LumiNet Publishing
Corporation
1905 Treetop Lane
Suite 41
Silver Spring, MD 20904
USA

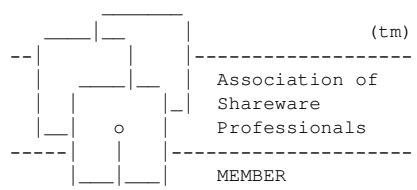
(301) 622-9642

LumiNet Tools-1(TM)

WordPerfect Document ToolKit

Create WordPerfect documents
from C and Clipper Programs

LumiNet(tm) Publishing Corporation



LumiNet-Tools-1(TM) complies with the standards of the
Association of Shareware Professionals.

LumiNet(tm) Publishing Corporation is an approved member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware principle works for you. If you are unable to resolve a shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products. Please write to the ASP Ombudsman at P.O. Box 5786, Bellevue, WA 98006 or send a Compuserve message via easyplex to ASP Ombudsman 70007,3536"

What is Shareware?

Shareware distribution gives you a chance to try software before buying it. If you try a Shareware program and continue using it, you are expected to register.

Copyright laws apply to both Shareware and commercial software, and the copyright holder retains all rights, with a few specific exceptions as stated below.

Shareware is a distribution method, not a type of software. You should find software that suits your needs and pocketbook, whether it's commercial or Shareware. The Shareware system makes fitting your needs easier, because you can try before you buy. And because the overhead is low, prices are low also. Shareware has the ultimate money-back guarantee -- if you don't use the product, you don't pay for it.

DISCLAIMER - AGREEMENT

Users of LumiNet-Tools-1(TM) must accept this disclaimer of warranty:
LumiNet-Tools-1(TM) is supplied as is. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author assumes no liability for damages, direct or consequential, which may result from the use of LumiNet-Tools-1(TM).

LumiNet-Tools-1(TM) is a "shareware program" and is provided at no charge to the user for evaluation. Feel free to share it with your friends, but please do not give it away altered or as part of another system. The essence of "user-supported" software is to provide personal computer users with quality software without high prices, and yet to provide incentive for programmers to continue to develop new products. If you find this program useful and find that you are using LumiNet-Tools-1(TM) and continue to use LumiNet-Tools-1(TM) after a 30 day trial period, you must make a registration payment of $55.00 to LumiNet(tm) Publishing Corporation. The $55.00 registration fee will license one copy for use on any one computer at any one time. You may distribute applications that use LumiNet-Tools-1(TM) royalty free. You must treat this software just like a book. An example is that this software may be used by any number of people and may be freely moved from one computer location to another, so long as there is no possibility of it being used at one location while it's being used at another. Just as a book cannot be read by two different persons at the same time.

Anyone distributing LumiNet-Tools-1(TM) for any kind of remuneration must first contact LumiNet(tm) Publishing Corporation at the address below for authorization. This authorization will be automatically granted to distributors recognized by the (ASP) as adhering to its guidelines for shareware distributors, and such distributors may begin offering LumiNet-Tools-1(TM) immediately (However LumiNet(tm) Publishing Corporation must still be advised so that the distributor can be kept up-to-date with the latest version of LumiNet-Tools-1(TM).

You are encouraged to pass a copy of LumiNet-Tools-1(TM) along to your friends for evaluation. Please encourage them to register their copy if they find that they can use it.

The LumiNet-Tools-1(TM) Shareware diskette, containing a copy of this manual may be freely copied and shared, but printed copies of this document may not be copied in any way without permission in writing from LumiNet(tm) Publishing Corporation. Thank you.

Registering LumiNet-Tools-1(TM)

LumiNet(tm) Publishing Corporation distributes LumiNet-Tools-1(TM) as Shareware. Should you continue to use LumiNet-Tools-1(TM) beyond a 30-day evaluation period, you MUST register your LumiNet-Tools-1(TM) copy for $55.00.

Please send THIS PAGE and a check or money order payable to:

LumiNet Publishing Corporation

And, mail to:

Patrick Todd
LumiNet Publishing Corporation
1905 Treetop Lane, Suite 41
Silver Spring, MD 20904

Your Name: _____

Company: _____

Address: _____

_____

City: _____

State: _____

Zip Code: _____

Phone: ( ) _____

Comments: _____

_____

Thanks from us for your support and encouragement!

Trademark Acknowledgments

Blinker is a registered trademark of Blink, Inc.
Borland is a registered trademark of Borland Corporation
Clipper is a registered trademark of Computer Associates, Inc.
LumiNet Tools-1(TM) and LumiNet(TM) are trademarks
of LumiNet Publishing Corporation
Microsoft is a registered trademark of Microsoft Corporation
MS-DOS is a registered trademark of Microsoft Corporation
WordPerfect is a registered trademark of WordPerfect Corporation

Table of Contents

What, Who and Why?

