# Epifiet

## Communication via the printerport

**MSX Computer & Club Magazine nummer 76 - juni 1995**

**Rob van Gans**

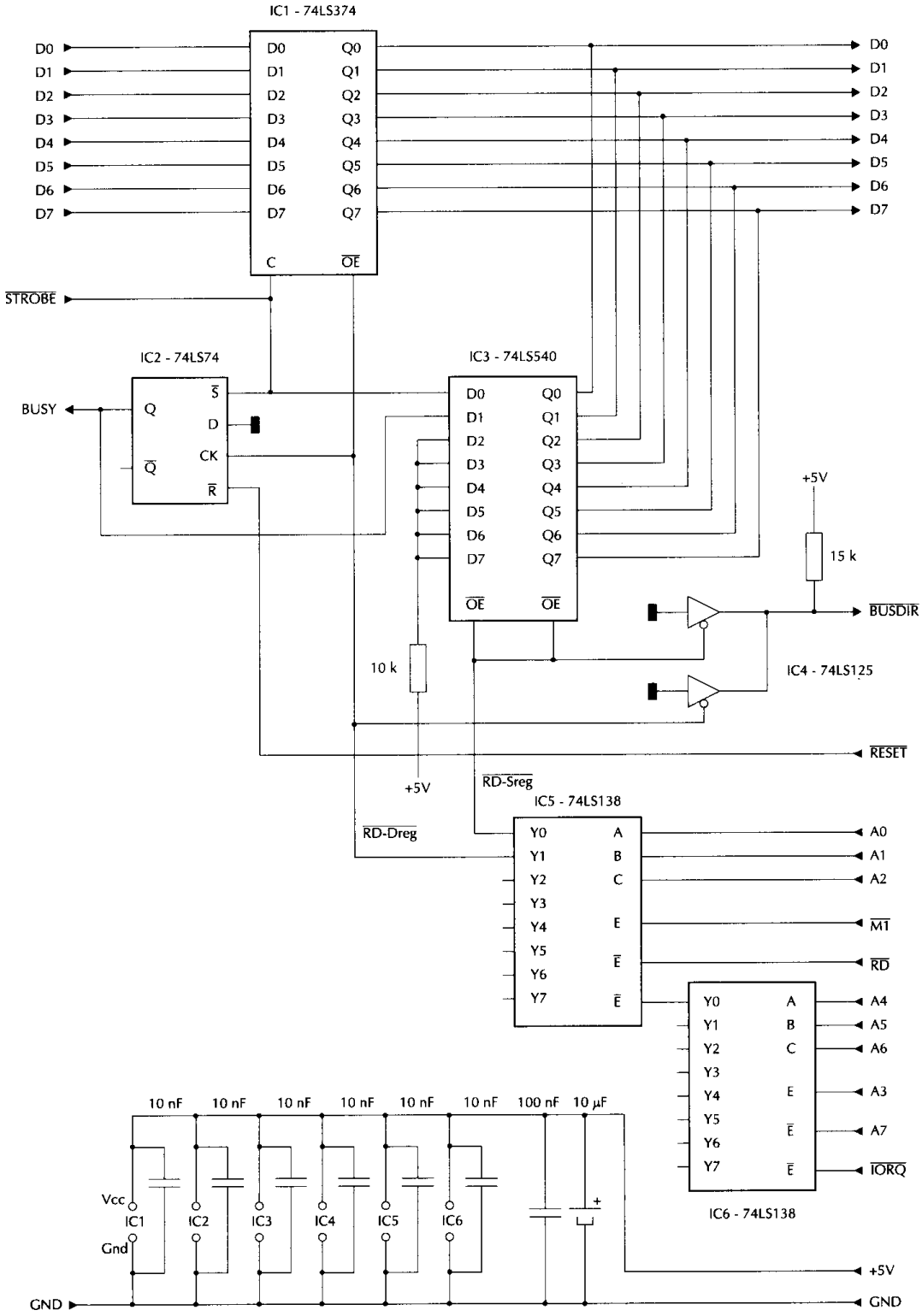
***Scanned, ocr'ed and translated and converted to PDF by HansO, 2001***

An interface for the printerport, whereby the MSX can read another computer. With the right software you can control the other computer. Do it yourself.

The name 'Epifiet' is chosen to make clear that this interface is not a parasite for its host. On the contrary, it lives in good harmony with the host. It is even possible to trasnfer Basic programs  and with MSX-Dos there are numerous possibilities.

**Do it yourself**

The interface is easy to build for low cost. With a housing it still is cheap.
The circuit diagram may look complicated but is easy to understand if you divide it in a few functional blocks.

