TFH-Berlin

Einführung Mikrocomputer

EMC 45

Teil 8: Schnittstellenbaustein UART

Studiengang Technische Informatik (TI)
Prof. Dr.-Ing. Alfred Rożek

nur für Lehrzwecke
Vervielfältigung nicht gestattet

Schnittstellenbaustein UART

TFH-Berlin

- **Schnittstellenbaustein UART (Universal Asynchronous Receiver Transmitter)**
  - Versionen des UART (8250, 16450 und 16550)
  - Pinbelegung
  - Interne Struktur des UART
  - Integration in ein Mikrocomputersystem
  - Anwendungsbeispiel
  - RS-232C Level Converter
- **Programmierung des UART**
  - Zugriff mittels DOS
  - Zugriff mittels BIOS
  - Direkter Zugriff mittels Registerprogrammierung
- **Anwendungsbeispiele ADNP (Advanced DIL/NetPC)**

UART UART - Universal Asynchronous Receiver Transmitter

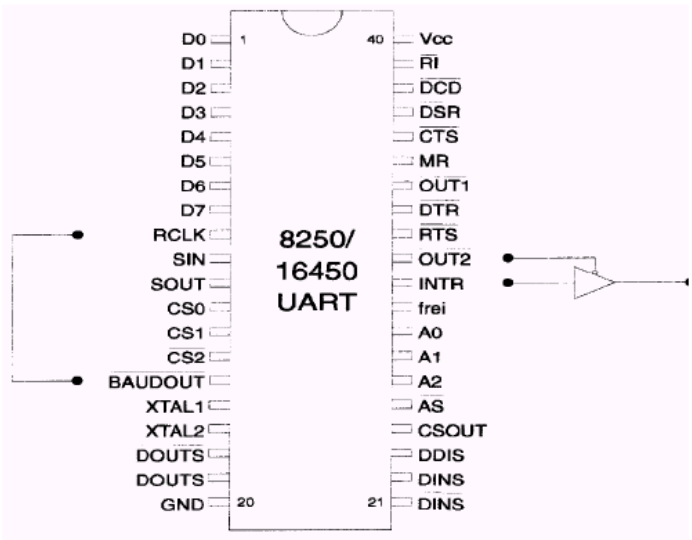
TFH-Berlin

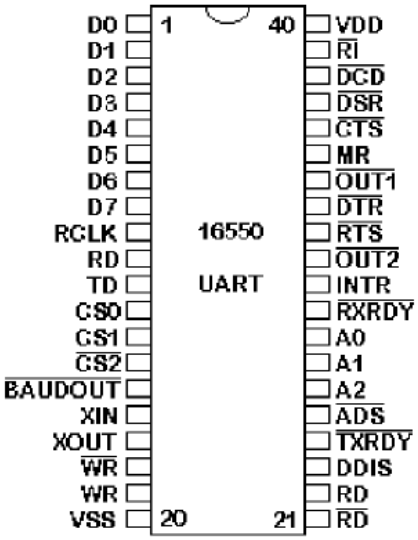
Schnittstellenbaustein für die serielle Eingabe (Receiver) und Ausgabe (Transmitter) von Zeichen. Der Baustein verwaltet die Codierung und Decodierung von (parallelen) Zeichen in einen seriellen Datenstrom, die Paritätsprüfung und überwacht die seriellen Start- und Stoppsignale bei der asynchronen Datenübertragung.

8250	Erster UART der Serie. Arbeitet bis 9.600 Bit/s
8250A	Busseitig schnellere Version des 8250 (bei seinerzeit schnellen Prozessoren (2 MHz Taktfrequenz und mehr, mussten z. B. 3 NOP – Befehle zwischen zwei Buszugriffen von der CPU eingefügt werden). Interrupt-Bug des 8250 behoben.
8250B	Fast identisch zum 8250 UART. Interrupt-Bug des 8250 aus „Kompatibilitätsgründen“ wieder eingebaut!
16450	Eingesetzt in AT's (Buszugriffszeiten verbessert). Arbeitet bis 38.400 Bit/s. Heute noch am Markt. Softwareseitig identisch mit dem 8250.
16550	Erste Generation gepufferter UARTs (FIFO). FIFO umfasst 16 Byte buffer, fehlerhafter Chip, ersetzt durch 16550A.
16550A	Heute üblicher UART, geeignet für Hochgeschwindigkeits-Modems, 28.000 Bit/s und mehr. FIFOs funktionieren.
16650	Sehr aktueller UART. 32 Byte FIFO, programmierbare XON / XOFF Zeichen, unterstützt Power Management.
16750	Enthält 64 Byte FIFO.

UART 8250/16450 und 16550 - Anschlussbelegung

TFH-Berlin

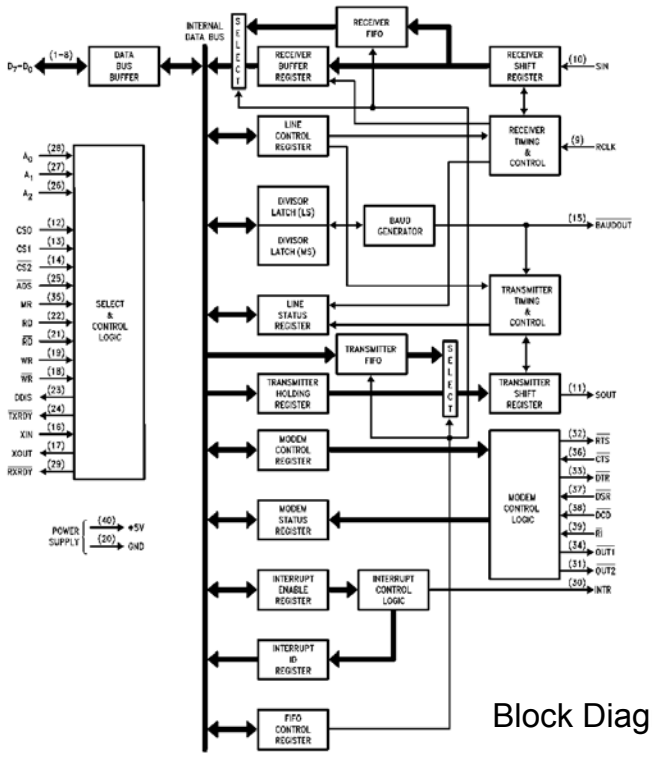




Üblicherweise sind BAUDOUT und RCLK miteinander verbunden, Sender und Empfänger arbeiten also mit derselben Baudrate. Der INTR-Ausgang wird durch ein Gatter mit OUT2 verknüpft (16550 = UART mit FIFO)

