# The RS232 - Interface

The heart of the serial interface in the PC is the serial module UART 8250 (*Universal Asynchronous Receiver Transmitter*). This module allows serial data transmission, during which it takes over every conversion of data from parallel to serial and from serial to parallel.

The 8250 contains over 10 internal registers which allow control of the serial interface lines and transmitting and receiving of data. The major part of these registers are used during initializing of the module, while during data transmission, usually only one or two registers are used.

The registers of the 8250 are accessed with the assembler command OUT and IN.

After initializing the interface, the data, which is transferred to the 8250 through the transmitter register (*Transmitter Holding Register*), automatically transmits serial so that the main program can devote itself to other tasks. It merely has to check the status register (*Line Status Register*) before it can transfer another character to the transmitter register

# The BIOS-Interface

The BIOS of the PC contains functions which support a serial data transmission. These functions can be used to - without much effort - initialize the serial interface and transmit data.

The BIOS functions are invoked through a software interrupt (INT14h). Upon entering, the register AH must include the function number and DX the serial interface. A detailed explanation can be found in the corresponding BIOS-literature.

For simple uses these BIOS functions should be sufficient, nevertheless they are not flexible and quick enough to overcome complex tasks. A system programmer must therefore concern himself in more detail with the programming of the UART if he wishes to realise a data transmission in the background in his appliaction which should run interruption-driven with maximum speed and security, complicated flow control and under various protocols.

## Programming the 8250

The 8250 contains 10 internal registers which at first appear to be quite complicated. This complexity vanishes though as soon as one looks closer at these registers. Furthermore, not all registers are used in each situation.

The following table lists the registers and their addresses.

| Register | Adress (Offset) |
|---|---|
| Transmitter Holding Register (THR) | 0 1) |
| Receive Data Register (RDR) | 0 1) |
| Baud Rate Divisor LSB | 0 2) |
| Baud Rate Divisor MSB | 1 2) |
| Interrupt Enable Register (IER) | 1 1) |
| Interrupt Identification Register (IIR) | 2 |
| Line Control Register (LCR) | 3 |
| Modem Control Register (MCR) | 4 |
| Line Status Register (LSR) | 5 |
| Modem Status Register (MSR) | 6 |

1) Bit 7 in the Line Control Register = 0
2) Bit 7 in the Line Control Register = 1

## Initializing

The following 5 registers must be initialized before the serial interface is accessible. These registers are:

- ❑ Baud Rate Divisor LSB
- ❑ Baud Rate Divisor MSB
- ❑ Line Control Register
- ❑ Modem Control Register
- ❑ Interrupt Enable Register

Please be aware that Bit 7 (DLAB) ot the *Line Control Register* is set on 1, in order to address the *Baud Rate Divisor Register*. In all other cases this Bit = 0.

In the further course of the data transmission the individual data is transmitted through the *Transmitter Holding Register* and received through the *Receiver Data Register*. The *Line Status Register* informs the application program whether a character can be transmitted or whether a character was received and is ready for fetching.

In order to transmit a character, Bit 5 in the *Line Status Register* is checked. If Bit 5 = 1 then the *Transmitter Holding Register* is empty, i.e. the 8250 is ready to accept and transmit another character. Whether a character is ready for fetching is ascertained through Bit 0 of the *Line Status Register*. Here Bit 0 must equal 1.

## Baud Rate Divisor

The transmission speed of the serial interface is set by the two *Baud Rate Registers*.

As a result the divisor value (1 word) is determined through the exact division

Divisor = 115200 / Baud rate

and the LSB part of the divisor value transferred into the *Baud Rate Register* LSB and the MSB part to the *Baud Rate Register* MSB.

For initializing in 9600 Baud the above formula gives

115200 / 9600 = 12 = 0Ch.

Thus the *Baud Rate Register* LSB is initialized with OCh and the Baud Rate Register MSB with 00h. As already mentioned, for initializing the *Baud Rate Register* Bit 7 (DLAB) of the *Line Control Register* must be set to 1.

Example:

```
MOV    AL, 80h      ; Bit 7 Mask
MOV    DX, IO Reg+3 ; Line Control Register
OUT    DX, AL       ; Set Bit 7
MOV    DX, IOReg    ; Baud Rate Register LSB
MOV    AX, Divisor  ; divisor value
OUT    DX, AL       ; LSB
MOV    DX, IOReg+1  ; Baud Rate Register MSB
MOV    AL, AH
OUT    DX, AL       ; MSB
```

## Line Control Register

After completing the setting of the transmission speed, setting the protocol (character length, number of stop Bits, parity etc.) through the *Line Control Register* must take place.

