32-bit Power Page v2.02

Compuserve# 75170,776
Internet: ron@ntpage.com
Web: http://www.inventiveweb.com
Ron Tanner (303) 689-0720

New in Version 2: MODEM SHARING, PCS compatibility

What is Power Page 32?

Power Page is a 32-bit Windows DLL that allows you to send Alpha-numeric (full text) messages to pagers. You can integrate the power of paging into your application easily with a single function call from virtually any 32-bit language including:

Visual Basic 4.0, 5.0
Delphi 2,3
Visual FoxPro 3.0, 5.X
PowerBuilder 5,6
Access for Windows 95 (Access 97)
Visual C++ version 2.0 or above
or any other development tool that supports calling 32-bit DLL functions.

Other versions of Power Page available separately:
16-bit Windows DLL
DOS Version with CLIPPER / C examples
(includes DOSPAGE - a DOS command line utility to send pages)
Windows NT Console EXE – command line utility

See www.ntpage.com for information on a complete server based solution.

The DLL function is defined as follows:

send_alpha_page(port, baud, modem_init_str, modem_dial_str,access_number, pager_id, max_chars_per_blk, msg, debug_file, retmsg)

By making this simple call, you can send a message to anyone anywhere.

Other functions available in the POWERP32.DLL: (REGISTERED users only)

PowerPage() - allows you to put your own test in the display window or allows you to run PowerPage minimized
Beeper() - use your modem to automate dialing into a standard numeric beeper's phone number. (Unregistered version replaces digits with 1's)
MultiTaskPowerPage() - spawns a thread to send the page and then returns

control back to the calling program
GetPowerPageStatus() - Gets the current status of the executing thread
CancelPowerPage() - Stops the executing thread
PowerPageFile() -Same as PowerPage function except the parameters are passed via a file. This is handy if the parameter sizes exceed the maximum size allowed.

NOTE: THE UNREGISTERED VERSION OF THIS SOFTWARE LIMITS THE MESSAGE TO 40 CHARACTERS. THE REGISTERED VERSION ALLOWS FOR AN UNLIMITED MESSAGE SIZE.

You could use Power Page to:
- Send Time-critical information.
- Notify support technicians of an error or warning.
- Integrate messaging into your application.
- Send alerts or reminder notes. (Wake up calls)
- Send scheduling or meeting notices.
- Provide customers an easy way to ask technical support questions.
- Integrate with dispatch applications.

Power Page makes paging integration a cinch!

What is required?

Microsoft Windows 95 or Windows NT

POWERP32.DLL installed in a directory on your path or the Windows System directory.

A modem and phone line connected to the computer (COM1 - COMX)

In addition, you will need to know some information about the paging company and pager of the person you wish to send a message to. This is described later.

What does it cost? (and ORDER information)

UNREGISTERED USERS:

This is not free software. The license allows you to use this software for evaluation purposes without charge for a period of 20 days. If you use this software after the 20 day evaluation period a registration fee is required.

The cost of a licenses of Power Page 32 is $35 (US) per license. A license is for a single computer per the licensing terms stated below. An unlimited royalty-free distribution license with source code is $350 (US).

If you are a current registered user of Power Page, you can upgrade to the 32-bit DLL for 40% off the new price ($21 US per CPU) or ($210 US for the unlimited version with source code).

Quantity discounts, site licenses, or custom modifications are also available by contacting the author:

Ron Tanner
4955 E. Preserve Court
Greenwood Village, CO 80121
Phone: 303-689-0720
Fax: 303-689-0730

To order, please call the author at the above number. Charge cards are accepted. (Visa / MasterCard/American Express)

You may also charge it to your compuserve account. Just GO SWREG and enter one of the registration id numbers below. You will receive your registered version of the Power Page DLL via Compuserve e-mail within 48 hours.

Windows 16-bit DLL ($25) - #4920
Windows 32-bit DLL ($35) - #7794
Windows NT Console EXE ($35) - #8355
DOS Version (includes DOSPAGE command line utility) ($25) - #6077
Windows 16-bit DLL Unlimited license with source code ($300) - #7795
Windows 32-bit DLL Unlimited license with source code ($350) - #7796
Windows NT Console EXE Unlimited license with source ($350) - #8356

UNREGISTERED USERS:
Please see the file ORDER.TXT for more information on how to order.

What is the difference between the registered and unregistered version?

The registered version offers the following enhancements:

1. The 40 character message limitation has been lifted. You can send a message up to 1000 characters long. If the message is bigger than the maximum characters per block limit of your paging company, then Power Page will split the message up into multiple pages.

2. You may send multiple messages per call. For example, if you want to send a message to all the salesmen, this is possible now with one phone call. You do this by passing several pager id's and one or

several messages separated by the ^ (shift-6) character.

3. The PowerPage(), MultiTaskPowerPage() and associated functions are available to be called.

How do I Install Power Page?

To install Power Page, simply copy the POWERP32.DLL file to the Windows System directory. (For NT: \WINNT\SYSTEM32)

What do I need to know about the Pager?

The only thing you need to know about the pager is the pager phone number or the Pager ID (Alpha PIN). For example, I have a Motorola Advisor on AT&T in Colorado. The phone number to my pager is 8260590. (Do not pass a dash in the phone number when you call the function)

You will need to know the following about the paging company of the pager. (In my case, the paging company is AT&T):

1. Access Number. You will need to know the paging terminals alphanumeric paging access telephone number. This is the phone number of the modem at the paging company that will receive the message using the TAP protocol. This number is the same for everybody that has a pager from that company. In my case, AT&T's access phone number is 303-799-0055. Many carriers will offer 800 numbers - especially the nationwide guys like SkyPage or Mobilcomm.

