Ingenieurbüro / Vertrieb
Klaus Kohl
Postfach 1173 - 86406 Mering
Zeppelinstr. 10 - 86415 Mering
Tel. 08233/30524
Fax 08233/9971

PC-Meßtechnik

Kennung: PC-MT04
Datum: 05.05.96
Stichworte: PC, Meßtechnik, RS232, V24

Klaus Kohl

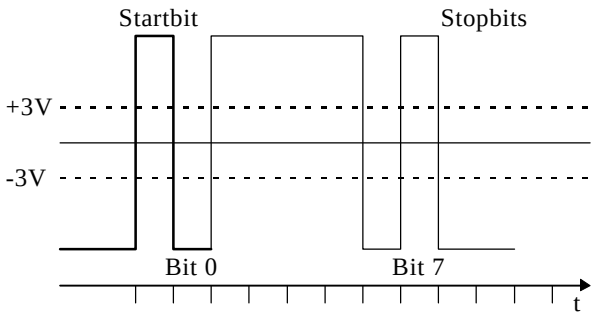
Die serielle Schnittstelle

Da bei der nächsten Folge mit dem Joystick-Port schon unmittelbar ein A/D-Wandler angesprochen wird, soll hier zum letzten Mal ein nicht so streng auf Meßtechnik ausgerichteter Artikel den Aufbau und die Programmierung der RS232-Schnittstelle des PC´s erklären. Da diese Schnittstelle ebenfalls Interruptfähig ist, werden wir sie in späteren Folgen der Serie für die Ansteuerung externer Meßsysteme nutzen. Als Beispiel werden wir hier ein einfaches, interruptgesteuertes Terminalprogramm entwerfen und damit ein Multimeter zur Messung von Spannungen abfragen.

1. Einführung

Inzwischen steht jedem PC über einen 9- oder 25poligen Stecker eine RS232-Schnittstelle zur Verfügung. Neben den beiden für die bidirektionale Datenübertragung notwendigen Pins sind weitere Ein- und Ausgänge vorhanden, die für den Handshake genutzt werden können. Wie wir später noch sehen werden, ist dabei die Ansteuerung dieser Leitungen reine Programmsache und kann nicht automatisch vom verwendeten SIO-Baustein 8250 abgenommen werden.

1.1. Begriffserklärungen



Zuerst muß hier der prinzipielle Ablauf der Übertragung gezeigt werden. Dabei ist vor allem zwischen den 1 (+5V) und 0 (0V) der Datenbits und der tatsächlich an den Datenleitungen TD und RD der seriellen Schnittstelle anstehenden Spannungen zu unterscheiden. Nach Vorschrift sind Spannungen über +3V bis +15V als Startbit oder als 0-Datenbits (SPACE) verwendet. Stopbits oder 1-Datenbits (MARK) werden durch Spannungen von -3V bis -15V gekennzeichnet. Früher wurde durch geeignete Treiber meist ±12V erzeugt. Durch Verwendung integrierter Bausteine wie MAX232 beträgt die Spannung meist nur noch ±10V oder bei 3.3V-Hardware sogar nur noch ±6V. Meist wird dann einfach eine Spannung über +3V als Startbit bzw. 0-Datenbit und Spannungen unter +3V als 1-Datenbit erkannt. Bei der Übertragung ist die Datenleitung außerhalb des PC´s im Ruhezustand auf -12V. Ein +12V-Impuls mit angegebenen Bitlänge (= 1/Baudrate) markiert den Start der Übertragung. Anschließend folgen ohne Pause ebenfalls mit der gleichen Dauer die Datenbits, wobei

mit dem niederwertigsten Bit begonnen wird. Im Anschluß an die 7 bzw. 8 Datenbits folgt evtl. noch ein Paritybit und 1, 1.5 oder 2 Stopbits.

Baudrate / Baud

Die Baudrate gibt an, wieviel Bits pro Sekunde übertragen werden. Üblich sind hier Werte von 300 (alte Modems) bis 115200 Baud (höchster Wert für PC), wobei zur Zeit folgende Werte am häufigsten verwendet werden:

2400 Baud	PC-Mäuse
9600 Baud	Langsame Modems, externe Peripherie
38400 Baud	Verbindung zu schnellen Mikrocontroller

Eine Baudrate von 9600 Baud besagt, daß ein Bit für eine Dauer von 1/9600 Sekunden (entspricht 0.104ms) ausgegeben wird. Da wir für die Ausgabe eines Bytes (8 Bit) einschließlich Start- und zweier Stopbits mindestens 11 Bits benötigen, können maximal 872 Zeichen pro Sekunde übertragen werden.

