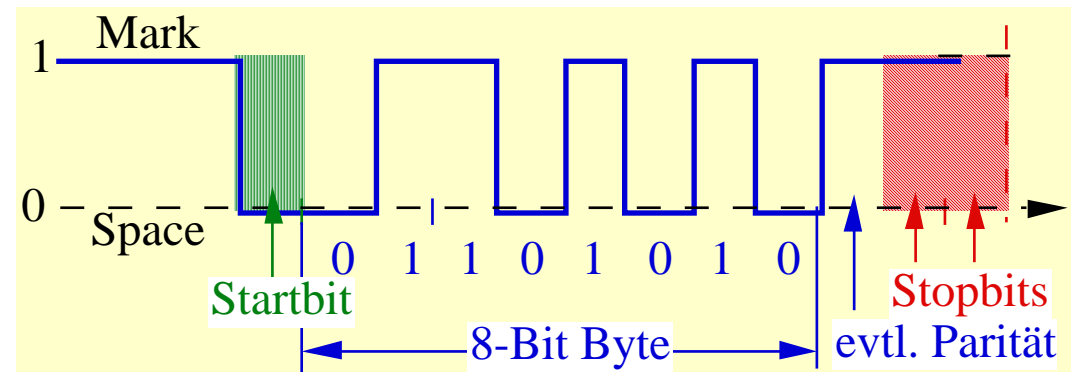


7. Die serielle Schnittstelle

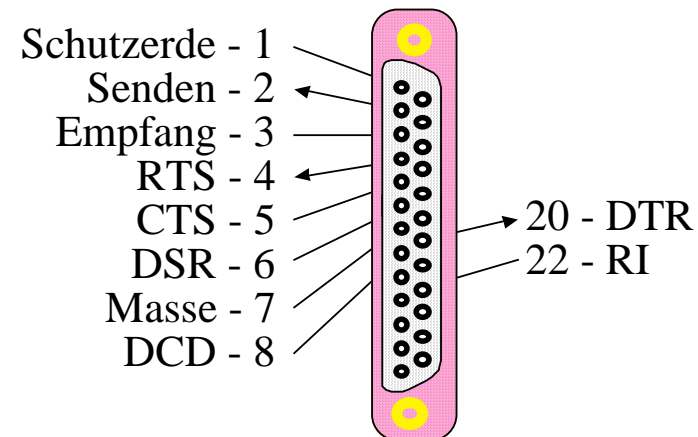
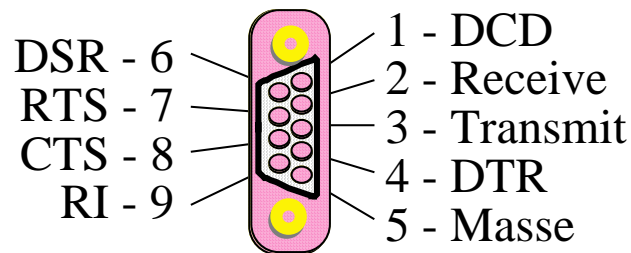
7.1 Asynchrone Signalform

- Beginn einer Zeichenübertragung zu einem beliebigen asynchronen Zeitpunkt:
- Ähnliche Technik wie beim Tastaturanschluß:
 - Datenraten von 300 Bit/sec bis ~1 MBit/sec.
 - minus 12 Volt bedeutet eine binäre 1,
 - Ruhezustand ist ebenfalls -12 Volt,
 - plus 12 Volt bedeutet eine binäre 0,
 - Startbit, Stopbits, Parity,
 - keine separate Taktleitung,
- Parität:
 - gerade, ungerade, keine, eins, null,
 - even, odd, none, one, zero.



