

Epifiet

Communicatie via printerpoort

MSX Computer & Club Magazine nummer 76 - juni 1995

Rob van Gans

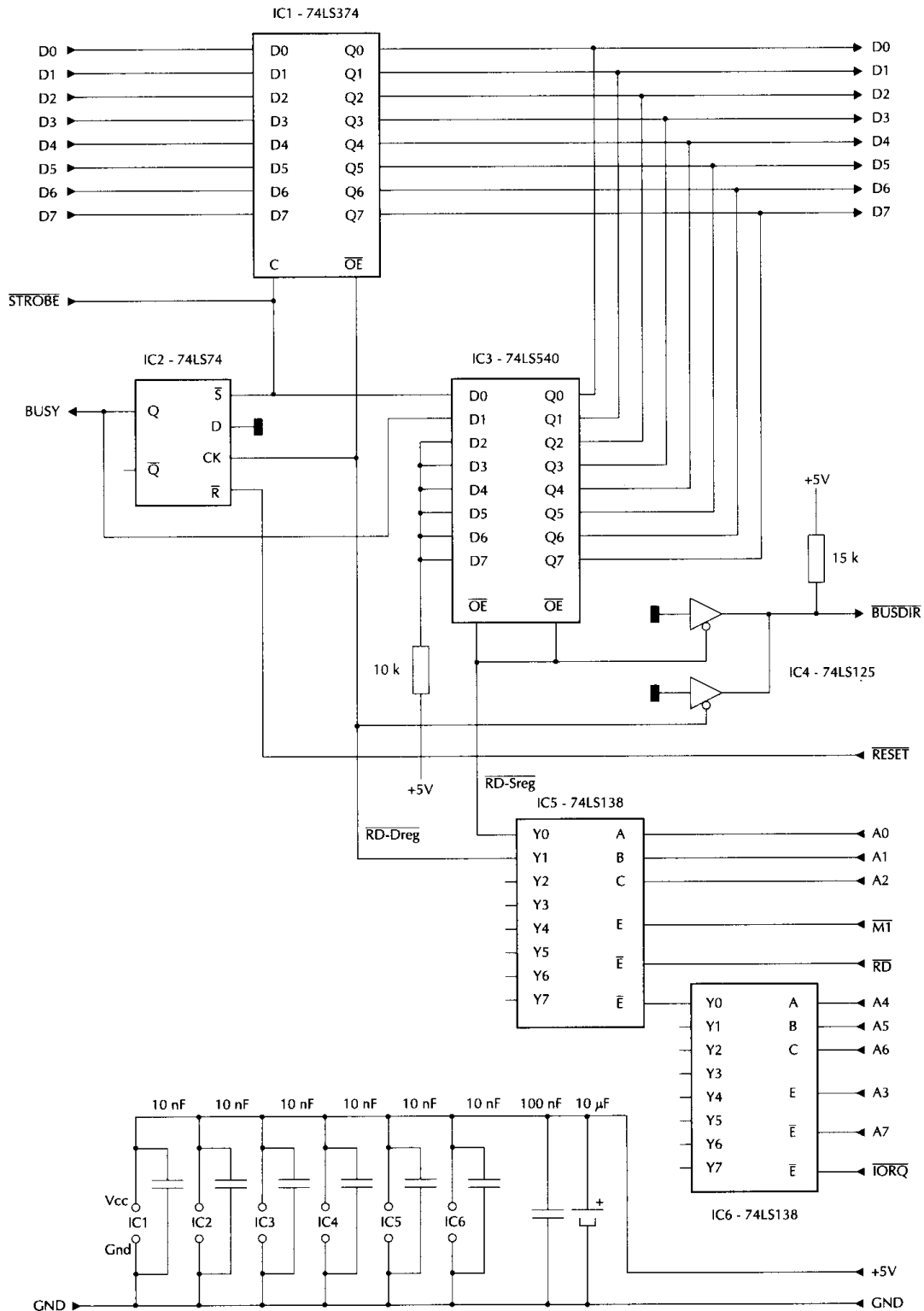
Scanned and converted to PDF by HansO, 2001

Een interface voor de printerpoort, waarmee uw MSX een andere computer kan uitlezen. Met de software is die andere computer zelfs te besturen. Aanbevolen voor zelfbouwers.

