wird gelöscht, sobald der Prozessor mit dem Befehl IN AL,DX dieses Empfangsschieberegister ausliest. Es kann auch durch Schreiben in diesen Port verändert werden.

Bit 1 Overrun Error: Ist dieses Bit 1 gesetzt, so zeigt es an, daß ein neues Empfangsbyte in das Schieberegister geschoben wurde, BEVOR der Prozessor das alte Byte ausgelesen hat. Dieses Bit wird zu 0 gesetzt, wenn der Prozessor dieses LSR ausliest.

Bit 2 Parity Error : Dieses Bit wird 1 gesetzt, wenn bei der Übertragung ein Parity-Check vereinbart wurde (siehe Initialisierung!) und die Quersumme nicht den erforderlichen geraden bzw. ungeraden Wert aufweist. Gelöscht wird dieses Bit durch das Lesen des LSR.

Bit 3 Framing Error : Dieses Bit wird gesetzt, wenn ein ankommendes Zeichen nicht in den gewählten Datenrahmen paßt und dabei ein ungültiges Stopbit auftritt: Das Stopbit muß ja immer eine

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

negative Spannung, also eine (bei Daten) logische 1 sein.

Bit 4 Break Interrupt Indicator : Dieses Bit wird gesetzt, wenn sich im Empfangsregister längere Zeit nichts rührt: Sobald der Zeitraum "Startbit - Datenbits - Stopbit" verstrichen ist, dann ist zu vermuten, daß der Sender seine Datenübertragung abgebrochen hat ("Break").

Diese 5 Bits von 0 bis 4 können bei entsprechender Programmierung des Interrupt Enable Registers IER (Adresse \$3F9) einen Interrupt des Prozessors auslösen. In unserem Fall werden wir aber von der Möglichkeit einer interruptgesteuerten Kommunikation keinen Gebrauch machen.

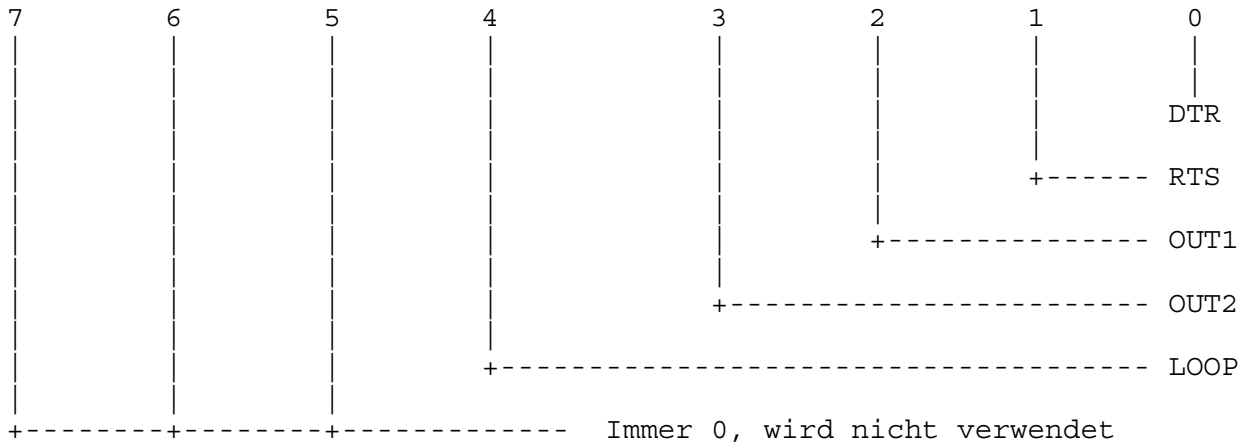
Bit 5 Transmitter Holding Register (THR) empty:
Die Daten des Prozessors gelangen nicht unmittelbar an das Sende-Schieberegister, vielmehr ist dazwischen ein Puffer-Register, das "Holding-Register", geschaltet. Es hat die Aufgabe, Daten des Prozessors erst dann in das Sende-Schieberegister zu kopieren, wenn dieses mit dem Senden fertig ist.
Sobald das THR seinen Inhalt in das Schieberegister kopiert hat, wird dieses Bit auf 1 gesetzt, so daß der Prozessor, wenn er diese Bit abfragt, weiß, wann er das nächste Sendebyte in dieses THR schreiben darf. Sobald dies erfolgt ist, wird dieses Bit auf 0 gesetzt.
Darüber hinaus kann dieses Bit auch einen Interrupt auslösen, wenn er im Interrupt Enable Register (IER) freigegeben wurde.