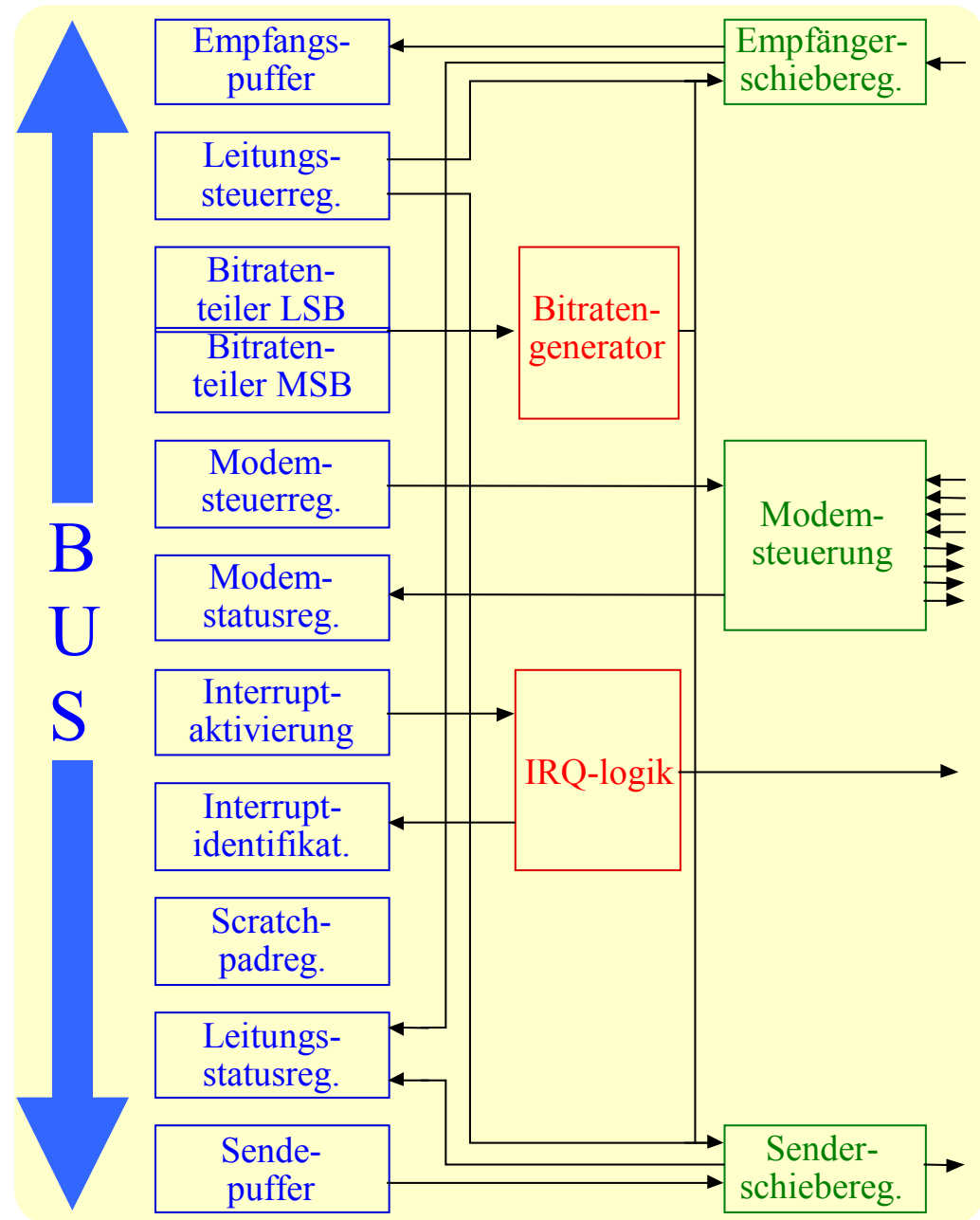
7.2 Steckerformat

- COM1-Schnittstelle meist als 9-poliger Trapezstecker mit Stiften:
 - auf Wunsch IRQ#4, Port-Register bei \$3f8,
 - separate Sende- & Empfangsleitung,
 - Steuersignale zur Flusskontrolle etc.
 - Bereitschaftszustände.
- COM2 - Schnittstelle meist als 25-poliger Trapezstecker, weiblich (\$2f8, IRQ#3).
- Stiftbelegung als DTE:



7.3 8250 Chip-Architektur

- Meist zusammen mit weiteren Supportchips in Südbrücke integriert.
- Funktionale Erweiterung:
 - Hitachi 16450 / 16550,
 - Mit zusätzlichem Puffer.
- Steuerung der Schnittstellenpins:
 - Bereitschaftszustände,
 - Anruferanzeige,
 - Flusskontrolle.