De naam Epifiet is met opzet gekozen: net als epifyten, mossen die op bomen groeien zonder te parasiteren, bevindt deze interface zich namelijk wel bij zijn gastheer, maar is deze niet tot last. Integendeel, hij leeft in goede harmonie met hem samen. Het is zelfs mogelijk om met deze interface BASIC programma's over te zenden en ook onder MSXDOS zijn er diverse mogelijkheden.

Zelfbouw

Het is goed te doen de interface zelf te bouwen. De kosten vallen best mee. Alles bij elkaar schat ik ongeveer f 25,-aan materiaalkosten. Doe je er een luxe doosje omheen, wordt het geheel een paar gulden duurder. Het schema lijkt ingewikkeld, maar is goed te begrijpen als je het in een aantal functionele blokken opdeelt. Ik ga daar nu dieper op in.



Opzet schema

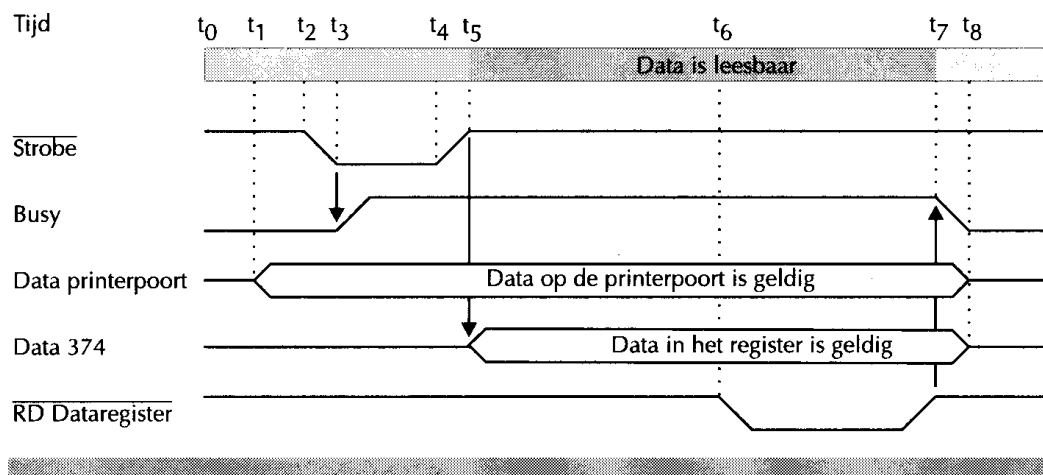
Aan de linkerkant bevindt zich een kabel met een stekker voor in de printerpoort van de eerste computer, die net wordt aangesloten alsof het een printer is. Aan de rechterkant zit de aansluiting naar de slotconnector van de andere computer. IC1 is een achtvoudige D-flip-flop met tristate uitgang. De printerpoort zet hier zijn data in.

Dit IC slaat de aangeboden data op, als de Clock-in-gang (C) van laag naar hoog gaat. De afzonderlijke toestanden hoog en laag doen niets. De opgeslagen data verschijnt niet meteen op de Q-uitgangen: het IC staat nog in de tristate mode, de uitgangen zijn nog hoogohmig. Pas als de Output Enable-ingang (OE) laag is komt de opgeslagen data op de Q-uitgangen te staan. IC2 is een Set-Reset-D-flipflop. Dit IC zorgt voor het juiste BUSY signaal. In dit IC bevinden zich twee van deze flipflop's waarvan er maar één wordt gebruikt.

IC3 is een achtvoudige, inverterende tristate busbuffer. Hiermee wordt de toestand van het STROBE en BUSY signaal bekeken. Dit IC geeft de data op de D-ingangen omgekeerd—laag wordt hoog en hoog wordt laag—aan de Q-uitgangen door als beide Output Enable-ingangen (OE) laag zijn; anders bevinden de Q-uitgangen zich in de tristate mode en zijn ze dus hoogohmig. De IC's 5 en 6 zorgen samen voor de adressering van IC 1 en IC3. Een instructie INP (8) maakt uitgang YO Read Status register (RD-Dreg) laag en een instructie INP (9) maakt uitgang Y1 Read Data register (RD-Dreg) laag.

