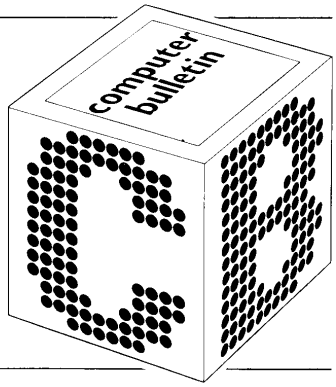
COMPUTER BULLETIN

een supplement van RB gewijd aan microprocessoren en aanverwante onderwerpen

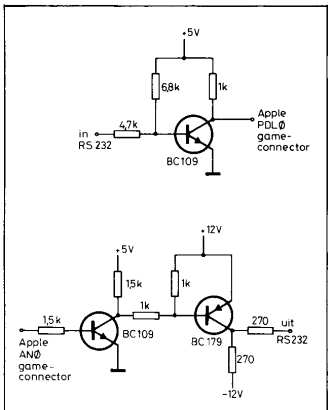


Apple als terminal

RS232-interface voor de Apple

H. J. C. Otten

Een nuttige uitbreiding van de in/uit-mogelijkheden van de Apple is een RS232-interface. We kunnen daarop een printer of een modem aansluiten of, zoals hier als toepassing wordt beschreven, de Apple gebruiken als een terminal voor een andere computer. Om deze RS232-interface te realiseren kan een van de fraaie maar dure prints worden gekocht, die in de handel zijn en vele faciliteiten bieden, of gebruikmaken van de in- en uitgangen van de gameconnector, aangevuld met wat software. Zeker voor eenvoudig gebruik is dit een afdoende mogelijkheid om asynchrone seriële transmissie met een Apple te bedrijven.
In dit artikel wordt de zeer eenvoudige hardware van de RS232-interface beschreven en de machinetaalroutines die de asynchrone seriële communicatie verzorgen. De baudrate van de communicatie is software-instelbaar. Als toepassing wordt een eenvoudige routine gegeven om de Apple als terminal aan bijvoorbeeld een single board computer te hangen.



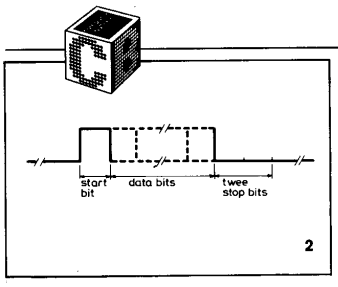
Afb. 1 RS232-interface tussen de gameconnector van de Apple en een randapparaat.

RS232-interface

Voor een eenvoudige RS232-interface is één ingangs- en één uitgangspoort voldoende. De gameconnector van de Apple heeft drie programmeerbare ingangen en drie uitgangen, zodat genoeg in/uit-faciliteiten aanwezig zijn om controlelijnen zoals CTS en DSR toe te voegen. We beperken ons echter tot één seriële ingang en één seriële uitgang.
In afb. 1 is de hardware-interface te zien. Een paar transistoren en de in de Apple aanwezige + en −12 V zijn voldoende.
De hardware kan het beste op een insteekvoet in de gameconnector worden gemonteerd.

RS232-software

De hardware is niet intelligent, zodat het benodigde parallel naar serie en serie naar parallel omzetten



***Afb. 2** Bitpatroon voor seriële asynchrone communicatie.*
***Lijst** Complete programma voor de terminalsimulatie.*

door de software moet worden uitgevoerd. Meestal wordt bij RS232-interfaces van asynchrone communicatie gebruik gemaakt. Daarbij bestaan tussen zender en ontvanger vaste afspraken over zendsnelheid en protocol. Omdat de communicatie serieel is, worden de databits achter elkaar op de lijn gezet, ieder gedurende een vastgelegde tijd: de bittijd. De snelheid wordt uitgedrukt in baud, dat is het omgekeerde van de bittijd.
Het protocol van de asynchrone communicatie maakt het mogelijk dat de ontvanger detecteert dat een nieuw karakter wordt verzonden. Daartoe wordt eerst een startbit (hoog) op de lijn gezet, daarna volgen de databits met de bijbehorende logische toestand van de lijn. Het zenden van een karakter wordt afgesloten met een of twee stopbits die de lijn op het lage logische niveau brengen, dat ook tussen karakters op de lijn aanwezig is. De opeenvolging van niveauwisselingen is in afb. 2 te zien.