Da ab der steigenden Flanke des Startbits nur noch die Dauer eines Bits über die richtige Erkennung der Daten entscheidet, müssen bestimmte Toleranzen eingehalten werden. Falls nur in der Mitte eines Bits der Wert beachtet wird, muß der Frequenzfehler bei einem Startbit und 8 Datenbits kleiner als 0.5/9 oder 5,6 % sein. Meist arbeiten die SIO-Bausteine mit einer bis zu 16fachen Überabtastung und einer 9aus16-Erkennung und erlauben deshalb nur Fehler von 7/16/10 oder 4,8%.

Datenformate

Bei vielen (Modem-)Anschlüsse ließt man die Angabe 1200-7-N-2. Diese bedeutet, daß bei Verbindungsaufnahme mit diesem Empfänger bei einer Geschwindigkeit von 1200 Baud nur 7 Datenbit gesendet werden. Das N steht für "None Parity" und zeigt an, daß kein Prüfbit erwartet oder gesendet wird. Weitere Möglichkeiten sind hier E für "Even Parity" und O für "Odd Parity". Mit der 2 am Schluß wird noch angegeben, daß zwei Stopbits erwartet werden. Dies gibt langsamen Empfänger die Zeit zur Auswertung des Zeichens.

Halbduplex / Fullduplex

Einige Controller (z.B. der RTX-2000) lösen die Ansteuerung der RS232-Schnittstelle in reiner Software. Diese sind nicht in der Lage, gleichzeitig zu Senden und zu Empfangen und arbeiten deshalb in "Halbduplex". Über Handshake-Leitungen wird dann dem Kommunikationsparter angezeigt, ob der Empfänger bereit ist. Serielle Bausteine können dagegen gleichzeitig Senden und Empfangen und Arbeiten deshalb "Fullduplex". Bei intelligenten Bausteinen oder geeigneter Software wird über die Handshake-Leitungen angezeigt, daß der Empfangspuffer voll ist.

Mark

Der Ruhezustand der Leitung (-12V = 1-Datenbit) wird als MARK bezeichnet. In seltenen Fällen wird auch als Parity-Bit M angegeben und dann ein 1-Datenbit ausgegeben.

Parity (Odd/Even)

Zur Kontrolle, ob das Zeichen richtig übertragen wird, dient ein zusätzliches Bit. Bei "Odd Parity" wird dieses Bit so gewählt, daß die Anzahl der 1-Bits einschließlich des Paritybits ungerade ist. Bei "Even Parity" muß dagegen die Zahl der gesetzten Bits gerade sein. In dem PC übernimmt sogar der Baustein die Prüfung dieses Bits, wobei auch MARK- oder SPACE-Parity angebbar ist.

Protokolle

Nicht unmittelbar von den einzelnen Bits abhängig sind die vereinbarten Protokolle. Diese dienen bei der Übertragung größerer Datenmengen durch zusätzliche Prüfsummen für bessere

Fehlerkontrolle und können zum Teil automatisch diese Fehler durch erneute Anforderung diese Daten korrigieren. Bekannte Protokolle sind z.B. Kermit , X- oder Z-Modem.

RS-232 / V.24 / DIN66020

Historisch ist das hier angesprochene Übertragungsverfahren erstmals 1969 von der EIA (Electronic Industries Association) als RS-232 formuliert worden. Sie diente zur Definition des Datenaustauschverfahrens zwischen Computer oder zwischen Computer und Modem. Sie wurde später nochmals überarbeitet und existiert noch heute in der entgültigen Form als RS-232-C. Von der internationalen Standardisierungsorganisation CCITT wurde das Gegenstück V.24 herausgegeben. Sie enthält jedoch nur funktionelle Eigenschaften. Die ergänzenden elektrischen Kennwerte zur RS-232-C sind in einer eigenen Empfehlung V.28 zusammengefaßt. Die deutsche Version des RS232-Standards hat die Bezeichnung DIN66020.

Space

Das Gegenstück zum Mark-Bit ist das Space-Bit, welches identisch mit den Stopbits oder den 0-Datenbits ist. In seltenen Fällen wird es auch als Paritybit genutzt.

Startbit

Zur Synchronisierung der Datenübertragung wird zuerst mit dem +12V-Signal ein Startbit übertragen. Es wird sonst nicht weiter berücksichtigt, aber muß die Dauer von genau einem Bit haben.

Stopbit

Um genügend Raum zur Synchronisation mit dem nächsten Datenbit zu lassen oder bei reinen Software-Lösungen Zeit zur Auswertung des erfaßten Wertes zu lassen, wird mindestens 1 Stopbit nach den Daten gesendet. Es hat den Pegel der in Ruhe befindlichen Leitung.