2. The baud rate. You need to know the paging company's modem baud rate associated with the access number. Believe it or not, most access modems still receive alphanumeric messages at 300 baud. Many have "upgraded" to 2400 baud. It doesn't really matter that much what rate they have - pager messages tend to be relatively short so the speed of transfer is not really that important.

3. Characters per message block. Most paging carriers limit how many characters can be sent in one message. A lot of carriers limit the message size to 80 characters. Many others have expanded their service to 230 characters per message. The size of message really becomes a factor of the carrier. You will need to find out this information or learn this through trial and error.

4. Password? Does your paging carrier require a password. Most US companies DO NOT. In this case, Power Page sends six zeros as the

password. This is the standard way of doing it. If you require a special password, see the note under "access number" below.

REGISTERED USERS NOTE:
You can send any size of message to Power Page. Power Page will automatically split them up into several messages to fit the characters per message block. Unregistered user's messages are chopped off at 40 characters.

What else do I need to know?

You will also need to know what port your modem is connected to. You may need to know what kind of modem is attached and perhaps custom configure it by passing Power Page a modem initialization string.

How do I make the DLL function call?

send_alpha_page() Description

The following describes the "send_alpha_page" function call contained in the POWERP32.DLL.

Note: All strings must be ASCIIZ strings (Null Terminated)

```
int send_alpha_page(port, baud, modem_init_str, modem_dial_str,
        access_number, pager_id, max_chars_per_blk, msg, debug_file, retmsg)

char *port;                 /* Port that modem is connected          */
char *baud;                 /* Baud rate of the paging company       */
char *modem_init_str;       /* Any special setup string your modem needs */
char *modem_dial_str;       /* Any special dial string your modem needs  */
char *access_number;        /* The paging company's alpha access phone # */
char *pager_id;             /* The pager phone number                */
char *max_chars_per_blk;    /* The paging company's character limit / msg*/
char *msg                   /* The message to send to the pager      */
char *debug_file,           /* If you have problems give this a file name*/
char *retmsg,               /* The return message of how the initialization went */
```

Parameter	Description
port	Specifies the port that your modem is connected to. This can be "COM1" through "COM4" or “TAPI” (see modem sharing section below)

baud	Specifies the baud rate that the paging company's alpha access port communicates at. This can be any of the following: "300", "1200", "2400", "4800", "9600", "19200". In very rare cases, I have seen the paging terminal require N,8,1 rather than E,7,1. (Parity, data bits, stop bits) If this is the case, precede the baud rate with an N. (i.e. "N300") If you preceed the baud rate with “S” this is a psuedo Direct connect to the terminal. This is the “Stay Connected” option. When a page is to be sent, if Modem carrier is low, it dials the paging terminal, sends the page but does not hang up. It leaves the COM port open. When you send a second page, if carrier is still high, then it does not dial, but simply sends the page. (i.e. “S300”)
modem_init_str	Start off by leaving this string NULL. If the value is NULL, Power Page will send your modem an AT + a return. The modem must then respond with an "OK". If the default doesn't work, you will need to find the modem's manual and figure out a string that will work. Any time you need a <cr> replace it with the tilde character (~) (i.e. AT&F~) If you need to pause, the hat character (^) will give you a 1 second pause. (i.e. AT&F~^^AT~) The maximum length of the modem initialization string is 80 characters. (See the enclosed MODEMS.TXT file for a list of suggested modem init strings for various modems. THESE WERE INCLUDED FOR YOUR CONVENIENCE AND I DO NOT GUARANTEE THAT THEY WILL ALL WORK!
modem_dial_str	Start off by leaving this string NULL. If the value is NULL, Power Page will send your modem an ATDT when it is ready to dial. If only pulse dialtone is available, you will have to pass "ATDP" in this parameter.
access_number	This is the telephone number for the paging company's modem that receives alphanumeric messages. THIS IS DIFFERENT FROM THE PAGER TELEPHONE NUMBER. You may include dashes and comma's like any other modem phone number. (A comma usually pauses 2 seconds - i.e. 9,444-4444) (See above for more information) Note: For those paging carriers that require a specific password, this is where you put it. After the access phone number place the ^ hat character (shift 6) and then place the password. For example, if the paging carrier required the

password of L23456, then the access_number string would be: access_number$="9,444-4444^L23456"

By putting "DIRECT" in this string, Power Page assumes a direct connection. No dialing takes place and DTR is NEVER dropped.

pager_id — This is usually the pager's phone number or PIN number. DO NOT INCLUDE ANY DASHES in this number.

You may send multiple pages with one phone call. You do this by separating the pager_id's and messages by the ^ (hat - shift 6) character. For example, if I wanted to send myself two messages, I would set pager_id="8260590^8260590" and the msg field to msg="This is message 1.^This is message 2". Note: that there had better be the same number of ^ characters in the pager_id field as there is in the message field.

max_chars_per_blk — This is the number of characters the paging company allows per message block. This number varies widely from company to company. Some are as small as 60 characters and some are as big as 1000 characters. The norms seem to be 80 and 230 characters per block. If the message you pass is bigger than the character per block limitation, Power Page automatically splits your message up into multiple blocks, so the recipient gets the entire message. (Registered version only) However, he/she does get a different page for each block. (See above for more information)

msg — This is the message that goes to the pager. The un-registered version truncates this message to 40 characters.