Zendroutine

De taak van de zendroutine is het in de accumulator ontvangen karakter op de boven beschreven manier verpakt te verzenden door op de juiste tijdstippen de uitgang laag of hoog te maken. De zendroutine is in de lijst van regel 1170 tot 1420 te vinden.
Er moeten 11 bits worden verstuurd, zodat de routine 11 maal een lus uitvoert. Eerst wordt in de lus de carry-vlag naar de uitgang gebracht, vervolgens een bittijd lang gewacht en dan het karakter met een ROR-instructie geroteerd. Daarbij wordt de eerst hoog gemaakte carry in het LSB van het karakter geschoven en het MSB in de carry.
Door de lus te beginnen met de carry laag wordt het startbit op de lijn gezet. Door het schuiven komen alle databits vervolgens op de lijn en na de negende maal schuiven de stopbits, omdat de carry voor het schuiven hoog was.

Ontvangroutine

De ontvangroutine, in de lijst te vinden op regel 910 tot 1130, probeert eerst het startbit te detecteren, in feite de niveauverandering die het begin van het startbit aangeeft. Als het startbit is gedetecteerd wordt een halve bittijd gewacht en daarna een hele bittijd. We zitten nu midden in het eerste databit, dat in het te ontvangen karakter wordt geschoven. Door in een lus acht maal in het midden van een databit het niveau van het databit te testen wordt het hele karakter ontvangen.
Voor de zekerheid wachten we nu tot het midden van het eerste stopbit. Door in het midden van een databit de waarde van het bit te testen, zijn we een stuk minder gevoelig geworden voor snelheidsverschillen tussen zender en ontvanger. Deze methode van detectie laat snelheidsverschillen van 5 % toe. Aan het begin van ieder karakter synchroniseert de ontvangroutine weer.

Timing

Zowel de zend- als de ontvangroutine maken gebruik van een wachtroutine (DELAY) om een bittijd te wachten. De ontvangroutine gebruikt ook nog een wachtroutine die een halve bittijd duurt (DEHALF). Deze routines, in de lijst te vinden op regel 1460 – 1590, geven een programmeerbare vertraging door een constante in de zeropagelocatie BAUD te zetten. Routine DELAY duurt BAUD × 5 + 1 microseconden. Voor een transmissiesnelheid van 1200 baud moet BAUD de waarde 166 ($A6) hebben (DELAY duurt dan 1/1200 = 833 microseconden), voor andere transmissiesnelheden moet een an-