Synchron / Asynchron

Bei der RS232-Schnittstelle spricht man immer von einer asynchronen Schnittstelle. Im Gegensatz zu der synchronen Schnittstelle hat man hier keine Taktleitung, auf den sich die Übertragung "synchronisiert". Statt dessen wird das Startbit und ab dann nur noch eine Zeitsteuerung verwendet.

BREAK

Als Break wird eine Folge von Startbits bzw. 0-Bits länger als eine Zeichenlänge erkannt. Die Ursache dazu kann z.B. eine Leitungsunterbrechung sein. Die meisten Bausteine erkennen dies automatisch und erlauben einen Interrupt bei diesen Zustand. Ansonsten führt dieser Zustand zur ständigen Erkennung von 0-Bytes.

1.2. Pinbelegung

Folgende Liste zeigt die Pinbelegung des 9- und 25poligen Steckers. Bei manchen älteren Schnittstellen ist auch noch Pin 23 des 25poligen Steckers für die Erkennung der Baudrate reserviert, wird aber bei den PC´s nicht genutzt.

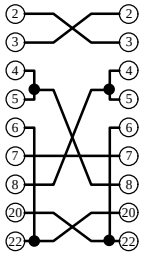
Pin (25polig)	Pin (9polig)	Richtung	Verwendung
1			Masse für Abschirmung
2	3	Ausgang	TD = Transmitted Data Vom PC ausgehende Datenleitung
3	2	Eingang	RD = Received Data Zum PC gehende Datenleitung
4	7	Ausgang	RTS = Request to Send Ein +12V-Signal zeigt an, daß Daten bereit stehen
5	8	Eingang	CTS = Clear to Send

			Ein +12V-Signal zeigt an, daß der Empfänger bereit ist
6	6	Eingang	DSR = Data Set Ready Früher hat ein Modem (="Data Set") über diese Leitung angezeigt, daß es vorhanden und bereit ist.
7	5		Signalmasse
8	1	Eingang	DCD = Data Carrier Detect
20	4	Ausgang	DTR = Data Terminal Ready
22	9	Eingang	RI = Ring Indicator Modems zeigen über diese Leitung das Klingeln des Telefons an.

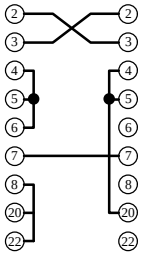
1.3. Anschlußart

Da die RS232-Leitungen von den üblichen Empfängern wie Drucker oder Modem meist nicht vollständig unterstützt werden, hat sich ein großer Vielfalt von Anschlußbelegungen gebildet. Bestimmte Verknüpfungen sind oft notwendig, weil ein Gerät einen bestimmten Pegel an seinen Eingängen oder das sendende PC-Programm einen Handshake erwartet. In der Praxis werden dabei die folgenden beiden Versionen am häufigsten eingesetzt:

Nullmodem



Smart Modem



1.4. Speicheradressen

Beim Bootvorgang sucht der PC automatisch auf vier Adressen ($3F8, $2F8, $3E8 und $2E8) nach einem seriellen Baustein. Damit aber intelligente Karten weitere RS232-Ports installieren und Anwenderprogramme diese Adressen ermitteln können, gibt es in den BIOS-Variablen vier Speicherstellen für die Basisadresse der vorhandenen Schnittstellen. Nicht verfügbare Ports werden mit dem Wert 0 gekennzeichnet.

Adresse der BIOS-Variable	**Verwendung**
$0040:$0000/1	COM1-Basisadresse (meist $03F8)
$0040:$0002/3	COM2-Basisadresse (meist $02F8)
$0040:$0004/5	COM3-Basisadresse
$0040:$0006/7	COM4-Basisadresse

Bei jedem angegebenen RS232-Port werden bis zu 8 Adressen für die Programmierung belegt. Zusätzlich werden üblicherweise die Interrupts 3 (für COM2 und COM4) und 4 (für COM1 und COM3) reserviert.

Die 8 Adressen verwalten sogar 11 Register, wobei Zugriffe auf Offset 0 beim Schreiben die Ausgabe eines Zeichens starten und beim Lesen des gleichen Ports das zuletzt erkannte Zeichen auslesen. Über ein Bit im Line-Control-Register (Offset 3, Bit 7) kann der Teiler für den Baudratengenerator auf Offset 0/1 eingeblendet werden.