This time Bit 7 must be equal to 0 (DLAB), so that it does not address to the *Baud Rate Register* MSB

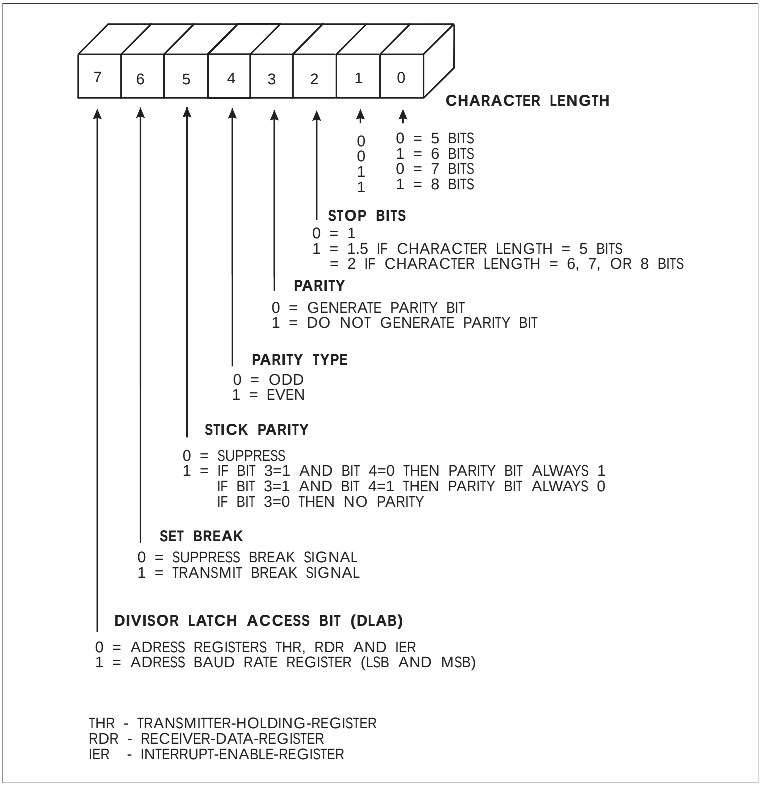

7 6 5 4 3 2 1 0
CHARACTER LENGTH
0 0 = 5 BITS
0 1 = 6 BITS
1 0 = 7 BITS
1 1 = 8 BITS
STOP BITS
0 = 1
1 = 1.5 IF CHARACTER LENGTH = 5 BITS
= 2 IF CHARACTER LENGTH = 6, 7, OR 8 BITS
PARITY
0 = GENERATE PARITY BIT
1 = DO NOT GENERATE PARITY BIT
PARITY TYPE
0 = ODD
1 = EVEN
STICK PARITY
0 = SUPPRESS
1 = IF BIT 3=1 AND BIT 4=0 THEN PARITY BIT ALWAYS 1
IF BIT 3=1 AND BIT 4=1 THEN PARITY BIT ALWAYS 0
IF BIT 3=0 THEN NO PARITY
SET BREAK
0 = SUPPRESS BREAK SIGNAL
1 = TRANSMIT BREAK SIGNAL
DIVISOR LATCH ACCESS BIT (DLAB)
0 = ADRESS REGISTERS THR, RDR AND IER
1 = ADRESS BAUD RATE REGISTER (LSB AND MSB)
THR - TRANSMITTER-HOLDING-REGISTER
RDR - RECEIVER-DATA-REGISTER
IER - INTERRUPT-ENABLE-REGISTER

Figure 2-1 Line Control Register

**Example:**

```
MOV  AL,00000011b ; Bit mask for 8 data
                  ; Bits,
                  ; 1 Stop Bit
                  ; no parity
MOV  DX,IOReg+3   ; Line Control Register
OUT  DX,AL        ; initialize
```

## Modem Control Register

The *Modem Control Register* serves to control the modem line and is initialized in the uninterruptable operating mode with 03h (DTR = 1 and RTS = 1), which for most cases is correct and, if no modems are connected, doesn't interfere.

If the serial interface should operate in the interrupt mode (communication in the background), Bit 3 (OUT 2) is to be set to 1. In the Loop Mode, however, Bit 4 is to be set to 1.

In the interrupt mode an IRQ line must exist for the interface, through which the interrupt can be invoked. Additionally the 8259 Interrupt Controller must be set (appropriate Bit set to 0) such that an interrupt is invoked through the IRQ line.