Bit 6 Transmitter Shift Register (TSR) empty:
Dieses Bit wird gesetzt, wenn nicht nur das Pufferregister, sondern auch das Sendeschieberegister leer, also das Byte schon über die Leitung gegangen ist.

Bit 7 wird nicht genutzt und ist immer 0

Steuerung der Ausgangsleitungen RTS, DTR, und weiterer Hilfssignale

Bedeutung der Bits des Modem Control Registers MCR:



Bit 0 Data Terminal Ready

Wird dieses Bit vom Programm gesetzt, dann geht die Leitung DTR (Pin 20) in den aktiven Zustand auf +10V

Bit 1 Request to Send

Hier gilt dasselbe, wie oben, nur betrifft es hier die Leitung RTS (Pin 4)

Bit 2 Out 1

Dieses Bit ist mit einem Ausgang des integrierten V24-Bausteins verbunden. Dieser Ausgang wird selten bei PC's verwendet. Dagegen ist das nächste Bit äußerst wichtig für die Auslösung eines Interrupts:

Bit 3 Out 2

Ist dieses Bit auf 0 gesetzt, dann wird KEIN Interrupt vom V24-Baustein an den Interrupt-Controller des PC's weitergeleitet, auch wenn er auf dem Baustein selbst signalisiert wurde. Erst wenn diese Bit auf 1 gesetzt ist, dann kann ein auf dem Baustein ausgelöster Interrupt an den PC weitergeleitet werden.

Bit 4 LOOP

Dieses Bit dient zum Test und zur Diagnose der Funktion des V24-Controller-Bausteins. Dazu werden die Ausgänge RTS und DTR mit den zugehörigen Eingängen CTS und DSR sowie die Sendeleitung TD mit der Empfangsleitung RD verbunden

Es gibt noch ein weiteres Ausgaberegister, das Line Control Register. Mit diesem Register wird bestimmt, wie viele Bits ein gesendetes Zeichen hat (die alten Fernschreiber hatten z.B. nur 5 Bits!), wie viele Stopbits ein gesendetes und ein empfangenes Byte haben müssen (z.B. die alten mechanischen Fernschreiber benötigten zwei davon, heute ist nur ein einziges üblich), ob ein Parity-Bit mitgeschickt wird und, wenn ja, ob es eine gerade oder eine ungerade Anzahl von Bits signalisieren soll.

Dieses Line Status Register wird aber auch durch die Initialisierung über das BIOS-Programm INT(\$14) belegt (s.o.), so daß wir uns nicht weiter darum

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

kümmern wollen.

[illegible]

Begin

```
ClrScr;           { Bildschirm löschen }
V24_Initialisieren(8);
```

ASM

```

*      { Adresse $3FC, Modem Control Register
*      {
*      { Inhalt dieses Registers holen und dafür sorgen, daß}
*      { nur die Bits 0 und 1 gelöscht werden, ohne die an- }
*      { deren zu verändern                                }
*      {
*      { Inhalt von AL ins Modem Control Register MCR      }
*      { RTS zunächst löschen: Am Pin 4 -10V ausgeben    }

```

```
{ Nun dafür sorgen, daß die Leitung auch wirklich gesetzt wird. Warum
durch Lesen des Line-Status-Registers eine Blockade der Weiterleitung
der Signale des Modem-Control-Registers aufgehoben wird, ist unklar!
Jedenfalls wurde dies nach 12 Stunden mit Hilfe der "Sherlock-Holmes-
Programmiertechnik" entdeckt. Wenn es einmal funktioniert, dann braucht
man deswegen dieses Register NICHT mehr auszulesen }
```

```
♣ { Adresse des Line Status Registers LSR laden }
♣ { Inhalt dieses Registers einlesen und zur Anzeige }
♣ { in die Variable lsr abspeichern }
```