Offset	Richtung	Bit	Bedeutung
0	Schreiben	0-7	Zeichenausgabe an TD starten
0	Lesen	0-7	Abfrage des zuletzt an RD erfaßten Zeichens
1			Interupt Enable Register
		0	1: Interrupt, wenn Daten verfügbar (RxRDY)
		1	1: Interrupt, wenn Sendepuffer leer ist (TBE)
		2	1: Interrupt bei Übertragungsfehler
		3	1: Interrupt bei Änderung der Eingangsleitungen
		4-7	immer auf 0
2			Interupt Identification Register Bit 0=0: Interrupt steht an Wert 0: RS232-Eingangsleitungen geändert Wert 2: TBE Wert 4: RxRDY Wert 6: BREAK oder Serialize Error
3			Line Control Register (Übertragungsprotokoll)
		0/1	Anzahl Datenbits (%00=5, %01=6, %10=7 und %11=8)
		2	Anzahl Stopbits (0=1 und 1=2 Stopbits)
		3..5	Parity (0=None, 1=Odd, 3=Even, 5=Mark, 7=Space)
		6	Break Controll (0=aus, 1=an)
		7	DLAB: bei 1 liegt Baudratenteiler auf Register 0/1
4			Modem Control Register (steuert Ausgänge)
		0	DTR (Pin 20)
		1	RTS (Pin 4)
		2	GP01
		3	GP02 (Interrupt Enable; siehe unten)
		4	Loopback (für Tests)
		5-7	immer auf 0
5			Line Status (Bit 0-5 können Interrupt auslösen)
		0	RxRDY (Empfangspuffer voll)
		1	Overrun Error
		2	Parity Error
		3	Framing Error
		4	Break Detect
		5	TBE (Sendepuffer leer, Übertragung kann noch laufen)
		6	TXE (Ausgabepuffer leer, Übertragung beendet)
		7	immer 0
6			Modem Status (Bit 0..3 zeigt Änderung seit letztem Lesen)
		0	Flag für Änderung an CTS
		1	Flag für Änderung an DSR
		2	Flag für Änderung an RI
		3	Flag für Änderung an DCD
		4	Aktueller Status an CTS (RTS bei Loopback)
		5	Aktueller Status an DSR (DTR bei Loopback)
		6	Aktueller Status an RI (GP02 bei Loopback)
		7	Aktueller Status an DCD (GP01 bei Loopback)
7			freies RAM (wird nicht genutzt)
8			Auf Register 0 gelegt, wenn Bit DLAB gesetzt ist enthält niederwertige Byte des Baudratenteilers
9			Auf Register 1 gelegt, wenn Bit DLAB gesetzt ist enthält höherwertige Byte des Baudratenteilers

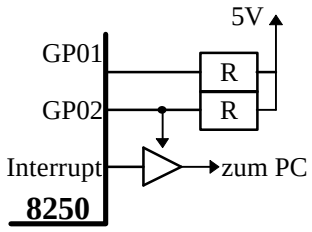
Der Baudratenteiler für die Übertragungsfrequenz wird durch einen 1.8432MHz-Quarz gespeist, der früher vom 4,77MHz-Bustakt geteilt durch 3 kam. Nach einer Teilung durch 16 (für die oben angesprochene Überabtastung) können folgende sinnvollen Werte eingestellt werden:

Teiler	1	3	6	9	12	24	48	96	384
Baudrate	115200	38400	19200	12800	9600	4800	2400	1200	300

Der 12800Baud-Wert wurde z. B. für eine Mikrocontroller-Entwicklung benötigt, wo von einem 8MHz-Quarz die eingebaute SIO gespeist wurde. Die dort einstellbare Frequenz von 12500 Baud liegt noch innerhalb der Fehlertoleranz. Ähnlich können natürlich auch andere Teilerwerte genutzt werden.

1.5. RS232-Interrupt beim PC

Wie schon im ersten Artikel dieser Serie erwähnt, muß man bei einem PC außer den Register des entsprechenden Bausteins noch die Interrupttabelle (zur Festlegung der Adresse des Interruptprogrammes) und den Interruptcontroller (zur Freigabe der Leitung) programmieren. Im PC ist es üblich, für die erste serielle Schnittstelle die Interruptleitung 4 und für die zweite RS232 die Interruptleitung 3 zu nutzen. Falls noch mehr Schnittstellen zur Verfügung stehen, muß man sich diese Leitungen teilen. Damit man einen RS232-Controller von der Interruptleitung abschalten kann, wurde schon bei den IBM-Entwicklern einer der beiden nicht genutzten 8250-Portleitungen (GP02) zur Ansteuerung eines Treiberbausteins herangezogen. Nur wenn diese Leitung auf 1 ist, kann der 8250 auch einen Interrupt auslösen.



