Anton Ruepp

EMUF als serielles Interface

Drucker am seriellen Commodore-Bus

Genaugenommen ist diese Applikation für alle Computer verwendbar, die den seriellen (IEC-ähnlichen) Bus besitzen, also VC-20, C-64 und Nachfolger. Mit EMUF-Unterstützung lassen sich nun auch Drucker mit anderen Schnittstellen anschließen, die aber rechnerseitig ohne zusätzliche Software mit den normalen Befehlen angesprochen werden können. Das vorliegende Beispiel zeigt die Lösung für eine serielle 20-mA-Schnittstelle, die leicht in eine V.24-Schnittstelle umgewandelt werden kann.

Der EMUF [1] bedient die serielle Schnittstelle des Computers und gibt die empfangenen Zeichen in der gewünschten Form an den angeschlossenen Drukker aus. Hierbei lassen sich auch eventuelle Zeichenumcodierungen oder Sonderzeichen realisieren. Im vorliegenden Fall handelt es sich um eine TTY-Schnittstelle, also eine 20-mA-Schleife, mit einer Übertragungsrate von 150 Baud und ungerader Parität.
Es lassen sich natürlich auch andere Schnittstellen realisieren. Dazu ist entweder nur eine Hardware-Anpassung erforderlich (V.24), oder aber das Zeichenausgabe-Programm muß umgeschrieben werden (Centronics).

Programmablauf

Nach der Initialisierung wartet der EMUF auf seine Primäradresse, die im Falle des Druckers 4 lautet. Sobald er diese erkannt hat, fühlt er sich so lange angesprochen, bis der Rechner „Unlisten" ausgibt. Sekundäradressen werden ignoriert. Eine ausführliche Beschreibung des seriellen Busses findet sich in [2].
Die Routine GBYTE des Programmes (*Bild 1*) liest ein Zeichen vom Bus und legt es in Zero-Page in der Zelle SHIFT ab, die bei der Ausgabe als Schieberegister verwendet wird. Bei Empfang des Unlisten-Zeichens erfolgt ein Neustart. Die Routine AUSGA fügt bei Erkennen eines Carriage Return (CR) noch ein Linefeed (LF) ein, wenn der Pegel an PA0 dies verlangt. Das Programm OUTALL

```
0000                    *=0
0000                PARYB   *=*+1               ;PARITY-ZAEHLER
0001                TIMER   *=*+1               ;EOI TIME REGISTER
0002                SHIFT   *=*+1               ;SCHIEBEREGISTER
0003                OUTBD   *=*+1               ;ZAEHLER FUER BAUD GENER
0004                ATNFG   *=*+1               ;ATNFLAG,TRUE (LOW) ==> $FF
0005                PRTA=$800
0005                DDRA=$801
0005                PRTB    =$802
0005                DDRB    =$803
0005                CLOCK   =%00100000
0005                ;
0005                        *=$FFC              ;RESET
0FFC  00 0C                 .WOR START
0FFE                        *=$FFE              ;IRQ
0FFE  00 0C                 .WOR START          ;UNUSED
1000                ;
1000                        *=$C00
0C00  78            START  SEI                 ;DO NOT DISTURB
0C01  A2 FF                 LDX #$FF            ;ORDNE
0C03  9A                    TXS                 ;STACK
0C04  D8                    CLD                 ;NO DEC
0C05                ;
0C05  20 36 0C              JSR INIT            ;INITIALISIERUNG
0C08  20 54 0C              JSR ATNHI           ;WARTE BIS ATN NEUTRAL IST
0C0B  20 4A 0C      HAUPT   JSR ATNLO           ;UND JETZT BIS ATN TRUE
0C0E  AD 00 08              LDA PRTA            ;WENN CLOCK HI
0C11  29 20                 AND #%00100000      ;IST, VERSUCHEN WIR'S
0C13  D0 F6                 BNE HAUPT           ;NOCHMALS
0C15  20 62 0C              JSR SDLOW           ;SETZE DATA LOW
0C18  20 5A 0C              JSR CLOHI           ;UND WARTE AUF CLOCK HI
0C1B  20 8A 0C              JSR GBYTE           ;HOL DAS ZEICHEN
0C1E  20 70 0C              JSR MYADS           ;BIN ICH GEMEINT ?
0C21  B0 DD                 BCS START           ;NEIN
0C23  20 8A 0C      MYLA    JSR GBYTE           ;EIN BYTE VOM BUS HOLEN
0C26  24 04                 BIT ATNFG           ;WAR ES ETWA
0C28  F0 06                 BEQ CONT            ;UNTER ATN ?
0C2A  A5 02                 LDA SHIFT           ;WENN JA,  -WAR ES
0C2C  C9 3F                 CMP #$3F            ;VIELLEICHT "UNLISTEN" ?
0C2E  F0 D0                 BEQ START           ;JA  ALSO SCHLUSS
0C30  20 D9 0C      CONT    JSR AUSGA           ;ZEICHEN AUSGEBEN
0C33  4C 23 0C              JMP MYLA            ;NAECHSTES HOLEN
0C36                ;
0C36  A9 00         INIT    LDA #0
0C38  8D 01 08              STA DDRA            ;ALLES INPUTS
0C3B  A9 01                 LDA #1
0C3D  8D 02 08              STA PRTB            ;PRINTERSTROM
0C40  A9 FF                 LDA #$FF
0C42  8D 03 08              STA DDRB
0C45  A9 01                 LDA #1              ;ODD PARITY
0C47  85 00                 STA PARYB
0C49  60                    RTS
0C4A                ;WARTET AUF ATN UND CLOCK TRUE
0C4A  AD 00 08      ATNLO   LDA PRTA
0C4D  30 FB                 BMI ATNLO
0C4F  29 20                 AND #CLOCK
0C51  D0 F7                 BNE ATNLO
0C53  60                    RTS
```