Ten slotte IC4: de tristate buffers. Dit IC maakt BUSDIR laag als een van beide registers wordt gelezen. Dit is nodig omdat de computer anders niet weet in welk slot deze zich bevinden. In dit IC zitten vier van deze buffers, waarvan er maar twee worden gebruikt.

Een van de belangrijkste aspecten van deze interface, is dat hij zich richting printerpoort moet gedragen als een printer. Hiervoor zorgt IC2. Allereerst de eigenschappen van dit IC. De niet-Q uitgang—Q met een streepje er boven—is altijd het omgekeerde van de Q-uitgang. De niet Q-uitgang wordt hier niet gebruikt. Als de Set-ingang (S) laag is, wordt de Q-uitgang hoog, als de Reset ingang R laag is, wordt de Q-uitgang laag. Set en Reset mogen nooit beide laag zijn; dit is een verboden toestand. Dat is natuurlijk begrijpelijk, want wat moet Q worden, hoog of laag? Beide hoog mag wel, dan wordt de laatste toestand vastgehouden. Op dit vasthouden is een uitzondering: als de Clock-ingang (C) van laag naar hoog gaat, krijgt Q de waarde van de Data-ingang (D). In ons geval is D altijd laag. Voor deze Clock-ingang geldt hetzelfde als voor die van IC1. Bekijk bij het nu volgende ook het tijddiagram.



Tijdbalk

Bovenaan staat een tijdbalk met de aanduidingen t_0 tot en met t_8 . Hier verwijs ik regelmatig naar. Als de computer wordt aangezet of de resetknop wordt ingedrukt, wordt het signaal RESET even laag. De RESET van IC2 dus ook. Als gevolg hiervan wordt Q en dus ook BUSY laag. Dit is de beginsituatie t_0 . De computer die iets naar

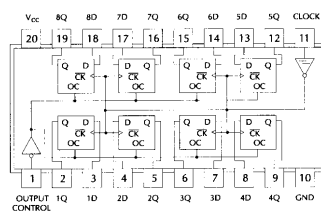
de printerpoort wil sturen, bekijkt eerst of BUSY wel laag is. Dit is zo, dus er mag data worden verstuurd. Als BUSY hoog is, doet hij dat niet, maar blijft BUSY testen totdat BUSY alsnog laag is. De volgende stap t1 is dat de computer data op zijn printerpoort zet. Daarna, t2, maakt de computer de STROBE laag, met als gevolg t3, STROBE Set IC2 waardoor Q en dus ook BUSY hoog wordt. Vervolgens t4 maakt de computer de STROBE weer hoog, met als gevolg, op t5, dat de DATA in IC1 wordt opgeslagen. Deze toestand blijft staan zolang de andere computer het data-register IC1 niet uitleest. Dit is het tijdvak tussen t5 en t6.

Nu gaan we de zaak vanaf de andere kant bekijken. Zoals we hebben kunnen zien, is er pas geldige data in IC1 aanwezig als zowel BUSY als STROBE hoog zijn. Deze zijn verbonden met DO en DI van IC3. De ingangen D2 tot en met D7 zijn altijd hoog, doordat ze via een weerstand zijn verbonden met +5V.

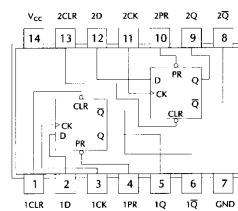
Hoe nu verder? Allereerst moet met INP (8) worden gekeken of er wel geldige data aanwezig is. RD-Sreg wordt tijdens het lezen laag waardoor IC3 zijn D-ingangen op het slot zet. Het resultaat moet 0 (NUL) zijn. Immers IC3 keert alles om. Als dit inderdaad 0 oplevert, mag het data-register worden gelezen. Dit doen we met INP (9). Terug naar het tijddiagram t6. Tijdens het lezen van data-register IC1 wordt RD-Dreg laag; de data wordt op het slot gezet en in de computer gehaald. Bij t7 wordt RD-Dreg weer hoog en dus ook Clock van IC2, het gevolg hiervan t8 is dat Q en dus ook BUSY weer laag wordt. Hoewel de data in het data-register nog aanwezig is, moeten we het toch als ongeldig beschouwen omdat we door middel van BUSY-laag te kennen geven dat er nieuwe data naar de printerpoort mag worden verstuurd.