Neben der üblichen Bestätigung am Interruptcontroller muß auch auf dem 8250 eine geeignete Aktion zur Beendigung des Interruptzustandes erfolgen. Folgende Möglichkeiten sind vorgesehen:

Interruptart	**Notwendige Aktion**
Receive Error oder Break	Line Status Register auslesen
Receive Data	Datenregister lesen
TBE	Neue Daten schreiben oder Interrupt Identification Register lesen
RS232 Input Change	Modem Status Register lesen

2. Ein kleines PC-Terminalprogramm in FORTH

Eigentlich stellt das Betriebssystem des PC´s über Interrupt 14 einige Funktionen zum Senden und Empfangen von Daten über die RS232-Schnittstelle zur Verfügung. Da aber dies nur bis zu einer Geschwindigkeit von 9600 Baud geht und die Steuerregister nicht besser angesteuert werden können, wird bisher immer auf eigene Routinen zurückgegriffen. Das folgende Listing ist aus einem Hardware-Testprogramm in KK-FORTH entnommen. Urspüngliche Quelle der Kernroutinen war ein Terminalprogramm von Klaus Schleisiek zum volksFORTH-PC. Der hier ebenfalls aufgelistete Beispielscreen (Screen # 15) enthält ein einfaches Terminalprogramm TERMINAL , daß empfangene Zeichen ausgibt und Tastatureingaben zur Schnittstelle schickt. Die Baudrate und die Schnittstelle muß vorher mit SETBAUD gesetzt werden. Das ebenfalls vorhandene Programm MULTIMETER wird weiter unten beschrieben.

```
       Screen # 0
\\ Verwaltung der COM-Schnittstelle bei PC      ( 05.05.96/KK )

System:         KKF_PC V1.2/2
Änderungen:     16.08.93  KK: aus KKFT2 entnommen

                    Hinweise:
 * Verwendet kleinen 1K-Buffer im FORTH-Speicher
 * Keine Handshake's verwendet

 Quelle für SIO-Programmierung: PC-volksFORTH 3.81.41 (KS)

 ACHTUNG:  Register BX wird im KKF_PC als TOS verwendet
           Spoolergröße: 128 Byte

       Screen # 1
\ Loadscreen                                    ( 15.04.94/ki )

  \needs Assembler  2 loadfrom ASM8086.SCR

  &02 capacity 1- thru

  5 baud# !  1 port# !                    \ COM2 mit 9600 Baud

       Screen # 2
\ Tabellen für COM-Port                         ( 05.05.96/KK )
  Create baudtab    ( Tabelle mit der Baudrate/100 )
    &1152 ,  &576 ,  &384 ,  &192 ,  &128 ,
     &96 ,   &24 ,   &12 ,  &288 ,    &3 ,
  Create diftab     ( Tabelle mit den Teilern )
       1 ,     2 ,     3 ,     6 ,     9 ,
     &12 ,   &48 ,   &96 ,     4 ,  &384 ,
  Create porttab    ( Tabelle mit Portadressen )
    $03f8 , $02f8 , $03e8 , $02e8 ,
  Create ileveltab  ( Tabelle mit Interrupt-Level )
     $10 ,   $08 ,   $10 ,   $08 ,
  Create inttab     ( Tabelle mit Interrupt-Adresse )
     $30 ,   $2c ,   $30 ,   $2c ,

       Screen # 3
\ Variablen für den COM-Port                    ( 05.05.96/KK )
  2Variable oldivec                ( Alter Interrupt-Vektor )
  &50000 Constant Timeout      ( Abbruch nach 50000 Schleifen )
\ Einstellungen
  Variable baud#                        ( Nummer der Baudrate )
  Variable baud                              ( Baudrate / 100 )
  Variable port#                        ( Nummer des Port - 1 )
\ Dieser SYSVAR-Bereich ist frei und hat feste Speicheradressen
  Create (baud      2 allot             ( Teiler für Baudrate )
  Create (portaddr  2 allot              ( Addresse des Ports )
  Create (ilevel    2 allot                        ( I-Maske )
  Create (iaddr     2 allot               ( Interrupt-Adresse )

       Screen # 4
\ Neue Baudrate oder Port setzen                ( 16.08.93/KK )
  : setbaud     ( baud port -- )
    dup port# !  2*  dup inttab    + @      (iaddr    !
                     dup ileveltab + @  not (ilevel   !
                         porttab   + @      (portaddr !
    dup baud# !  2*  dup baudtab   + @      baud      !
                         diftab    + @      (baud     ! ;

