TBKCOMM.DLL

TBKCOMM.DLL provides functions you can use to control up to four serial ports. Like the other DLLs supplied with ToolBook, TBKCOMM.DLL can be distributed with your ToolBook applications.

Functions in TBKCOMM.DLL:

clearComBreak()	*isComTxReady()*	*setComPortTxXlate()*
closeComPort()	*openComPort()*	*SetComRxNotify()*
comRxNotifyMsgID()	*readComPort()*	*TBK_COMMVersion()*
flushComRxBuffer()	*setComBreak()*	*writeComPort()*
flushComTxBuffer()	*setComPort()*	
isComRxReady()	*setComPortRxXlate()*	

For information about linking and declaring DLL functions, and for details about functions in the other DLLs licensed for distribution by the Authors Resource Kit, see the ToolBook DLL Reference booklet. A copy of this booklet is included with the Developer Utility Package and with your ToolBook documentation.

Using COM3 and COM4

Using COM3 and COM4 can be difficult, and in some cases impossible, due to physical limitations of the PC AT architecture; TBK-COMM.DLL cannot override errors or limitations in the hardware configuration. Refer to the file SYSINI.TXT created by the Windows Setup program in your Windows directory for more information. Generally speaking, if you can open and close a port as well communicate through it using the Windows application TERMINAL, you should be able to use that port with TBK-COMM.DLL.

clearComBreak()

Syntax

clearComBreak(<COM number>)

Declaration

INT clearComBreak(WORD)

Description

Restores character transmission suspended by a call to setComBreak() and places the transmission line in a nonbreak state.

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

0 or a positive number	The function was successful
A negative number	The port is not open or an error occurred

Example

```
if get SetComBreak(PortNo) >= 0
    pause 30 ticks     -- pause 0.3 seconds
    get ClearComBreak(PortNo)
end
```

closeComPort()

Syntax

closeComPort(<COM number>)

Declaration

INT closeComPort(WORD)

Description

Closes the port designated by <COM number>. If the port was not opened, closeComPort() has no effect.

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

A positive number	The function was successful
0	An error occurred

Example

```
get closeComPort(1)    -- close COM1:
```

comRxNotifyMsgID()

Syntax

comRxNotifyMsgID

Declaration

WORD comRxNotifyMsgID()

Description

Returns the ID number of the Windows message used when incoming characters notification is turned on with setComRxNotify(). This message ID may vary from session to session.

Parameters

None

Returns

A number in the range 0..65535.

flushComRxBuffer()
flushComTxBuffer()

Syntax

flushComRxBuffer(<COM number>)
flushComTxBuffer(<COM number>)

Declaration

INT flushComRxBuffer(WORD)
INT flushComTxBuffer(WORD)

Description

flushComRxBuffer() flushes out any characters in the receive buffer for a COM: port. flushComTxBuffer() flushes out any characters in the transmit buffer. The characters are lost.

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

0 or positive number	The function was successful
Negative number	An error occurred

Example

```
get flushComRxBuffer(1) -- flush receive queue of COM1:
get flushComTxBuffer(1) -- flush transmit queue
                        -- of COM1:
```

isComRxReady()
isComTxReady()

Syntax

isComRxReady(<COM number>)
isComTxReady(<COM number>)

Declaration

WORD isComRxReady(WORD)
WORD isComTxReady(WORD)

Description

isComRxReady() checks if there are characters in the receive buffer for a COM: port. isComTxReady() checks if there is room in the transmit buffer for a COM: port. Use this function to avoid long waits, because writeComPort() will loop as long as necessary to avoid overrunning the output buffer.

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

isComRxReady() returns a positive number representing the number of bytes in the receive buffer, 0 if the receive buffer is empty, or a negative number if the port number is invalid.
isComTxReady() returns the number of free bytes in the transmit buffer. If the transmit buffer is empty, this number is equal to the size of the buffer.

Example

```
to handle idle
    if isComRxReady(1) > 0 then
        put readComPort(1) after text of field "Input"
    end
end
while isComTxReady(1) < 1
    if sysTime  MarkTime > 1 then
        request "Time out error on COM1:"
        break to system
    end
end while
get writeComPort(1, TheText)
```

openComPort()

Syntax

```
openComPort(<COM number>,<input buffer size>,\
            <output buffer size>)
```

Declaration

```
INT openComPort(WORD,WORD,WORD)
```

Description

Opens a serial port for communications.