What is LumiNet Tools-1(TM)

LumiNet Tools-1(TM) is a collection of functions that can be used by a C or Clipper developer to create WordPerfect v5.1 documents. LumiNet Tools-1(TM) provides access to WordPerfect's page layup, fonts and text formatting capabilities. LumiNet Tools-1(TM) provides the developer with document creation and formatting functions that are fully compatible with WordPerfect version 5.1 standards, and can be accessed from any C or Clipper program.

All WordPerfect codes created from LumiNet Tools-1(TM) are identical to any formatting codes that you would create directly from within a WordPerfect document. These codes can be edited and deleted, just as if they were originally created from within WordPerfect!

Who Should Use LumiNet Tools-1(TM)

Any developer who is writing applications using C or Clipper, and has a requirement to merge application information into a WordPerfect document!

Why Should I Use LumiNet Tools-1(TM)

Seamless Interface! For any developer who is merging information from a database into WordPerfect documents, LumiNet Tools-1(TM) is a MUST!

If you have ever tried to write and debug a complex macro from within WordPerfect, you will appreciate this opportunity to avoid using WordPerfect's internal Macro features - and continue to write programs in the platform you are most comfortable!

Sure, you can create a WordPerfect macro that merges ASCII delimited database information into your 'preset form'...but what happens if your macro runs slowly (especially on a LAN), or your users cancel from the macro before it has completed? You end up with a partially completed or an empty WordPerfect document file!

The routines that make up LumiNet Tools-1(TM) allow you to merge database or text information from your application directly into a WordPerfect document - Quickly and Accurately! No more truncated macros, no more missing data! No more creating macros within WordPerfect!

New in LumiNet Tools-1(TM) Version 1.5

* Font changing capability! With LumiNet Tools-1(TM) v1.5 you have access to 13 different fonts that can be inserted into your WordPerfect document through the use of the wpfont() function.

* Page headers and footers can be created. Similiar to WordPerfect's Header A, Header B, Footer A and Footer B, you can now create headers and footers and include text, graphics, etc into this page layup feature. Page headers and footers are activated with the wphfstrt() function, and de-activated with the wphfend() function.

* Figure Graphics. You can now retrieve and size WordPerfect figure graphics (.WPG) files into your document using the wpfig() function.

* Page Orientation. Landscape and portrait orientation can be specified for page orientation with the wporient() function.

* Newspaper Columns. A dual column (equal sized) effect can be created through the use of the wpclmon() and wpclmoff() functions.

* Tab Settings. Tab spacing can now be specified for 10 horizontal positions along the width of your document with the wpsettab() function.

* Convert a WordPerfect file into an ASCII text file! By specifying the name of the existing WordPerfect (v5.1) file, it can be converted to an ascii file - that you can access from within your application. Use the wpascii() function!

argument..... parameter passed to the function

<arg1c...arg#c> character argument, with the first argument identified with a '1', the second argument identified with a '2', etc.

<arg1n...arg#n> numeric argument, with the first argument identified with a '1', the second argument identified with a '2', etc.

In some cases, the argument specified is in the format of a WordPerfect Unit (wpu). A WordPerfect Unit is an integer that is calculated by multiplying the desired value (in inches) by 1200. For example if the end result of 1/2 inch is required, then it's WordPerfect Unit equivalent is 1/2 * 1200, or 600.

All numeric values that specifically require the conversion to a WordPerfect Unit will be stated along with other details required for the argument.

System Requirements

The minimum equipment requirements to run LumiNet Tools-1(TM): IBM PC XT, AT, PS/2 or compatible computer with a hard disk, 512KB of RAM, MS/PC DOS 3.3 or later and, one of the following compilers:

- Clipper Summer '87 - CAClipper 5.2 from Computer Associates
- Microsoft C (up through version 6.0)
- Borland C,C++ (up through version 3.1)

Installation

LumiNet Tools-1(TM) is supplied on one 5.25 inch or one 3.5 inch diskette and consists of the following files:

CLP2WPC.LIB.....	Clipper Library
MSC2WPCS.OBJ.....	Microsoft C Small Model Object File
MSC2WPCM.OBJ.....	Microsoft C Medium Model Object File
MSC2WPCL.OBJ.....	Microsoft C Large Model Object File
BCC2WPCS.OBJ.....	Borland C,C++ Small Model Object File
BCC2WPCM.OBJ.....	Borland C,C++ Medium Model Object File
BCC2WPCL.OBJ.....	Borland C,C++ Large Model Object File
LumiNet1.DOC.....	LumiNet Tools-1(TM) Manual
READ.ME.....	Notes on last minute changes

To install LumiNet Tools-1(TM) files, copy the appropriate files (depending on your type of compiler) to your designated library directory.