       Screen # 5
\ Queue und notwendige Befehle                  ( 16.08.93/KK )
( Datenqueue mit 128 Bytes und zwei Zeiger für Interuptroutine)
( Queue+0: Anzahl der gespeicherten Zeichen )
( Queue+2: Offset zum nächsten auszugebenden Zeichen )
```

```
  $0084 Constant qlen       ( Bedarf für Queue )
6   Create queue  qlen allot  ( Queue )
  $20 Constant I-ctrl           \ 8259-Register
  $21 Constant I-mask           \ 8259-Maske

       Screen # 6
\ tx? = Statusabfrage für Zeichenausgabe        ( 05.05.96/KK )
( Test, ob ein Zeichen ausgegeben werden kann )
  Code tx?      ( -- f )        \ f=-1, wenn bereit
   tos push,                \ TOS(=Top Of Stack) zum Datenstack
   (portaddr #) dx mov,   5 # dx add,           \ Statusadresse
   dx in,                   \ Port zum Register A holen
   tos tos xor,             \ TOS auf 0 setzen
 \ $1020 # ax and,          \ Ausgabe mit warten auf Handshake
   $0020 # ax and,          \ Ausgabe ohne Handshake
 \ $1020 # ax cmp,          \ dann Test, ob diese Bits gesetzt
   $0020 # ax cmp,          \ dann Test, ob diese Bits gesetzt
   0= IF, tos dec, THEN,    \ Zeichenausgabe erlaubt ?
   next,                    \ Next kompiliert Next-Makro
  End-Code

       Screen # 7
\ (tx  tx  = Zeichenausgabe                     ( 16.08.93/KK )
( Unbedingte Zeichenausgabe direkt in den 8250-Port )
  Code (tx      ( char -- )
   tosl al xchg,        \ Zeichen zum AL-Register bringen
   (portaddr #) dx mov, \ Datenadresse in's D-Register
   dx byte out,         \ AL ausgeben
   tos pop,             \ nächsten Stackwert ins D-Register
   next,                \ Next kompiliert Next-Makro
  End-Code
( Wartet, bis letztes Zeichen ausgegeben wurde )
  : tx          ( char -- )
   timeout                              \ Schleifenanzahl
   BEGIN tx? IF drop (tx exit THEN
         1- ?dup 0=                     \ Timeout abgelaufen ?
   UNTIL -1 Error" Transmit-Timeout" ;

       Screen # 8
\ -DTR  +DTR  = Steuerleitung ein/aus           ( 16.08.93/KK )
( DTR-Leitung auf +12 V = logisch Null )
  Code -dtr     ( -- )
   (portaddr #) dx mov,  4 # dx add,   \ Modem-Kontrollregister
   dx byte in,             \ Inhalt zum AL-Register
   $1c # al and,           \ DTR und RTS auf 0 setzen = +12 V
   dx byte out,            \ AL wieder in Port schreiben
   next,                   \ nächster FORTH-Befehl
  End-Code
( Analog dazu DTR und RTS wieder auf 1 setzen = -12 V )
  Code +dtr     ( -- )
   (portaddr #) dx mov,  4 # dx add,
   dx byte in,   $fc # al and,    1 # al or,   dx byte out,
   next,
  End-Code

       Screen # 9
\ s-int = Interruptroutine; rx?                 ( 05.05.96/KK )
| Proc s-int
     dx push,   si push,   ax push,           \ Register retten
     cs seg,  queue # si mov,             \ Queue-Adresse holen
     cs seg,  si ) ax mov,        \ Anzahl gespeicherte Zeichen
     ax dx mov,   ax inc,   $007f # ax and,        \ +1 MOD 128
     cs seg,  ax si ) mov,   dx si add,       \ Zurückschreiben
     cs seg,  (portaddr #) dx mov,  dx byte in, \ Zeichen holen
     cs seg,  al 4 si d) mov,      \ und in den Queue schreiben
     $20 # al mov,   i-ctrl #) byte out,         \ EOI für 8259
     ax pop,   si pop,   dx pop,   iret,       \ Register holen
   End-Proc
| Code rx?  ( -- f ) \ Flag, ob Zeichen im Puffer sind
```

```
     tos push,   queue # di mov,
     di ) tos mov,   2 di d) tos xor,
     0<> IF,  -1 # tos mov,  THEN,   NEXT,            End-Code

        Screen # 10
\ (rx  rx  = Zeichen aus Queue holen           ( 16.08.93/KK )
( Holt Zeichen aus dem Queue, verändert evtl. die Zeiger )
  Code (rx      ( -- char )
     tos push,   si push,
     queue 2+ # si mov,
     cs seg,  si ) ax mov,
     ax dx mov,   ax inc,   $007f # ax and,
