Manual

RS232 Portable Buffer

Release	1.0
Type	88031
	88127
	88511

W&T

Instructions

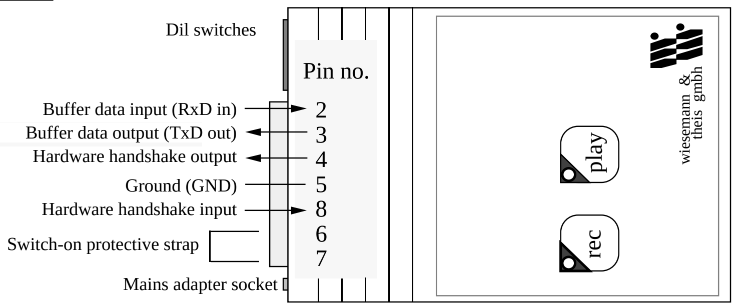
Cabling

To connect up your buffer you´ll need a cable that matches the pin-out of the buffer and your computer/peripheral device. See the illustration and the two examples given.

Note

To prevent you from accidentally switching the buffer on and running the batteries down, a plug that incorporates a protective strap between pin 6 and pin 7 has to be inserted before you can power the buffer up.

Pin-out


Dil switches
Pin no.
Buffer data input (RxD in) 2
Buffer data output (TxD out) 3
Hardware handshake output 4
Ground (GND) 5
Hardware handshake input 8
Switch-on protective strap 6 7
Mains adapter socket
play
rec
wiesemann & theis gmbh

Power supply

The mains power adaptor supplied delivers 5 volts, stabilized to ±5%. You should use no other adaptor. Away from the mains supply, power is provided by batteries that will hold your data intact, which the buffer switched off, for about 14 days. With the buffer switched on, the batteries will run the buffer for a maximum of 24 hours. Recharging, with the buffer connected to the mains adapter, takes 24 hours.

Example: connection to a PC

Function (IBM end)	DB25 socket (IBM end) Pin no.		DB9 socket (Buffer end) Pin no.	Function (Buffer end)
Data out	2	→	2	Data in
Data in	3	←	3	Data out
Handshake in	5	←	4	Handshake out
Ground	7	—	5	Ground
Handshake out	20	→	8	Handshake in
	4 ┐		┌ 6	Protective strap
	6 ┘		└ 7	

The strap from pin 4 to pin 6 sets "ready" level on the Data Set Ready (DSR) input of the IBM PC

Example: connection to a terminal

Function (Terminal end)	DB25 socket (Terminal end) Pin no.		DB9 socket (Buffer end) Pin No.	Function (Buffer end)
Data out	2	→	2	Data in
Data in	3	←	3	Data out
Handshake in	5	←	4	Handshake out
Ground	7	—	5	Ground
Handshake out	4	→	8	Handshake in
	6 ┐		┌ 6	Protective strap
	8 ┤		└ 7	
	20 ┘			

The strap from pin 6 to pin 8 and 20 sets "ready" level on the terminal´s ready-inputs.

Settings

All communications parameters can be set up by means of the DIL-switches, which are easily accessible from the outside. There´s no need to set number of stop bits: the buffer accepts any number of stop bits and will always send at least two. This makes it compatible with any receiving device.

The buffer executes software and hardware handshakes automatically. If you don´t want to use one of these handshake methods, just leave the lines concerned unconnected.

With S7 (the mode switch) you can specify whether the buffer is to respond to control codes (S7=On) or only to the built-in buttons (S7=Off). It´s true that software control gives you more scope, but you also have to program the attached device so that it sends the correct control sequences to the Portable Buffer. Keyboard control is substantially simpler.

Setting up the DIL switches

Baud	S 1	S 2	S 3
150	on	on	on
300	off	off	off
600	on	off	off
1200	off	on	off
2400	on	on	off
4800	off	off	on
9600	on	off	on
19200	off	on	on

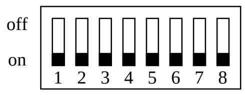
Data	S 4
7	on
8	off

Mode	S 7
key	off
Soft	on

Auto-OFF: S8 to "on".