Lijst

```
      HJCO 81 6502 ASSEMBLER V2.0    PAGE 01

0010:
0020:              ********************************************************
0030:              *                                                      *
0040:              *        APPLE II                                      *
0050:              *                                                      *
0060:              *  TERMINAL SIMULATION                                 *
0070:              *                                                      *
0080:              *  ASYNCHR RS232 INPUT OUTPUT ROUTINES                 *
0090:              *                                                      *
0100:              *  COMPUTER BULLETIN 1981  HJC OTTEN                   *
0110:              ********************************************************
0120:
0130:              ; DEFINES
0140:
0150: 2000                 ORG   $2000
0160:
0170:              ; APPLE ZERO PAGE USE
0180:
0190: 2000         CH      =     $0024  ; RELATIVE CURSOR POSITION
0200: 2000         BASL    =     $0028  ; BASE ADDRESS SCREEN
0210: 2000         BAUD    =     $00FB  ; BAUD RATE CONSTANT
0220: 2000         CHAR    =     $00FE  ; DATAFIELD
0230:
0240:              ; APPLE I/O ADRESSES
0250:
0260: 2000         KB      =     $C000  ; KEYBOARD INPUT
0270: 2000         KBSTR   =     $C010  ; KEYBOARD STROBE
0280: 2000         MARK    =     $C058  ; GAME AN0 FLIP FLOP SET
0290: 2000         SPACE   =     $C059  ; GAME AN0 FLIP FLOP RESET
0300: 2000         PDLSW   =     $C061  ; GAME SW0 INPUT IN BIT 7
0310:
0320:              ; APPLE MONITOR USE
0330:
0340: 2000         BS      =     $FC10  ; NON DESTROYING BACKSPACE
0350: 2000         VIDOUT  =     $FDF0  ; PRINT CHARACTER IN A ON SCREEN
0360:
0370:
0380:              ; MAIN ROUTINE
0390:              ;  TERMINAL SIMULATION ROUTINE
0400:
0410: 2000 A4 24   TERSIM LDY   CH     ; BEGIN  * TERMINAL SIMULATION *
0420: 2002 B1 28          LDAIY BASL   ;  REPEAT
0430: 2004 48             PHA          ;    SHOW CURSOR POSITION
0440: 2005 29 3F          ANDIM $3F    ;    IF KEY PRESSED THEN
0450: 2007 91 28          STAIY BASL   ;      REMOVE CURSOR FROM SCREEN
0460: 2009 2C 00 C0 TEST  BIT   KB     ;      CHARACTER := KEYCODE
0470: 200C 30 11          BMI   KEVIN  ;      OUTPUT (CHARACTER)
0480: 200E 2C 61 C0       BIT   PDLSW  ;      SHOW (CHARACTER)
0490: 2011 30 F6          BMI   TEST   ;    ENDIF
0500: 2013 68             PLA          ;    IF CHARACTER AT INPUT THEN
0510: 2014 91 28          STAIY BASL   ;      REMOVE CURSOR FROM SCREEN
0520: 2016 20 5F 20       JSR   INPUT  ;      INPUT (CHARACTER)
0530: 2019 20 30 20       JSR   SHOW   ;      SHOW (CHARACTER)
0540: 201C 4C 00 20       JMP   TERSIM ;    ENDIF
0550: 201F 91 28   KEVIN  STAIY BASL   ;  UNTIL FOREVER
0560: 2021 AD 00 C0       LDA   KB     ; END * TERMINAL SIMULATION *
0570: 2024 2C 10 C0       BIT   KBSTR  ;
0580: 2027 20 30 20       JSR   SHOW   ;
0590: 202A 20 68 20       JSR   OUTPUT ;
0600: 202D 4C 00 20       JMP   TERSIM ;
0610:
0620:              ; SHOW CHARACTER ON APPLE SCREEN ROUTINE
0630:
0640: 2030 C9 8A   SHOW   CMPIM #8A    ; BEGIN * SHOW CHARACTER ON SCREEN *
0650: 2032 D0 07          BNE   BACKSP ;
0660: 2034 A0 00          LDYIM #00    ;   IF CHARACTER = LINEFEED THEN
0670: 2036 84 24          STY   CH     ;     COLUMNPOSITION := 0
0680: 2038 4C F0 FD       JMP   VIDOUT ;     SEND (LINEFEED)
0690: 203B C9 88   BACKSP CMPIM #88    ;   ENDIF
0700: 203D D0 0D          BNE   CARRET ;   IF CHARACTER = BACKSPACE THEN
0710: 203F 48             PHA          ;     * MAKE IT DESTROYING BACKSPACE *
0720: 2040 20 10 FC       JSR   BS     ;     SEND (BACKSPACE)
0730: 2043 A9 A0          LDAIM #A0    ;     SEND (SPACE)
0740: 2045 20 F0 FD       JSR   VIDOUT ;     SEND (BACKSPACE)
0750: 2048 68             PLA          ;   ENDIF
0760: 2049 4C F0 FD       JMP   VIDOUT ;   IF CHARACTER = CARRIAGE RETURN THEN
0770: 204C C9 8D   CARRET CMPIM #8D    ;     COLUMNPOSITION := 0
0780: 204E D0 05          BNE   ESC    ;   ENDIF
```

dere waarde voor BAUD worden berekend of beneden 600 baud een tragere wachtroutine worden gegeven.

Terminalsimulatie

Met de zend- en ontvangroutine en een derde routine om een karakter op het scherm te zetten is eenvoudig een terminal te simuleren met de Apple. De terminalsimulatie, in de lijst vanaf regel 410 tot 600 te vinden, bestaat uit een oneindige lus. Een terminal heeft twee functies: de bij een ingedrukte toets behorende karaktercode moet naar de computer worden verstuurd en een van de computer ontvangen karakter moet op het scherm worden getoond.
Deze twee functies voert de terminalroutine uit door met twee BIT-testen te controleren of er een toets is ingedrukt (BIT KB) of een startbit van een te ontvangen karakter is aangekomen (BIT PDLSW). Als een toets is ingedrukt wordt de karaktercode verzonden. Deze terminal werkt in half-duplex, zodat de toets ook op het scherm wordt gezet door de „toon karakter"-routine aan te roepen. Voor full-duplex moet dit worden verwijderd.
Bij de detectie van een startbit wordt naar de ontvangroutine gesprongen en daarna het ontvangen karakter op het scherm gezet. Zolang er nog geen toets is ingedrukt of een karakter wordt ontvangen, wordt een cursor op het scherm getoond (niet knipperend) op de huidige cursorpositie. Bij een actie wordt de cursor van het scherm verwijderd. In de terminalsimulatielus wordt het binnenhalen van een toetsindruk onmiddellijk afgehandeld. Het tonen van een karakter op het scherm is in een aparte routine, „toon karakter" ondergebracht die ook de cursorpositie bijwerkt.