     cs seg,  ax si ) mov,   dx si add,
     cs seg,  2 si d) al mov,   0 # ah mov,
     ax tos mov,   si pop,   next,
  End-Code
( Holt Zeichen, wartet, bis Zeichen bereit )
  : rx          ( -- char )  rx? IF (rx exit THEN
   Timeout  BEGIN rx? IF drop (rx exit THEN
                1- DUP 0=              \ Timeout abgelaufen ?
            UNTIL DROP -1 ERROR" Receive-Timeout";
        Screen # 11
\ s-init  = Initialisierung der Schnittstelle   ( 15.04.94/ki )
| Code s-init   ( -- )
   tos push,  ds push,           \ TOS und DS-Register sichern
   ax ax xor,  (iaddr #) w mov,
   ax ds mov,   s-int # w ) mov,        \ Interuptvektor setzen
   cs ax mov,  ax 2 w d) mov,       \ und DS wieder zurückladen
   ds pop,  (portaddr #) dx mov,  3 # dx add,
   $80 # al mov,   dx byte out, \ Baudratenregister einschalten
   2 # dx sub,  (baud #) ax mov,   al ah xchg,       \ Baudrate
   dx byte out,  dx dec,  al ah xchg,  dx byte out,    \ setzen
   3 # dx add,  $a06 # ax mov,  dx out,  \ 7bit, noP, +RTS +OUT
   2 # dx sub,  1 # al mov,  dx byte out,  \ Interrupt erlauben
   i-mask #) byte in,
   (ilevel #) al and,                          \ 8259 aktivieren
   i-mask #) byte out,  tos pop,  next,
  End-Code

      Screen # 12
\ sioinit  -sioinit  = Initialisierung/Reset    ( 16.08.93/KK )
( Löscht Queue-Zeiger und Initalisiert Baustein/Interupt )
  : sioinit     ( -- )
    baud# @  port# @  setbaud                     \ Einstellung
    0 (iaddr @ l2@  oldivec 2!        \ Interrupt-Vector merken
    queue off  queue 2+ off                    \ Puffer löschen
    s-init  +dtr ;
( Blockiert Interupt, Schaltet RTS u. DTR ab )
  : -sioinit    ( -- )
    0 (portaddr @ 1+  pc!               \ 8250-Interupt sperren
    0 (portaddr @ 4 + pc!               \ -RTS/-rts/-out2
    i-mask pc@ (ilevel @ not or i-mask pc! \ 8259 sperren
    oldivec 2@  0 (iaddr @  l2! ;       \ Vektor zurücksetzen

        Screen # 13
\ comval@                                       ( 05.05.96/KK )
  Create comval$   &15 allot        \ Platz für "DC-00.000 V  "
  : com$>val    ( -- val )
    0
    comval$ 3 +  1 FOR >r  &10 *  r@ c@ $30 -  +  r> 1+  NEXT
    1+           2 FOR >r  &10 *  r@ c@ $30 -  +  r> 1+  NEXT
    7 -          c@   Ascii -  =  IF negate THEN ;

      Screen # 14
\ comval@                                       ( 05.05.96/KK )
  : (comval@    ( -- val ) \ $7fff bei Fehler; $8000 bei Taste
    BEGIN rx? WHILE rx drop REPEAT              \ Puffer löschen
    Ascii D  tx                          \ Startbefehl ausgeben
    0   &10000                        \ Offset, Schleifenzähler
```

```
    BEGIN key? IF 2drop $8000 exit THEN   \ Abbruch durch Taste
          rx? IF drop rx $7f and  \ Zeichen holen und Maskieren
                 $0d case?
                 IF &13 = IF com$>val ELSE $7fff THEN exit THEN
                 over comval$ +  c!  1+
               &14 case? IF $7fff exit THEN  &10000 \ Zu Lang
              ELSE 1- ?dup 0= IF drop $7ffe exit THEN     \ Neu
              THEN
    REPEAT ;
  : comval@     ( -- val ) \ Mit erneutem Versuch bei Timeout
    BEGIN (comval@ $7ffe case? 0= UNTIL ;

       Screen # 15
\ Mini-Terminal                                    ( 05.05.96/KK )
  : terminal    ( -- ) \ Terminalmodus
    sioinit
    BEGIN key?  IF  key  3 case? ?exit
                    tx?       IF tx ELSE drop THEN  THEN
          rx?   IF  rx $7f and
                    $20 case? IF Ascii . THEN
                    $0d case? IF cr ELSE emit THEN  THEN
    REPEAT
    -sioinit ;
  : multimeter  ( -- ) \ Werte vom Multimeter anzeigen
    base push  decimal   sioinit    comval$ &14 blank
    BEGIN comval@  $8000 <>
    WHILE cr  comval$ &14 type   ."  -> "  6 .r  ." mV  "
    UNTIL key drop  -sioinit ;
```