END; { ASM }

```
Got oxy(1,1);
```

```
Bit_aus('LSR nach RTS und DTR = -10V ',lsr,4,1);
```

	Marke 1	

```
{ Nun nacheinander DTR und anschließend RTS auf +10V setzen. Dabei muß ge-  
gewährleistet sein, daß NUR, und auch wirklich NUR das jeweilige Bit im  
MCR gesetzt wird, ohne daß ein anderes Bit geändert wird }
```

ASM

```

♣      { Adresse des MCR laden                                     }
♣      { Byte holen                                              }
♣      { Bit setzen                                              }
♣      { und abschicken. Hier muß Pin 20 auf +10V gehen         }

```

	Marke 2	

```
{ Nun Bit Nr. 0 wieder löschen:}
```

♣
♣
♣
♣ { und Bit Nr. 1 setzen, ohne andere Bits zu ändern }

END ;

{ Marke 3 }

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

```
{|
{ Sie spielen jetzt mit dem Sende-Schieberegister und dem Line-Status-Register.
  Als erstes schicken Sie ein Byte auf die Reise und beobachten die ent-
  sprechende LED. Damit sie erst einmal richtig flackert, wird das Muster
  0000 0000 abgeschickt. Als zweites kommt das Muster 1111 1111
  Beobachten Sie den Unterschied im Leuchtverhalten der LED
}
```

ASM

♣
♣
♣

♣
♣

END; {ASM}

```
{ Erklären Sie den Unterschied! }
```

```
{ Nun untersuchen Sie die Wirkung von Bit Nr. 6 im Line Status Register.
  Dieses Bit wird gelöscht, sobald Sie in die Adresse $3F8 (= Puffer des
  Sende-Schieberegisters) ein Byte schreiben und wieder auf 1 gesetzt,
  wenn das Schieberegister selbst nach Übernahme aus dem Puffer die Bits
  vollständig auf die Leitung geschoben hat und zur Übernahme eines neuen
  Zeichens bereit ist.}
```

```
delay(500);          { Warten, bis das Senderegister sicher leer ist! }
```

ASM

```
♣          { Adresse des Line Status-Registers laden }
♣          { Inhalt des Line Status Registers holen }
♣          { Register AL in die Variable LSR schieben und }
```

END;

```
clrscr;          { Bildschirm löschen }
```

```
Bit_aus('LSR vor dem Absenden eines Byte ',lsr,1,1);
```

```
{ Nun beim ersten Befehl nach Marke 4 einen Breakpoint setzen und
  mit <Strg><F9> das Programm ohne Verzögerung bis dorthin laufen lassen }
```

ASM

```
♣          { Adresse des Sende-Registers laden }
♣          { Eine Eins, also 1x +10V als Startbit,
              dann 1x -10V als 1 und dann 7x +10V und abschließend
              wieder -10V als Stopbit }
♣          { auf die Reise Schicken }
```

END;

```
{ Sofort nach dem Laden des Sendepuffers LSR auslesen:}
```

ASM

```
♣          { Adresse des Line Status-Registers laden }
♣          { Inhalt des Line Status Registers holen }
♣          { Register AL in die Variable LSR schieben und ... }
```

END;

```
Bit_aus('LSR nach Laden des Sendepuffers ',lsr,2,1);
```

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

```
Zykl := 0;
{ ... nun laufend LSR überwachen und abschließend die Anzahl der durchlaufenen
  Wartezyklen anzeigen: }
While (LSR AND $40) = 0 do { Schieberegister noch nicht leer }
  Begin
    inc(zykl);          { Zykluszähler für While }
    ASM
      ♣                 { Adresse des Line Status-Registers laden }
      ♣                 { Inhalt des Line Status Registers holen }
      ♣                 { Register AL in die Variable AL schieben }
    END;
  End;
Bit_aus('LSR nach dem Leeren des TR: ',lsr,3,1);
Gotoxy(1,4); Write('
                                zykl:1, ' While-Zyklen');

Gotoxy(1,6);
Writeln('Zum Fortfahren bitte Taste drücken');
Readln;

{ _____ Marke 4 _____ }
{ |_____| }
```

