Index & shortform manual

Installation & Programming

On one side the interface is connected to C64/C128´s serial port, the other side to the RS232 port.

The enclosed cassette port cable is used which must be plugged into the C64/128 cassette port, while the computer is switched off to supply power. The cassette recorder can still be plugged on top of the interface connector. As an alternative the interface can be powered by a seperate AC adaptor which is available as an option. Do not use any other AC adaptor - the interface needs a stabiliszised supply of 5V ± 5%.

RS232 pinout

The pinout of the RS232 connector must be adapted to your particular RS232 device. Details of the pinout are on page 9 The interface provides either Hardware handshake (DTR/CTS) or XON/XOFF handshake can be selected during the programming sequence.

Programming

The interface has an internal EEPROM which saves all programmings even after power down. Programming can be done easily by using simple PRINT# commands directly from C64/C128. The command channel is accessed via secondary-address 15 (e.g. OPEN 1,6,15:PRINT#1,"...").

The programming process can be repeated in practice any time you wish to change settings. Each programming changes only the items which are addressed. All other parameters remain unchanged.

Page 9

Normal Operation

After the interface has been configured once, it will be operational immediately after each power-up. There is a 64KByte buffer inside, which is divided into 2x32KBytes for input and output seperately. Thus, the computer must no longer wait for the end of a serial output. When inputting, data are not lost due to slow computer operation (with 32KByte input-buffer, a buffer-overflow will not occur with most normal applications). If the interface is powered by the optional ac-adaptor, it can even receive data with the computer being switched off.

Output: C64/128 > V24	OPEN 1,6:PRINT#1,"OUT"	Outputs are made to secondary-address 0 with normal PRINT#-commands. Inputs with INPUT#-command are also done using secondary-address 0. The GET#-command must only be used with secondary-address 1.
Input : V24 > C64/128	OPEN 1,6:INPUT#1,A$	
	OPEN 1,6,1:GET#1,A$	

GET#-commands allow to input all possible 8-bit-codes, but only maximum one byte at a time. After each GET#-command the status-variable ST must be checked. If ST=0 the GET#-command has been successful and one valid databyte has been entered. If ST<>0 no valid databyte has been read (because the interface´s inputbuffer is empty) and the GET#-command should be repeated. Note, that even if ST<>0 there may be characters in the GET#-command´s variable but these are not usable.

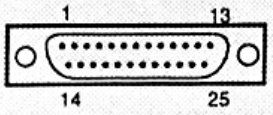
The INPUT#-command reads all data, that are present in the interface´s input-buffer until the next CR-code CHR$(13) or until the next comma (see page 10!). The INPUT#-.command can only be used for entry of characters and numbers, because the computer does not accept every 8-bit-code and even the interface must suppress the LF-code CHR$(10) to achieve normal operation.

Additional functions

By pressing the Clear key both the input- and output buffer will be cleared.
If the Init key is pressed while the interface is switched on or while it is cleared, the interface will work on address 6 regardless of any other setting. This function is important if the address on which the interface was programmed is unknown.

Pinout, Handshake

V24-plug



Pin	Function	marked as
2	Data output of the interfaces	TxD
3	Data input of the interfaces	RxD
5	Hardware-Handshake input of the interfaces	CTS
7	Ground	GND
20	Hardware-Handshake output of the interfaces	DTR

The interface transmits data on pin 2 if there is a positive voltage >3V on pin 5. If XON/XOFF handshake is activated (see "programming"), it will transmit only if it has not received an XOFF code last.
The interface receives data from the RS232 device on pin 3. It output a positive voltage > 3V on pin 20 as long as data can be received (as long as the buffer is not full). If XON/XOFF is activated the interface will transmit XOFF CHR$(19) to stop the RS232 transmitting and XON CHR$(17) if it is ready to receive data. The interface will signal "STOP" some 2Kbyte before its buffer is really full. This high buffer tolerance will give the RS232 transmitter enough time to react and avoid data lost.

Programming

When the interface is shipped, it is configured to address 6. For programming all functions secondary address 15 is used. In the following examples the necessary OPEN and CLOSE commands are also stated each time. Each program function begins with its function letter followed by the necessary parameters. Everything is sent with a normal PRINT# command. After each programming a pause of about 1secound is required to give the interface time to transfer the data to its constant memory. Programmed data will be valid until they are changed again.

Device address

Any address between 0 and 30 is selectable. In the following example the address is changed from 6 to 7:

```
OPEN 1,6,15:PRINT#1,"ADR7":CLOSE1
```

Code-conversion

The interface is capable of two code-conversions, which are active in both directions:

CN: no conversion
CA: CBM upper/lower case and German "Umlaute" are converted to "German-ASCII"
CI: CBM upper/lower case and German "Umlaute" are converted to IBM-PC-code.