We zijn nu het cirkeltje rond en weer bij t0 aangekomen. Als je het tijddiagram bekijkt, zou je denken dat het opslaan van de data in IC1 overbodig is. Er zijn echter computers die in de periode t5 tot t6 alvast nieuwe data op hun printerpoort zetten en alleen wachten met het in de printer clocken tot BUSY laag is. Dan is vlak na t5 de data op de printerpoort niet meer geldig.

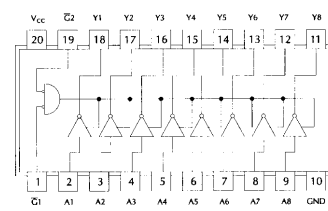
Tot slot nog een opmerking over IC3. Hier is een 74LS540 gebruikt, maar je kunt ook een 74LS240 gebruiken. Beide IC's hebben voor- en nadelen. De 540 is iets duurder maar heeft alle D-in-gangen aan de ene en alle Q-uitgangen aan de andere kant zitten. Dit is erg gemakkelijk bij zelfbouw. Hoewel de 240 iets goedkoper is, heeft deze het nadeel dat de D-ingangen en Q-uitgangen door elkaar zitten. Bij massafabricage is dit geen probleem en speelt de prijs mee.



■ IC1: 74LS374

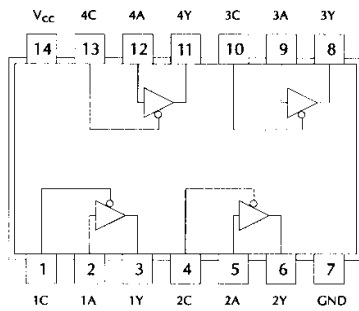
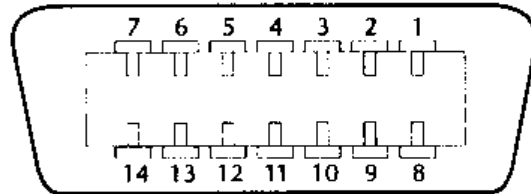


■ IC2: 74LS74

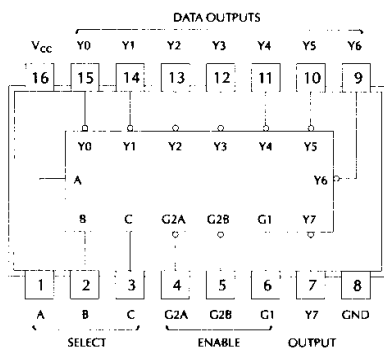


■ IC3: 74LS540

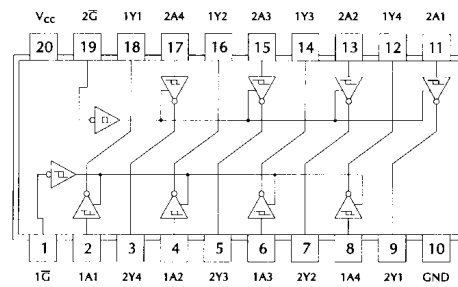
Pin	Name	I/O
1	PSTB	O
2	PDB0	O
3	PDB1	O
4	PDB2	O
5	PDB3	O
6	PDB4	O
7	PDB5	O
8	PDB6	O
9	PDB7	O
10	NC	
11	BUSY	I
12	NC	
13	NC	
14	GND	