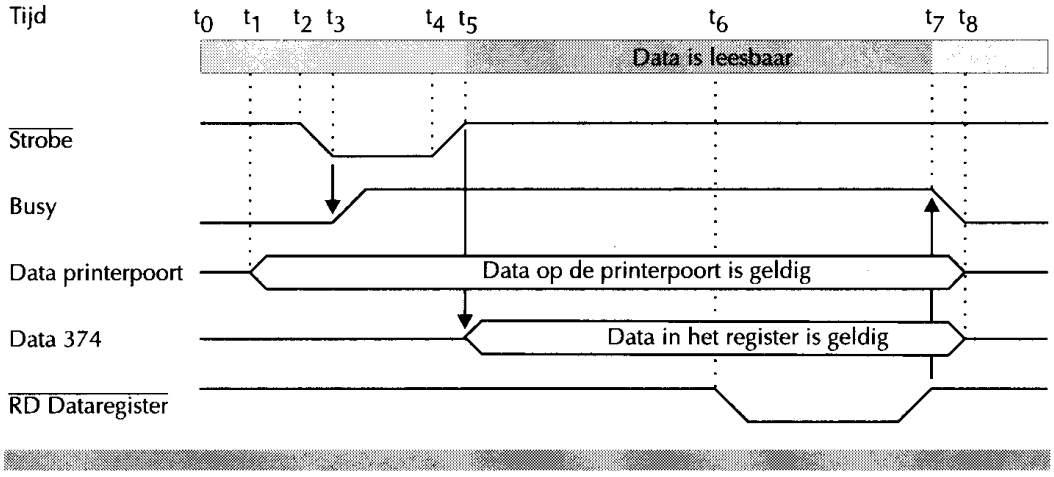



## The circuit

At the left you seen a cable with connector to be connected to the printerport of the first computer, just as a printer.
To the right there is the connection to the slotconnector of the other computer.
IC1 is a eight D-flopflop with tri-state output. The printerport records its data here.

This IC stores the data offered at the moment when the Clock input (C) goes from low to high state. The data dows not appear on the Q outputs: the IC is in tri-state mode and the output are high impedance.
When the Output Enable OE is low the data appears at the Q outputs.
IC2 is a Set-Reset D-flipflop. This IC creates the BUSY signal. Only one flipflop is used.
IC3 is a tri-sate inverting busbuffer. This IC is used to check the state of BUSY and STROBE.. The IC gives the data inverted to the Q outout if both OE inputs are low, otherwise the outputs are in high-impedance tri-state.
IC5 and 6 take care of the addressing of IC1 and IC3.
Instruction INP(8) will make Y0 RD-SREG low, INP(9) will make Y1 RD-DREG low.
IC4 is a tri-state buffer. This is used to make BUSDIR low if any of the two registers is read to ensure the correct setting of the internal databus buffers of some MSX-2 computers.

The Epifietinterface must behave like a printer. IC2 is used to ensure this. If the Set input is low, then Q becomes high, if R is low then Q becomes low.
If the Clock inout goes from low to high then Q is given the value of the D input, which in this case is permanently low.




**Timing**

At the top the timing is shown with indications t0 to t8, which will be referred to often hereafter.
When the computer is switched on or the resetbutton pressed the signal RESET becomes low for a short moment, so the RESET input of IC2 too. As a consequence the output Q and therefere BUSY becomes low.This is the start situation t0.
A computer that wants to send data to the printerport first checks if BUSY is low. If so, data may be transmitted.If BUY is high, the computer keeps testing BUSY until it becomes low.
The computer stores the data on the printerport at the next step t1. At t2 the computer sets STROBE to low, whichleads to t3 where STROBE set IC2, making Q and BUSY high.
At t4 the computyer makes STROBE high again and at t5 the data is stored in IC1.

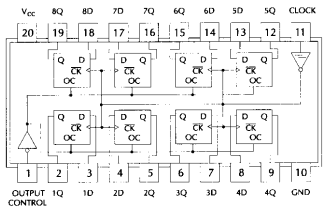
This state remains as long as the other computer does not read the data register IC1, between t5 and t6.

No look at this from the other side. There is only valid data in OC1 if BUSY and STROBE is high. These are connected via D0 and D1 to IC3. Th inputs D2 to D7 are always high with the pull resistor to +5V.

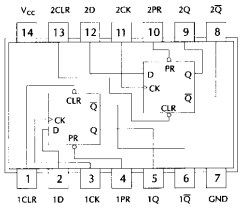
The next step is to check with INP(8) if there is valid data. RD-SREG becomes low during a read, causing IC3 to put his D ports in the databus of the slot. The result has to be zero because IC3 inverts the data. Now the data register can be read with INP(9).
Back to t6. During thr reading of  the data register IC1 RD-REG becomes low, the data is put on the databus of the slot and read into the computer. At t7 RD-REG becomes high, so Clock of IC2. At t8 Q and BUSY become low. Although the data is still in the data register, it is to be considered invalid because BUSY indicates that new data may be send to the printerport.
The cycle is now complete and back at t0. It may look that storing the data in IC1 is unnecesssary, but there ar ecomputers that first put the data on the printerport before aiting for BUSY to go low for clocing the data in the printer. This means that just after t5 the data on the printerport is not valid anymore.