Please note that you may send multiple messages separated by the ^ (hat - Shift 6) character. Please see pager_id description above for more information. If you want to send the same message to the entire list of pagers, insert the single message and terminate it with ^^. (i.e. Msg$="Test Page^^")

debug_file — Normally, this parameter is the empty string “”. In the event you are having trouble successfully completing a page, specify a valid file name for this parameter. Power Page will append to the end of the file some information on where the problem is. This file will also help me to determine where the problem is if you want to E-Mail me the file.

retmsg — This is a string returned by the PowerPage() function telling you what happened. See below for possible return codes. The PowerPage function returns the value of the message number. (0 = Successful) Make sure you pre-allocate 100 bytes for this string BEFORE you make the function call.

PowerPage() Description

The following describes the "PowerPage" function call contained in the POWERP32.DLL.

Note: All strings must be ASCIIZ strings (Null Terminated)

```
int PowerPage(port, baud, modem_init_str, modem_dial_str, access_number,
        pager_id, max_chars_per_blk, msg, debug_file, retmsg, line1, line2, line3,
        minimize)

char *port;                 /* Port that modem is connected              */
char *baud;                 /* Baud rate of the paging company           */
char *modem_init_str;       /* Any special setup string your modem needs */
char *modem_dial_str;       /* Any special dial string your modem needs  */
char *access_number;        /* The paging company's alpha access phone # */
char *pager_id;             /* The pager phone number                    */
char *max_chars_per_blk;    /* The paging company's character limit / msg*/
char *msg                   /* The message to send to the pager          */
char *debug_file,           /* If you have problems give this a file name*/
char *retmsg,               /* The return message of how the initialization went */
char *line1;                /* The text to display on line 1 of the box  */
char *line2;                /* The text to display on line 2 of the box  */
char *line3;                /* The text to display on line 3 of the box  */
char *minimize;             /* Y to minimize window initially            */
```

Parameter Description

The parameters for PowerPage() are identical to the send_alpha_page parameters with the following additions:

line1 — This is the first line of the Power Page Window. If this parameter is NULL or empty, "Power Page" will be displayed. It is up to you to add preceeding spaces in order to center the text.

line2 — This is the second line of the Power Page Window. If this parameter is NULL or empty, my copyright

will be displayed. It is up to you to add preceeding spaces in order to center the text.

line3 This is the third line of the Power Page Window. If this parameter is NULL or empty, "Registerd Version" will be displayed on this line. It is up to you to add preceeding spaces in order to center the text.

minimize If the first character in this string is "Y" then the power page window will not be displayed. A window is is not created if 'Y.

Beeper() Description

The following describes the "Beeper" function call contained in the POWERP32.DLL. The Beeper() function is provided so that you can send messages to "Numeric Only" pagers. It does not use the TAP protocol. This function simply takes the phone line off-hook, dials the pager phone number, waits x number of seconds, touch tones the digits to be displayed on the pager and hangs up.

If possible, I would suggest that you use the other function calls to send numeric pages through the paging carrier's TAP port. Some allow for this, some do not. The TAP protocol guarantees delivery to the paging company. You are notified when it fails. This IS NOT the case with this Beeper function. It just dials and hope it works. IT HAS NO WAY OF KNOWING WHETHER IT WAS SUCCESSFUL!

Note: All strings must be ASCIIZ strings (Null Terminated)

```
int Beeper(port, baud, modem_init_str, modem_dial_str, pager_phone_number,
        delay_seconds, digits_to_send, termination_str, retmsg)

char *port;                 /* Port that modem is connected          */
char *baud,                 /* Baud rate to communicate with modem   */
char *modem_init_str,       /* Any special setup string your modem needs  */
                            /* Leave this blank unless you have problems */
char *modem_dial_str;       /* Any special dial string your modem needs  */
                            /* Leave this blank unless you have problems */
char *pager_phone_number, /* The beepers phone number */
char *delay_seconds,        /* How long to delay from the time the beeper number
                               dialed and the display digits are dialed     */
char *digits_to_send,       /* What digits will display on the pager*/
char *termination_str,      /* The modem command used to terminate call */
```

```
                        /* Leave this blank unless you have special needs */
char *retmsg,           /* The return message of how the initialization went */
```

Parameter	Description
port	Specifies the port that your modem is connected to. This can be "COM1" through "COM4" or “TAPI” (See modem sharing section below for more information)
baud	Specifies the baud rate that will be used to communicate with your modem. This can be any of the following: "300", "1200", "2400", "4800", "9600", "19200".
modem_init_str	Start off by leaving this string NULL. If the value is NULL, Power Page will send your modem an AT + a return. The modem must then respond with an "OK". If the default doesn't work, you will need to find the modem's manual and figure out a string that will work. Any time you need a <cr> replace it with the tilde character (~) (i.e. AT&F~) If you need to pause, the hat character (^) will give you a 1 second pause. (i.e. AT&F~^^AT~) The maximum length of the modem initialization string is 80 characters.
modem_dial_str	Start off by leaving this string NULL. If the value is NULL, Power Page will send your modem an ATDT when it is ready to dial. If only pulse dialtone is available, you will have to pass "ATDP" in this parameter.
pager_phone_number	This is the telephone number for the beeper. This is the number on the beeper that people call to touch tone in digits.
delay_seconds	How many seconds to delay before touch toning the digits that will appear on the pager.
digits_to_send	This is the digits that will appear on the pager.
termination_str	Normally, this parameter is NULL. When this parameter is NULL, it uses a #,H; as the termination string. The # is normally used when you have finished touch-toning in the digits to send to the pager. The comma tells the modem to wait 2 seconds before hanging up. The H tells the modem to hang up when its done. the semi-colon puts the modem in command mode and sends an OK to the computer when it has finished dialing.

retmsg	This is a string will return “Successful”. The Beeper() function has no way of knowing if anything went wrong. Make sure you have pre-allocated space for this variable before you make the function call.