{ Nun werden einige Zeichen mit der Baudrate 110Bd gesendet, damit die Gegenstation die Reaktion des Empfangsregisters RD und des Line-Status-Registers untersuchen kann. Bitte sprechen Sie sich mit der Gegenstation ab, wann Sie das nächste Zeichen senden! }

```
For i := 1+65 to 3+65 do
  Begin
    ASM
      ♣                 { Adresse des Senderegisters ins zuständige Register }
      ♣                 { Zeichen 66, 67 und 68 nacheinander ins AL-Register }
      ♣
    End; { ASM }
    Gotoxy(1,23); Write('Noch ',3+65-i, ' Zeichen zu übertragen');
    Delay(300);      { 0.3 Sekunden warten, damit das Zeichen auch
                      sicher über die Leitung gegangen ist }
  End; { For }
```

```
{ _____ Marke 5 _____ }
{ |_____| }
```

{ Die Gegenstation probiert nun die Erkennung eines Rahmenfehlers aus, indem sie ihre Empfangs-Baudrate auf 9600 Baud gestellt hat. Hier wird dazu mit 110 Baud das Byte NULL gesendet, das 9 x +10V als Ausgangssignal hat, so daß die Gegenstation an der Stelle des Stopbits (-10V werden erwartet) sicher +10V feststellt }

```
ASM
  ♣                 { Adresse des Senderegisters ins zuständige Register }
  ♣                 { Nur Nullen senden vom ersten bis zum letzten Bit }
  ♣                 { damit die Leitung auf +10V bleibt, Stopbit wäre -10V }
```

End; { ASM }

Höcht, 29.4.2002

Nun werden die Register untersucht, die auf die Eingangsleitungen der Schnittstelle reagieren:

Delta-CTS, CTS, Delta-DSR, DSR, Datenempfangsregister RD,

Zusätzlich untersuchen Sie die Reaktion des Line Status Registers auf Empfangsdaten

Zu diesem Zweck muß die Gegenstation dieses Programm bis hierher zusammen mit Ihnen durchspielen, während Sie ab hier das Programm schrittweise weiter abarbeiten

ClrScr;

```
{ Untersuchung der Reaktion des Modem Status Registers auf Änderung und Zustand der Leitungen CTS und DSR }
{ Zunächst Initialisierung des Modem Status Registers durch Auslesen dieses Registers }
```

```
{ Zuerst muß die Gegenstation beide Leitungen auf -10V legen, siehe oben...
Dazu muß die Gegenstation ihr Programm starten und bis zu
_____ Marke 1 _____ das Programm schrittweise abarbeiten! }
```

Als nächstes wird das eigene MSR initialisiert durch einmaliges Auslesen, damit die Delta-Bits Nr. 0 und Nr.1 gelöscht werden: }

ASM

```
♣ { Adresse des MSR laden }
♣ { und Inhalt ins AL einlesen }
```

End;

```
{ Nun wird das Modem Status Register nochmals ausgelesen und mit den gelöschten Delta-Bits am Bildschirm angezeigt. }
```

ASM

```
♣ { Adresse des MSR laden }
♣ { und Inhalt ins AL einlesen }
♣ { Inhalt von AL in die Variable MSR zur Anzeige kopieren }
```

End;

Bit_aus('MSR Anfangszustand ',MSR,1,1);

```
{ An dieser Stelle muß die Gegenstation die Leitung DTR auf 1 setzen.
Dazu muß sie das Programm bis zu _____ Marke 2 _____ abarbeiten! }
```