7.4 Beispiel: UartPoll.pas

```
program UartPoll; uses dos,crt;
const   com   = $3f8;           (* com1=$3f8, com2=$2f8*)
        ier   = 1+com;         (* interrupt enable reg. *)
        iir   = 2+com;         (* interupt identify reg. *)
        lcr   = 3+com;         (* line control register *)
        mcr   = 4+com;         (* modem control reg. *)
        lsr   = 5+com;         (* line status register *)
        msr   = 6+com;         (* modem status register *)
        rxd   = $01;           (* Receivebuffer is full *)
        txe   = $40;           (* Transmittbuffer empty*)
var status, inp : shortint;    zlr : longint;
procedure InitCom( rate: longint);
const   clock = 115200;   stopBit2   = 4;
        noParity   = 0;     data8     = 3;
        Lsb        = com;     (* low byte divisor *)
        msb        = 1+com;   (* high byte divisor *)
begin   port[Lcr ] := $80;    (* select divisorLatch *)
        port[Lsb ] := (clock div rate) mod 256;
        port[msb ] := (clock div rate) div 256;
        port[Lcr ] := stopBit2 +noParity +data8;
end;
begin   InitCom( 300 );
        for zlr:=0 to 9999999 do begin
            status := port[ Lsr ];
            if (status and $1e)>0 then writeln('erro');
            if (status and rxd)>0 then write(port[com]:2)
            if (status and txe)>0 then port[com]:=2;
        end end.
```

7.5 Registersemantik

7.5.1 Empfangs- & Senderegister

- Auch Empfangspuffer bzw. Sendehaltereregister genannt.
 - Unstrukturierte 8-Bit Register.
 - Bei kürzeren Zeichen Inhalt maskieren.
 - Die Register kommunizieren mit den jeweiligen Schieberegistern.
 - FiFos im Chip 16450 und 16550 unsichtbar.

7.5.2 Leitungsstatusregister (LSR)

- Auch Serialisierungs-Status-Register genannt.

[0]	RxRD:	Empfangsdaten bereit,
[1]	ÜBLF:	Empfangsdatenüberlauf,
[2]	PARF:	Paritätsfehler,
[3]	FRMF:	Framing-Fehler.
[4]	BREK:	Break festgestellt,
[5]	TBE:	Transmitpuffer leer,
[6]	TXE:	Transmitshifter leer.

7.5.3 Leitungsteuerregister (LCR)

- Auch Datenformatregister genannt.
 - [0..1] D-Bits: (5Bit, 6Bit, 7Bit, 8Bit),
 - [2] STOP2: zwei Stopbits,
 - [3..5] Parität: (none, odd, , even, 1, , 0).
 - [6] BRK: Break erzeugen bitte,
 - [7] DLAB: Divisorregister selektieren.

7.5.4 Modemstatusregister (MSR)

- Zustand der RS-232C Schnittstelle.
 - [0] DCTS: CTS-Änderungsanzeige,
 - [1] DDSR: DSR-Änderungsanzeige,
 - [2] DRI: RI-Änderungsanzeige,
 - [3] DDCD: DCD-Änderungsanzeige.
 - [4..7] CTS, DSR, RI, DCD: Signalpegel.

7.5.5 Modemsteuerregister (MCR)

- Steuerung des Modems & Flusskontrolle.
 - [0] DTR: Betriebsbereitschaftsanzeige,
 - [1] RTS: Empfangsbereitschaft,
 - [2] OUT1: erster Hilfsausgang (RI),
 - [3] OUT2: Ausgang für Interruptgatter.
 - [4] LOOP: Prüfschleife bilden bitte,
 - [5..7] immer null setzen.

7.5.6 Interruptaktivierungsregister (IER)

- Auch Interrupt-Enable Register genannt.
- Interrupt auslösen bei folgenden Ereignissen:
 - [0] RXD: Byte im Empfangspuffer (->RxRD),
 - [1] TBE: Transmitpuffer leer,
 - [2] ERBK: Paritätsfehler oder Break,
 - [3] SINP: Zustandsänderung am RS-232C.
 - [4..7] immer null setzen.

7.5.7 Interruptidentifizierungs-Register (IIR)

- Enthält genauere Ursache der Unterbrechung:
 - [0] not PEND: keine Unterbrechung anhängig,
 - [1..2] Typ (SINP, TBE, RXD, ERBK),
 - [3..7] ergibt immer null.
- Unterbrechungen im UART zurücksetzen:
 - nach Paritätsfehler LSR lesen,
 - nach Receive den Empfangspuffer lesen,
 - nach Transmit: Byte schreiben oder IIR lesen,
 - nach Modem-Event MSR lesen.