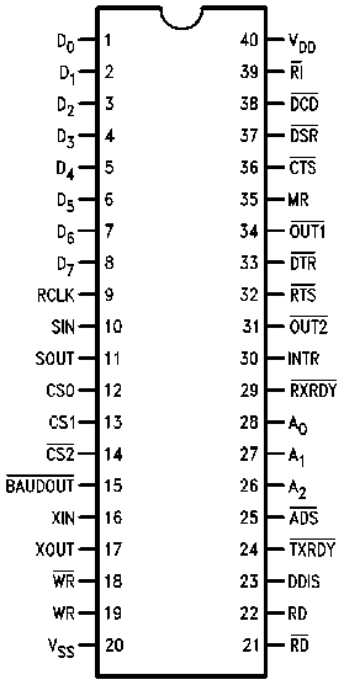
UART 16550

TFH-Berlin



Block Diagram

Dual-In-Line Package

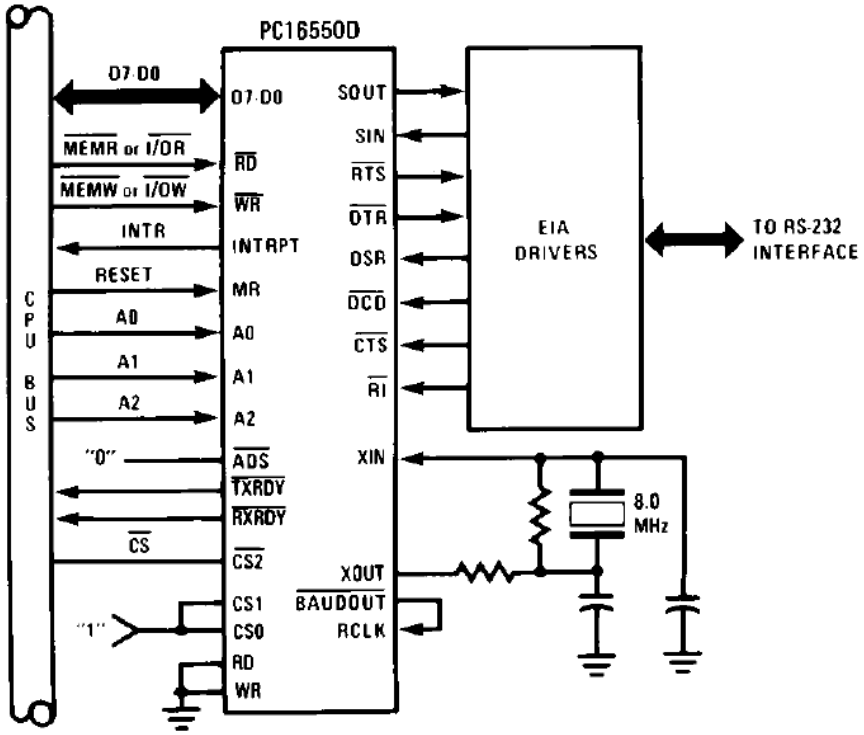


Quelle: National Semiconductor, Datasheet PC16550D

© Prof. Dr.-Ing. Alfred Rożek

UART 16550 Basic Konfiguration

TFH-Berlin



Integration in ein Mikrocomputersystem

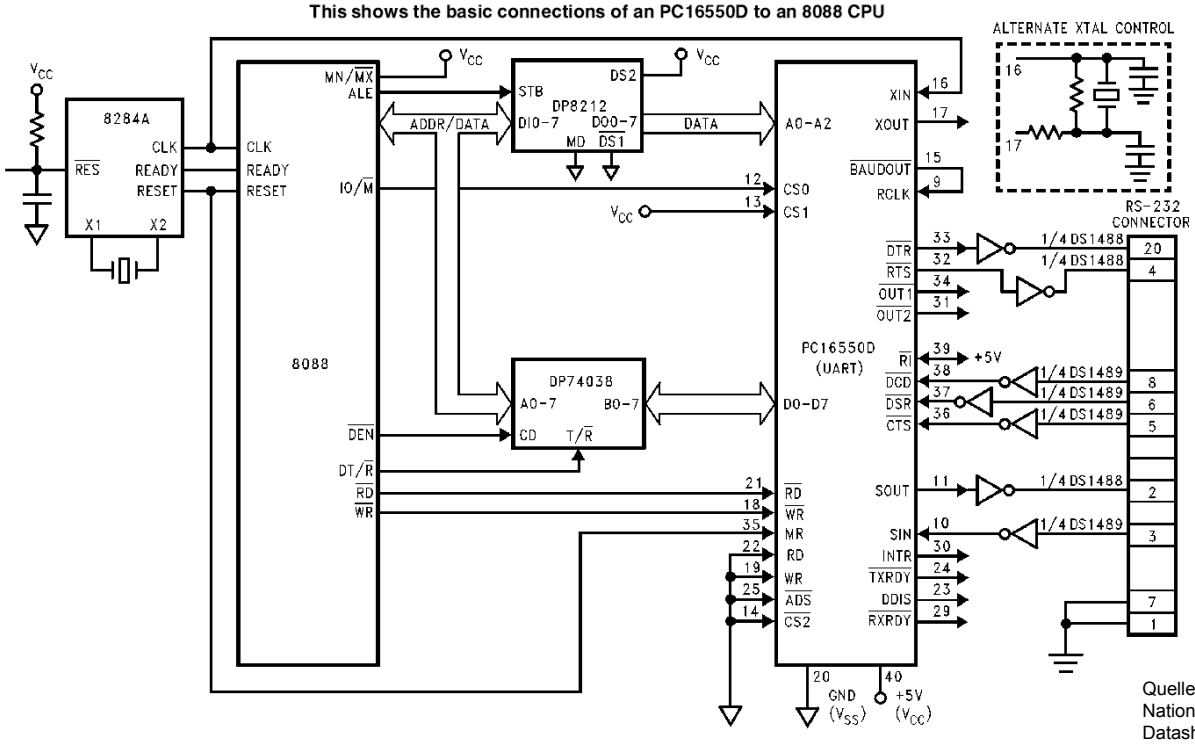
Quelle: National Semiconductor, Datasheet PC16550D

© Prof. Dr.-Ing. Alfred Rożek

Anwendungsbeispiel 16550

TFH-Berlin

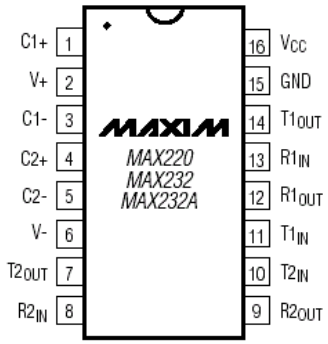
This shows the basic connections of an PC16550D to an 8088 CPU



Quelle:
National Semiconductor,
Datasheet PC16550D

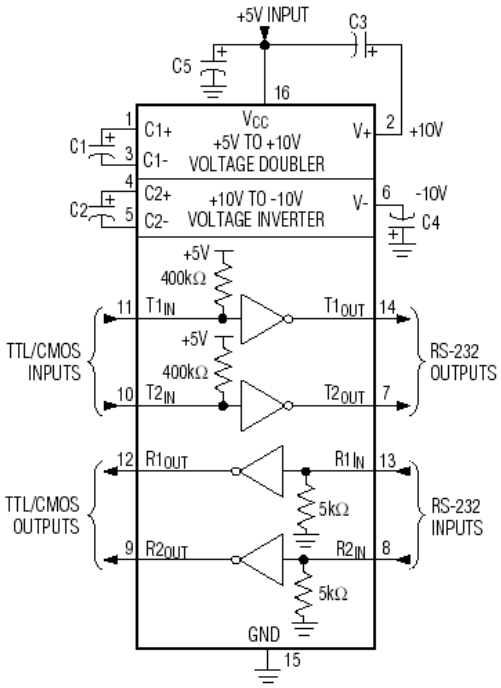
RS-232C Level Converter$_1$

TFH-Berlin



DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



Quelle: Maxim Datasheet MAX220-MAX249

RS-232C Level Converter$_2$

TFH-Berlin

Two common RS-232 Level Converters are the 1488 RS-232 Driver and the 1489 RS-232 Receiver. Each package contains 4 inverters of the one type, either Drivers or Receivers. The driver requires two supply rails, +7.5 to +15v and -7.5 to -15v. As you could imagine this may pose a problem in many instances where only a single supply of +5V is present.

However the advantages of these I.C's are they are cheap.

Another device is the MAX-232. It includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This I.C. also includes two receivers and two transmitters in the same package. This is handy in many cases when you only want to use the Transmit and Receive data Lines. You don't need to use two chips, one for the receive line and one for the transmit.

However all this convenience comes at a price, but compared with the price of designing a new power supply it is very cheap. There are also many variations of these devices.

Quelle: Maxim Datasheet MAX220-MAX249

Programmierung der seriellen Kommunikation

TFH-Berlin

Möglichkeiten der Programmierung der seriellen Kommunikation

- mit Hilfe der **DOS / BIOS-Funktionen** des Betriebssystem. Da die **DOS und BIOS-Funktionen (INT 14h)** für die serielle Übertragung jedes einzelnen Zeichens den gesamten Hardware-Handshake-Zyklus der Kommunikation zwischen DTE (Rechner) und DCE (Modem) durchführen, ist die Übertragung mit diesen Funktionen vergleichsweise langsam (max. 9600 Bd) und erfordert unbedingt ein universelles Nullmodemkabel zwischen den beiden Rechnern (eine einfache 3-Draht-Verbindung reicht nicht aus).
  Die Interrupt-Betriebsart wird durch DOS und BIOS nicht unterstützt
- direkte Programmierung des seriellen **Schnittstellenbausteins UART** ohne Betriebssystemfunktionen benötigt keinen Steuersignal-Hardware-Handshake (Software-Handshake möglich) und damit auch kein Nullmodemkabel (3-Draht-Verbindung reicht aus) und somit kann eine sehr schnelle serielle Übertragung, auch unter Verwendung der Interrupt-Betriebsart, aufgebaut werden.

DOS-Funktionen

TFH-Berlin

Da die **DOS-Funktionen (INT 21H)** für die serielle Übertragung jedes einzelnen Zeichens den gesamten Hardware-Handshake-Zyklus der Kommunikation zwischen DTE (Rechner) und DCE (Modem) durchführen, ist die Übertragung mit diesen Funktionen vergleichsweise langsam und aus historischen Gründen auf max. 9.600 Bit/s begrenzt. Sie erfordert unbedingt ein **Nullmodemkabel** zwischen den beiden Rechnern (eine einfache 3-Draht-Verbindung reicht nicht aus).

Das Betriebssystem DOS stellt für die Programmierung der seriellen Schnittstelle nur zwei spezielle Funktionen zur Verfügung. Mit ihnen kann nur die reservierte E/A-Einheit AUX angesprochen werden, die standardmäßig mit dem Gerät COM1, d.h. mit der seriellen Schnittstelle Nummer eins, verbunden ist.

Zeichen von der seriellen Schnittstelle AUX einlesen

```
Aufruf:        AH = 03H ;Funktionscode
Rückkehr:      AL = empfangenes Zeichen
```

Zeichen über die serielle Schnittstelle AUX ausgeben

```
Aufruf:        AH = 04H
               DL = zu sendendes Zeichen
```

Als Default-Werte für die Schnittstellenparameter gibt DOS (meist) folgende Werte vor:
2400 Baud, 8 Datenbit, 1 Stopbit, kein Paritätsbit. Andere Werte können im Kommando-Interpreter durch das Kommando MODE eingestellt werden. Die DOS–Funktionen sind für Kommunikation nur bedingt brauchbar („Aufhängen“ des Rechners oder „verlorene“ Zeichen beim Lesen wg. ungepuffertem Betrieb). Außerdem sind die „High-Level“-Funktionen 3FH, 30H (Datei lesen/schreiben) anwendbar.

Beispiele: Zugriffe unter DOS

TFH-Berlin

Funktion 03H - Zeichen von serieller Schnittstelle einlesen (AUX = COM1)

```
MOV AH,03H
INT 21H          ;Zeichen in AL - Wartet bis Zeichen eintrifft !
```

Funktion 04H - Zeichen über die serielle Schnittstelle ausgeben

```
MOV AH,04H
MOV DL,'A'
INT 21H
```

Die beiden folgenden Funktionen verwenden Handles und können Zeichenketten lesen / schreiben. AUX hat standardmäßig Handle 03. Alle anderen COMx sind zuvor mit Funktion 3DH (als Datei) zu öffnen.

Funktion 3FH – Datei / Einheit lesen

```
MOV AH,3FH       ;Funktionscode (DS = Datensegment des Puffers)
MOV BX,03H       ;Handle
MOV CX,20        ;20 Zeichen lesen
MOV DX,OFFSET PUFFER ;Datenpuffer, hier 20 Byte gross
INT 21H ;Rueckkehr: CY=0: O.K.; CY=1: Fehler, Code in AX
```

Funktion 40H – Datei / Einheit schreiben

```
MOV AH,40H
MOV BX,HANDLE
MOV CX,01H ;1 Zeichen schreiben
MOV DX,OFFSET PUFFER ;Datenpuffer, hier 20 Byte gross
INT 21H;Rueckkehr: CY=0: O.K.; CY=1: Fehler, Code in AX
```

BIOS- und Portadressen, IRQ-Belegung im PC

TFH-Berlin

Adresse	Größe	Aufbau	Inhalt	Bedeutung
		76543210		
40:00	Wort		Basisadresse	COM1
40:02	Wort		Basisadresse	COM2
40:04	Wort		Basisadresse	COM3
40:06	Wort		Basisadresse	COM4
40:11	Byte	...xxx.	installierte Hardware	Zahl der seriellen Schnittstellen
				(000=0,001=1,010=2,011=3,100=4)
40:7C	Byte		Zeitüberschreitung COM1	Time-out-Wert in Sekunden
40:7D	Byte		Zeitüberschreitung COM2	Time-out-Wert in Sekunden
40:7E	Byte		Zeitüberschreitung COM3	Time-out-Wert in Sekunden
40:7F	Byte		Zeitüberschreitung COM4	Time-out-Wert in Sekunden

BIOS-Datenbereich für die serielle Schnittstelle

PORT	Adresse	IRQ
COM1	3F8H	4
COM2	2F8H	3
COM3	3E8H	4
COM4	2E8H	3

PC-Basisadressen und Interrupt-Kanäle für die RS-232 Schnittstelle

BIOS-Funktionen$_1$

TFH-Berlin

Bei der Nutzung der BIOS-Funktionen ist ein 3-Draht-Nullmodemkabel mit seinen gebrückten Steuerleitungen am besten geeignet (Terminal-Emulation). Mit den gekreuzt durchverbundenen Steuersignalen des universellen Nullmodemkabels können bei Verwendung der BIOS-Funktionen Probleme auftreten!

Insgesamt stehen sieben Funktionsaufrufe (INT 14H) zur Verfügung:

Register	Aufrufwert	Rückkehrwert
Funktion 00H - Serielle Schnittstelle initialisieren		
AH	00H	Übertragungsstatus[1)]
AL	Parameterbyte[1)]	Modemstatus[1)]
DX	Schnittstellennummer	
Funktion 01H - Zeichen über serielle Schnittstelle ausgeben		
AH	01H	Übertragungsstatus[1)]
AL	Zeichen	Modemstatus[1)]
DX	Schnittstellennummer	

1) Erläuterung auf gesondertem Blatt

Schnittstellennummer: 0 = COM1 = AUX, 1 = COM2, 2 = COM3, 3 = COM4

BIOS-Funktionen$_2$

TFH-Berlin

Register	Aufrufwert	Rückkehrwert
Funktion 02H - Zeichen über serielle Schnittstelle einlesen		
AH	02H	Übertragungsstatus[1)]
AL		Zeichen
DX	Schnittstellennummer	
Funktion 03H - Status der seriellen Schnittstelle ermitteln		
AH	03H	Übertragungsstatus[1)]
AL		Modemstatus[1)]
DX	Schnittstellennummer	

1) Erläuterung auf gesondertem Blatt

Schnittstellennummer: 0 = COM1 = AUX, 1 = COM2, 2 = COM3, 3 = COM4

BIOS-Funktionen$_3$

TFH-Berlin

Register	Aufrufwert	Rückkehrwert
Funktion 04H - Serielle Schnittstelle erweitert initialisieren		
AH	04H	Übertragungsstatus[1)]
AL	0 = kein Break; 1 = Break	Modemstatus[1)]
BH	Parität	
BL	Stoppbit	
CH	Datenbit	
CL	Baudrate	
DX	Schnittstellennummer	

1) Erläuterung auf gesondertem Blatt

Parität: 0 = keine, 1 = ungerade, 2 = gerade, 3 = Mark, 4 = Space
Stoppbit: 0 = ein Stoppbit, 1 = zwei Stoppbit, 1½ bei 5 Datenbit
Datenbit: 0 = 5, 1 = 6, 2 = 7, 3 = 8 Datenbits
Baudrate: 00 = 110, 01 = 150, 02 = 300, 03 = 600, 04 = 1200
05 = 2400, 06 = 4800, 07 = 9600, 08 = 19200
Break: Break erzeugt an SOUT des UART ein logisches NULL, d.h. es wird für den Empfänger eine Unterbrechung der Verbindung simuliert
Schnittstellennummer: 0 = COM1 = AUX, 1 = COM2, 2 = COM3, 3 = COM4

BIOS-Funktionen$_4$

TFH-Berlin

Register	Aufrufwert	Rückkehrwert
Funktion 05H, Unterfunktion 00H - Modemsteuerregister lesen		
AH	05H	
AL	00H	
BL		Modemsteuerregister[1)]
DX	Schnittstellennummer	
Funktion 05H, Unterfunktion 01H - Modemsteuerregister schreiben		
AH	05H	Übertragungsstatus[1)]
AL	01H	Modemstatus[1)]
BL	Modemsteuerregister[1)]	
DX	Schnittstellennummer	

[1)] Erläuterung auf gesondertem Blatt

BIOS-Fktn: Übertragungsstatus- und Modemstatusbyte

TFH-Berlin

Übertragungsstatusbyte

7	6	5	4	3	2	1	0
TIM	TSR	THR	BRK	FRM	PAR	ÜBL	EMP

TIM: Zeitüberschreitung (Time-out)
1=Zeitüberschreitungsfehler
0=kein Fehler

TSR: Senderschieberegister
(Transmitter-Shift-Register)
1=frei 0=belegt

THR: Senderhalteregister
(Transmitter-Hold-Register)
1=frei 0=belegt

BRK: Unterbrechung (Break)
des Übertragungswegs
1=aufgetreten 0=kein Unterbrechung

FRM: Protokoll- (Framing-) Fehler
1=aufgetreten 0=kein Framing-Fehler

PAR: Paritätsfehler
1=aufgetreten 0=kein Paritätsfehler

ÜBL: Überlauffehler (Overrun-Fehler)
1=aufgetreten 0=kein Überlauffehler

EMP: Empfangsdaten
1=bereit 0=nicht vorhanden

Modemstatusbyte

7	6	5	4	3	2	1	0
DCD	RI	DTR	CTS	DDC	DRI	DDT	DCT

DCD: Datenträgersignal(Data Carrier Detect)
1=erfaßt
0=nicht erfaßt

RI: Läuten (Ring-Indikator)
1=erfaßt 0=nicht erfaßt

DTR: Empfängerbereitschaft (Data Terminal Ready)
1=bereit 0=nicht bereit

CTS: Sendebereitschaft (Clear to Send)
1=bereit 0=nicht bereit

DDC: Datenträgersignaländerung (Delta DCD)
1=erfaßt 0=nicht erfaßt

DRI: Änderung des Läutsignals erfaßt (Delta RI)
1=erfaßt 0=nicht erfaßt

DDT: Änderung der Empfängerbereitschaft
(Delta DTR)
1=erfaßt 0=nicht erfaßt

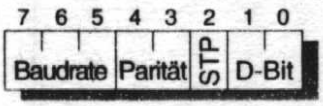
DCT: Änderung der Sendebereitschaft (Delta CTS)
1=erfaßt 0=nicht erfaßt

Quelle: Messmer, PC-Hardware, 1998

BIOS-Fktn: Parameterbyte

TFH-Berlin

Parameterbyte




```
Baudrate : 000=110 Baud    001=150 Baud    010=300 Baud    011=600 Baud
           100=1200 Baud   101=2400 Baud   110=4800 Baud   111=9600 Baud
Parität: 00=keine   01=ungerade   10=keine   11=gerade
STP: Stopbit
     0=1 Stopbit    1=2 Stopbit
D-Bit: Datenbit
     10=7 Bit       11=8 Bit
```

Quelle: Messmer, PC-Hardware, 1998

Beispiele für BIOS-Funktionen

TFH-Berlin

Status der ersten seriellen Schnittstelle ermitteln

```
MOV AH,03H    ;Funktionsnummer: Status ermitteln
MOV DX,00     ;Den Status von COM1 ermitteln
INT 14H       ;BIOS aufrufen
Rückkehr:
(AH = Übertragungsstatus)
(AL = Modemstatus)
```

COM2 initialisieren mit 4800 Baud, gerade Parität, 2 Stoppbit, 7Datenbit

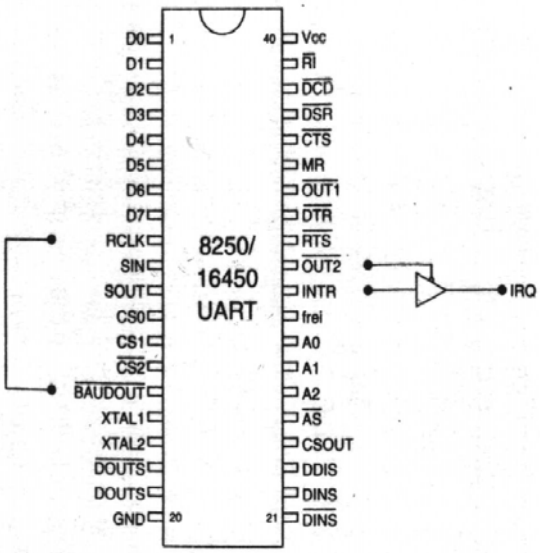
```
MOV AH,00     ;Funktionsnummer: Schnittstelle initialisieren
MOV AL,11011110B ;Parameterbyte
MOV DX,01h    ;COM2 wird angesprochen
INT 14H       ;BIOS aufrufen
Rückkehr: ...
```

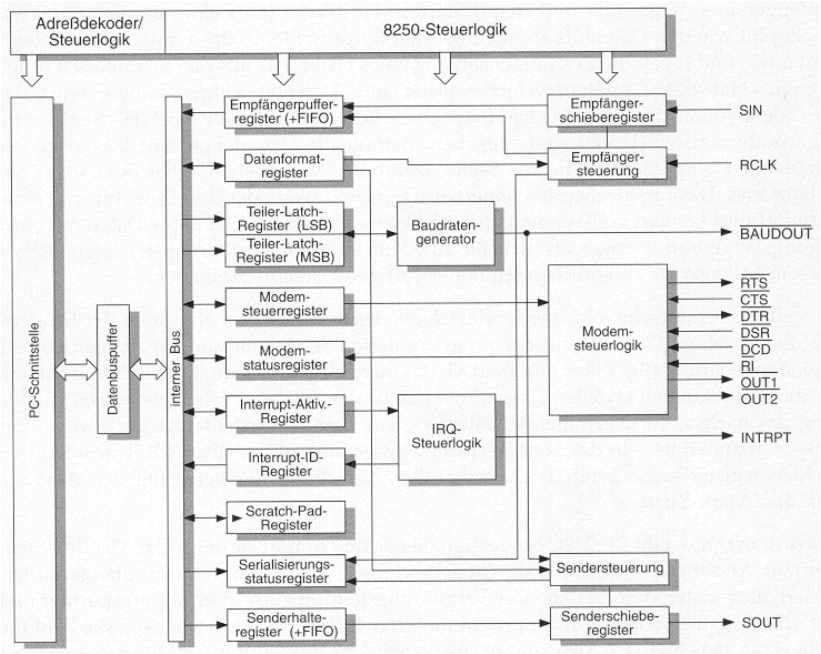
Ein Zeichen über das BIOS von COM3 einlesen

```
MOV AH,02     ;Funktionsnummer: Zeichen einlesen
MOV DX,02h    ;Das Zeichen von COM3 einlesen
INT 14h       ;Funktion wartet und übergibt Zeichen im Register
Rückkehr: ...
```

UART
Universal Asynchronous Receiver and Transmitter

TFH-Berlin





Anschlußschema des UART 8250/16450/16550

Blockdiagramm des UART 8250/16450/16550

Quelle: Messmer, PC-Hardware, 1998

UART$_1$
Zugriff auf die Register des UART

TFH-Berlin

Die 12 Register eines UART sind über verschiedene Ports zu erreichen, die sich jeweils an der Basis-Adresse der zugehörigen seriellen Schnittstelle orientieren. Im PC werden COM1 und COM2 mit den Basisadressen 3F8H und 2F8H bzw. COM3 und COM4, soweit präsent, mit 3E8H und 2E8H angesprochen.

Um die jeweils aktuelle Basisadresse zu erhalten, empfiehlt es sich, die vier BIOS-Variablen abzufragen, in denen die Basisadressen der maximal vier vom BIOS unterstützten seriellen Schnittstellen verzeichnet sind.

Da zwei Portadressen des UART (Basisport + 0 und Basisport + 1) von mehreren Registern belegt werden, wird das höchstwertige Bit im Line-Control-Register (DLAB) für die Unterscheidung der verschiedenen Register benutzt. Dieses Bit ist also vor dem Zugriff auf eines dieser Register entsprechend zu laden.

UART$_2$
Die internen Register der UARTs

TFH-Berlin

Register (deutsch)	Register (englisch)	Kürzel	Offset	DLAB	A2	A1	A0
Empfängerpufferregister (read)	Receiver Buffer Register	RBR	00H	0	0	0	0
Senderhalteregister (write)	Transmitter Holding Register	THR	00H	0	0	0	0
Interrupt-Aktivierungsregister (enable)	Interrupt Enable Register	IER	01H	0	0	0	1
Interrupt-Identifizierungsregister (read)	Interrupt Ident. Register	IIR	02H	X	0	1	0
FIFO-Controlregister (write)	FIFO Control Register	FCR	02H	X	0	1	0
Datenformatregister	Line Control Register	LCR	03H	X	0	1	1
Modemsteuerregister	Modem Control Register	MCR	04H	X	1	0	0
Serialisierungsstatusregister	Line Status Register	LSR	05H	X	1	0	1
Modemstatusregister	Modem Status Register	MSR	06H	X	1	1	0
Scratch-Pad-Register	Scratch Register	SCR	07H	X	1	1	1
Teiler-Latch-Register (LSB)	Divisor Latch LSB	DLL	00H	1	0	0	0
Teiler-Latch-Register (MSB)	Divisor Latch MSB	DLM	01H	1	0	0	1

DLAB
Divisor Latch Access Bit (Teiler-Latch-Zugriffsbit) im Datenformatregister

Registeradressen des UART 8250/16450/16550

UART$_3$
Zusammenfassung der Register (16550) und Belegung

TFH-Berlin

Bit No.	Register Address											
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Reg-ister	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.
Note 2: These bits are always 0 in the 16450 Mode.

UART$_4$
Empfängerpuffer-, Senderhalte- und Interrupt-Aktivierungsregister

TFH-Berlin

Empfängerpufferregister

7							0	
E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	RBR

Senderhalteregister

7							0	
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	THR

E7..E0: Empfangsbit 7 bis 0S7..S0: Sendebit 7 bis 0 **(Offset 00h)**

7	6	5	4	3	2	1	0
0	0	0	0	SINP	ERBK	TBE	RxRD

IER: Interrupt-Aktivierungsregister (Offset 01h)

SINP: RS-232-Eingabe
1=Interrupt bei Zustandsänderung einer RS-232-Eingangsleitung
0=kein Interrupt

ERBK: Fehler und Break (Error & Break)
1=Interrupt bei Paritäts-, Überlauf-, Framing-Fehler oder BREAK
0=kein Interrupt

TBE: Übertragungspuffer leer (Transmitter-Buffer Empty)
1=Interrupt, wenn Übertragungspuffer leer 0=kein Interrupt

RxRD: Daten empfangen (Received Data Ready)
1=Interrupt, wenn ein Byte im Empfängerpufferregister bereitsteht
0=kein Interrupt

Quelle: Messmer, PC-Hardware, 1998

UART$_5$
Interrupt-Identifizierungs- und Datenformatregister

TFH-Berlin

7				3	2	1	0
0	0	0	0	0	ID1	ID0	PND

IIR: Interrupt-Identifizierungsregister (Offset 02h)

ID1, ID0: Identifizierungsbit
00=Änderung eines RS-232-Eingangssignals (Priorität 3)
01=Übertragungspuffer leer (Priorität 2) 10=Daten empfangen (Priorität 1)
11=Serialisierungsfehler oder BREAK (Priorität 0)
0=höchste Priorität 3=niedrigste Priorität

PND: Pending-Bit
1=kein Interrupt anhängig 0=Interrupt anhängig, ID1, ID0 identifizieren Ursache

7	6	5	4	3	2	1	0
DLAB	BRK	Parität			STOP	D-Bit	

LCR: Datenformatregister (Offset 03h)

DLAB: Divisor-Latch Acces-Bit
1=Zugriff auf Teiler-Latch
0=Zugriff auf Empfangs-/Senderegister und Interrupt-Aktivierungsregister

BRK: BREAK
1=ein 0=aus

Parität: 000=keine 001=ungerade 011=gerade 101=mark 111=space

STOP: Anzahl der Stopbit
1=2 Stopbit 0=1 Stopbit

D-Bit: Anzahl der Datenbit
00=5 Datenbit 01=6 Datenbit 10=7 Datenbit 11=8 Datenbit

Quelle: Messmer, PC-Hardware, 1998

UART$_6$
Modemsteuer- und Serialisierungsstatusregister

TFH-Berlin

7	6	5	4	3	2	1	0
0	0	0	LOOP	OUT2	OUT1	RTS	DTR

MCR: Modemsteuerregister (Offset 04h)

```
LOOP:  Rückkoppelungsprüfung (Loopback-Prüfung)
       1=aktiv      0=Normalbetrieb
OUT2:  Vielzweckausgang 2 (General-Purpose-Output 2): im PC Master-Interrupt-
       Aktivierungsbit
       1=aktiv     0=deaktiviert
OUT1:  Vielzweckausgang 1 (General-Purpose-Output 1)
       1=aktiv     0=deaktiviert
RTS:   Request-To-Send
       1=aktiv     0=deaktiviert
DTR:   Data-Terminal-Ready
       1=aktiv     0=deaktiviert
```

7	6	5	4	3	2	1	0
0	TXE	TBE	BREK	FRMF	PARF	ÜBLF	RxRD

LSR: Serialisierungsstatusregister (Offset 05h)

```
TXE:   Sender leer (Transmitter-Empty)
       1=kein Byte im Senderhalteregister und Senderschieberegister
       0=ein Byte im Senderhalteregister oder Senderschieberegister
TBE:   Senderhalteregister leer (Transmitter-Buffer-Empty)
       1=kein Byte im Senderhalteregister  0=ein Byte im Senderhalteregister
BREK:  Break
       1=erfaßt                 0=kein Break
FRMF:  Framing-Fehler
       1=Fehler aufgetreten     0=kein Framing-Fehler
PARF:  Paritätsfehler
       1=Fehler aufgetreten     0=kein Paritätsfehler
ÜBLF:  Überlauffehler
       1=Fehler aufgetreten     0=kein Überlauffehler
RxRD:  Empfangsdaten bereit
       1=Empfangsdaten im Empfängerpufferregister   0=keine Empfangsdaten
```

Quelle: Messmer, PC-Hardware, 1998

UART$_7$
Modemstatus- und Scratch-Pad-Register

TFH-Berlin

7	6	5	4	3	2	1	0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

MSR: Modemstatusregister (Offset 06h)

```
DCD:   Data-Carrier-Detect
       1=DCD aktiv      0=DCD inaktiv
RI:    Ring-Indicator
       1=RI aktiv       0=RI inaktiv
DSR:   Data-Set-Ready
       1=DSR aktiv      0=DSR inaktiv
CTS:   Clear-To-Send
       1=CTS aktiv      0=CTS inaktiv
DDCD:  Delta-Data-Carrier-Detect
       1=DCD seit letztem Lesevorgang verändert  0=DCD gleich geblieben
       1=DCD seit letztem Lesevorgang verändert  0=DCD gleich geblieben
DRI:   Delta-Ring-Indicator
       1=RI seit letztem Lesevorgang verändert   0=RI gleich geblieben
DDSR:  Delta-Data-Set-Ready
       1=DSR seit letztem Lesevorgang verändert  0=DSR gleich geblieben
DCTS:  Delta-Clear-To-Send
       1=CTS seit letztem Lesevorgang verändert  0=CTS gleich geblieben
```

7	6	5	4	3	2	1	0
SPR_7	SPR_6	SPR_5	SPR_4	SPR_3	SPR_2	SPR_1	SPR_0

SCR: Scratch-Pad-Register (Offset 07h)

```
SPR7...SPR0: Bit 7 bis 0 im Register
```

Quelle: Messmer, PC-Hardware, 1998

Initialisierung des UART

TFH-Berlin

Es sollte bei der Einstellung der Kommunikationsparameter mit der Baudrate begonnen werden, denn beim Beschreiben der dazu nötigen Register setzt der UART die anderen Kommunikationsparameter im Line-Control-Register LCR zurück. Die gewünschte Baudrate wird mit den beiden Registern DLL und DLM eingestellt

$$\text{Baudrate} = \frac{\text{Hauptbezugsfrequenz}}{16 * \text{Teiler}} = \frac{\text{Bezugsfrequenz}}{\text{Teiler}}$$

Hauptbezugsfrequenz = 1,8432 MHz
Bezugsfrequenz = 115.200 Hz
(1,8432 MHz ➔Taktfrequenz des Ur-PC)

Baudrate	Teiler	DLM-Reg.	DLL-Reg.
50	2304	09H	00H
75	1536	06H	00H
110	1047	04H	17H
134,5	857	03H	59H
150	768	03H	00H
300	384	01H	80H
600	192	00H	C0H
1200	96	00H	60H
1800	64	00H	40H
2000	58	00H	3AH
2400	48	00H	30H
4800	24	00H	18H
7200	16	00H	10H
9600	12	00H	0CH
19200	6	00H	06H
38400	3	00H	03H
57600	2	00H	02H
115200	1	00H	01H

Alle weiteren Parameter wie die Wortlänge, die Anzahl der Stopbits und die Verwendung von Paritätsbits werden über das Line-Control-Register LCR eingestellt (durch Beschreiben des LCR). Für die Ermittlung der aktuellen Einstellungen kann das Register aber auch ausgelesen werden.

Beispiel: Initialisierung

TFH-Berlin

Beispiel: COM1 auf 300 Baud einstellen Teiler = 384 = 180H

```
MOV DX,3F8H+3   ;Adresse des Line Control Registers LCR
IN  AL,DX       ;Wert von LCR nach AL einlesen
OR  AL,80H      ;DLAB-Bit setzen
OUT DX,AL       ;LCR schreiben, d.h. ausser DLAB alle Bits erhalten!

MOV DX,3F8h     ;Adresse Divisor-Latch-Registers DLL (Low-Teil)
MOV AL,80H      ;Niederwertiges Teilerbyte nach AL laden
OUT DX,AL       ;Divisor-Latch-Register (Low-Teil) schreiben

INC DX          ;Adresse Divisor-Latch-Registers DLM (High-Teil)
MOV AL,01H      ;Hoeherwertiges Teilerbyte nach AL laden
OUT DX,AL       ;Divisor-Latch-Register (MSB) schreiben

MOV DX,3F8H+3   ;Adresse des Line Control Registers LCR
IN  AL,DX       ;Wert von LCR nach AL einlesen
AND AL,7Fh      ;DLAB-Bit loeschen
OUT DX,AL       ;LCR schreiben
```

Beispiel: Status ermitteln

TFH-Berlin

Das Line-Status-Register LSR liefert Information über den Status des Empfänger- und Senderteils

Beispiel:
Ermitteln, ob ein Datenbyte im Empfängerpufferregister bereitsteht und das Datenbyte ggf. lesen

```
   MOV DX,3F8H+5        ;Adresse Serialisierungsstatusregister LSR
WEITER:                 ;Sprungziel wenn auf Datenempfang gewartet wird
   IN  AL,DX            ;Serialisierungsstatusregister in AL einlesen
   AND AL,01H   ;RxRD = 1?, d.h. stehen Daten bereit?
   JZ  WEITER           ;Sprung, wenn RxRD gelöscht-keine Empfangsdaten
   MOV DX,3F8H          ;Adresse des Empfängerpufferregisters laden
   IN  AL,DX            ;Datenbyte aus Empfängerpufferregister einlesen
WEITER:                 ;Sprungziel wenn bis zum Datenempfang noch
                        ;andere Aufgaben erledigt werden sollen
```


Beispiel: Modem Control- und Modem Status-Register

TFH-Berlin

Beispiel Modem Control Register:
Das Master-Interrupt-Bit setzen, um individuelle Interrupts zuzulassen

```
MOV DX,3F8H+4           ;Adresse des Modemsteuerregisters laden
IN  AL,DX               ;Modemsteuerregister einlesen
OR  AL,00001000B        ;OUT2=Master-Interrupt-Aktivierungsbit
OUT DX,AL               ;Modemsteuerregister schreiben
```

Beispiel Modem Status Register:
Ermitteln, ob sich das Bit DCD (Data Carrier Detect) geändert hat

```
   MOV DX,3F8H+6        ;Adresse des Modemstatusregisters laden
   IN  AL,DX            ;Modemstatusregister nach AL einlesen
   AND AL,00001000B     ;DeltaDCD testen
   JNZ WEITER           ;Sprung, wenn Delta DCD gesetzt
   ;Programm „keine Änderung"
WEITER:
   ;Programm zur Behandlung der Änderung
```

Beispiel: Interrupt-ID-Register

TFH-Berlin

Ermitteln, ob für COM1 Interrupt anhängig ist und Sprung zum Handler entsprechend den Adressen in einem Sprungverteiler:

```
   MOV DX,3FAH                ;Adresse Interrupt-Identifizierungsregister
   IN  AL,DX                  ;Register nach AL einlesen
   AND AL,00000001B           ;Pending-Bit testen - Interrupt anhängig?
   JNZ WEITER                 ;Nein: Sprung zu WEITER, wenn PND = 1
   AND AX,0110B               ;ID-Bits isolieren (verkuerzte Schreibweise!)
   XCHG AX,BX                 ;ID1, ID0 als Index auf Sprungtabelle-->BX
   MOV BX,WORD PTR VERTEILER[BX] ;Zieladresse des Handlers holen
   JMP BX                     ;Sprung zum entsprechenden Handler fuer Interrupt
VERTEILER:                    ;Die Sprungverteiler-Tabelle
   DW HAND0                   ;Adresse Handler 0 (Index 0)
   DW HAND1                   ;Adresse Handler 1 (Index 2)
   DW HAND2                   ;Adresse Handler 2 (Index 4)
   DW HAND3                   ;Adresse Handler 3 (Index 6)
HAND0:
   ;Programm Handler No. 0
   JMP END_HANDLER
HAND1:
   ;Programm Handler No. 1
   JMP END_HANDLER
;Und so weiter
WEITER:
END_HANDLER:
```


FIFO-Steuerregister

TFH-Berlin

Die Nachfolger des 8250/16450 weisen je einen 16Byte großen Empfangs– und Sendepuffer (FIFO) auf, die dem Receiver-Buffer-Register (Empfang) und dem Transmitter-Holding-Register (Senden) vorgeschaltet sind. Um kompatibel zu den Vorgängern zu sein, sind diese internen Puffer zuerst einmal deaktiviert und müssen über das FIFO-Steuerregister eingeschaltet werden.

Das FIFO-Steuer-Register belegt die gleiche Adresse wie das Interrupt-ID-Register. Während das Interrupt-ID-Register aber nur ausgelesen werden kann, lässt sich das bei den 8250-Nachfolgern vorhandene FIFO-Register nur beschreiben.

Zu beachten ist, dass der Empfangspuffer innerhalb einer Interrupt-Service-Routine durch Polling geleert werden muss (Abfrage des Line-Status-Registers und Auslesen des RBR)

Aufbau des FIFO-Steuer-Registers beim 16550 und seinen Nachfolgern (Offset 02 h):

Bit 0 = 1: FIFO-Puffer an
Bit 1 = 1: Empfangspuffer löschen
Bit 2 = 1: Sendepuffer löschen
Bit 3-5 = immer 0
Bit 6,7 Anzahl der Zeichen, nach deren Empfang der UART einen Interrupt auslösen soll

- 00 = nach jedem Zeichen
- 01 = nach 4 Zeichen
- 10 = nach 8 Zeichen
- 11 = nach 14 Zeichen

Beispiel: Interrupt-Service-Routine für UART mit FIFO

TFH-Berlin

```
void interrupt PORT1INT() {
   unsigned char c;
   do {
      c = inportb(PORT1 + 5); //Serialisierungsstatusregister
      if (c & 1) {            //Empfangsdaten bereit?
         buffer[bufferin] = inportb(PORT1); //Daten lesen
         bufferin++;
         if (bufferin == 1024) bufferin = 0;
      }
   } while (c & 1);           //so lange bis FIFO leer ist
   outportb(0x20,0x20);       //EOI an PIC
}
```


Programmierhinweise für die Bausteine 16550 und 16550A

TFH-Berlin

Bei neueren Hardwarebaugruppen ist oft die Funktionalität des 16550 realisiert. Das heißt für die Programmierung eines **Hardware-Handshakes**, dass der im 16550 vorhandene FIFO berücksichtigt werden muss.

Das Problem: Die wesentliche Funktionalität des UART, bei Empfang eines Zeichens das Signal RTS automatisch rückzusetzen, ist für den 16550 so realisiert, dass erst nach dem Füllen des aktuell eingestellten Puffers (meist 4 Byte nach der Initialisierung) das RTS rückgesetzt wird. Es muss dann das Auslesen des Empfangspuffers so implementiert werden, dass die Bytes entsprechend der aktuellen Puffergröße ausgelesen werden.

Für Testzwecke ist es daher sinnvoll, einfach den FIFO zu deaktivieren. Damit verhält sich dann der 16550 wieder wie ein 16450 beziehungsweise 8250 und setzt das Signal RTS nach dem Empfang eines gültigen Zeichens zurück.

Das Deaktivieren des FIFO erfolgt, indem an Offset 2H eine 06H geschrieben wird (Interrupt-Identifizierungs-Register / FIFO-Steuer-Register).

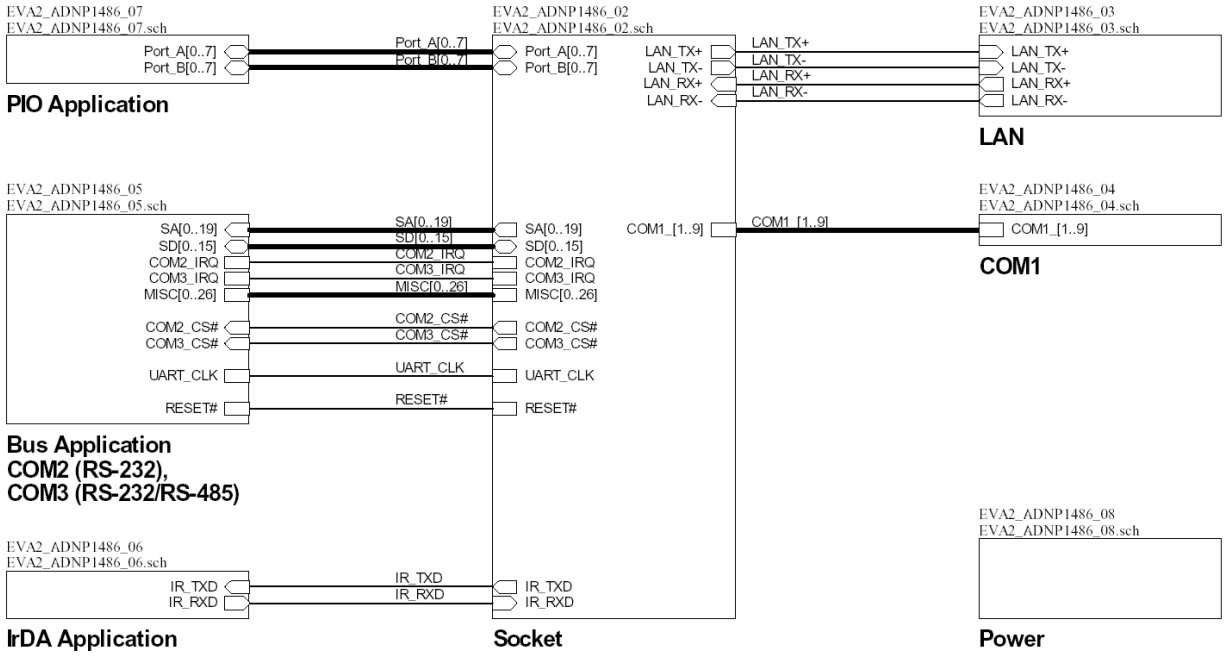
Literaturhinweise:

Drigalsky, Ingo: "Serielle Schnittstellentechnik und Protokoll-Analyzer-Anwendungen: serielle Schnittstellen und deren Protokolle verstehen, aufbauen und testen" IWT-Verlag, 1992

Elsing, Jürgen: "Schnittstellen-Handbuch: verständliche Erläuterung und Benutzung von Centronics, V24, IEC-Bus" IWT-Verlag, 1992

ADNP: Basisboard für ADNP/1486-3V

TFH-Berlin

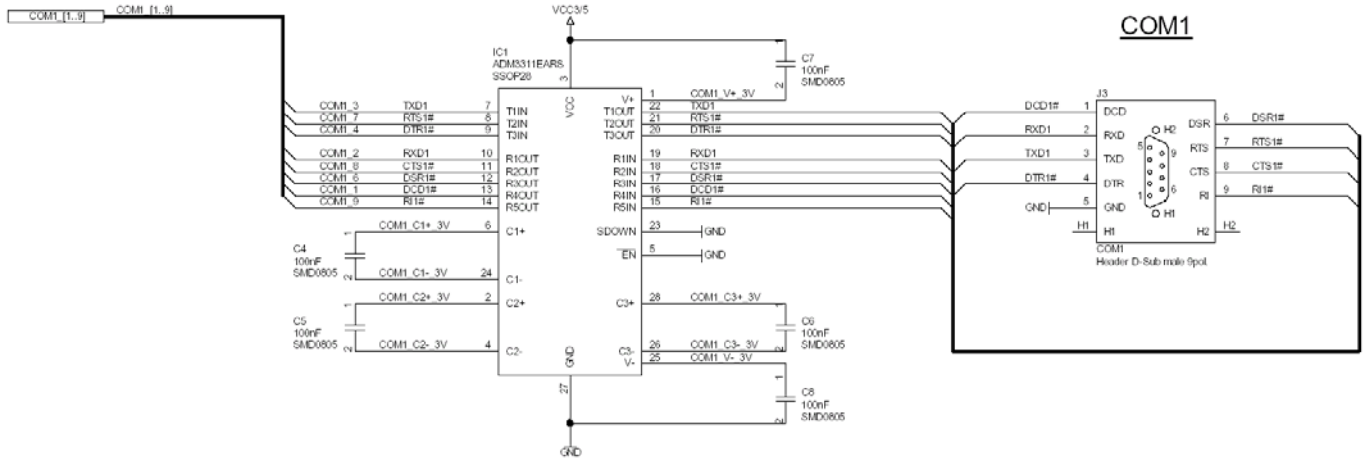


Basisboard basierend auf DNP/EVA2 Rev. 1.1

Quelle: SSV, CDROM V.1.16, USER-DOC\DIAGRAMS\Adnp\DNPEVA2-ADNP1486-P1.pdf

ADNP-Anwendungsbeispiel: COM1

TFH-Berlin

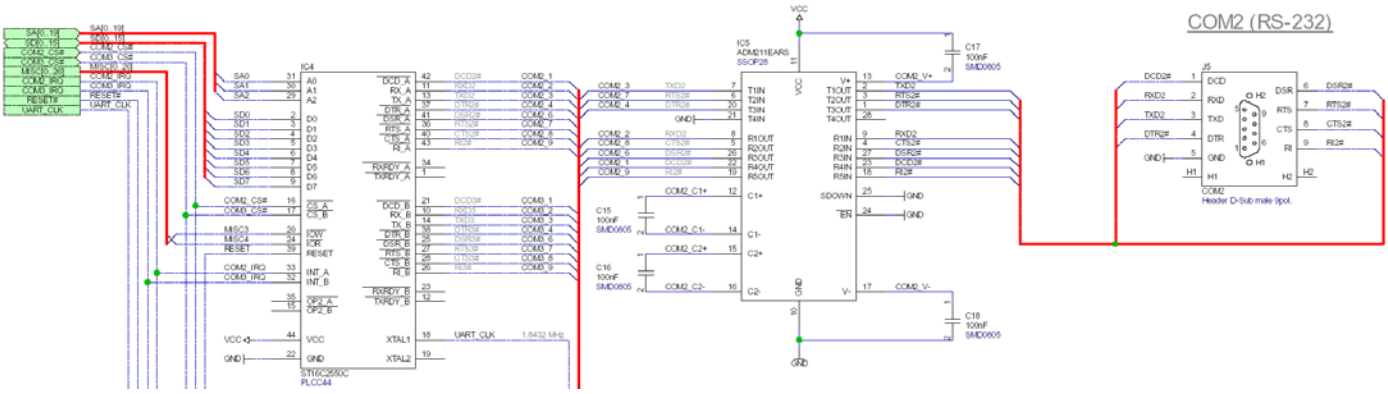


RS-232C-Datenübertragungsinterface des ADNPs

Quelle: SSV, CDROM V.1.16, USER-DOC\DIAGRAMS\Adnp\DNPEVA2-ADNP1486-P4.pdf

ADNP-Anwendungsbeispiel: COM2

TFH-Berlin

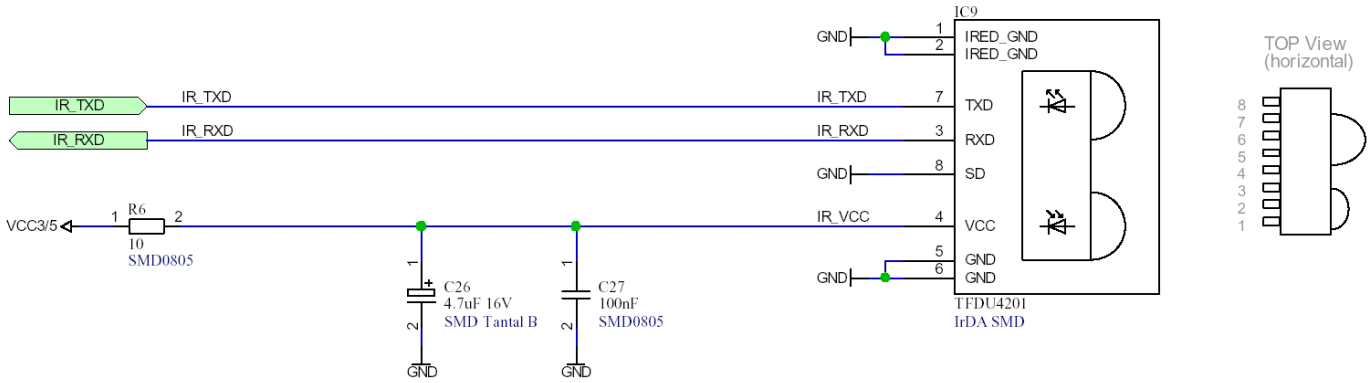


Schaltungsauszug zum RS-232C-Datenübertragungsinterface des ADNPs

Quelle: SSV, CDROM V.1.16, USER-DOC\DIAGRAMS\Adnp\DNPEVA2-ADNP1486-P5.pdf

ADNP-Anwendungsbeispiel: IrDA

TFH-Berlin



Infrarot-Datenübertragungsinterface des ADNPs

Quelle: SSV, CDROM V.1.16, USER-DOC\DIAGRAMS\Adnp\DnpEva2-IrDA.pdf