Lesen Sie anschließend zweimal das Modem-Status-Register aus:}

ASM

```
♣ { Adresse des MSR laden }
♣ { und Inhalt ins AL einlesen }
♣ { Inhalt von AL in die Variable MSR zur Anzeige kopieren }
```

End;

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

```
Bit_aus('MSR nach RTS = 1, erstes Auslesen ',MSR,2,1);
```

```
ASM
```

```
♣           { Adresse des MSR laden                               }
♣           { und Inhalt ins AL einlesen                           }
♣           { Inhalt von AL in die Variable MSR zur Anzeige kopieren }
```

```
End;
```

```
Bit_aus('MSR nach RTS = 1, zweites Auslesen ',MSR,3,1);
```

```
{ Beachten Sie insbesondere Bit Nr. 0 und 1, die für die Anzeige einer
  ÄNDERUNG verantwortlich sind }
```

```
{ Die Gegenstation ändert nun gleichzeitig DTR auf -10V (= logische 0) und
  RTS auf +10V (= logische 1), indem sie ihr Proramm bis _____ Marke 3 _____
  abarbeitet. }
```

```
Untersuchen Sie nun wieder zweimal hintereinander das Modem-Status-Regis-
ter. Beachten Sie, daß nun Delta-DSR = 1, aber DSR selbst gleich 0 ist! }
```

```
ASM
```

```
♣           { Adresse des MSR laden                               }
♣           { und Inhalt ins AL einlesen                           }
♣           { Inhalt von AL in die Variable MSR zur Anzeige kopieren }
```

```
End;
```

```
Bit_aus('MSR nach DTR = 0, RTS = 1, erstes Auslesen ',MSR,4,1);
```

```
ASM
```

```
♣           { Adresse des MSR laden                               }
♣           { und Inhalt ins AL einlesen                           }
♣           { Inhalt von AL in die Variable MSR zur Anzeige kopieren }
```

```
End;
```

```
Bit_aus('MSR Nach DTR = 0, RTS = 1, zweites Auslesen ',MSR,5,1);
```

```
{ Hier folgen Versuche mit dem Empfangsregister und dem Line Status Register}
```

```
{ Zur Initialisierung muß je einmal das Empfangsregister und das Line-Stat-
  us-Register ausgelesen werden, nachdem die Gegenstation bis zu
  _____ Marke 4 _____ im Programm gekommen ist }
```

```
ASM
```

```
♣           { Adresse des Empfangsregisters laden                 }
♣           { und Inhalt ins AL einlesen                           }
```

```
End;
```

```
ASM
```

```
♣           { Adresse des LSR laden                               }
♣           { und Inhalt ins AL einlesen                           }
```


Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

End;

```
{ Untersucht werden soll zunächst Bit 0 Data Ready bei korrektem Empfang
  eines Zeichens. Dazu wird zunächst nochmals das LSR ausgelesen, um zu
  sehen, ob ein Zeichen im Empfangsregister angekommen ist. }
```

ASM

```
♣           { Adresse des LSR laden }
♣           { und Inhalt ins AL einlesen }
♣           { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
```

End;

Bit_aus('LSR vor Empfang eines Zeichens ',lsr,7,1);

```
{ Jetzt muß die Gegenstation ein Zeichen senden, in dem es die Schleife nach
  _____ Marke 4 _____ EINMAL durchläuft.
  Darauf wird zuerst LSR, dann das Empfangsregister und schließlich nochmals
  das LSR ausgelesen, um das Verhalten von Bit Nr. 0 zu beobachten }
```

ASM

```
♣           { Adresse des LSR laden }
♣           { und Inhalt ins AL einlesen }
♣           { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
```

End;

Bit_aus('LSR nach Empfang vor Auslesen eines Zeichens ',lsr,8,1);

ASM

```
♣           { Adresse des Empfangsregisters RD laden }
♣           { und Inhalt ins AL einlesen }
♣           { AL in die Variable RD kopieren zur Anzeige am Bildschirm }
```

End;

Bit_aus('Inhalt des Empfangsregisters ',RD ,7,1);

Gotoxy(65,7);Write('Zeichen: ',chr(rd));