Toon karakter

De Apple heeft voor het tonen van een karakter op het scherm een routine in ROM (VIDOUT) die bijna alle werk, dat we aan „toon karakter" willen overlaten, uitvoert. Alleen de verwerking van de controlekarakters linefeed, backspace en carriage return vereisen een aparte aanpak.
Als een linefeed wordt ontvangen, wordt eerst een carriage return uitgevoerd door de cursor-kolompositie op nul te zetten en dan pas de linefeedroutine van VIDOUT.
De backspaceroutine van de Apple zet de cursor een positie op de regel terug maar laat het karakter, dat we willen verbeteren, staan. Een backspace, die het karakter wegveegt, wordt hier gesimuleerd door eerst een Apple-backspace uit te voeren, dan een spatie te printen die het karakter wegveegt en om de cursor op de goede positie te krijgen weer een Apple-backspace.
De carriage returnroutine van de Apple voert ook een linefeed uit; hier beperken we ons tot een echte carriage return door alleen de cursor-kolompositie op nul te zetten.
Om de terminallus te kunnen verlaten, wordt het escapekarakter gebruikt. Daartoe moet de terminalsimulatie als een subroutine worden aangeroepen, bijvoorbeeld met een CALL vanuit Basic. Escape forceert een terugkeer uit de subroutine terminalsimulatie.

Gebruik als terminal

Aan de softwareoplossing kleven een paar kleine bezwaren die aan de verbonden computer een simpele eis stellen. Het probleem zit in de linefeedroutine van de Apple, die vrij veel tijd kost (2 ms). Gedurende het uitvoeren van een linefeed is de terminal doof voor invoer. Merkwaardigerwijze is dit een zelfde situatie, als bij printende terminals zoals teletypes optreedt. De oplossing is simpel; na het versturen van een linefeed moet de zender even wachten met het zenden van een volgend karakter. De meeste computers voorzien hier al in, anders is een toevoeging aan de zendroutine, die na een linefeed even wacht, voldoende.

```
0790: 2050 A9 00          LDAIM $00     ;   IF CHARACTER = ESCAPE THEN
0800: 2052 85 24          STA   CH      ;     + COMPLETE ESCAPE FROM TERMINAL SIMULATION
0810: 2054 60             RTS           ;     EXIT
0820: 2055 C9 9B    ESC   CMPIM #$9B    ;   ENDIF
0830: 2057 F0 03          BEQ   EXIT    ;   IF CHARACTER = NORMAL THEN
0840: 2059 4C F0 FD       JMP   VIDOUT  ;     SEND (CHARACTER)
0850: 205C 68       EXIT  PLA           ;   ENDIF
0860: 205D 68             PLA           ; END * SHOW CHARACTER ON SCREEN *
0870: 205E 60             RTS           ;
0880:
0890:               ; INPUT CHARACTER FROM GAME SW0 ROUTINE
0900:
0910: 205F 8A       INPUT TXA           ; BEGIN * INPUT CHARACTER *
0920: 2060 48             PHA           ;   * READ CHARACTER FROM AN0 ASYNCHR *
0930: 2061 98             TYA           ;   * AN0 = BIT 7 #C061              *
0940: 2062 48             PHA           ;
0950: 2063 A0 08          LDYIM $08     ;   BITCOUNT := 8
0960: 2065 2C 61 C0 IN    BIT   PDLSW   ;   WHILE NOT STARTBIT DO TEST FOR STARTBIT
0970: 2068 30 FB          BMI   IN      ;   DELAY BITTIME
0980: 206A 20 AE 20       JSR   DELAY   ;   DELAY HALF BITTIME
0990: 206D 20 B4 20       JSR   DEHALF  ;   WHILE BITCOUNT > 0 DO
1000: 2070 AD 61 C0 INP   LDA   PDLSW   ;     READ BIT
1010: 2073 29 80          ANDIM #80     ;     SHIFT BIT IN CHARACTER
1020: 2075 46 FE          LSR   CHAR    ;     BITCOUNT := BITCOUNT - 1
1030: 2077 05 FE          ORA   CHAR    ;   ENDWHILE
1040: 2079 85 FE          STA   CHAR    ;   DELAY BITTIME
1050: 207B 88             DEY           ;   SET MSB OF CHARACTER
1060: 207C D0 F2          BNE   INP     ;   * FOR APPLE VIDEO ROUTINE *
1070: 207E 20 AE 20       JSR   DELAY   ; END * INPUT CHARACTER
1080: 2081 68             PLA           ;
1090: 2082 A8             TAY           ;
1100: 2083 68             PLA           ;
1110: 2084 AA             TAX           ;
1120: 2085 09 80          ORAIM $80     ;
1130: 2087 60             RTS           ;
1140:
1150:               ; OUTPUT CHARACTER TROUGH GAME AN0
1160:
1170: 2088 48       OUTPUT PHA          ; BEGIN * OUTPUT CHARACTER *
1180: 2089 98             TYA           ;   * SEND CHARACTER ASYNCHR TROUGH AN0 *
1190: 208A 48             PHA           ;   * AN0 FLIPFLOP OUTPUT C058 C059    *
1200: 208B 8A             TXA           ;   * 1 STARTBIT , 8 DATABITS , 2 STOP  *
1210: 208C 48             PHA           ;
1220: 208D BA             TSX           ;   CARRY := STARTBIT
1230: 208E BD 03 01       LDARX $0103   ;   BITCOUNT := 11
1240: 2091 A0 0B          LDYIM #0B     ;   WHILE BITCOUNT > 0 DO
1250: 2093 18             CLC           ;     OUTPUT := CARRY
1260: 2094 48       OUT   PHA           ;     DELAY BITTIME
1270: 2095 B0 05          BCS   MRKOUT  ;     CARRY := STOPBIT
1280: 2097 AD 59 C0       LDA   SPACE   ;     SHIFT CHARACTER RIGHT
1290: 209A 90 03          BCC   WAIT    ;     LSB CHAR := CARRY
1300: 209C AD 58 C0 MRKOUT LDA  MARK    ;     CARRY := MSB CHAR
1310: 209F 20 AE 20 WAIT  JSR   DELAY   ;   ENDWHILE
1320: 20A2 68             PLA           ;
1330: 20A3 38             SEC           ; END * OUTPUT CHARACTER *
1340: 20A4 6A             RORA          ;
1350: 20A5 88             DEY           ;
1360: 20A6 D0 EC          BNE   OUT     ;
1370: 20A8 68             PLA           ;
1380: 20A9 AA             TAX           ;
1390: 20AA 68             PLA           ;
1400: 20AB A8             TAY           ;
1410: 20AC 68             PLA           ;
1420: 20AD 60             RTS           ;
1430:
1440:               ; DELAY BITTIME ROUTINE
1450:
1460: 20AE A4 FB    DELAY LDY   BAUD    ; BEGIN * DELAY BITTIME *
1470: 20B0 CA       XD    DEX           ;   DELAY (BAUD)*5 + 1 MICROSEC
1480: 20B1 D0 FD          BNE   XD      ; END * DELAY BITTIME *
1490: 20B3 60             RTS           ;
1500:
1510:               ; DELAY HALF BIT TIME ROUTINE
1520:
1530: 20B4 A5 FB    DEHALF LDA  BAUD    ; BEGIN * DELAY HALF BITTIME *
1540: 20B6 4A             LSRA          ;   DELAY (BAUD/2)*5 + 6 MICROSEC
1550: 20B7 AA             TAX           ; END * DELAY HALF BITTIME *
1560: 20B8 CA       XDH   DEX           ;
1570: 20B9 D0 FD          BNE   XDH     ;
1580: 20BB 60             RTS           ;
1590:
```

Rectificatie

In het artikel „Apple als terminal" zijn een paar storende fouten in de programmalijst geslopen. Met onderstaande verbeteringen is het programma wel te gebruiken.

Geheugen-locatie	**Hex-inhoud**	**Symbool**
205F	EA	NOP
2060	EA	NOP
2083	A5	LDA
2084	FE	CHAR
20AE	A5	LDX