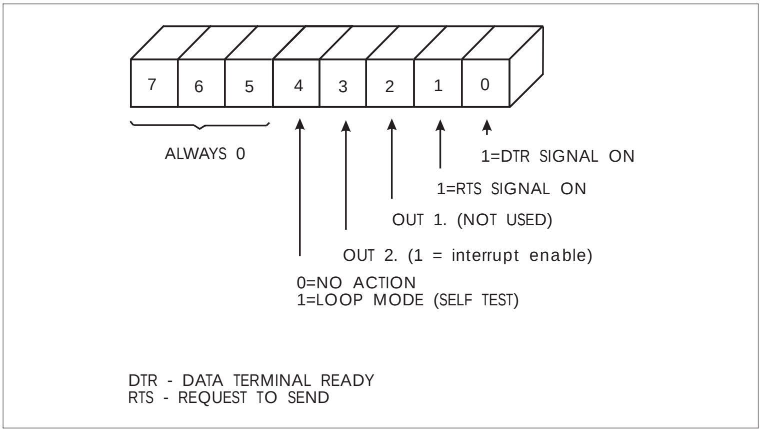


Figure 2-2 Modem Control Register

In the Loop Mode (test mode) each transmitted character is received again by the interface. Thus the operating mode of the interface may be tested.

**Example:**

```
MOV   AL,00001011b ; Bit mask for DTR, RTS
                   ; OUT2
MOV   DX,IOReg+4   ; Modem Control Register
OUT   DX,AL        ; initialize
```

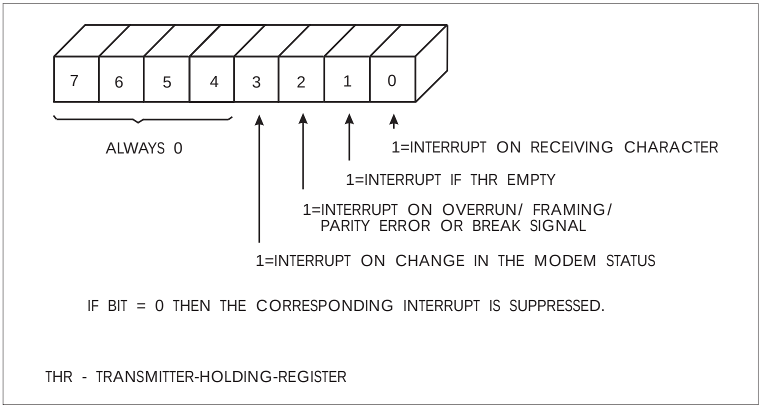

7
6
5
4
3
2
1
0
ALWAYS 0
1=INTERRUPT ON RECEIVING CHARACTER
1=INTERRUPT IF THR EMPTY
1=INTERRUPT ON OVERRUN/ FRAMING/ PARITY ERROR OR BREAK SIGNAL
1=INTERRUPT ON CHANGE IN THE MODEM STATUS
IF BIT = 0 THEN THE CORRESPONDING INTERRUPT IS SUPPRESSED.
THR - TRANSMITTER-HOLDING-REGISTER

Figure 2-3 Interrupt Enable Register

## Interrupt Enable Register

The 8250 contains several interrupt sources, but can only release one interrupt itself.

If OUT2 in the *Modem Control Register* was set to 1 then the events which would lead to an interruption, can be stopped through the *Interrupt Enable Register*.

The events possible are:

❑ Interrupt after receiving a character

❑ Interrupt if *Transmitter Holding Register* empty

❑ Interrupt if error while transmitting or if a break signal is detected.

❑ Interrupt after a change in the status of the modem line.

**Example:**

```
MOV   AL,00001111b   ; Bit mask for all
                     ; interrupt sources
MOV   DX,IOReg+1     ; Interrupt Enable
                     ; Register
OUT   DX,AL          ; initialize
```

## Data transmission with the 8250

Three registers are very important during the data transmission:

❑ Line Status Register

❑ Transmitter Holding Register

❑ Receiver Data Register

Each piece of data is transmitted through the *Transmitter Holding Register* and received through the Receiver Data Register.

To transmit a character, Bit 5 in the *Line Status Register* must be checked. If Bit 5 = 1, then the *Transmitter Holding Register* is empty and the 8250 is ready to accept and transmit another character.

Bit 5 has the value 0 as long as a character is being transmitted.