NOTE: If you are using the PowerPage() function to run minimized or to put your own copyright notice in the window, there is a way to send a Beeper() message using the PowerPage() function. Call PowerPage() as you usally do, but replace the following parameters with the following values:

baud - baud rate should be preceeded with a B (i.e. B1200) This tell PowerPage to go into beeper mode.

access_number - replace this with the termination string described above. (Usually this will be NULL)

pager_id - Put in the beepers phone number. (pager_phone_number described above)

max_chars_per_blk - Put the delay_seconds described above into this parameter.

msg - Put the digits_to_send described above into this parameter.

PowerPageFile()

The following describes the “PowerPageFile” function call contained in the POWERPG DLL. The PowerPageFile() function functions just like the PowerPage() function, except the parameters are passed in a file.

Note: All strings must be ASCIIZ strings (Null Terminated)

int PowerPageFile(filename, retmsg,start,msgs_sent)

char *filename;	/* The full path and name to the file */
char *retmsg,	/* The return message of how the initialization went */
char *start,	/* Which message in the file you wish to send first */
char *msgs_sent,	/* Return value of how many messages were sent during session */

Parameter	Description
filename	The full path and file name to the specially formatted file needed for this function. (i.e. c:\powerpg\page.txt)

retmsg	This is a string returned by Power Page telling you what happened during the call. See below for a complete description of the possible values in this string. Make sure you pre-allocate 100 bytes for this string BEFORE you make the function call.
start	Normally you will pass a “0” or “1” to this function. This parameter gives you the option to start sending the Nth message in the file. This is particularly helpful when you had a previous phone connection loss but several pages had already been sent before the disconnection. This would eliminate the situation of the customers getting the same message twice.
msgs_sent	The function RETURNS the number of messages sent to the the paging terminal during the call. If everything worked fine, then this would be the number of messages in the file. If you lost your phone connections, this parameter would return how many messages were sent before the failure. Make sure you pre-allocate 5 bytes for this string BEFORE you make the function call.

The contents of the “filename” are described below:

NOTE: a semi-colon (;) in the first column indicates a comment.

The first non-commented line must contain “Header=“. Following the Header=, all the PowerPage() parameters (except pager_id and message) must be comma delimited.

NOTE: if you need to put a comma in the middle of a parameter (i.e. Pause for access_number), you must replace the comma with a tilde (~).

Following Header= must be: port, baud, modem_init_str, modem_dial_str, access_number, max_chars, debug_file, line1, line2,l ine3, minimize. (See the descriptions of send_alpha_page() and PowerPage() above for a detailed description of these parameters.

The Header= line must be terminated by a <CR><LF> pair.

Each line following the Header= line is a message to be delivered to a pager. It consists of a pager_id and a message - separated by a comma and terminated by a <CR><LF>.

If the carrat (^) character is substituted for the message, then the message from the previous line is sent to that pager.

An example file would be as follows:

```
;*********************************************************************
; This is a sample file that is used with PowerPageFile()
;*********************************************************************
;
;Port,Baud,modem_init,modem_dial,access_number,max_chars,
;debug_file,line1,line2,line3,minimize
Header=COM1,300,,,9~7990055,230,,,,,N
8260590,Test Message
2334444,^
8260590,Different Message
2334444,Totally different message
```

MultiTaskPowerPage() Description

The following describes the "MultiTaskPowerPage" function call contained in the POWERP32.DLL.

The MultiTaskPowerPage() function validates the parameter values and opens the COM port. It is up to the programmer to call the GetPowerPageStatus() function to determine when the page completes. (See Multi-task Power Page Message Communications below)

Note: All strings must be ASCIIZ strings (Null Terminated)

```
int MultiTaskPowerPage(port, baud, modem_init_str, modem_dial_str,
        access_number, pager_id, max_chars_per_blk, msg, debug_file, retmsg,
        whndl, stat_chg_msgid, finished_msgid)

char *port;                 /* Port that modem is connected              */
char *baud;                 /* Baud rate of the paging company           */
char *modem_init_str;       /* Any special setup string your modem needs */
char *modem_dial_str;       /* Any special dial string your modem needs  */
char *access_number;        /* The paging company's alpha access phone # */
char *pager_id;             /* The pager phone number                    */
char *max_chars_per_blk;    /* The paging company's character limit / msg*/
char *msg                   /* The message to send to the pager          */
char *debug_file,           /* If you have problems give this a file name*/
char *retmsg,               /* The return message of how the initialization went */
long whndl,                 /* The calling Window's Handle               */
unsigned int stat_chg_msgid,  /* WM_USER message id for status change */
```

unsigned int finished_msgid); /* WM_USER message id when finished */

Parameter Description

The parameters to the MultiTaskPowerPage() function are identical to the send_alpha_page() function with the following additions:

whndl	This is the Window handle of your program that is calling the PowerPage function. This is used to optionally send you a Windows message upon status change or completion. You may specify 0 for this parameter if you do not wish Power Page to send you any WM_USER messages. If you receive a WM_USER, you must call the GetPowerPageStatus() function to receive the message number and string. (See "Power Page Message Communication" below for more information.
stat_chg_msgid	If this parameter is greater than 0, then Power Page will send you a Windows message every time the status string changes. (i.e. Dialing, Connected, Sending Block, etc.) This parameter specifies the Windows Message number that Power Page will use. (i.e. 1024 = WM_USER, 1025 = WM_USER+1 etc.)
finished_msgid	If this parameter is greater than 0, then Power Page will send you a Windows message when the page is complete. This parameter specifies the Windows Message number that will be posted. (i.e. 1024 = WM_USER, 1025 = WM_USER+1 etc.)

Multi-Task Power Page Message Communication

The MultiTaskPowerPage function returns control immediately after validating the parameter values and opening the COM port. It is then up to the application programmer to make a call to the GetPowerPageStatus() function to get the current status or completion status. (See below for a description of this function)

Once you have called the MultiTaskPowerPage() function and received a zero return code, there are two different ways to monitor the progress of the call and determine the completion and success of the page. Both methods involve calling the GetPowerPageStatus() function. GetPowerPageStatus() returns a 0 if it is in-progress and a 1 when the page is completed. Once the page is completed, you must examine the message code to determine if the page was successful.

Method 1

Make a call to the PowerPage function passing your window handle and message id's for the status change and finished parameters as follows:

ret = MultiTaskPowerPage("COM1","300","","","7990055","8260590","230","Test Page","",retmsg,hwnd,1024,1025);

If the MultiTaskPowerPage() function is successful, it will create its own thread and immediately start sending the message(s) to the paging terminal.

Every time the status message changes, a WM_USER message (1024) will be sent to your window. In your Window message handling procedure define the WM_USER case as follows (the way you do this varies on the tool)

ret = GetPowerPageStatus(port, &message_code, retmsg);:

if ret equals zero (In-progress) display the retmsg in a dialog or status line bar.

When the new MultiTaskPowerPage thread has completed, then it will send a WM_USER+1 message to your window with the completion code.

Method 2 - (Polling)

If you are having difficulty getting the window handle or receiving the WM_USER messages, then you must create a polling mechanism to periodically call GetPowerPageStatus().

You can do this with a timer control or any other method that you desire. Periodically call GetPowerPageStatus until the return code is greater than zero.

GetPowerPageStatus() Description

The following describes the "GetPowerPageStatus" function call contained in the POWERP32.DLL.

The GetPowerPageStatus() function returns the current status of the call or the completion code of the call. You should call MultiTaskPowerPage() and receive a 0 return code before calling this function.

Note: All strings must be ASCIIZ strings (Null Terminated)

int GetPowerPageStatus (port, message_code, retmsg)

```
char *port;                 /* The COM port you wish to inquire on  (i.e. COM1)
int *message_code,          /* The message code (see below for list */
```

char *retmsg); /* The message string */

Parameter	Description
port	The COM port that you wish to inquire on. It is possible to have several COM ports active at one time, and this differentiates which port you wish to get the status on. Example: “COM1”, “TAPI”, “TAPIA12” (see modem sharing section below for more information)
message_code	Specifies the current status or completion code. This is a number corresponding to the message numbers below.
retmsg	This is a string returned by the function telling you the current status or completion status of what’s happened during the sending the page. Make sure you pre-allocate 100 bytes for this string BEFORE you make the function call.

Return codes and strings for the GetPowerPageStatus() function

RETURNS:
0 = Call in Progress
1 = Call completed

The possible values for the message_code and retmsg parameters are as follows:

Call In Progress codes

Msg Code	Retmsg
30	Initializing Modem
31	Dialing <phonenumber>
32	Connected
33	Sending Password
34	Send Msg <msg#> Block <blk#>
35	Password Accepted
36	Disconnecting

For call completed codes, see below.

CancelPowerPage() Description

The following describes the "CancelPowerPage" function call contained in the POWERP32.DLL.

The CancelPowerPage() function cancels a call in progress. The function always returns a value of 0. You must call the GetPowerPageStatus() function until the call is actually terminated. Check the return of the GetPowerPageStatus() until it is non-zero. Then you know the call has been terminated.

int CancelPowerPage(port)

char *port; /* The COM port you wish cancel (i.e. COM1)

Parameter	Description
port	The COM port that you wish to cancel. It is possible to have several COM ports active at one time, and this differentiates which port you wish to cancel. Example: “COM1”.

RETURNS: 0

IsCarrier() function

The following describes the "IsCarrier" function call contained in the POWERP32.DLL.

The IsCarrier() function determines the state of the carrier detect signal on the modem. The function returns a 1 if it is HIGH or ON and a 0 if it is LOW or OFF. This function is not usually used but may be helpful for certain situations.

int IsCarrier(port)

char *port; /* The COM port you wish inquire on (i.e. COM1)

Parameter	Description
port	The COM port that you wish to check. It is possible to have several COM ports active at one time, and this differentiates which port you wish to check. Example: “COM1”.

RETURNS:
1 if carrier is high
0 if carrier is low
-1 if port is not active

DropCarrier() function

The following describes the "DropCarrier" function call contained in the POWERP32.DLL.

The DropCarrier() function is used for those who use the “Stay Connected” feature. (See baud rate description) This function allows you to hang up and close the COM port when you are finished with the connection.

int DropCarrier(port)

char *port; /* The COM port you wish inquire on (i.e. COM1)

Parameter	Description
port	The COM port that you wish to terminate. It is possible to have several COM ports active at one time, and this differentiates which port you wish to check. Example: “COM1”.

RETURNS:
0 if successful
-1 error

Return Codes:

Return codes and strings for the PowerPage functions (during initialization)

RETURNS:
0 = Successful
>0 = Failure

The possible values for the retmsg parameters are as follows:

Ret Code	Retmsg
0	Successful.
4	The number of pager id’s does not match the number of messages.
5	Error opening COM port - rc: <Win32 error code>
6	Error GetCommState - rc: <Win32 error code>
7	Error SetCommState - rc: <Win32 error code>
8	Error SetCommTimeouts - rc: <Win32 error code>
9	Error SetupComm - rc: <Win32 error code>
26	You passed a zero length message.
27	You passed a zero block limit.
28	Returning with init error - The PowerPage DLL is busy
29	You are not allowed to have a <CR> character in the message

Call In Progress codes

Msg Code	Retmsg
30	Initializing Modem
31	Dialing <phonenumber>
32	Connected
33	Sending Password
34	Send Msg <msg#> Block <blk#>
35	Password Accepted
36	Disconnecting

Call Completed codes

0	Successful.
1	Operator interrupted session.
16	CloseComm Error.
17	Your modem on COM<x> did not respond to the following init string: <init string>
18	Timeout waiting for CONNECTion.
19	Lost Connection during call.
20	Timeout waiting for ID=.
21	Timeout waiting for goahead string.
22	Lost Connection during call.
23	Block was NAK'd five times.
24	XX Invalid Pager ID's - Bad List: XX XX .. XX .

Note: This error occurs if at least one of the pager ID's are invalid. A pager could be invalid if it is an incorrect pager ID or the pager has been turned off - such as when a customer stops service.

Since Power Page supports sending several messages in a single phone call, Power Page will attempt to send all the messages that you have sent - even if one of pager numbers is no longer valid.

This error message will return the number of messages that were bad - along with the list of which messages failed.

For example, If I sent the following:

pager_id$="8260590^6666666^8260590"
msg$="One,One^Two,Two^Three,Three"
and assuming 6666666 was not valid in the paging terminal,
I would get the following return:

01 Invalid Pager ID's - Bad List: 02 .

The 02 refers to which message failed.

Modem Sharing

New to version 2.0 is the ability for multiple applications to share the same modem. This is very useful if you want a fax application to wait for faxes on the same port that you want to send out pages. If you have assigned the COM port to RAS (under NT), Power Page can now use that port as well.

In order to do "modem sharing", Power Page had to talk to the COM ports using the TAPI interface. The TAPI interface handles all the details once a program "talks TAPI" through a more difficult than should be interface.

To turn on modem sharing, simply pass "TAPI" for the port parameter. Power Page will search through all the compatible devices in the system and attempt to send the page. If one port doesn't work, it will try the next port until it has exhausted all the compatible TAPI ports.

If you wish to force Power Page to a specific device, you can pass a device number. For example, "TAPI1" would use only the first modem in your system. You can get the list of modems by double-clicking on the Modems icon in the control panel.

If you want to run multiple simultaneous ports, you may still assign each port to "TAPI". The second thread will simply see the first one is busy and select the next one. However, I recommend that each thread COM port is named something unique. Power Page uses the port name that you pass to keep track of the various threads. One simple way to control things is to assign a device to each thread such as "TAPI1", "TAPI2" etc. If you wish for everything to be dynamic, then start naming your ports "TAPIA", TAPIB", etc.

If Power Page does not see a numeric character after the TAPI, it assumes you want to use any and all of the available compatible devices.

For most applications, simply pass "TAPI" as the port parameter and modem sharing is enabled.

Troubleshooting (Common Problems)

Error 21 - Timeout waiting for ID=

By far the most common problem is the failure to communicate with the paging terminal. This is also returned when there is "NO DIAL TONE" or some other modem problem.

The solution to this problems is sending the appropriate modem initialization string. Many paging terminals use old 300 baud modems. The do not take too kindly to the error correcting and data compression protocols of the newer V.XX modems. Therefore, your goal is to find a modem initialization string that will turn off all of that and set your modem to "normal" Bell 212 mode. (if possible) There are too many modems to accumulate a list of these strings - and they vary very widely. Some of the commercially available packaged alpha dispatching software has a list of these strings.

May I also suggest that you try your new string using a COMM program like the Windows terminal program. Set the baud rate to 300 baud, Even parity, 7 data bits, and 1 stop bit. XON / XOFF flow control should be OFF. Dial manually and after you see the CONNECT message, press ENTER (CR). If you are communicating properly, you will see the string "ID=". Play with it until you do, then try Power Page again.

Enclosed also is the MODEMS.TXT file which lists the suggested initializations strings for the various modems. These were included for your convenience. I do not guarantee that they all work.

GPF's

If you got a GPF during the call to PowerPage() or GetPowerPageStatus() it is most likely caused because you did not pre-allocate space for the retmsg$ parameter.

Power Page returns the message strings into the retmsg$ parameter. In many languages, string space is allocated as needed - so when you simply define a string, the space is not yet allocated. Therefore, you should pre-allocate the retmsg$ string with a command equivalent to: Retmsg$=Space$(100)

Getting stuck during sending block (NAK 5 times message)

Most likely you have a unacceptable pager ID. PageNet assigns separate alpha PIN numbers than the pager phone number. The alpha pin number is what you should use in the Pager ID parameter.

EXAMPLES DLL CALLS

The setup and call from your favorite language would be similar to how you would make a call to a Windows API (SDK) function. Some specific examples follow:

NOTE: The Power Page DLL uses the "Pascal" or "Standard Call" calling convention (as opposed to the "C" calling convention. (this is the standard for the

Win32 API) If your page is going through, but are having problems when PowerPage returns, then this could be the problem. Please contact the author.

Check out the web page http://www.inventiveweb.com **for working examples.**

PowerBuilder for NT Example

Tested using PowerBuilder for NT (Intel) 4.0

Declare Global External Functions:

```
FUNCTION int send_alpha_page(ref string port, ref string baud, &
        ref string modem_init_str, ref string modem_dial_str, &
        ref string access_number, ref string pager_id, &
        ref string max_chars_per_blk, ref string msg, &
        ref string debug_file, ref char retmsg[100]) &
         LIBRARY "powerp32.dll"
```

Script:

```
int rc
char retmsg[100]
string port, baud, modem_init_str, modem_dial_str
string access_number, pager_id, max_chars_per_blk, msg, debug_file

port="COM1"
baud="300"
modem_init_str=""
modem_dial_str=""

max_chars_per_blk="80"
access_number="7990055"
pager_id="8260590"

msg="Test page from Ron Tanner"
debug_file="\ron.txt"

rc= send_alpha_page(port, baud, modem_init_str, &
        modem_dial_str, access_number, &
        pager_id, max_chars_per_blk, msg, debug_file, retmsg)

MessageBox("Done",retmsg)
```

VISUAL BASIC 4 or Access for Windows 95 Example

Example tested using Visual Basic 4.0 32-bit mode

In the declarations area add the following line (put on one line):

```
Declare Function send_alpha_page Lib "POWERP32.DLL" (ByVal port$, ByVal baud$,
    ByVal modem_init_str$, ByVal modem_dial_str$, ByVal access_number$,
```

```
    ByVal pager_id$, ByVal max_chars_per_blk$, ByVal msg$, ByVal debug_file$,
    ByVal retmsg$) As Integer
```

A Sample script to setup and send a page:

```
port$ = "COM1"
baud$ = "300"
modem_init_str$ = ""
modem_dial_str$ = ""
access_number$ = "7990055"
pager_id$ = "8260590"
max_chars_per_blk$ = "80"
msg$ = "Howdy partner."
debug_file$ = ""
retmsg$ = Space$(100)

Rem -- Put the following on one line:
ret% = send_alpha_page(port$, baud$, modem_init_str$, modem_dial_str$, access_number$,
    pager_id$, max_chars_per_blk$, msg$, debug_file$, retmsg$)

MsgBox retmsg$
```

VISUAL FoxPro Example

Note: Example tested using Visual FoxPro 3.0

```
LOCAL port, baud, modem_init_str, modem_dial_str, access_number, pager_id, ;
        max_chars_per_blk, msg, debug_file, retmsg, retval

DECLARE INTEGER send_alpha_page ;
        IN POWERP32 ;
        STRING @port, STRING @baud, STRING @modem_init_str, ;
        STRING @modem_dial_str, STRING @access_number, STRING @pager_id, ;
        STRING @max_chars_per_blk, STRING @msg, ;
        STRING @debug_file, STRING @retmsg

port="COM1"
baud="300"
modem_init_str=""
modem_dial_str=""
access_number="7990055"
pager_id="8260590"
max_chars_per_blk="80"
msg="Howdy partner."
debug_file=""
retmsg=space(100)

retval=send_alpha_page(@port, @baud, @modem_init_str, @modem_dial_str,  ;
        @access_number, @pager_id, @max_chars_per_blk, @msg, ;
        @debug_file, @retmsg)
? retmsg
```

C or C++ Example

Tested using Visual C++ 2.2

```
int sample_function(void)
{
        char retmsg[100];
        char dispmsg[120];
        HANDLE hLib = NULL;
        typedef int (pascal *PFNDLL1)();
        PFNDLL1 gpfnDLLFunction1 = NULL;
        int ret;

        if (!(hLib = LoadLibrary ("POWERP32.DLL"))) {
          MessageBox (0,(LPCTSTR) "MainWndProc(): LoadLibrary () failed",
            (LPCTSTR) "Err! - LOADTEST",MB_OK | MB_ICONEXCLAMATION);
          return(-1);
        }
        else {
          gpfnDLLFunction1 = (PFNDLL1) GetProcAddress (hLib, "PowerPage");
        }

        ret=gpfnDLLFunction1("COM1","300","","","7990055","8260590","230",
                "Test Page","c:\\sw\\powerpg\\nt\\101.00\\ron.dbg",retmsg,"This is line1",
                "This is line2","This is line3","N");
        sprintf(dispmsg,"%d - %s",ret,retmsg);
        MessageBox(NULL,(LPCTSTR) dispmsg,(LPCTSTR)"Return Status",MB_OK);
        FreeLibrary(hLib);
        return(0);
}
```

MultiTaskPowerPage and GetPowerPageStatus Example

Visual Basic and Access

In general object of a "module":

```
Declare Function MultiTaskPowerPage Lib "POWERP32.DLL" (ByVal port$, ByVal baud$, ByVal
modem_init_str$, ByVal modem_dial_str$, ByVal access_number$, ByVal pager_id$, ByVal
max_chars_per_blk$, ByVal msg$, ByVal debug_file$, ByVal retmsg$, ByVal whndl&, ByVal
stat_chg_msgid%, ByVal finished_msgid%) As Integer
Declare Function GetPowerPageStatus Lib "POWERP32.DLL" (ByVal port$, ByRef mc%, ByVal
retmsg$) As Integer
```

In a command button of a form:

```
port$ = "COM1"
baud$ = "300"
modem_init_str$ = ""
modem_dial_str$ = ""
access_number$ = "799-0055"
pager_id$ = "8260590"
```

```
max_chars_per_blk$ = "220"
msg$ = "Test page from Windows Messenger for Windows95"
debug_file$ = "ppgerror.txt"
retmsg$ = Space$(100)
whndl& = 0
stat_chg_msgid% = 0
finished_msgid% = 0 '(I don't need any of these three)

ret% = MultiTaskPowerPage(port$, baud$, modem_init_str$, modem_dial_str$,
access_number$, pager_id$, max_chars_per_blk$, msg$, debug_file$, retmsg$, whndl&,
stat_chg_msgid%, finished_msgid%)

MsgBox retmsg$
```

The call to GetPowerPageStatus:

```
retmsg$ = Space(100)
ret% = GetPowerPageStatus(port$, mc%, retmsg$)
```

LEGAL MATTERS

Of course the usual disclaimers still apply. We are not responsible for anything at all. Nothing. Even if we are held responsible, the limit of our liability is the licensing fees you paid.

SHAREWARE LICENSE - END USER

The Power Page software is not and has never been public domain software, nor is it free software.

Non-licensed users are granted a limited license to use our software on a 20-day trial basis for the purpose of determining whether the software is suitable for their needs. Any use of our software, except for the initial 20-day trial, requires registration. The use of unlicensed copies of our software, outside of the initial 20-day trial, by any person, business, corporation, government agency or any other entity is strictly prohibited.

SHAREWARE LICENSE - FOR DISTRIBUTION OF SHAREWARE FILES, USER GROUPS, BBS's, ONLINE SERVICES, SHAREWARE VENDORS, and OTHERS

A limited license is granted to copy and distribute our shareware software only for the trial use of others, subject to the following limitations:

1) The software must be copied in unmodified form, complete with the file containing this license information.

2) The full machine-readable documentation must be included with each copy.

3) Our software may not be distributed in conjunction with any other product with out a specific license to do so from Ron Tanner.

4) Vending of our software products in retail stores (by "shareware rack vendors") is specifically prohibited without prior written authorization.

5) No fee, charge, or other compensation may be requested or accepted, except as authorized below:

A) Non-profit user groups may distribute copies of the our products to their members, subject to the above conditions, without specific permission. Non-profit groups may collect a disk duplication fee not to exceed five dollars.

B) Operators of electronic bulletin board systems (sysops) may make our products available for downloading only as long as the above conditions are met. An overall or time-dependent charge for the use of the bulletin board system is permitted as long as there is not a specific charge for the download of our software.

C) Mail-order vendors of shareware software may distribute our products, subject to the above conditions, without specific permission. Vendors may charge a disk duplication and handling fee, which, when pro-rated to each individual product, may not exceed eight dollars.

LICENSED COPIES OF OUR SOFTWARE ARE GOVERNED BY THE FOLLOWING:

THIS SOFTWARE IS NOT FOR SALE: The software is subject to the following license terms and conditions.

SOFTWARE LICENSE granted, when required fees paid, by Ron Tanner, with its mailing address at 4955 E. Preserve Court, Greenwood Village, CO 80121. The software contained in this package is licensed to you as the end user. It is not sold.

LICENSE:

1.0 The software contained in this package (hereafter referred to as "the Software") is copyrighted material owned by Ron Tanner. Payment of the single copy license fee authorizes one named person to use the Software on one computer provided this copyright is not violated and provided the rules outlined herein are observed.

1.1 One person may use the Software on any single computer. This license can be transferred only once in any twenty-four hour period. You must pay for additional

copies of the Software if more than one person uses it at one time, or if the Software is used on two or more computers. Neither concurrent use on two or more computers, nor use by more than a single individual on a network is permitted without authorization and payment of other license fees.

1.2 You may make copies of the software for backup purposes, as long as all such copies, along with the original, are kept in your possession or control.

1.3 You may not make any changes or modifications to the Software, including, but not limited to, decompiling, disassembling, or otherwise reverse engineering it. You may not rent or lease it to others. You may not use it on a computer network if more than one user can use it on more than one computer during any one twenty-four hour span of time.

LIMITED WARRANTY

Ron Tanner guarantees your satisfaction with this product for a period of 90 days from the date of original purchase. If you are unsatisfied with the product within that time period, return the package in salable condition to the place of purchase for a full refund.

Ron Tanner warrants that all disks provided are free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase.

Ron Tanner warrants that the program will perform in substantial compliance with the documentation supplied with the software product. If a significant defect in the product is found, the purchaser may return the product for a refund. In no event will such a refund exceed the purchase price of the product.

EXCEPT AS PROVIDED ABOVE, RON TANNER DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE PRODUCT. SHOULD THE PROGRAM PROVE DEFECTIVE, THE

PURCHASER ASSUMES THE RISK OF PAYING THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL RON TANNER BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING WITHOUT LIMITATION DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OR THE INABILITY TO USE THIS PRODUCT EVEN IF RON TANNER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Use of this product for any period of time constitutes your acceptance of this agreement and subjects you to its contents.

TRADEMARKS

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.
Windows, Windows 95, Windows NT is a trademark of Microsoft Corporation.

Revision History

1.02 - 03 Nov 1995

Added Beeper() function.

1.03 - 27 Dec 1995

Changed the PowerPageFile() file so it wasn't limited by memory allocations. The line length can now be up to 1000 characters. Previously, it was limited to 100 bytes. Added two parameters to the call.

Added Universal Separator mode of TAP.

1.04 May 28, 1996

Added IsCarrier() function and dial once stay connected "S" option.
Added concurrent multi-port capability

1.05 October 18, 1996

Bug in splitting of messages caused by Universal Separator fix.

1.06 October 31, 1996

When a Visual Basic passes a Text control with no data in it, it passes a NULL pointer which caused POWERP32 to GPF. POWERP32 now tests for a NULL pointer.

1.08 June 14, 1997

On certain hardware, Power Page was unable to communicate with the COM port. Changed the order of the COMM API calls to insure the port would never be reset.

2.01 September 1, 1997

Added TAPI support (modem sharing). Tested on 95 and NT 4.0 with RAS loaded.

2.02 December 4, 1997

Bug fixes on port not closing properly, dial once stay connected.