Bild 1. Das EMUF-Programm im Quellenformat. Änderungen für andere Baudraten sind in der Tabelle zu finden

```
0C54  AD 00 08     ATNHI  LDA PRTA          ;WARTET AUF ATN HI
0C57  10 FB               BPL ATNHI
0C59  60                  RTS
0C5A                ;WARTET BIS CLOCK HIGH IST
0C5A  AD 00 08     CLOHI  LDA PRTA
0C5D  29 20               AND #CLOCK
0C5F  F0 F9               BEQ CLOHI
0C61  60                  RTS
0C62                ;SET DATA LOW
0C62  A9 00        SDLOW  LDA #0
0C64  8D 00 08            STA PRTA
0C67  AD 01 08            LDA DDRA
0C6A  09 40               ORA #%01000000    ;PA6 AUSG
0C6C  8D 01 08            STA DDRA
0C6F  60                  RTS
0C70                ;CLEAR CARRY WENN ADR. OK
0C70  A5 02        MYADS  LDA SHIFT
0C72  C9 24               CMP #$24          ;PRIMAERADRESSE  4
0C74  18                  CLC
0C75  F0 01               BEQ FINI
0C77  38                  SEC
0C78  60           FINI   RTS
0C79                ;WARTEN AUF CLOCK LOW
0C79  AD 00 08     CLOLO  LDA PRTA
0C7C  29 20               AND #CLOCK
0C7E  D0 F9               BNE CLOLO
0C80  60                  RTS
0C81  AD 01 08     DAINP  LDA DDRA          ;PA6 INPUT (=HIGH)
0C84  29 BF               AND #%10111111
0C86  8D 01 08            STA DDRA
0C89  60                  RTS
0C8A                ;HOLT EIN BYTE VOM BUS  GIBT EOI
0C8A                ;HANDSHAKE WENN NOETIG.
0C8A                ;SETZT ATN-FLAG ENTSPRECHEND (ATN  TRUE= $FF)
0C8A  20 5A 0C     GBYTE  JSR CLOHI         ;WARTE BIS CLOCK HIGH
0C8D  20 81 0C            JSR DAINP         ;MACH PA6 INPUT & HIGH
0C90  A9 0F               LDA #$0F
0C92  85 01               STA TIMER         ; "TIMER" FUER EOI ACK
0C94  AD 00 08     GBY1   LDA PRTA
0C97  29 20               AND #CLOCK
0C99  F0 0F               BEQ CL            ;CLOCK LOW . KEIN EOI
0C9B  C6 01               DEC TIMER
0C9D  D0 F5               BNE GBY1          ;NOCH KEIN EOI
0C9F  20 62 0C     EOI    JSR SDLOW         ;QUITTIERE EOI MIT DATA LOW
0CA2  A2 0C               LDX #12
0CA4  CA           D1     DEX
0CA5  D0 FD               BNE D1
0CA7  20 81 0C            JSR DAINP         ;DATA WIEDER HIGH
0CAA  A2 08        CL     LDX #8
0CAC  20 79 0C     CL2    JSR CLOLO
0CAF  20 5A 0C            JSR CLOHI
0CB2  AD 00 08            LDA PRTA
0CB5  29 40               AND #%01000000    ;DATA
0CB7  0A                  ASL A
0CB8  0A                  ASL A             ;DATA IN CARRY
0CB9  66 02               ROR SHIFT         ;CARRY INS SCHIEBEREGISTER
0CBB  CA                  DEX
0CBC  D0 EE               BNE CL2
0CBE  AD 00 08            LDA PRTA
0CC1  30 04               BMI NIXAT
0CC3  A9 FF               LDA #$FF
0CC5  D0 02               BNE ATNJN         ;JA, ATN
0CC7  A9 00        NIXAT  LDA #00
0CC9  85 04        ATNJN  STA ATNFG

0CCB  20 79 0C             JSR CLOLO        ;EIN BYTE IST IM KASTEN