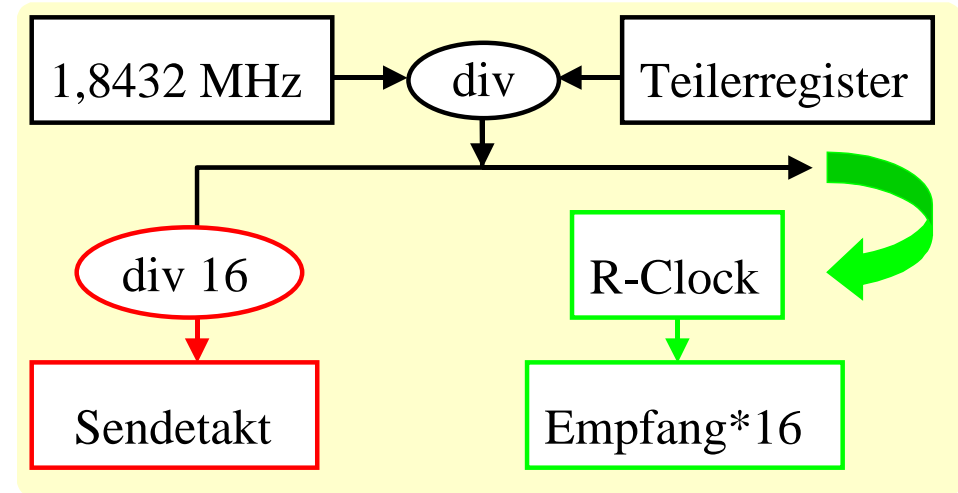
7.5.8 Teilerregister (LSB, MSB)

- Datenrate:

- Primärteiler = 65535 -> 28,1 Hz,
- **Primärteiler = 1** -> **115200 Hz.**
- MSB: more significant Byte (Teiler high),
- LSB: less significant Byte (Teiler low),

- Sendetakt auf der Leitung:

- 16-facher Sekundärteiler zum Senden,
- Hauptreferenztakt 1,8432 Mhz,
- 115200 Hz.



- Empfangstakt zur mehrfachen Abtastung des empfangenen Signals.

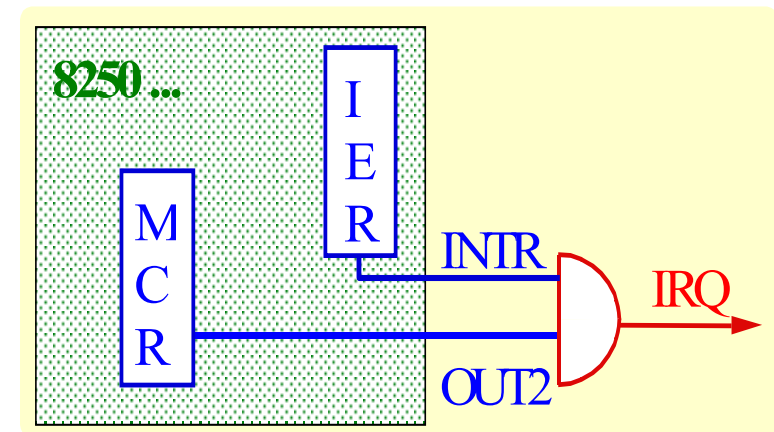
- Jedes Bit zur Erhöhung der Übertragungssicherheit 16 mal abgetastet.

- Ausnahmsweise mit 921600 Khz:

- Nur 2* abtasten pro Bit.
- Sekundärerteilung=2,

7.6.1 Interrupt-Aktivierung:

- Handler muss als Interrupt bezeichnet sein, da andere Codegenerierung:
 - alte Flags werden per IRET wieder geladen,
 - IRET wird vom Pascal-Compiler erzeugt,
 - alten Handler sicherstellen (wie gehabt).
- Gewünschte Interrupttypen im Interrupt-Enable Register (IER) wählen:
 - Paritätsfehler etc. (\$04, Priorität=0 bzw. hoch),
 - Receive-Interrupt (\$01, Priorität=1),
 - Transmit-Interrupt (\$02, Priorität=2),
 - Modemsteuerleitungen (\$08, Priorität=3),
- Externes Interruptgatter mit OUT2 öffnen:
 - OUT2 ist ein Hilfssignal im MSR,
 - leitet Unterbrechung an den Interruptkontroller:
- Interrupt #3 oder #4 im 8259 demaskieren.