Parity	S 5	S 6
no	off	off
even	on	off
odd	on	on

Soft-Parameter
S5=off, S6=on



S8 (**Auto-OFF**) serves to avoid an unintentional discharge of the batteries. When S8=On, the buffer will switch itself off automatically after about four minutes of inactivity, i.e. if (in input mode) no data have been received or (in output mode) the data receiver has not been ready to receive (this being indicated via the handshake).

Soft parameters: If you select this function the buffer will respond to the parameters specified by software commands (see the explanation given under "Software mode"), ignoring all other DIL switch settings.

The Portable Buffer lets you collect any kind of data at one point and output it somewhere else. In between collecting and reading the data you can change the baud rate, the data format and the handshake protocols. You can control your data collection and read back with the built-in keys, or with software commands passed via the RS232 interface. Software control offers more flexibility than keyboard control (for example, random access to data.

Keyboard control (S7=Off)

a) Recording
Start by pressing and holding down the REC key, then press the PLAY button too (just like a cassette recorder).

If you press these keys only briefly, the REC light-emitting diode (LED) will blink. This is to tell you that any following text you input will be appended to the existing contents of the buffer.

If you press both keys and hold them down till the REC indicator glows continuously, any text previously held in the buffer will be wiped: new data will be accepted to overwrite it.

b) Replay
If you press the PLAY button only briefly, the full contents of the buffer will be output once. The buffer will then switch itself off automatically.

An overview of the key functions

Function	Keys	LED display
Input (append)	First REC,then PLAY as well,short push	PLAY, REC blink
Input(from new)	First REC,then PLAY as well,long push	PLAY, REC
Output once	PLAY, short push	PLAY
Output cyclically	PLAY, long push. Stop: press PLAY	PLAY blinks
Switch off	Press PLAY	all extinguished

When the buffer has only 4 kilobytes free space left it tells the data sender to "Stop", using the XOFF code and (simultaneously) putting a voltage of <-3V on its handshake output. If the data sender ignores this handshake signal, the REC indicator will be extinguished as soon as the buffer is full. It will then blink, to signal that the buffer is still switched on. If the buffer is full and has been switched off, and then you try to start inputting (i.e. appending) more data, the buffer will immediately output a handshake stop (XOFF and the hardware handshake signal).

If you hold down the PLAY button for longer than 3 seconds, the text will be output cyclically over and over until you press PLAY once more. During cyclical output, the PLAY indicator will blink.

Software control (S7=On)

You can use software control as long as the quipment that you connect to the buffer can be programmed to send the commands that the buffer requires. You can then define all parameters, including those for data transmission format, in software. In software mode, you switch the buffer off and on by pressing the PLAY button.

Many commands require you to input an address or a number of bytes. You should first convert any such number to hexadecimal form and then split it into three bytes for transmission (least significant byte first). Example:

n=2000 decimal = 00 07 d0 hexadecimal

d0=CHR$(208) (least significant byte)
07=CHR$(7)
00=CHR$(0) (most significant byte)

The detailled application can be learned from the examples included.

Programming the data format: You can define three different data transmission formats, for use in each of the buffer´s three modes - input under key control, output under key control, and software-controlled operation. These parameteres will subsequently be selected automatically, provided S5=off and S6=on. Otherwise the buffer will use, for all modes, the parameters defined by the other DIL switches. Follow the plain-text examples given here, using the sequence: baud-rate (75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200), parity ("E"=even, "O"=odd, "N"=none), and number of data bits ("7" or "8").

Error string: you can use the "ESC E" command to make the buffer output a 9-digit string (ending with CR + LF - see the example program) that will warn you of any input errors. If the string consists of all "0", there is no error. In all other cases, a letter is included in the error string to indicate the following conditions: "A": The address input after ESC A is too large. "B": ESC W was specified with an excessive number of bytes. "C": ESC B was followed by an address that was too high. "D": ESC R requested too many bytes. "E": an incorrect baud-rate was specified. "F": an incorrect parity was specified. "G": an incorrect number of data bits was specified.