0CCE  A9 00                LDA #0
0CD0  8D 00 08             STA PRTA
0CD3  A9 40                LDA #%01000000
0CD5  8D 01 08             STA DDRA         ;DATA ACCEPTED
0CD8  60                   RTS
0CD9                ; HIER WERDEN ALLE UNTER ATN ANKOMMENDEN
0CD9                ; ZEICHEN AUSGEFILTERT
0CD9  A5 04        AUSGA   LDA ATNFG
0CDB  D0 18                BNE ENT          ;EXIT WENN ATN FLAG CLEAR
0CDD  A5 02                LDA SHIFT        ;HOLE BYTE
0CDF  29 7F                AND #$7F
0CE1  C9 0D                CMP #$0D         ;CARRIAGE RET
0CE3  D0 0D                BNE NOL          ;NEIN, GIBS AUS
0CE5  48                   PHA              ;SAVE IT
0CE6  AD 00 08             LDA PRTA         ;WIE STEHT DER
0CE9  4A                   LSR A            ;CR-CRLF SWITCH ?
0CEA  90 05                BCC NOLF         ;KEIN ZUSAETZLICHER LF
0CEC  A9 0A                LDA #$0A         ;LF
0CEE  20 F6 0C             JSR OUTALL       ;AUSGEBEN
0CF1  68           NOLF    PLA              ;ORIG. CHAR. HOLEN
0CF2  20 F6 0C     NOL     JSR OUTALL
0CF5  60           ENT     RTS
0CF6                ;ZEICHEN SERIELL AUSGEBEN
0CF6  A2 07        OUTALL  LDX #$07         ;LADE ZAEHLER FUER 7 BIT
0CF8  18                   CLC
0CF9  48                   PHA
0CFA  A9 00                LDA #0           ;SENDE STARTBIT
0CFC  8D 02 08             STA PRTB
0CFF  20 2A 0D             JSR DELAY
0D02  68                   PLA
0D03  8D 02 08     QQQ     STA PRTB         ;SENDE DATENBIT
0D06  4A                   LSR A            ;SCHIEBE DAS NAECHSTE HEREIN
0D07  48                   PHA
0D08  20 2A 0D             JSR DELAY
0D0B  A5 00                LDA PARYB
0D0D  69 00                ADC #0           ;ADDIERE GESENDETES ZU PARY
0D0F  85 00                STA PARYB
0D11  68                   PLA
0D12  CA                   DEX
0D13  D0 EE                BNE QQQ          ;SCHON ALLE GESENDET ?
0D15  A5 00                LDA PARYB
0D17  8D 02 08             STA PRTB         ;GIB PARITY AUS
0D1A  20 2A 0D             JSR DELAY
0D1D  A9 01                LDA #1           ;ODD PARITY
0D1F  85 00                STA PARYB        ;WIEDER INIT
0D21  A9 FF                LDA #$FF
0D23  8D 02 08             STA PRTB         ;SENDE 1 STOPPBIT
0D26  20 2A 0D             JSR DELAY
0D29  60                   RTS
0D2A                ;BAUDRATENGENERATOR
0D2A                ;DELAY. VERZOEGERUNGSSCHLAUFE FUER
0D2A                ; 6,67 MS BITDAUER (1/BAUDRATE)
0D2A                ; =150 BD
0D2A  A0 10        DELAY   LDY #$10         ;NUR INNERHALB EINER PAGE
0D2C  A9 33        DLY1    LDA #$33         ;OUT - BAUD
0D2E  85 03                STA OUTBD
0D30  C6 03        DLY2    DEC OUTBD
0D32  D0 FC                BNE DLY2
0D34  88                   DEY
0D35  D0 F5                BNE DLY1
0D37  60                   RTS
0D38
0D38                       .END
```