Anmerkung zu Listing

Die Pegel der Steuerleitungen sind hier für die Kommunikation zu Einplatinencomputer vorbereitet. Ähnlich wie hier bei der Leitung DTR beschrieben können auch die anderen Steuerleitungen geändert werden. Da der SIO-Baustein selbst keine Auswertung der Handshake-Leitungen durchführt, muß der Ausgabeteil TX bzw. TX? dies übernehmen. Meist kommt man ohne Handshake aus und verwendet nur die Abfrage, ob das nächste Zeichen ausgegeben werden kann. Soll auch noch das RTS berücksichtigt werden, muß das entsprechende Bit (wie in Screen 6 in der ausmaskierten Zeilen) getestet werden. Die Größe des Empfangspuffer ist in Abhängigkeit der Baudrate und der erwarteten Programmzeiten entsprechend groß zu wählen.

Anmerkung zu DOS-Programmen in Windows

Unter Windows3.1 einschließlich Windows 95 hat sich in der Praxis gezeigt, daß bei sehr schneller bidirektionaler Übertragung manchmal ein Fehler auftritt. Es wird dann ein schon empfangenes Zeichen erst dann dem (Interrupt-)DOS-Programm gemeldet, wenn es das nächste Zeichen ausgibt. Unter Windows NT scheint dieser Fehler aber wieder korrigiert zu sein.

3. Anschluß eines Multimeters

Bei der Hardware vieler Anbieter von Meßsystemen ist eine serielle Schnittstelle zur Einstellung der Parameter oder Abfrage der Daten schon integriert. Das nachfolgende Beispiel verwendet dazu ein z.B. bei Conrad erhältliches Multimeter, daß über diese Schnittstelle den aktuellen Meßwert ausgibt und deshalb z.B. zur Kalibrierung von Analogausgängen genutzt werden kann.

Dabei erwartet das Multimeter das Zeichen "D" und liefert nach der Messung ein String im Format "DC-00.000 V " zurück. Das hier gelistete COMVAL@ empfängt diesen String, wertet ihn aus und liefert die Spannung in Millivolt zurück. Dabei werden mit den nicht auftretenden Meßwerten -

32766 ($7fff) ein Empfangsfehler und mit $8000 ein Abbruch mittels beliebiger Taste kodiert. Da man beim Start des Testprogrammes häufig vergißt, das Multimeter einzuschalten, wird nach einer Timeout-Zeit (hier willkürlich 10000 Schleifendurchläufe) das Startzeichen nochmals ausgegeben.

4. Einsatzmöglichkeiten

Die oben geschilderte Abfrage von Multimeter ist nur eine der Möglichkeiten, die mit Hilfe eigener Schnittstellenprogramme realisiert werden kann. Etwas größere Meßsysteme (leider meist auch entsprechend teuer) können sogar von sich aus Messungen durchführen und liefern nach geeigneter Abfrage ganze Signalverläufe zurück. Oft muß man diese Systeme erst mittels der RS232-Schnittstelle initialisieren.

Leider kenne ich bisher keine A/D-Wandler, der automatisch Signale mit RS232-konformen Timing liefert. Meist sind sie für entspechende RISC-Prozessoren vorbereitet und erwarten einen externen Takt zur Erzeugung des Datenstroms. Doch gibt es einige Schaltungsbeispiele, bei denen die Steuerleitungen der RS232-Schnittstelle dafür "mißbraucht" werden. Neben den Taktsignal wird wie schon in der vorherigen Folge dieser Serie (Centronicsport) die Stromversorgung für das Meßsystem geliefert. Man hat hier sogar den Vorteil höherer Spannungen, da mindestens ±3 bis ±15V von den Schnittstellenbausteinen ausgegeben werden.

5. Literatur zur seriellen Schnittstelle

Joe Campbell
C Programmer´s Guide to Serial Communications
Howard W. Sams & Company

Günther Klotz
Bits im Gänsemarsch
c´t 1986, Heft 12, Seite 185ff

Wolfgang Hartung, Michael Felsmann, Andreas Stiller
PC-Bausteine: Der UART 8250 als Tor zur seriellen Welt
c´t 1988, Heft 5, Seite 204ff

Der Interrupt 14 für serielle Schnittstellen
DOS Extra 1´87/88

Thom Hogan
Die PC-Referenz für Programmierer
Microsoft Press

Klaus Dembowski
PC-gesteuerte Meßtechnik
(Buch mit zwei Disketten und 2 Platinen für Meßsysteme an Centronics und RS232)
(ist zur Zeit als Restposten für ca. 30DM erhältlich, Neupreis: 129DM)
ISBN 3-87791-516-7
Markt&Technik Buch- und Software-Verlag GmbH&Co