7.6.2 Interrupt-Betrieb:

- Ausdecodieren, welcher Chip-Interrupt:
 - aus dem Interrupt-Identifikationsregister,
 - ausnahmsweise aus dem Leitungsstatusregister.
- Abgehandelte Interrupts im UART-Chip zurücksetzen:
 - nach Paritätsfehler LSR lesen (ERBK),
 - nach Receive den Empfangspuffer lesen,
 - nach Transmit: Byte schreiben oder IIR lesen,
 - nach Modem-Event MSR lesen (SINP).
- Interrupt im 8259 zurücksetzen (EOI).
- Vorsicht bei Nutzung derselben Variablen im Interrupt & im Programm.
- Ende des Interrupt-Betriebs:
 - Handler deinstallieren, alten Handler wieder einsetzen.
 - Je nachdem Interrupt abschalten oder weiter laufen lassen?
 - Interrupts im UART über Out2 deselektieren?

7.6.3 Beispiel mit Interrupt

```
program UARTInterrupt;      uses dos,crt;
const   com = $3f8; (* com1=$3f8, com2=$2f8*)
        ier  = 1+com;      (* interrupt enable *)
        iir  = 2+com;      (* interrupt identify*)
        lcr  = 3+com;      (* line control      *)
        mcr = 4+com;      (* modem control  *)
        lsr  = 5+com;      (* line status   *)
        msr = 6+com;      (* modem status  *)

        txe  = $40;      (* shiftregister empty *)
        tbe  = $20;      (* Transmitbuffer empty *)
        rxd  = $01;      (* Receivebuffer full  *)
        pck = '123456789012345678901234567890';

var oldHandler : procedure;
procedure InitCom( rate: longint );
const   clock      = 115200;
        Lsb        = 0+com; (* low byte divisor *)
        msb        = 1+com; (* hig byte divisor *)
        stopBit2   = 4; noParity = 0;
        data8      = 3;
begin port[ Lcr ] := $80; (* select divisorlatch *)
      port[ Lsb ] := (clock div rate) mod 256;
      port[ msb ] := (clock div rate) div 256;
      port[ Lcr ] := stopBit2 + noParity + data8;
      port[ ier ] := $05;      (* frameErr, rxd *)
      port[ mcr ] := $08;      (* out2 for iGate *)
      port[ $21 ] := port[ $21 ] and not $10 ;
end;
```

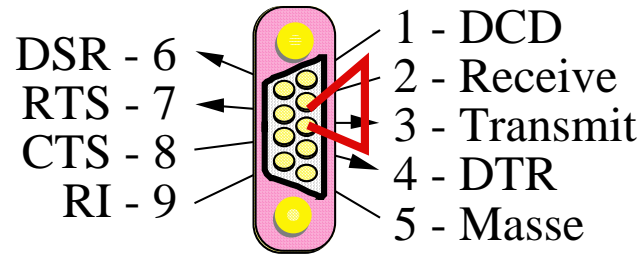
```

procedure ComHandler; interrupt;
begin
    case port[ iir ] of (* Int.-Identifikation *)
        1: (* no interrupt pending *);
        0: writeln('modem', port[msr]);
        2: writeln('transmit empty' );
        4: write ( chr( port[com] ) );
        6: writeln('frame', port[Lsr]);
    end;
    port[ $20 ] := $20;          (* EOI für 8259 *)
end;
procedure SendChar( ch: char);
begin    repeat until (port[Lsr] and txe)>0;
        port[com]:=ord(ch);
end;
procedure Sendpacket( packet: String; len: integer);
var pos : integer;
begin    for pos:=1 to len
        do SendChar( packet[ pos ] )
end;
begin    GetIntVec( 12, addr( oldHandler ) );
        SetIntVec( 12, addr( ComHandler ) );
        InitCom( 20 );
        SendPacket( pck, 25 );
        write( '<FIN>' );
        repeat until keypressed;
        writeln; writeln;
        port[ mcr ] := $00;      (* deselektieren *)
        SetIntVec( 12, addr( oldHandler ) );
end.

```

7.6.4 Testszenarium:

- Echodraht am lokalen Stecker:



- Ausgabe des Programmes:

12345678901234567890123<FIN>45

- Als Alternative den Loopmodus aktivieren:

- chipinterne Umschaltung für Diagnosezwecke,
- Sender- und Empfangsschieberegister verbunden.
- DTR mit DSR und RTS mit CTS verbunden,
- Out2 mit DCD und Out1 mit RI verbunden.

- Die Parallel-Serie Wandlung bewirkt ca. 2 Zeichen Verzögerung:

