Table of Contents

Foreword 0

Part I About MegaPipe Win32 DLL 3

1 Introduction 3

2 How to Use It 3

Trial Version 3

Full Version 3

3 How to Distribute It 4

Part II Reference Guide 4

1 Serial Communication 4

ClosePort Function 4

EscapeCommFunc Function 5

GetCommStatus Function 5

GetFeedback Function 6

GetInputDataCount Function 7

GetPortStatus Function 7

OpenPort Function 7

ReceiveData Function 8

SendData Function 9

SetPortParam Function 9

2 Modem 11

CloseTAPI Function 11

DropCall Function 12

InitTAPI Function 12

GetLineStatus Function 13

GetModemCount Function 13

GetModemName Function 14

GetModemPort Function 14

MakeCall Function 15

WaitForCall Function 16

3 File Transfer 17

GetXferStatus Function 17

XferAddFile Function 17

XferClearAllFiles Function 18

XferGetCurrBytes Function 18

XferGetCurrFileName Function 19

XferGetCurrFileSize Function 19

XferSetDstFile Function 20

XferSetParam Function 20

XferSetWorkDir Function 21

XferStart Function 22

XferStop Function 22

Part III License 22

Index

0

1 About MegaPipe Win32 DLL

1.1 Introduction

MegaPipe Win32 DLL is a reliable and powerful library for handling serial communication, modem operation and file-transfer (XModem Checksum, XModem CRC, XModem 1K, YModem, YModem-G, ZModem and Kermit), it is usable with a variety of popular application development environments such as VC++, VB, VB.Net, C#.Net, Access, Delphi and Borland C++.

MegaPipe Win32 DLL supports multiple port/phone line (up to 8) communications simultaneously, each communication channel is uniquely identified by a channel ID (0-7).

1.2 How to Use It

1.2.1 Trial Version

It is very easy to use the trial version MegaPipe Win32 DLL in your application:

1. Finish the installation for the trial version.

2. Go to "VC++" sub-folder in the installation destination folder (e.g. "C:\Program Files\MW6 MegaPipe Win32 DLL\Trial Version\"), copy MegaPipeWin32.dll (VC++ IDE also needs MegaPipeWin32.lib) to your application folder, take a look at VC++ Demo, VB Demo or Delphi Demo for all function declarations, copy and paste them to your application.

3. Now you can call MegaPipe Win32 DLL functions in your application, the API documentations for all functions are in "Reference Guide" section of this documentation.

1.2.2 Full Version

Follow the instructions listed below to use the full version MegaPipe Win32 DLL in your application:

1. Uninstall the trial version and complete the installation for the full version,

2. Go to "VC++" sub-folder in the installation destination folder (e.g. "C:\Program Files\MW6 MegaPipe Win32 DLL\Full Version\VC++"), copy MegaPipeWin32.dll (VC++ IDE also needs MegaPipeWin32.lib) to your application folder to replace the trial version file(s).

3. The license key function MegaPipeSetKey should be called before you call other MegaPipe functions.

```
BOOL MegaPipeSetKey(LPCTSTR lpKey);
```

Parameters

lpKey

A Pointer to a null-terminated string containing 10 digits license key obtained from us.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Examples

```
BOOL bValidKey;
bValidKey = MegaPipeSetKey("XXXXXX-XXXX");
```

1.3 How to Distribute It

If you want to redistribute MegaPipe Win32 DLL as part of your application, on the target machine, simply put MegaPipeWin32.dll into the windows 32-bit system folder (e.g. "c:\windows\system32" or "c:\winnt\system32") for 32-bit Windows OS, or the SysWow64 folder (e.g. "c:\windows\SysWow64) for 64-bit Windows OS.

2 Reference Guide

2.1 Serial Communication

2.1.1 ClosePort Function

Closes the serial port for a communication channel (direct serial connection or modem connection).

```
BOOL ClosePort(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

See Also

OpenPort Function

2.1.2 EscapeCommFunc Function

Performs an extended function for a communication channel (direct serial connection or modem connection).