The example will select IBM-conversion:

```
OPEN 1,6,15:PRINT#1,"CI":CLOSE1
```

On page 12 you will find more information on the way code conversions are performed and how toprogram your own conversion.

Dataformat

7 Databits: `OPEN 1,6,15:PRINT#1,"D7":CLOSE1`
8 Databits: `OPEN 1,6,15:PRINT#1,"D8":CLOSE1`

Parity

None: `OPEN 1,6,15:PRINT#1,"PN":CLOSE1`
Even: `OPEN 1,6,15:PRINT#1,"PE":CLOSE1`
Odd: `OPEN 1,6,15:PRINT#1,"PO":CLOSE1`

Stopbits

1 Stopbit:

```
OPEN 1,6,15:PRINT#1,"S1":CLOSE1
```

2 Stopbits:

```
OPEN 1,6,15:PRINT#1,"S2":CLOSE1
```

CBM transmission speed (only RS232>CBM)

The TL and TH functions allow to modification to the speed of the CBM's serial bus - don't confuse this with the baudrate! This function is needed only for high-speed. The lowest possible TL and TH values must be determined by experiment. The following example shows the standard-setting:

```
OPEN 1,6,15:PRINT#1,"TL60 TH37":CLOSE1
```

Baud-Rate

Baudrates between 225 and 57600 are selectable.
Example: 9600 Baud

```
OPEN 1,6,15:PRINT#1,"B9600":CLOSE1
```

The baudrate is internally stored as a number between 0 and 255 that has the following correspondance:

$$\text{Baudrate} = \frac{57600}{256 - N}$$

Any baud rate can be selected that complies with the above formula for integer N with $0 \le N \le 255$. N is only used internally in the interface. When programming the real baudrate is used.
If a baud rate is selected that does not comply with the above formula the next smaller rate is used.

XON/XOFF-Handshake

If you do not use hardware-handshake, the CTS- and DTR-pin's of the interface can be unconnected.
XON/XOFF is activated as follows:

activate XON/XOFF:

```
OPEN 1,6,15:PRINT#1,"HS":CLOSE1
```

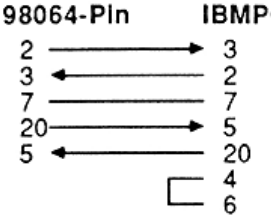
deactivate XON/XOFF:

```
OPEN 1,6,15:PRINT#1,"HH":CLOSE1
```

Applications

Connection to IBM-PC

The 98064-interface can be used to transfer data between IBMPC and C64/C128. For this application the following cable connection is needed:



If the IBMPC sends data it can be stopped using Pin 5 of the IBMPC. For this reason IBM-Pin 5 is connected to 98064 pin 20.

In the other direction (CBM>IBM) a simple handshake is not available because the IBMPC has normally no input handshake implemented. As auxilary handshake the IBMPC can be programmed to send everything it has received back and the CBM computer can be made to send the next data string only after it has received the "echo" of the previous one.

The following example programs transfer a sample text continuously either from CBM to IBM or vice versa. On the left hand side you see the CBM program, on the right hand side the IBM program.

C64/C128-Program:

```
10 OPEN 1,6,15
20 PRINT#1,"B9600,D8,S2,HH,CN,TL60,TH37"
30 FOR I=1 TO 2000:NEXT
40 CLOSE 1:OPEN 1,6
50 INPUT "1=Send, 2=Receive";A$
60 IF A$="2" GOTO 200
100 PRINT#1,"C64 >>> IBM-PC"
110 INPUT#1,I$:GOTO 100
200 INPUT#1,I$:PRINT I$:GOTO 200
```

In line 10 the command channel is opened. Line 20 programs all RS232 parameters in one single PRINT# command. The delay loop in line 30 is necessary to give the interface about 1 secound for programming. Line 40 closes the command channel and opens the data channel. Line 50 and 60 will go to line 100 -110 for sending and to line 200 for receiving data. When sending, line 100 sends the data string. Line 110 waits until an echo is received from the IBMPC and the returns to line 100 to send the next string.
The echo itself is not checked because the IBMPC has a small input buffer and thus it is sufficient to check whether there is any echo at all. If you program further checking mechanisms here you can easily program reliable protocolls.
The receiving program in line 200 waits for a data string, prints it on the screen and then waits for the next one.