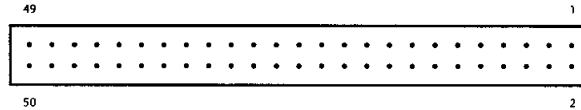
■ IC4: 74LS125



■ IC5: 74LS138



■ Alternatief voor IC3: 74LS540



Pin	Name	I/O	Pin	Name	I/O
1	$\overline{CS1}$	O	2	$\overline{CS2}$	O
3	$\overline{CS2}$	O	4	\overline{SLTSL}	O
5	Reserved	-	6	RFSH	O
7	WAIT	I	8	INT	I
9	\overline{MT}	O	10	\overline{BUSDIR}	I
11	\overline{TORQ}	O	12	\overline{MERQ}	O
13	\overline{WR}	O	14	RD	O
15	RESET	O	16	Reserved	-
17	A9	O	18	A15	O
19	A11	O	20	A10	O
21	A7	O	22	A6	O
23	A12	O	24	A8	O
25	A14	O	26	A13	O
27	A1	O	28	A0	O
29	A3	O	30	A2	O
31	A5	O	32	A4	O
33	D1	I/O	34	D0	I/O
35	D3	I/O	36	D2	I/O
37	D5	I/O	38	D4	I/O
39	D7	I/O	40	D6	I/O
41	GND	-	42	CLOCK	O
43	GND	-	44	SW1	-
45	+5V	-	46	SW2	-
47	+5V	-	48	+12V	-
49	SOUNDIN	I	50	-12V	-

Software

Epifiet is gemakkelijk in de omgang, je hoeft niets in te stellen. Maar, zoals eigenlijk met alle hardware het geval is, er gebeurt niets zonder de juiste programmatuur. Er is één regel die je met Epifiet altijd moet volgen: lees eerst het Status register met INP (8) om te kijken of er wel geldige data is. Als het resultaat 0 (nul) is, dan is er geldige data. Pas als je zo hebt geconstateerd dat er inderdaad geldige data is, lees je de data in met INP (9). Bedenk hierbij wel dat je deze data maar één keer per karakter kunt inlezen. Het handigste is om de data meteen in een variabele te zetten. Dan kun je het daarna altijd nog bewerken, er een stringvariabele van maken, het aan een andere string toevoegen of wat dan ook.

Testen

We gaan nu eens kijken of alles het wel doet. Hiervoor heb ik een paar kleine programmaatjes gemaakt. Ze zijn alleen maar bedoeld om je een paar dingen te laten zien en een indruk te geven van hoe je de interface kunt gebruiken. Tik de twee programma's EPI-BRON.BAS en EPI-DOEL.BAS in en zet ze op een schijf. Diskabonnees vinden de files kant en klaar op de A-disk. De commentaarregels 10 tot en met 30 mag je zondermeer weglaten.

Steek de interface in een slot van een MSX computer—dit is de doelcomputer—en sluit de stekker van de interface aan op de printerpoort van een tweede computer, de broncomputer. Zet beide computers aan. Als eerste laad en run je EPI-DOEL.BAS in de doelcomputer. Geef nu op de broncomputer het commando om iets op de printer af te drukken, bijvoorbeeld:

```
LPRINT "Hallo dit is een test"
```

Nadat je de returntoets hebt ingedrukt, zal deze tekst via de printerpoort naar Epifiet worden verstuurd en zo op het scherm van de doelcomputer verschijnen. Als dit allemaal goed gaat, werkt de interface!

De volgende stap

We gaan een stapje verder. Laad en run EPI-BRON.BAS in de broncomputer. Nu wordt iedere karaktercode van het toetsenbord van de broncomputer direct op zijn eigen scherm en via de printerpoort en Epifiet naar het scherm van de doelcomputer gestuurd. Ook de bijzondere toetsen zoals de cursortoetsen, home, esc, tab en del. We gebruiken nu alleen het toetsenbord van de broncomputer. Probeer bijvoorbeeld shift+home en beide schermen zijn leeg.

Doordat MSX de VT52-escape sequences ondersteunt, zijn deze codes ook te gebruiken. Zet eens een scherm vol met tekst, desnoods volstreekte onzin, zet de cursor ergens in het midden en druk na elkaar escape en de hoofdletter L in. Er wordt nu een lege regel tussengevoegd. Er is een hele rij van dit soort VT52-escape codes. Verscheidene handboeken geven hierover meer informatie.

Machinetaalroutine

We gaan nu wat ingewikkelder te werk met programmatuur voor de doelcomputer. Voer de listing POKENEW.BAS in, of haal hem van het diskabbonnement. Zorg er in ieder geval voor, dat je het programma op schijf hebt staan, want voor je het weet ben je het weer kwijt.

Het programma POKENEW doet twee dingen: het verhoogt de onderkant van het geheugengebied voor BASIC, waardoor er ruimte vrijkomt voor een stukje machinetaal, en plaatst een tekst in de toetsenborbuffer. Deze tekst zorgt er na afloop van het programma voor, dat automatisch een volgend programma wordt gestart. Er staan nogal wat commentaarregels in om duidelijk te maken hoe dat in zijn werk gaat. Laad en run nu in de doelcomputer het programma POKENEW. Dit levert de foutmelding/J7e notfound. Dat klopt, want er wordt geprobeerd het bestand EPIFKEYB.BIN op te starten en dat hebben we nog niet gemaakt. Van deze foutmelding trekken we ons voor deze keer niets aan. Ons belangrijkste doel voor nu, een stukje geheugen vrijmaken voor machine taal, is wel gelukt.

Tik nu het programma EPIFKEYB.BAS in en save ook dit eerst, voor je het weer kwijt bent, of haal het van het diskabbonnement. Dit programma poke't data in het vrijgemaakte geheugen en zet deze machinetaal meteen op schijf. Voor de kenners staat de oorspronkelijke pro-grammalisting in assembleertaal hiernaast afgedrukt. Als EPIFKEYB.ASM is de source ook op het diskabbonnement te vinden.

Reset de doelcomputer en laad en run POKENEW.BAS. Nu moet alles vanzelf goed komen. Wat is het resultaat van dit alles? Er is, door het ombuigen van de hook HKEYI, een stukje programma tussen de afhandeling van de interrupts gezet, zodanig dat behalve naar het toetsenbord, nu ook gekeken wordt of Epifiet iets aan te bieden heeft. Is dat het geval, worden de door Epifiet opgehaalde karakters in de toetsbordbuffer gezet. Hierdoor is er geen BASIC programma meer voor nodig om Epifiet uit te lezen. Dit biedt uiteraard meer mogelijkheden.

Carriage return en line feed

Nog een opmerking: als je op de broncomputer de returntoets indrukt, wordt er

behalve de return, CHR\$(13), ook een LineFeed, CHR\$(10), naar de printerpoort gestuurd. Om ongewenste effecten te voorkomen, wordt er in het programma EPIFKEYB.BIN getest of de uit Epifiet gehaalde karaktercode een LineFeed is. Als dat zo is, wordt deze onderdrukt, oftewel niet naar de toetsenbordbuffer doorgestuurd. In de volgende paragrafen ga ik er van uit, dat nadat de doelcomputer is aangezet of gereset, het programma POKE-NEW—en dus ook EPIFKEYB.BIN—is ingeladen en gerund. De doelcomputer volgt daarmee alles van de bron, zodat de verdere bediening vanaf die ene computer kan geschieden.

Voorbeelden

De mogelijkheden van Epifiet zijn bijna onbeperkt. Een paar ideeën worden hieronder uitgewerkt.

Versturen van BASIC programma's

Het versturen van een BASIC programma kan via Epifiet als volgt.

Laad een BASIC programma in de broncomputer:

```
LOAD "filenaam.bas" Toets daarna in:
```

```
LLIST
```

Dit commando drukt normaal gesproken een listing op de printer af. Het programma wordt nu naar de doelcomputer gestuurd en komt in zijn geheugen te staan alsof je het op de doelcomputer hebt ingetoetst.

Stop een schijfje in de doelcomputer en toets op de broncomputer in:

```
LPRINT "SAVE"+CHR$(34)+ "filenaam.bas"+CHR$(34)
```

Verbaas je, niet maar het programma dat je zojuist naar de doelcomputer hebt verstuurd, wordt ook op die computer op diskette gezet. Vervolgens voer je—nog altijd op de broncomputer—in:

```
LPRINT "RUN"
```

Het zojuist verstuurde programma wordt daar nog gestart ook. Als je dat programma met het intoetsen van een karakter (dus niet via control-stop) kunt beëindigen, dan kan dat met behulp van een geschikte LPRINT ook vanuit de broncomputer.

Directory bekijken op andere computer

Zorg dat er een schijfje met een aantal bestanden erop in de doelcomputer zit. Type nu op de broncomputer:

```
LPRINT "FILES"+CHR$(13)
```

Op het scherm van de doelcomputer verschijnt vervolgens de directory van dat schijfje. Omdat de communicatie via Epifiet één kant op werkt, is het niet mogelijk de inhoudsopgave op het scherm van de broncomputer te zien te krijgen.

Schermuitvoer onder DOS omleiden

Ook onder MSXDOS is Epifiet bruikbaar. Laad MSXDOS in de broncomputer en toets control-P. Nu stuurt DOS alle schermuitvoer ook naar de printer en dus ook naar Epifiet. Probeer dit bijvoorbeeld maar eens:

DIR

De directory van het schijfje in de broncomputer komt ook op het scherm van de doelcomputer.

Tekst direct invoeren in tekstverwerker

Als vierde voorbeeld nog een erg handige optie. Ik heb het ROM-programma MSXTRA in de broncomputer gezet. Dat is een assembler/disassembler die een gedisassembleerde listing ook naar printer kan sturen. Nadat de doelcomputer is opgestart en POKENEW.BAS gerund is, laad en run ik de tekstverwerker Tasword2. Als ik nu MSXTRA een disassembler listing naar de printer laat sturen, komt die zelfs binnen bij Tas-word2. Met deze optie heb ik ook de listing van EPIFKEYB.ASM samengesteld. Waarom zou je die nog met de hand intypen?

Tot slot

Er zijn natuurlijk nog veel meer mogelijkheden; experimenteer naar hartelust. Ik wens de bouwers van Epifiet veel plezier met dit goedkope, nuttige en handige stukje hardware.

Listing EPI-BRON

```
10 ' EPI-BRON.BAS
20 ' Bij Epifiet, MCCM 76
30 '
40 CLS:LOCATE ,,1
50 ON STOP GOSUB 90:STOP ON
60 D$=INKEY$
70 IF D$="" GOTO 60
80 LPRINT D$;:PRINT D$;:GOTO 60
90 LOCATE ,,0:STOP OFF:END
```

Listing EPI-DOEL

```
10 ' EPI-DOEL.BAS
20 ' Bij Epifiet MCCM 76
30 '
40 CLS:LOCATE ,,1
50 ON STOP GOSUB 90:STOP ON
60 S=INP(8):IF S<>0 GOTO 60
70 D=INP(9)
80 PRINT CHR$(D);:GOTO 60
90 LOCATE,,0:STOP OFF:END
```

Listing EPIFKEYB

```
10 ' EPIFKEYB.BAS
20 ' Bij Epifiet MCCM76
30 '
40 AD=&H8000:DL=&H4A
50 FOR I=0 TO DL:READ D$:POKE AD+I,VAL("&H"+D$):NEXT I
60 BSAVE "EPIFKEYB.BIN",&H8000,&H804A,&H802D
100 DATA DB,08,A7,20,22,2A,F8,F3,54,5D,23,7D,FE,18,20,03
110 DATA 21,F0,FB,3A,FA,F3,BD,28,0E,DB,09,FE,0A,28,08,00
120 DATA 00,00,00,12,22,F8,F3,C9,C9,C9,C9,C9,21,9A,FD
130 DATA 11,27,80,01,05,00,ED,B0,21,46,80,11,9A,FD,01,05
140 DATA 00,F3,ED,B0,FB,C9,CD,00,80,C9,C9
```

Listing POKE-NEW

```
10 ' POKENEW.BAS
20 ' Bij Epifiet MCCM76
30 '
40 ' Poke in (KEYBUF) volgende statement
50 ' verhoog (BOTTUM)
60 '
70 ' Adres keybuffer en tekst
80 KB=&HFBF0
90 KB$="BLOAD"CHR$(34)"EPIFKEYB.BIN"CHR$(34)",R"CHR$(13)
100 ' Poke in keybuffer
110 FOR I=0 TO LEN(KB$)-1
120 POKE KB+I,ASC(MID$(KB$,I+1,1)) AND 127
130 NEXT
140 ' Aanpassen keybuffer pointers
150 ' Zet (GETPNT) op begin keybuffer
160 POKE &HF3FA,&HF0:POKE &HF3FB,&HFB
170 ' Bereken plaats laatste teken keybuffer
180 KE=&HF0+KL
```

```

190 POKE &HF3F8,KE MOD 256 '      lo (PUTPNT)
200 POKE &HF3F9,&HFB+KE\256 '     hi (PUTPNT)
210 ' Verhoog (BOTTOM)
220 POKE &HF676,&H81:POKE &HF677,&H80
230 POKE &H8080,0:NEW

```

assembler listing

```

BEGIN      8000 DB 08      IN   A,(08)      lees status-register "EPIFIET"
           8002 A7        AND   A              geldige data aanwezig?
           8003 20 22     JR    NZ,22 8027     nee, dan naar OHKEYI

           8005 2A F8 F3   LD   HL,(F3F8)   lees *>PUTPNT<*
           8008 54        LD   D,H  ---- | bewaar *>PUTPNT<*
           8009 5D        LD   E,L  ---- |
           -----ga naar volgende plaats in *>KEYBUF<*
           800A 23        INC   HL              verhoog met 1
           800B 7D        LD   A,L
           800C FE 18     CP   18              boven hoogste adres van *>KEYBUF<* ?
           800E 20 03     JR    NZ,03 8013     nee, ga naar BUFVOL
           8010 21 F0 FB   LD   HL,FBF0     ja,dan begin *>KEYBUF<*
           -----is de buffer vol?
BUFVOL     8013 3A FA F3   LD   A,(F3FA)   lees *>GETPNT<*
           8016 BD        CP   L
           8017 28 0E     JR    Z,0E 8027     als buffer vol, ga dan naar OHKEYI

           8019 DB 09     IN   A,(09)      lees data-register "EPIFIET"
           801B FE 0A     CP   0A              is dit een LineFeed?
           801D 28 08     JR    Z,08 8027     ja, dan negeer dit en ga naar OHKEYI
           801F 00        NOP  - - - - -
           8020 00        NOP  |reserve ruimte voor nog
           8021 00        NOP  |een test en negeer
           8022 00        NOP  - - - - -
           8023 12        LD   (DE),A      zet data in *>KEYBUF<*
           8024 22 F8 F3   LD   (F3F8),HL  geef *>PUTPNT<* nieuwe waarde

OHKEYI     8027 C9        RET  - - - - -   op deze plaats komt tijdens het
           8028 C9        RET  |installeren van dit programma de
           8029 C9        RET  |originele inhoud van de hook
           802A C9        RET  | *>HKEYI<* te staan
           802B C9        RET  - - - - -
           802C C9        RET  |
           -----installatie programma
START     802D 21 9A FD   LD   HL,FD9A     bron->HKEYI<*
           8030 11 27 80   LD   DE,8027     doel-OHKEYI
           8033 01 05 00   LD   BC,0005     aantal byte = 5
           8036 ED B0     LDIR           copyeer blok
           8038 21 46 80   LD   HL,8046     bron-NHKEYI
           803B 11 9A FD   LD   DE,FD9A     doel->HKEYI<*
           803E 01 05 00   LD   BC,0005     aantal byte = 5
           8041 F3        DI              zet interrupt uit
           8042 ED B0     LDIR           copyeer blok
           8044 FB        EI              zet interrupt aan
           8045 C9        RET  |
           |
NHKEYI     8046 CD 00 80   CALL 8000 - - - de hook *>HKEYI<* krijgt
           8049 C9        RET  |dit als nieuwe inhoud
           804A C9        RET  - - - - -

```