Commands for use in Software Control

Command	Function	Beispiel
ESC W n1 n2 n3	Write n bytes into the buffer	PRINT CHR$(27); "W"; CHR$(208); CHR$(7); CHR$(0); Writes the following 2000 bytes starting at the write position.
ESC R n1 n2 n3	Read n bytes from the buffer	PRINT CHR$(27); "R"; CHR$(208); CHR$(7); CHR$(0); Reads 2000 bytes starting at the read position.
ESC A n1 n2 n3	Position the write pointer at a buffer address	PRINT CHR$(27);"A"; CHR$(208); CHR$(7); CHR$(0); Positions the write pointer at address 2000.
ESC B n1 n2 n3	Position the read pointer at a buffer address	PRINT CHR$(27);"B"; CHR$(208); CHR$(7); CHR$(0); Positions the read pointer at address 2000.
ESC I	Set RS232 parameters for data input	PRINT CHR$(27); "I 9600, E, 8"; Sets the parameters for data input in keyboard mode (S7=off) to 9600 bauds, even parity and 8 data bits.
ESC O	Set RS232 parameters for data output	PRINT CHR$(27); "O 4800, O, 7"; Sets the parameters for data output in keyboard mode (S7=off) to 4800 bauds, odd parity and 7 data bits.
ESC S	Set RS232 parameters for software control	PRINT CHR$(27); "S 2400, N, 8"; Sets the parameters for software mode (S7=on) to 2400 bauds, no parity and 8 data bits.
ESC E	Read and clear the error register	PRINT CHR$(27);"E"; Causes the buffer to send its error string.
ESC X	Switch buffer off	PRINT CHR$(27);"X"; Switches the buffer off.

Sample BASIC Program to run on PCs

Command	Comments
	Preliminary settings: S1=on, S2=off, S3=on, S4=off, S5=off, S6=off, S7=on, S8=on at the buffer. The buffer must be connected to the PC´s number 1 serial port (see the cable in the example given in these instructions). Switch the buffer on by pressing the PLAY button.
10 A$="This sentence will be stored in the buffer starting at address 100"	
20 OPEN"COM1:9600,N,8" AS #1	Open serial channel 0. Parameters as set by DIL switches.
30 PRINT#1,CHR$(27);"A";:A=100:GOSUB 200	Position write pointer at address 100.
40 PRINT#1,CHR$(27);"W";:A=LEN(A$):GOSUB 200	Prepare buffer to receive length of A$.
50 PRINT#1,A$;	Send A$ itself.
60 GOSUB 300	Call the error string.
70 PRINT#1,CHR$(27);"B";:A=100:GOSUB 200	Position read pointer at address 100.
80 PRINT#1,CHR$(27);"R";:A=LEN(A$):GOSUB 200	Command buffer to send the length of A$ in bytes.
90 PRINT INPUT$(A,#1)	Receive and display these bytes.
100 GOSUB 300	Call the error string.
110 END	
200 N3=INT(A/(256*256)) 210 N2=INT((A-N3*256*256)/256) 220 N1=A-N3*256*256-N2*256 230 PRINT#1,CHR$(N1);CHR$(N2);CHR$(N3); 240 RETURN	Subprogram: accept a number in A and pass it to the buffer in the form the buffer requires. This number can represent either an address or a number of bytes.

300 PRINT#1,CHR$(27);"E";
310 IF INPUT$(9,#1)<>"0000000"+CHR$(13)+CHR$(10) THEN
      PRINT "ERROR"
320 RETURN
	Read and display the error string.
	
	Note: Don´t forget the semicolon (";") after each PRINT command. This ensures that the PC doesn´t insert a carriage-return and line-feed (2 unwanted bytes) after each PRINT.