**Remark on IC3**

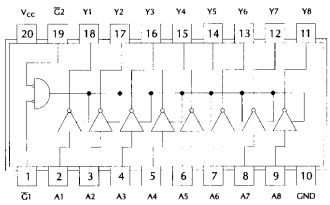
Instead of type 74LS540 for IC3 the more cheaper 74LS240 could have been used. The advantage of the 540 is the placing of all D inputs on one side and all Q outputs on the other. This is a clear advantage at constructing on exp board.




■ IC1: 74LS374

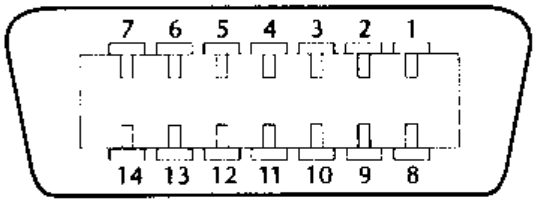


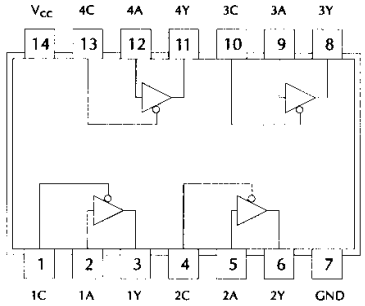

■ IC2: 74LS74



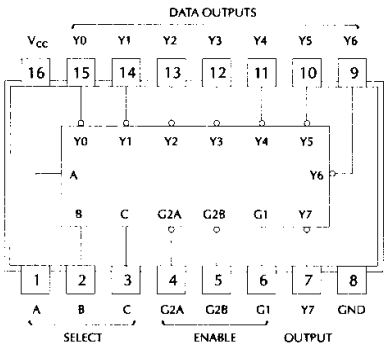

■ IC3: 74LS540

| Pin | Name | I/O |
|---|---|---|
| 1 | $\overline{\text{PSTB}}$ | 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 | |

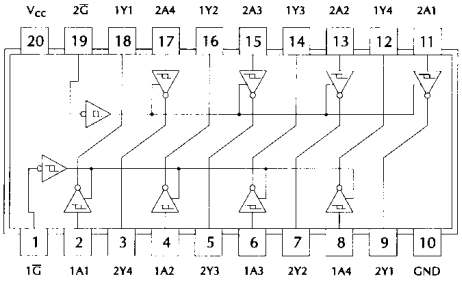


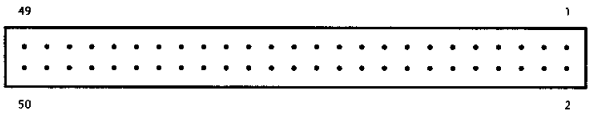

## IC4: 74LS125




## IC5: 74LS138




## Alternatief voor IC3: 74LS540




| Pin | Name | I/O | Pin | Name | I/O |
|---|---|---|---|---|---|
| 1 | $\overline{CS1}$ | O | 2 | $\overline{CS2}$ | O |
| 3 | $\overline{CS12}$ | O | 4 | $\overline{SLTSL}$ | O |
| 5 | Reserved | – | 6 | $\overline{RFSH}$ | O |
| 7 | $\overline{WAIT}$ | I | 8 | $\overline{INT}$ | I |
| 9 | $\overline{M1}$ | O | 10 | $\overline{BUSDIR}$ | I |
| 11 | $\overline{IORQ}$ | O | 12 | $\overline{MERQ}$ | O |
| 13 | $\overline{WR}$ | O | 14 | $\overline{RD}$ | O |
| 15 | $\overline{RESET}$ | 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**

No configuration is required for the interface.But without software the Epifiet does not do anything.
The rules to follow with Epifiet are
*Always read the status register with INP(8) to check if there is valid data. If the result is 0 (zero) then read the data with INP(9)*
*The data can be read only once, store in a variable immediately.*

**Test**

To ensure all works well a number of simple testprograms are available: EPI-BRON. BAS (BRON is dutch for source)and EPI-DOEL.BAS.(DOEL is dutch for target)
Store the interface in a slot of an MSX, the target computer. Connect a printercable to the other MSX: the source computer.
Then switch the computers on.
Start EPI-DOEL.BAS in the target computer.
Now type at the source computer

LPRINT "Hello this is a test"

After pressing return this string will appear on the screen of the target computer. This indicates that the interface works!