Important: You must call closeComPort() before exiting your application. Otherwise, the port remains unavailable to other applications and will remain unavailable even after you exit Windows.

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

<input buffer size> The size of the input buffer the driver will reserve for this port. If <input buffer size> is 0, a default buffer size of 1,024 bytes will be assumed. Minimum buffer size is 32 characters; maximum is 32,767 characters, unless you use the setComPortRxXlate() function, in which case you should not use a buffer of more than 16,383 characters.

<output buffer size> The size of the output buffer reserved for this port. If <output buffer size> is 0, a default buffer size of 128 bytes is assumed. Minimum buffer size is 32 characters; maximum is 32,767 characters.

Returns

A Windows device handle	The function was successful
A negative number	The function failed

Note: The Windows device handle is not used by the other functions in this DLL, but it is returned in case you need to refer to the port in calls to functions in other DLLs.

Example

```
if openComPort(1,256,0) >= 0 then
    request "COM1: opened successfully"
else
    request "Could not open COM1:"
end if
```

readComPort()

Syntax

readComPort(<COM number>)

Declaration

STRING readComPort(WORD)

Description

Reads the characters in the receive buffer for a COM: port. The received string is truncated if it contains a character with a null value (ANSI character zero).

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

A string of characters or a null string if the receive buffer is empty.

Example

```
set Echo to readComPort(1)
```

setComBreak()

Syntax

setComBreak(<COM number>)

Declaration

INT clearComBreak(WORD)

Description

Suspends character transmission and places the transmission line in a break state until clearComBreak() is called.

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

Returns

0 or a positive number The function was successful

A negative number The port is not open or an error occurred

Example

```
if get SetComBreak(PortNo) >= 0
    pause 30 ticks     -- pause 0.3 seconds
    get ClearComBreak(PortNo)
```

setComPort()

Syntax

```
setComPort(<COM number>,<baud>,<byte size>,\
           <stop bits>,<parity>,<handshake>)
```

Declaration

```
INT setComPort(WORD,WORD,WORD,WORD,WORD,WORD)
```

Description

Sets the parameters for a COM: port. The port must have been previously opened with openComPort(). setComPort() will also reset the port completely, if possible.

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.
<baud>	Any baud rate supported by Windows and the COM: port hardware, typically 110, 300, 600, 1200, 2400, 4800, 9600 or 19200.
<byte size>	The number of bits in each transmitted character, typically 7 or 8.
<stop bits>	The number of stop bits that follow a transmitted character.
<parity>	0 for no parity, 1 for odd parity, or 2 for even parity.
<handshake>	0 for no handshaking, 1 for hardware handshaking, or 2 for Xon/Xoff handshaking. When hardware handshaking is in effect, RTS is used for transmit flow control and DTR is used for receive flow control. Both are set to time out after 4 seconds.

Returns

0 or a positive number	The function was successful
1	The port is not open or an error occurred
202	<baud> is invalid
203	<byte size> is invalid
204	<stop bits> is invalid
205	<parity> is invalid
206	<handshake> is invalid

Example

```
-- Open COM1: with default buffer sizes
get openComPort(1, 0, 0)
if it >= 0 then
    get setComPort(1, 9600, 8, 1, 0, 0)         -- 9600
baud, no parity, no handshake
end
if it < 0 then
    request "Could not open COM1:"
end
```

setComPortRxXlate()

Syntax

setComPortRxXlate(<COM number>, <state>)

Declaration

INT setComPortRxXlate(WORD, INT)

Description

setComPortRxXlate() turns on or off the translation of each incoming null character (ASCII value 0) into the character sequence "\0". The port must be open. This is the only way to detect incoming nulls, because OpenScript uses the null character as a string terminator internally .

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.