Compatibility

LumiNet Tools-1(TM) is backward compatible with the previous version (named docMaker) - 1.0.

WordPerfect (version 5.1)

LumiNet Tools-1(TM) creates WordPerfect 5.1 compatible document files.

Clipper (Summer '87 & 5.2)

Using the LumiNet Tools-1(TM) library file CLP2WPC.LIB linked with your application provides compatibility with Clipper Summer '87 through CAClipper version 5.2.

See Appendix B - Linking for more information on linking object files with your application.

Microsoft C (through version 6.0)

Using the LumiNet Tools-1(TM) small, medium or large model object file (MSC2WPCS.OBJ, MSC2WPCM.OBJ, MSC2WPCL.OBJ), linked with your application provides compatibility with Microsoft C (up to and including version 6.0).

See Appendix B - Linking for more information on linking object files with your application.

Borland C, C++ (through version 3.1)

Using the LumiNet Tools-1(TM) small, medium or large model object file (BCC2WPCS.OBJ, BCC2WPCM.OBJ, BCC2WPCL.OBJ), linked with your application provides compatibility with Borland C and C++ (up to and including version 3.1).

See Appendix B - Linking for more information on linking object files with your application.

LumiNet Tools-1(TM)
Technical Reference
Guide

The following is a list of LumiNet Tools-1(TM) functions:

```
wpadd().....   Add text
wpattoff()...  Attributes off (see Appendix A)
wpatton()...   Attributes on  (see Appendix A)
wpascii()....  Convert WordPerfect files to ASCII files
wpclmoff()..   Newspaper Columns (End)
wpclmon()..    Newspaper Columns (Start)
wpclose()...   Close WordPerfect file
wpcntr().....  Centering
wpdate()....   System Date
wpfig()....... Figure Graphics
wpfont().....  Change Font
wphfend()..    Header/Footers (End)
wphfstrt()...  Header/Footers (Start)
wphline()....  Horizontal Line
wphpage().     Hard page break
wphrtn().....  Hard Return
wpindent()..   Indent
wpjust()...... Justification
wplrmarg().    Left Right Margins
wpopen()...    Create and leave Open WordPerfect file
wporient()...  Portrait/Landscape Page Orientation
wppagen()..    Page Numbering
wpsettab()..   Set Tabs
wptab()......  Tab
wptbmarg().    Top Bottom Margins
wpvline()..... Vertical Line
```

Function.................. wpadd(<arg1c>)
Description.............. Appends the text string to the document specified in <arg1c>.
Arguments................. <arg1c>=text string
Return..................... Nothing
Comments.................. This function appends the text string into the WordPerfect document - that was opened with the wpopen() call. This string can contain the contents of a character field from a text or database file, providing the developer with capability to create customized forms and letters based on the contents of other application files. Keep in mind that all characters passed to the wpadd() function must be in character format. Note: Use only ASCII characters in the range of 32 through 127!!

The wpadd() function does not take into consideration the need for hard and soft line returns. Your application will specifically need to issue a hard line return (see wphrtn()) as required.