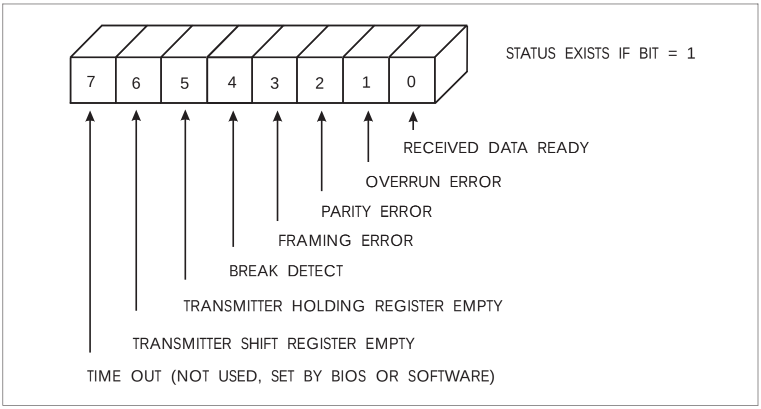

STATUS EXISTS IF BIT = 1
7
6
5
4
3
2
1
0
RECEIVED DATA READY
OVERRUN ERROR
PARITY ERROR
FRAMING ERROR
BREAK DETECT
TRANSMITTER HOLDING REGISTER EMPTY
TRANSMITTER SHIFT REGISTER EMPTY
TIME OUT (NOT USED, SET BY BIOS OR SOFTWARE)

Figure 2-4 Line Status Register

Bit 0 of the *Line Status Register* must be checked for a character to be received. If this Bit is 1, then a character was received and it is ready to be fetched. Bit 0 is reset when the received character has been read out from the *Receiver Data Register.*

A transmission error has occured if Bits 1,2 or 4 of the Line Status Register have the value of 1. A recognition break signal is given through Bit 4. Possible errors during data transmission are:

❑ Framing error if e.g. wrong word length, wrong number of data Bits or stop Bits is used.

❑ Parity error if the parity setting doesn't match

❑ Overrun error if received characters have been overwritten by newly received characters

The status of the modem line may be found through the *Modem Status Register*. Bits 0...3 show the delta state of the CTS, DSR, RI and DCA lines and Bits 4 to 7 contain the current status of the CTS, DSR, RI and DCD lines.

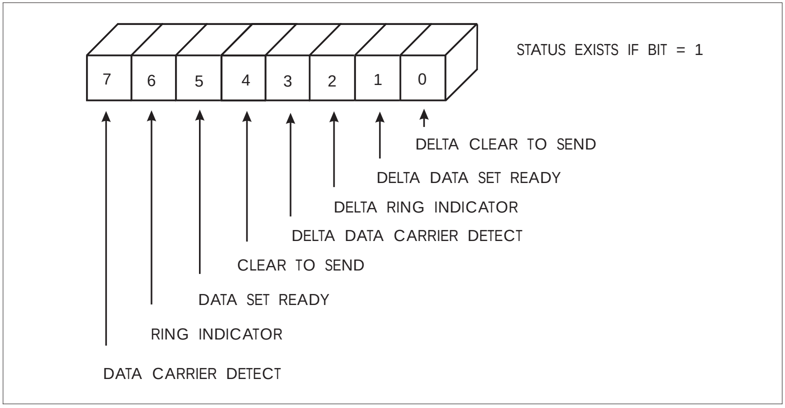


Figure 2-5 Modem Status Register

## Modem Signals

### From computer to modem

**DTR:** informs the modem that the computer is switched on and ready

**RTS:** informs the modem that the program in the computer wants to transmit data

### From modem to computer

**DSR:** the modem informs the computer that it is switched on and ready.

**CTS:** the modem informs the computer that it is ready to receive data.

**DCD:** the modem informs the computer that it has set up a connection with a remote modem

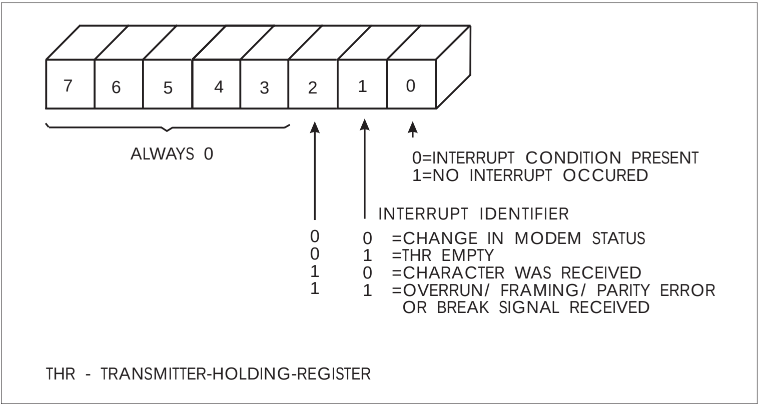
**RI:** the modem informs the computer that it has identified an incoming call on the telephone line.