<state> 0 to turn the translation off for the port, 1 to turn the translation on

Returns

0 or a positive number	The function was successful
A negative number	An error occurred

Example

```
get setComPortRxXlate(1, 1) -- turn on incoming null
                            -- translation for COM1:
```

setComPortTxXlate()

Syntax

setComPortTxXlate(<COM number>,<state>)

Declaration

INT setComPortTxXlate(WORD,WORD)

Description

Toggles the translation of transmitted escape character sequences for the specified port. The port must be open. By default, this setting is off when you open the port. setComPortTxXlate() allows you to embed escape sequences that will be transmitted as nulls, because null characters cannot be embedded directly in OpenScript strings.

Supported escape character sequences

Sequence	***Transmitted as***	***Hexadecimal equivalent***
\0 (1)	null	00h
\a	bell (Alert)	07h
\b	backspace	08h
\f	form feed	0Ch
\\	backslash character	5Ch

(1) The second character in \0 is the digit zero, not an uppercase letter O.

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.
<state>	1 to turn translation on, 0 to turn it off.

Returns

0 or a positive number	The function was successful
A negative number	The port is not open or an error occurred

Example

```
-- Open COM1: with default buffer sizes
get openComPort(1, 0, 0)
if it >= 0 then
        -- set to 9600 baud, no parity, no handshake
    get SetComPort(1, 9600, 8, 1, 0, 0)
        -- Turn on outgoing translation of \0 for COM1
    get setComPortTxXlate(1,1)
end
if it < 0 then
    request "Could not open COM1:"
else
    -- transmit some nulls, transmit normal characters,
    -- and make the other terminal beep
    get writeComPort(1, "\0\0\0\0\0\0\0\0\0\0")
    get writeComPort(1, "Hello, anyone there?\a"\
         & CRLF)
end
```

SetComRxNotify()

Syntax

SetComRxNotify(<COM number>, <window handle>, <state>)

Declaration

INT SetComRxNotify(WORD, WORD, INT)

Description

SetComRxNotify() turns on or off notification when incoming characters are detected at the specified COM port. The notification is in the form of a Windows message, and you must use an OpenScript translateWindowMessage control structure to start intercepting this message in your scripts. You can obtain the ID number of the message by calling ComRxNotifyMsgID(), also in TBKCOMM.DLL.
The port must be open prior to calling SetComRxNotify(). The port is checked by a Windows timer that runs every 20 milliseconds when Windows is idle. If this option is used for more than one port, only one timer is shared between the ports. Only one window can be specified for each port.

Parameters

<COM number> The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.
<window Handle> A window handle, (typically sysWindowHandle)
<state> 0 to turn the notification off
1 to turn the notification on

Returns

0 or a positive number The function was successful
-1 The specified port is invalid, or the window handle is invalid
-301 Function failed because no Windows timer is available
-302 Function failed because SetComRxNotify is already set for another window.

Example

```
-- Turn on incoming null translation for COM1
get SetComRxNotify(1, sysWindowHandle, 1)
-- Get the message ID
set ComNotifyMsgID to ComRxNotifyMsgID()
-- Activate translation into OpenScript message
translateWindowMessage for sysWindowHandle
    before ComNotifyMsg send ReceivedSomething
end
```

TBK_COMMVersion()

Syntax

TBK_COMMVersion()

Declaration

STRING TBK_COMMVersion()

Description

Returns a string containing the name of this DLL, the version number and the date this version was compiled.

Parameters

None

Returns

A string in the form "TBK-COMM.DLL Version 1.0 4/1/92".

writeComPort()

Syntax

writeComPort(<COM number>,<output>)

Declaration

INT writeComPort(WORD, STRING)

Description

Writes <output> to the transmit buffer of a COM: port. See also isComTxReady().

Parameters

<COM number>	The number of the COM: port to set such as 1 for COM1: 2 for COM2:, and so forth up to 4.
<output>	Anything that evaluates to a string. If <output> has a numeric value, the string representation of that value is sent.

Returns

A positive number	The number of characters actually placed in the transmit queue
-1	The port is not open or an error occurred
-16	A break was detected
-32	A CTS timeout was detected
-64	A DSR timeout was detected
-256	An attempt was made to queue a character to a full transmit queue (logical error internal to the DLL)

Other negative values are errors returned by the Windows COM driver; they are documented in the Microsoft Windows Programmer's reference manuals.

Example

```
if writeComPort(1, "Hello there" & CR) > 0 then
    request "It worked!"
end if
```