ASM

```
♣           { Adresse des LSR laden }
♣           { und Inhalt ins AL einlesen }
♣           { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
```

End;

Bit_aus('LSR nach Auslesen des Zeichens ',lsr,10,1);

```
{ Das LSR zeigt mit seinem Bit Nr. 1 an, ob ein zweites Zeichen angekommen
  ist, bevor das erste aus dem Empfangspuffer abgeholt wurde.
  Dies ist Inhalt dieses folgenden Versuchs.
  Die Gegenstation muß nun zwei Zeichen hintereinander senden, bevor LSR und
  RD ausgelesen werden, indem die Schleife nach _____ Marke 4 _____ ZWEI-
  MAL HINTEREINANDER durchläuft }
```

ASM

```
♣           { Adresse des LSR laden }
♣           { und Inhalt ins AL einlesen }
```

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

```
♣ { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
End;
Bit_aus('LSR NACH Empfang ZWEIER Zeichen OHNE Auslesen',lsr,12,1);
ASM
♣ { Adresse des Empfangsregisters RD laden }
♣ { und Inhalt ins AL einlesen }
♣ { AL in die Variable RD kopieren zur Anzeige am Bildschirm }
End;
Bit_aus('Inhalt des Empfangsregisters ',RD ,13,1);
Gotoxy(65,13);Write('Zeichen: ',chr(rd));
```

```
ASM
♣ { Adresse des LSR laden }
♣ { und Inhalt ins AL einlesen }
♣ { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
End;
Bit_aus('LSR nach Auslesen des 2. Zeichens ',lsr,14,1);
```

{ Abschließend untersuchen Sie, was geschieht, wenn das empfangene Zeichen kein gültiges Stopbit hat. Das Stopbit ist ja ein Pegel bei -10V. Ist dieser Pegel zum Zeitpunkt, an dem ihn der Empfänger bei -10V erwartet, aber bei +10V (also die Binärziffer 0), dann wird ein Rahmenfehler erkannt.
Dies läßt sich am besten dadurch simulieren, daß der Empfänger eine hohe Baudrate eingestellt hat, der Sender aber mit niedriger Geschwindigkeit, z.B. 110 Baud, lauter Nullen (also Pegel +10V) sendet.

Für den Rahmenfehler ist Bit 2 im Line-Status_Register verantwortlich.
Für diesen fehlerhaften Empfang müssen Sie Ihre eigene Schnittstelle abschließend mit z.B. 9600 Baud initialisieren.}

```
V24_Initialisieren(96);
```

{ Nach dieser Initialisierung muß die Gegenstation ein Nullbyte senden.
Dazu muß sie nach der Schleife bei _____ Marke 5 _____ weiterfahren.

Lesen Sie nun wieder das LSR und das RD-Register aus!}

```
ASM
♣ { Adresse des LSR laden }
♣ { und Inhalt ins AL einlesen }
♣ { AL in die Variable LSR kopieren zur Anzeige am Bildschirm }
End;
Bit_aus('LSR NACH Empfang mit Frame Error',lsr,16,1);
```

```
ASM
♣ { Adresse des Empfangsregisters RD laden }
♣ { und Inhalt ins AL einlesen }
♣ { AL in die Variable RD kopieren zur Anzeige am Bildschirm }
```

Serielle Schnittstelle 1, Elektrische Signale und Programmierung der Register

```
End;  
Bit_aus('Inhalt des Empfangsregisters ',RD ,17,1);  
Gotoxy(65,17);Write('Zeichen: ',chr(rd));
```

```
ASM
```

```
♣           { Adresse des LSR laden           }  
♣           { und Inhalt ins AL einlesen       }  
♣           { AL in die Variable LSR kopieren zur Anzeige am Bildschirm}
```

```
End;  
Bit_aus('LSR nach Auslesen des Zeichens mit falschen Rahmen ',lsr,18,1);
```

```
Gotoxy(1,20);  
Writeln('Zum Abschluß bitte Taste drücken');  
Readln;
```

```
End.
```