## The interrupt mode of the 8250

The interrupt mode of the 8250 is selected if the application doesn't have the time to serve the interface (interface status inquiry etc.) and the data transmission must therefore take place in the background.

In this case the interrupt mode of the 8250 is set up through the appropriate registers and communication takes place in the background. The 8250 generates an interruption, if the event on the serial interface is monitored by the *Interrupt Enable Register*. This interruption must then be processed by an interrupt routine.

The interrupt routine must be able to manage one or more tasks depending on the setup of the *Interrupt Enable Register.*



7
6
5
4
3
2
1
0
ALWAYS 0
0=INTERRUPT CONDITION PRESENT
1=NO INTERRUPT OCCURED
INTERRUPT IDENTIFIER
0 0 =CHANGE IN MODEM STATUS
0 1 =THR EMPTY
1 0 =CHARACTER WAS RECEIVED
1 1 =OVERRUN/ FRAMING/ PARITY ERROR OR BREAK SIGNAL RECEIVED
THR - TRANSMITTER-HOLDING-REGISTER

Figure 2-6 Interrupt Identification Register

As several events trigger only one Hardware-Interrupt, the interrupt routine is able to receive information through the *Interrupt Identification Register* about which event triggered the interruption and, depending on this, start the appropriate process. If several events occur simultaneously, then they must be worked off by the interrupt routine successively. In this case, the priority of the interrupt type determines which event is handled first. Bit 0 stays 0 until all events that caused the interruption are processed.

Only the first three Bits in the Interrupt Identification Register are important.

❑ Bit 0 has the value 0, if an interrupt exists.

❑ Bit 1 and 2 report the Interrupt Code, if Bit 0 = 0.

The Interrupt Code reports information about the interruption type and thus about the priority with which the event is to be treated. Each interrupt type has its own priority. The Interrupt Code 11 has the first priority, the Interrupt Code 10 the second, 01 the third and 00 the fourth priority.

The Interrupt Codes have the following interrupt types

| **Code** | **Priority** | **Typ** |
|---|---|---|
| 11 | 1 | Receiver error or break signal is received |
| 10 | 2 | Character was received |
| 01 | 3 | Transmitter Holding Register empty |
| 00 | 4 | Change in modem status |

For the interrupt mode to be possible, the initialization must occur with the following 6 steps.

**1.** Initialize Baud Rate Divisor LSB and MSB

**2.** Set the transmission characteristics in the Line Control Register (Data Bits, Stop Bits etc.)

**3.** Set Modem Control Register such that the interrupt mode is possible (OUT2=1)

**4.** Define the events in the Interrupt Enable Register which should be monitored

**5.** Install the Interrupt Routine on the appropriate interrupt vector

**6.** Free the appropriate IRQ line in the 8259 Interrupt Controller (e.g. Bit 4=0, für IRQ4 und COM1)

Assuming the *Interrupt Enable Register* was set up on the interrupt type 10 and the OUT2 line of the *Modem Control Register* and also the 8259 IC are activated, then the following Interrupt Routine is able, after an interruption from the serial interface, to file the received character in a designated buffer.

Example:

```
      TITLE COMISR
      LOCALS

DATA  SEGMENT  WORD PUBLIC

      ASSUME DS:DATA

      WrPtr  dw 0
      Buffer db 32 dup (?)

DATA  ENDS

IOReg EQU  3F8h              ; COM1
```

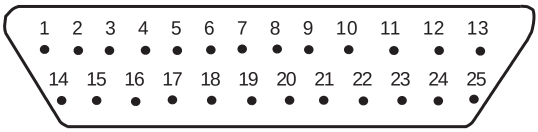
```
CODE  SEGMENT   BYTE PUBLIC

       ASSUME  CS:CODE
ISR    PROC FAR
       PUSH AX            ; save register
       PUSH BX
       PUSH DS
       MOV  AX,SEG Data  ; Set data segment of
                         ; the receiving buffer
       MOV  DS,AX        ; as new DS
       MOV  BX,WrPtr     ; fetch write pointer
       MOV  DX,IOReg     ; Port address of
                         ; serial
                         ; interface
       IN   AL,DX        ; fetch character
       MOV  Buffer [BX],AL ; file character in
                         ; buffer
       INC  BX           ; inc. write counter
       MOV  AL,020h      ; load EOI value
       OUT  20h,AL       ; transmit EOI to 8259
       POP  DS           ; recover register
       POP  BX
       POP  AX
       IRET              ; return to interrupted
                         ; program
  ISR  ENDP
  CODE ENDS
       END
```