IBM-PC-Program:

```
10 OPEN "COM1:9600,N,8,2" AS #1
20 INPUT"1=Send, 2=Receive";A$
30 IF A$="2" GOTO 200
100 PRINT#1,"IBM-PC >>> C64/C128":GOTO 100
200 INPUT#1,I$:PRINT#1,I$
210 PRINT I$:GOTO 200
```

In line 10 the serial channel is opened. Line 20 and 30 divert to line 200-210 for receiving and to line 100 for sending. When sending the sample text is sent continuosly. Handshake is automatic in this direction. In line 200 a data string is received from CBM and sent back to show correct acceptance. Then the string is displayed in line 210 and the computer will wait for another string.

Important note:

These programs can only transmit ASCII data.
The INPUT# command which is used on both sides cannot be used to transmit the full 8 bit code. If this is needed the GET#-command which is more complicated and much slower shoud be used.
Therefore it is recommended to make all transmissions with ASCII-codes. To transmit 8 bit code this way each code is simply convert to a 2 digit hex number which is then transmitted as two characters which the receiver will convert back to one byte.

Binary Transmission

The GET# command allows transmission of any 8 bit code. GET# does not wait until a character has been received and will get back "empty". Therefore, the ST variable must therefore be checked after each GET# command.

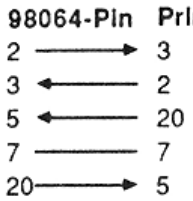
CBM-Program:

```
10 OPEN 1,6,1
20 GET#1,A$
30 IF ST<>0 GOTO 20
40 PRINT A$;
50 GOTO 20
```

The OPEN command in line 10 works with secondary address 1. Only then the GET# command can be used. In line 20 the interface is asked to transmit a character. If A$ has then been loaded with valid data the program will go to line 40. Otherwise the GET# command will be repeat until a character has been received. When typing in the command the ";" is important. Without it all characters would be printed underneath each other.

How to connect a printer

The 98064 interface can be used to connect a printer with RS232 input to a C64/C128. Most printers have their data input on pin 3 and their handshake output on pin 20.
In this case the connection cable looks as follows:



Printer and interface must be set to the same parameters. Use 9600 Baud, 8 Databits, 2 Stopbits, no XON/XOFF.
The command to program this in line 20 of the above example program for IBM/CBM connection. Instead of "CN" you will use "CA" if the printer has ASCII character set and "CI" if it is an IBM type printer. The interface will allow printing of normal letters and symbols but no special CBM symbols and no CBM graphics.

Any Problems?

No function

Keep the Init Key pressed during power up. With the following commands you can check all interface settings:

```
10 OPEN 1,6,15
20 INPUT#1,A$
30 PRINT A$
```

Now all settings can be seen the way they have been programmed.

(page). Instead of the baud rate there will be the internal number N from which the baud rate can easily be calculate.

If the settings can be read correctly most of the interface functions have been tested. Be sure that the program is set to the same address as the interface.

After this has been checked out and the RS232 device and the interface are set up to the same parameters most of the problems arise from the RS232 pinout.

If the interface does not send any data at all try connecting the interface pins 5 and 20 together. If data is sent now the problem is that your RS232 device does not allow data transfer. Check, on which pin the RS232 device shows it can accept data.

If the interface does not receive data and the RS232 device locks when tryinng to transmit, try to connecting all RS232 inputs on the RS232 device together and then to pin 20 of the interface.

GET#-Problems

GET# shoud only be used if the channel has been opened with secondary address 1. Then only the GET# and not the INPUT# command should be used.

GET# will accept any code from 0-255. There are a lot of codes in this area which will show special effects on the screen but are not printable.

Wrong characters

If there is no data transfer at all in any direction the problem will be the pinout. If other characters are received other than those sent the problem will be the dataformat (baudrate, databits, parity...). Be sure the interface and RS232 device are configured the same.

Check the C64/C128 transmissionspeed (page).
If the standard setting has been changed it will be possible for the incorrect characters to be transmitted in the direction RS232 > CBM.

Additional or missing codes

If the XON/XOFF handshake has been activated the interface will add XON/XOFF codes in the CBM>RS232 direction as necessary.
In the other direction RS232>CBM, XON/XOFF will be used for handshake but not transmitted to C64/C128.

The Interface will suppress the LF code CHR$(10) in the RS232>CBM direction if secondary address 1 (GET) is used.

The INPUT# command of the CBM and many other computers will suppress some codes (e.g. CHR$(13) and comma).

It is recommended that you use ASCII format for any kind of transmission. See note on page 10.

Problems with printer operation

The interface allows connection of a RS232 printer to a C64/C128. Although it is made for data transmission, it is not a dedicated printer interface and therefore, it is not possible to print special CBM characters or CBM graphics.

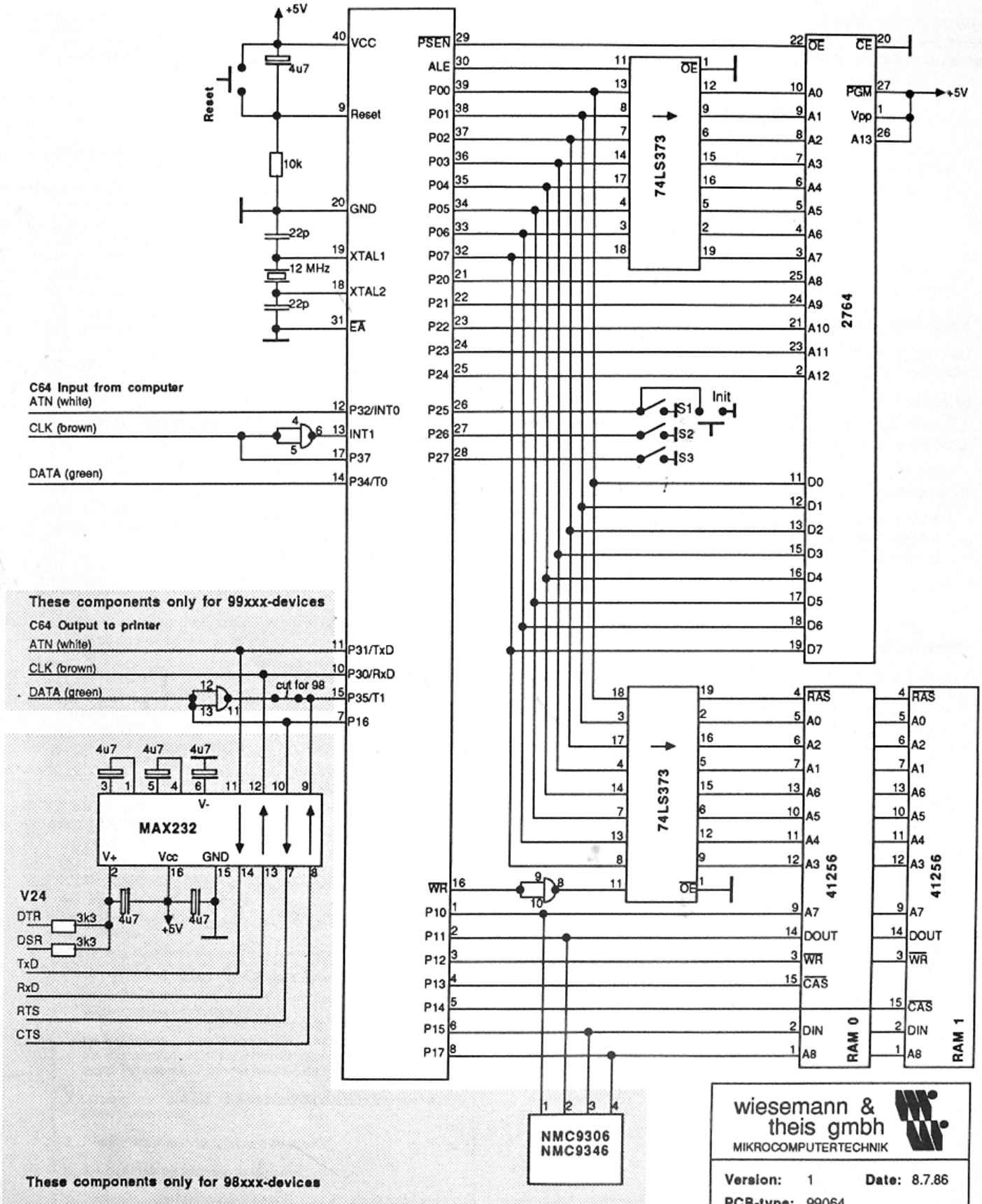
INPUT#-Problems

Data containing a comma cannot be used with the INPUT# command because the command will terminate each time the comma is received. The following program can be used instead:

```
10 OPEN 1,6,1
20 GET#1,A$:IF ST<>0 GOTO 20
30 IF A$=CHR$(13) PRINT S$:S$="":GOTO 20
40 S$=S$+A$:GOTO 20
```

This program reads data one byte after another and collects them in the S$ variable. After it has received CR (CHR$(13)) it will print the complete string on the screen and clear it for any new data entry.

Circuit diagram



wiesemann & theis gmbh
MIKROCOMPUTERTECHNIK

Version: 1 **Date:** 8.7.86

PCB-type: 99064