```
BOOL EscapeCommFunc(WORD nChannelID, DWORD dwFunc);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

dwFunc

Extended function to be performed, this parameter can be one of the following values:

dwFunc Value	Comment
3	Sets RTS (request-to-send) line
4	Clears RTS (request-to-send) line
5	Sets DTR (data-terminal-ready) line
6	Clears DTR (data-terminal-ready) line
7	Resets device if possible
8	Sets the device break line
9	Clears the device break line

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.1.3 GetCommStatus Function

Retrieves modem control-register values for a communication channel (direct serial connection or modem connection).

```
BOOL GetCommStatus(WORD nChannelID, LPDWORD lpModemStat);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpModemStat

A long pointer to a 32-bit variable that specifies the current state of the modem control-register values. This parameter can be a combination of the following values:

Value	Comment
MS_CTS_ON	The CTS (Clear To Send) signal is on.
MS_DSR_ON	The DSR (Data Set Ready) signal is on.
MS_RING_ON	The ring indicator signal is on.
MS_RLSD_ON	The RLSD (Receive Line Signal Detect) signal is on.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.1.4 GetFeedback Function

Retrieves the information for a communication channel (direct serial connection or modem connection) when an error or a warning occurs.

```
BOOL GetFeedback(WORD nChannelID, LPTSTR lpMsg, LPDWORD lpSize);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpMsg

A pointer to a buffer that receives a null-terminated character string containing the information.

lpSize

A pointer to the variable that receives the size (in characters) of the information.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.1.5 GetInputDataCount Function

Retrieves the number of incoming data bytes available in the input buffer for a communication channel (direct serial connection or modem connection).

```
DWORD GetInputDataCount(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

The number of incoming data bytes available in the input buffer.

See Also

ReceiveData Function

2.1.6 GetPortStatus Function

Retrieves the port status for a communication channel (direct serial connection or modem connection).

```
DWORD GetPortStatus(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

A return value of 1 indicates that the port is open, a return value of 0 indicates that the port is closed.

2.1.7 OpenPort Function

Opens the serial port for a communication channel (direct serial connection or modem connection).

```
BOOL OpenPort(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

See Also

ClosePort Function

2.1.8 ReceiveData Function

Reads data from the input buffer for a communication channel (direct serial connection or modem connection).

```
BOOL ReceiveData(
  WORD nChannelID,
  LPVOID lpBuffer,
  DWORD dwNumberOfBytesToReceive,
  LPDWORD lpNumberOfBytesReceived);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpBuffer

A pointer to the buffer that receives the data read from the input buffer.

dwNumberOfBytesToReceive

The minimum number of bytes to read.

lpNumberOfBytesReceived

A pointer to the variable that receives the number of bytes read.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is

zero.

2.1.9 SendData Function

Transmits data for a communication channel (direct serial connection or modem connection).

```
BOOL SendData(
  WORD nChannelID,
  LPVOID lpBuffer,
  DWORD dwNumberOfBytesToSend,
  LPDWORD lpNumberOfBytesSent);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpBuffer

A pointer to the buffer containing the data to be transmitted to the remote side.

dwNumberOfBytesToSend

Number of bytes to be transmitted to the remote side.

lpNumberOfBytesSent

A pointer to the variable that receives the number of bytes transmitted.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.1.10 SetPortParam Function

Sets up the serial port parameters for a communication channel (direct serial connection) or assigns a communication channel to a modem.

```
BOOL SetPortParam(
  WORD nChannelID,
  LPCTSTR lpPortName,
  DWORD dwBaudRate,
  WORD nDataBits,
  WORD nStopBits,
  WORD nParity,
  WORD nFlowControl,
  BOOL bUseTAPI,
  WORD nModemIndex);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpPortName

A pointer to a null-terminated string containing the serial port name (e.g. "COM1").

dwBaudRate

The baud rate of the transmission (e.g. 9600).

nDataBits

The data bits of the transmission, this parameter can be one the following values:

Value	Comment
4	4 data bits
5	5 data bits
6	6 data bits
7	7 data bits
8	8 data bits

nStopBits

The stop bits of the transmission, this parameter can be one the following values:

Name	Comment
0	1 stop bit
1	1.5 stop bits
2	2 stop bits

nParity

The parity of the transmission, this parameter can be one the following values:

Name	Comment
0	No parity
1	Odd parity
2	Even parity
3	Mark parity
4	Space parity

nFlowControl

The flow control of transmission, this parameter can be one the following values:

Value	Comment
0	No flow control
1	Xon/Xoff software control
2	Hardware control

bUseTAPI

Indicates whether a communication channel uses modem related Functions to handle the phone communication or not, if it is FALSE, ignore nModemIndex parameter.

nModemIndex

Used to assign a communication channel to a modem, this parameter is a 0-based index and a valid value must be between 0 and total number of modems - 1.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

If you use modem-related function (e.g. MakeCall Function and WaitForCall Function) to handle the phone communication, your application doesn't need to care about lpPortName, dwBaudRate, nDataBits, nStopBits, nParity and nFlowControl parameters, Micrsoft TAPI will take care of them automatically.