This interrupt routine is quite easy in its operation since it only shows one example how an interrupt routine can process the interruption of the 8250. It can be easily extended to intercept the overflow of the buffer, to execute flow control, to be able to process other events, variable transmit and receive buffers...

# Pin configuration of the RS232-Interface

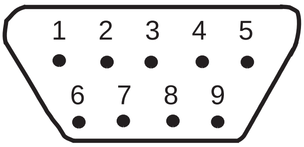
## DB 25 Connector plug



The pin configuration for the DB25 connector plug is as follows:

| Pin | DB25 |
|---|---|
| 1 | |
| 2 | TXD |
| 3 | RXD |
| 4 | RTS |
| 5 | CTS |
| 6 | DSR |
| 7 | GND |
| 8 | DCD |
| 9 | |
| 20 | DTR |
| 22 | RI |

## DB 9 Connector plug

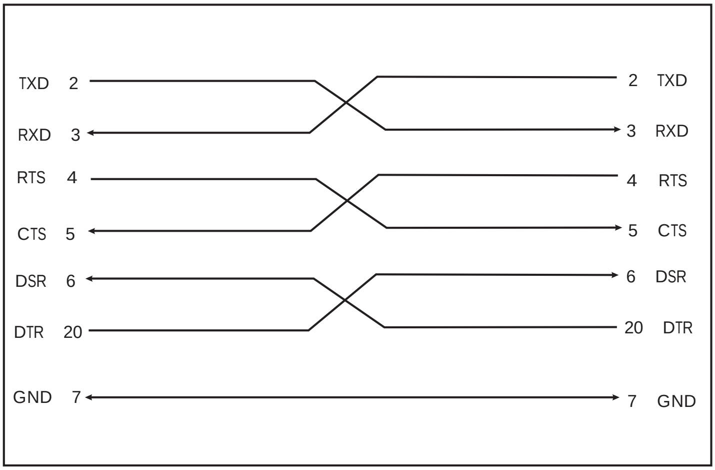


The pin configuration of the DB9 connector plug is as follows:

| Pin | DB9 |
|---|---|
| 1 | DCD |
| 2 | RXD |
| 3 | TXD |
| 4 | DTR |
| 5 | GND |
| 6 | DSR |
| 7 | RTS |
| 8 | CTS |
| 9 | RI |

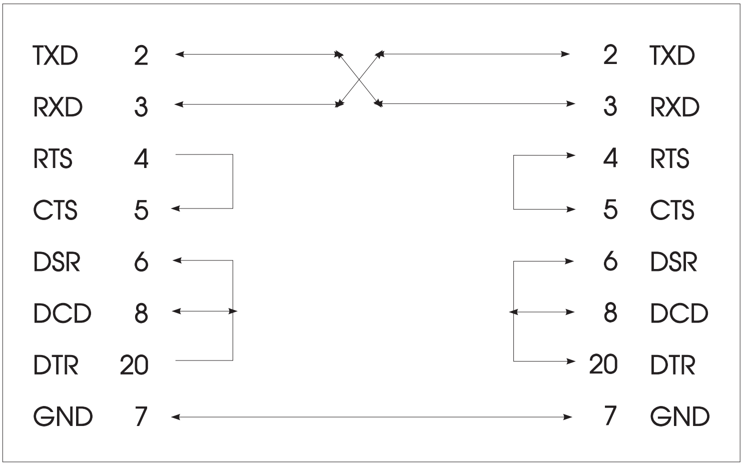
The abreviations used are explained in the glossary.

## NULL Modem Connection with DB25



## 3-wire NULL Modem with DB25

**Important:** If only a 3-wire connection is used then the modem line should be short circuited as depicted, otherwise blind interrupts for these lines will result, very often through interfering signals, and unnecessary load will thereby be produced which can negatively influence the data throughput.


TXD 2
RXD 3
RTS 4
CTS 5
DSR 6
DCD 8
DTR 20
GND 7
2 TXD
3 RXD
4 RTS
5 CTS
6 DSR
8 DCD
20 DTR
7 GND