**Next step**

Now start EPI-BRON.BAS in the source computer. From now on the code of every key pressed on the source computerwill be shown on the screen of the source and on

the screen of the target computer, including special keys such as cursor, home, VT52 escape codes etc. Try for example escape followed by L to insert an empty line.
MSX books describe all those codes.

**Machine language**

Store all programs on two floppy disks for both computers.
Run the program POKENEW.BAS ion the target computer.
This program performes two functions:

- the bottom memory of Basic is increased to make room for a piece of machine language
- stores a string in the keyboard buffer to automatically start the next program

See the comment lines.

The result is that, via the hook HKEYI not only the keyboard but also the Epifiet interface is checked. A character received by EPIFIET is placed in the keybaord buffer.

**Carriage return en line feed**

Pressing the Return key on the source computer will cause also the linefeed character to be send to the target computer. Therefore EPIKEYB.BIN suppresses linefeeds.

**Examples**

***Sending Basic programs.***

At the source computer:

- Load a Basic program
- Type LLIST, the program is stored now at the target in memory
- Type LPRINT "SAVE"+CHR$(34)+ "filenaam.bas"+CHR$(34) to store the program on the disk of the target
- Type LPRINT "RUN" to start the program on the target

***Show directory at target***

Insert a floppy in the drive of the target
At the source computer type LPRINT "FILES"+CHR$(13)
On the target now the directory is shown.

***Revert the screen output with MSX-DOS***

Start MSX-DOS at the source computer and press Control-P
Now MSX-DOS sends all screen output to the printerport and therefore via Epifiet to the target computer.

Try for example DIR to show the directory contents of the source computer at the target.

**Insert text in an editor.**

Some programs can send output only to the printer. With Epifiet you can capture that printer output at the target so that it can be put into a file.

## 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,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) next statement
50 ' increase (BOTTOM)
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 ' Adjustkeybuffer pointers
150 '(GETPNT) begin keybuffer
160 POKE &HF3FA,&HF0:POKE &HF3FB,&HFB
170 ' last char in keybuffer
180 KE=&HF0+KL
```

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

## assembler listing

```
BEGIN   8000 DB 08       IN   A,(08)     read status-register "EPIFIET"
        8002 A7          AND  A          valid data ?
        8003 20 22       JR   NZ,22 8027 no, then OHKEYI

        8005 2A F8 F3    LD   HL,(F3F8)  read *>PUTPNT<*
        8008 54          LD   D,H  ----| save *>PUTPNT<*
        8009 5D          LD   E,L  ----|
                         ----------------gnext in *>KEYBUF<*
        800A 23          INC  HL         increment 1
        800B 7D          LD   A,L
        800C FE 18       CP   18         above  *>KEYBUF<* ?
        800E 20 03       JR   NZ,03 8013 no, then BUFVOL
        8010 21 F0 FB    LD   HL,FBF0    yes than begin *>KEYBUF<*
                         ----------------buffer full?
BUFVOL  8013 3A FA F3    LD   A,(F3FA)   read *>GETPNT<*
        8016 BD          CP   L
        8017 28 0E       JR   Z,0E 8027  if buffer full, then OHKEYI

        8019 DB 09       IN   A,(09)     read data-register "EPIFIET"
        801B FE 0A       CP   0A         is LineFeed?
        801D 28 08       JR   Z,08 8027  yes ignore and goto HKEYI
        801F 00          NOP - - - - - -
        8020 00          NOP            |reserve space
        8021 00          NOP            |
        8022 00          NOP - - - - - -
        8023 12          LD   (DE),A     put data in *>KEYBUF<*
        8024 22 F8 F3    LD   (F3F8),HL  give *>PUTPNT<* new value

OHKEYI  8027 C9          RET - - - - - - store original content
        8028 C9          RET            |at install time
        8029 C9          RET            |hook
        802A C9          RET            |  *>HKEYI<*
        802B C9          RET - - - - - -
        802C C9          RET             end"EPIFKEYB" program
        ----             ----------------install program
START   802D 21 9A FD    LD   HL,FD9A    source-*>HKEYI<*
        8030 11 27 80    LD   DE,8027    target-OHKEYI
        8033 01 05 00    LD   BC,0005    nr of byte = 5
        8036 ED B0       LDIR            copy block
        8038 21 46 80    LD   HL,8046    source-NHKEYI
        803B 11 9A FD    LD   DE,FD9A    target-*>HKEYI<*
        803E 01 05 00    LD   BC,0005    nr of byte = 5
        8041 F3          DI             disable interrupt
        8042 ED B0       LDIR            copy block
        8044 FB          EI              enable interrupt
        8045 C9          RET             end install program

NHKEYI  8046 CD 00 80    CALL 8000 - - - hook *>HKEYI<*
        8049 C9          RET            |new contents
        804A C9          RET - - - - - -
```