2.2 Modem

2.2.1 CloseTAPI Function

Closes TAPI after you finish TAPI-related modem operation(s) for all communication channels.

```
BOOL CloseTAPI();
```

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

If InitTAPI API is called, this Function must be called in order to shut down TAPI functions properly.

If a phone line connection is established, be sure to set the DropCall property to true before call this function.

See Also

InitTAPI Function

2.2.2 DropCall Function

Cuts off current established phone line connection for a communication channel inexplicitly identified by a modem index.

```
BOOL DropCall(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.2.3 InitTAPI Function

Initializes TAPI before your application conducts TAPI-related modem operation(s) for all communication channels.

```
BOOL InitTAPI();
```

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is

zero.

2.2.4 GetLineStatus Function

Retrieves phone line status for a modem communication channel.

```
DWORD GetLineStatus(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

The return value can be one of the following values:

Value	Comment
1	There is an incoming call.
2	The call is proceeding.
3	The line is connected.
4	The line is disconnected.
5	The line is busy, please dial later.
6	No dial tone was detected.
7	The remote side does not answer.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.2.5 GetModemCount Function

Retieves the number of modems installed on PC.

```
DWORD GetModemCount();
```

Return Value

The return value is the number of modems installed on PC.

2.2.6 GetModemName Function

Retrieves the name of a modem explicitly identified by a modem index.

```
BOOL GetModemName(WORD nModemIndex, LPTSTR lpModemName, LPDWORD lpSize);
```

Parameters

nModemIndex

This parameter is a 0-based index and a valid value must be between 0 and total number of modems - 1.

lpModemName

A pointer to a buffer that receives a null-terminated character string containing the modem name.

lpSize

A pointer to the variable that receives the size, in characters, of the modem name string.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.2.7 GetModemPort Function

Retrieves the name of the port associated with a modem explicitly identified by a modem index.

```
BOOL GetModemPort(WORD nModemIndex, LPTSTR lpPortName, LPDWORD lpSize);
```

Parameters

nModemIndex

This parameter is a 0-based index and a valid value must be between 0 and total number of modems - 1.

lpPortName

A pointer to a buffer that receives a null-terminated character string containing the port name.

lpSize

A pointer to the variable that receives the size, in characters, of the port name string.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.2.8 MakeCall Function

Makes a phone call for for a modem communication channel.

```
BOOL MakeCall(WORD nChannelID, LPCTSTR lpPhoneNumber);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpPhoneNumber

A pointer to a null-terminated string containing the remote side phone number to dial in.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.2.9 WaitForCall Function

Waits for a call for a modem communication channel.

```
BOOL WaitForCall(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

Use SetPortParam function to define a relationship between a modem index and a communication channel ID (0-7).

See Also

SetPortParam Function | GetModemCount Function

2.3 File Transfer

2.3.1 GetXferStatus Function

Retrieves file-transfer status for a communication channel (direct serial connection or modem connection).

```
DWORD GetXferStatus(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

The return value can be one of the following values:

Value	Comment
1	A file-transfer session is aborted.
2	A file-transfer session is doing initialization.
3	Start to upload or download a file now.
4	One block of data are transferred successfully.
5	A file is uploaded or downloaded successfully.
6	A file-transfer session is finished successfully.

2.3.2 XferAddFile Function

Informs MegaPipe of the name of file which will be uploaded to the remote side for a communication channel (direct serial connection or modem connection).

```
BOOL XferAddFile(WORD nChannelID, LPCTSTR lpFileName);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpFileName

A pointer to a null-terminated string containing the relevant path name of file which will be uploaded

to the remote side.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks
All XModem protocols can only upload 1 file during one file-transfer session, your application only needs to call this API once for a communication channel.

YModem, YModem-G, ZModem and Kermit can upload multiple files during one file-transfer session, so your application maybe needs to call this API a few times if multiple files are uploaded for a communication channel.

2.3.3 XferClearAllFiles Function

Clears file name information in MegaPipe memory on the upload side for a communication channel (direct serial connection or modem connection).

```
BOOL XferClearAllFiles(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks
Call this API before you call XferAddFile() API.

2.3.4 XferGetCurrBytes Function

Retrieves the number of data bytes sent/received so far for a communication channel (direct serial connection or modem connection).

```
DWORD XferGetCurrBytes(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

The return value is the number of data bytes sent/received so far.

2.3.5 XferGetCurrFileName Function

Retrieves the name of file being transferred for a communication channel (direct serial connection or modem connection).

```
BOOL XferGetCurrFileName(WORD nChannelID, LPTSTR lpFileName, LPDWORD lpSize);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpFileName

A pointer to a buffer that receives a null-terminated character string containing the file name.

lpSize

A pointer to the variable that receives the size, in characters, of the file name string.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.3.6 XferGetCurrFileSize Function

Retrieves the size of file being transferred for a communication channel (direct serial connection or modem connection).

```
DWORD XferGetCurrFileSize(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

The return value is the size of file being transferred.

2.3.7 XferSetDstFile Function

Informs MegaPipe of the name of file which will be created on the download side for all XModem protocols for a communication channel (direct serial connection or modem connection).

```
BOOL XferSetDstFile(WORD nChannelID, LPCTSTR lpFileName);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpFileName

A pointer to a null-terminated string containing the relevant path name of file which will be created on the download side for all XModem protocols.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

Remarks

YModem, YModem-G, ZModem or Kermit doesn't need to touch this API, since the name of file on the download side is identical to the name of file on the upload side.

2.3.8 XferSetParam Function

Sets up file-transfer direction (upload or download) and file-transfer protocol type for a communication channel (direct serial connection or modem connection).

```
BOOL XferSetParam(WORD nChannelID, WORD nMode, WORD nProtocol);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

nMode

This parameters indicates file-transfer direction and it can be one of the following values.

Value	Comment
1	Upload file
2	Download file

nProtocol

This parameters indicates file-transfer protocol type and it can be one of the following values.

Value	Comment
0	XModem Checksum protocol
1	XModem CRC protocol
2	XModem 1K protocol
3	YModem protocol
4	YModem-G protocol
5	ZModem protocol
6	Kermit protocol

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.3.9 XferSetWorkDir Function

Sets up the work directory for a communication channel (direct serial connection or modem connection).

```
BOOL XferSetWorkDir(WORD Index, LPCTSTR lpWorkDir);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

lpWorkDir

A Pointer to a null-terminated string containing the work directory for a file-upload or a file-download session.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is

zero.

2.3.10 XferStart Function

Starts a file-transfer session for a communication channel (direct serial connection or modem connection).

```
BOOL XferStart(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

2.3.11 XferStop Function

Stops a file-transfer session for a communication channel (direct serial connection or modem connection).

```
BOOL XferStop(WORD nChannelID);
```

Parameters

nChannelID

A channel ID (0-7) used to identify a communication channel.

Return Value

If the function succeeds, the return value is a nonzero value, otherwise the return value is zero.

3 License

License agreement

1. License

* The Single Developer License allows 1 developer in your organization the royalty-free distribution (up to 10,000 users) of the software to the third parties, **each individual developer requires a separate Single Developer License as long as he or she needs access to MW6's product(s) and document(s)**.

* The 2 Developer License allows 2 developers in your organization the royalty-free distribution (up to 10,000 users) of the software to the third parties.

* The 3 Developer License allows 3 developers in your organization the royalty-free distribution (up to 10,000 users) of the software to the third parties.

* The 4 Developer License allows 4 developers in your organization the royalty-free distribution (up to 10,000 users) of the software to the third parties.

* The 5 Developer License allows 5 developers in your organization the royalty-free distribution (up to 10,000 users) of the software to the third parties.

* The Unlimited Developer License allows unlimited number of developers in your organization the royalty-free distribution (unlimited number of users) of the software to the third parties.

2. User Disclaimer

The software is provided "as is" without warrant of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or noninfringement. MW6 assumes no liability for damages, direct or consequential, which may result from the use of the software. Further, MW6 assumes no liability for losses caused by misuse or abuse of the software. This responsibility rests solely with the end user.

3. Copyright

The software and any electronic documentation are the proprietary products of MW6 and are protected by copyright and other intellectual property laws.