Änderungen für verschiedene Baudraten und Paritätsprüfung

Baudrate	Zeit in ms	$0D2B	$0D2D
50	20	10	9A
75	13,3	10	66
100	10	10	4D
150	6,7	10	33
300	3,3	10	19
Paritätsprüfung		$0D1E/0C3C	
Gerade		00	
Ungerade		01	

gibt die Zeichen seriell aus, wobei DELAY den Takt bestimmt. Eine Aufstellung für verschiedene Baudraten zeigt die *Tabelle*.

Etwas zusätzliche Hardware ist vonnöten

Bild 2 zeigt, daß wirklich sehr wenig Hardware zusätzlich zum EMUF erforderlich ist. Sie läßt sich sehr leicht auf dem Lochrasterfeld unterbringen. An PB0 liegt ein Darlington-Transistor, um die 20-mA-Stromschleife zu steuern. Der Schalter an PA0 schaltet die Auto-Linefeed-Funktion ein und aus.

Literatur

[1] *Feichtinger, H.*: Mädchen für alles. mc-EMUF-Sonderheft, Seite 10.

[2] *Löhr, R.*: IEC: Die seriellen Busroutinen. 65XX Micro Mag Nr. 35, Februar 1985, Seite 30.


```
                ;-----
                ;Z80- Befehle mit relativer Adressierung
00FF =          rel     equ     0ffh    ;Und- Maske
0010 =          djnz    equ     10h
0018 =          jr      equ     18h
0020 =          jrnz    equ     20h
0028 =          jrz     equ     28h
0030 =          jrnc    equ     30h
0038 =          jrc     equ     38h
                ;-----
                RDSEC:
0000 D5                 PUSH D
0001 C5                 PUSH B
0002 0ED0               MVI C,11010000B ;laufwerk 0 double dense mini
0004 0601               MVI B,1         ;READ
0006 CD1E00             CALL EXEC
0009 C1                 POP B
000A D1                 POP D
000B 3811               db      jrc,(BOOTERR-$-1) and rel
000D D5                 PUSH D          ;NEXT ADRESSE
000E 110001             LXI D,256       ; DOUBLE DENSE BOOT
0011 19                 DAD D
0012 D1                 POP D
0013 1C                 INR E           ;SEKTORANZAHL + 1
0014 7B                 MOV A,E
0015 FE11               CPI 16+1        ;1..16  erst 17 nicht ok
0017 3803               db      jrc,(RD1-$-1) and rel
0019 1E01               MVI E,1         ;START OVER
001B 14                 INR D
001C 10E2       RD1:    db      djnz,(RDSEC-$-1) and rel

                ;
001E =          BOOTERR EQU     $
001E =          EXEC    EQU     $
001E                    END
```

Relative Sprungbefehle am Beispiel der RDSEC-Routine im mc-CP/M-Computer

Relative Sprünge mit ASM

Zum Lieferumfang des Betriebssystems CP/M gehört ein 8080-Assembler mit dem Namen ASM.COM. Sofern man das BIOS (systemspezifischer Teil von CP/M) neu assemblieren möchte, kommt man um diesen Assembler nicht herum, da er eine durch DDT verarbeitbare HEX-Datei erzeugt. Auch das BIOS des mc-CP/M-Computers ist (deshalb?) unter 8080-Assembler geschrieben.

Sofern man einen Z80 in seinem CP/M-fähigen Computer hat, möchte man auch im BIOS die zusätzlichen Z80-Befehle verwenden. Beim Verfasser entstand der Wunsch, die vielen absoluten Sprünge durch relative Sprünge zu ersetzen, um Speicherplatz zu sparen.

Die Lösung ist zwar umständlich, aber einfach. Sie ist im *Bild* anhand des Programmabschnittes RDSEC aus dem CBIOS des mc-CP/M-Computers (vgl. mc 6/1985, Seite 92) demonstriert. Der Abschnitt ist von 36 auf 30 Byte verkürzt worden. Vier Bytes sind durch die Verwendung relativer Sprünge gespart, zwei weitere, weil die im Original enthaltene unnütze Sicherung des HL-Registerpaares bei dieser Gelegenheit mit gestrichen wurde.

Rolf-Fr. Matthaei

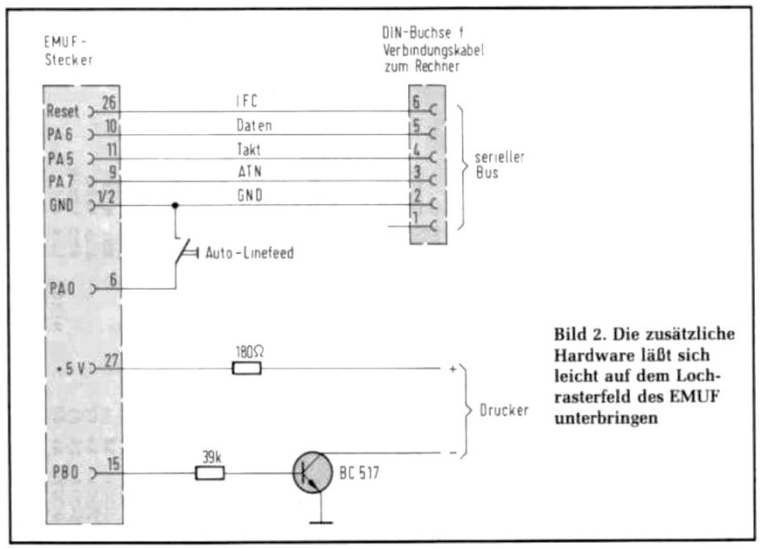

Bild 2. Die zusätzliche Hardware läßt sich leicht auf dem Lochrasterfeld des EMUF unterbringen