Code Examples:
In this example, the wpadd() function is used to handle both variable and fixed text. In the Clipper example, the XBASE.FIRSTNAME represents the field First Name in the applications XBASE.DBF file. The value contained in this field is passed as an argument to wpadd(). This call is then followed by two more calls to the wpadd() function that simply passes 'fixed' text.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     wpadd(Alltrim(XBASE.FIRSTNAME)+":  ")
     wpadd("Please tell your friends ")
     wpadd("that they should buy this product!")
     wpclose()
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {    /* Adds some text  to the document */;
          wpadd("Please tell your developer friends ");
          wpadd("that they should buy this product!");
          ...
          wpclose();
     }
```

Function..... wpattoff(<arg1n>)
Description..... Appends a WordPerfect font attribute code to the document, which 'turn's off' that attribute for all text following this code. This function works in coordination with wpatton()
Arguments..... <arg1n>=the number of the attribute code (see attribute code values)
Return..... logical
<TRUE>=successful
<FALSE>=unsuccessful
Comments..... This routine appends the WordPerfect compatible attribute code as specified in the argument. All text that follows this attribute will no longer contain the features of the attribute, as it has been 'turned off'. NOTE: The wpattoff() can only turn OFF features that have been turned ON with the wpatton() call.

This function uses the following values to the font attributes:

0 = extra large	6 = subscript	11=double underline
1 = very large	7 = outline	12=bold
2 = large	8 = italics	13=strike out
3 = small	9 = shadow	14=underline
4 = fine	10=redline	15=small caps
5 = superscript		

Code Examples:

In the following examples, bold and italic attributes are turned ON using the wpatton() function. Any text following this call takes on the attributes of any previous call to wpatton() that has not yet been turned OFF. When all successive calls to wpattoff() have been completed, the text is returned to NORMAL.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     wpatton(12)         && Activate text bold feature
     wpadd("This text now appears bold"
     wpatton(8)          && Activate text italic feature
     wpadd("This text now appears bold and italicized")
     wpattoff(12)        && Deactivate bold
     wpattoff(8)         && Deactivate italic
     wpclose()
Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               /*  Adds Bold attribute to the document */;
               /*  Any text added to the document will be bolded */;
               wpatton(DM_BOLD);
               wpadd("This text now appears bold")
               /*  Adds Italic attribute to the document */;
               /*  Any text added to the document will be bolded and italic */;
               wpatton(DM_ITALIC);
               wpadd("This text is now bold and italicized")
               wpattoff(DM_BOLD);
               wpattoff(DM_ITALIC)
               wpclose();
          }
```

Function.................. wpatton(<arg1n>)
Description.............. Appends a WordPerfect font attribute code to the document, which 'turn's on' that attribute for all text following this code.
Arguments................. <arg1n>=the number of the attribute code (see attribute code values)
Return..................... logical
<TRUE>=successful
<FALSE>=unsuccessful
Comments.................. This routine appends the font attribute code as specified in the argument. All text that follows this attribute will contain the features of the attribute, until the attribute has been turned off, through the use of the wpattoff() function. Adding font attributes through the wpatton() function is an ADDITIVE process. All previous attributes created with wpatton() will remain active until a wpattoff() has been issued. Any text added by the wpadd() function will assume all active font attributes.

This function uses the following values to the font attributes:

0 = extra large	6 = subscript	11=double underline
1 = very large	7 = outline	12=bold
2 = large	8 = italics	13=strike out
3 = small	9 = shadow	14=underline
4 = fine	10=redline	15=small caps
5 = superscript		

Code Examples:
In the following examples, bold and italic attributes are turned ON using the wpatton() function. Any text following this call takes on the attributes of any previous call to wpatton() that has not yet been turned OFF. When all successive calls to wpattoff() have been completed, the text is returned to NORMAL.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     wpatton(12)    &&  Activate bold attribute
     wpadd("This text now appears bold"
     wpatton(8)     && Activate italic attribute
     wpadd("This text now appears bold and italicized")
     wpattoff(12)   && De-activate bold attribute
     wpattoff(8)    && De-activate italic attribute
     wpclose()
Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               /*  Adds Bold attribute to the document */;
               /*  Any text added to the document will be bolded */;
               wpatton(DM_BOLD);
               wpadd("This text now appears bold")
               /*  Adds Italic attribute to the document */;
               /*  Any text added to the document will be bolded and italic */;
               wpatton(DM_ITALIC);
               wpadd("This text is now bold and italicized")
               wpattoff(DM_BOLD);
               wpattoff(DM_ITALIC)
               wpclose();
          }
```

Function.................. wpascii(<arg1c>,<arg2c>)
Description.............. Reads the exsiting WordPerfect v5.1 file, and converts all document text into an ascii text file.
Arguments................. <arg1c>=full filespec for existing WordPerfect document
<arg2c>=full filespec for name of ascii text file to be created
Return..... Nothing
Comments..... This function can be called to convert an existing WordPerfect document file into an ascii text file. Two arguments are required. The first argument identifies the full filespec of the existing WordPerfect file. The second argument identifies the name of the file to be created that will contain the ascii text.

The conversion from WordPerfect to ASCII will ONLY translate TEXT found in the WordPerfect document. It will NOT produce formatting codes.

Code Examples:

In this example, the wpascii() function is used to create the text file WPASCII.TXT from the existing WordPerfect file called WP.W51.

Clipper:

```
wpascii("WP.W51", "WPASCII.TXT")
If file("WPASCII.TXT")
     ?"Text file created!"
Endif
```

C:

```
#include "luminet1.h"
wpascii("WP.W51", "WPASCII.TXT");
```

Function..... wpclmoff()
Description..... Ends the last usage of newspaper columns in a document - one that had newspaper columns activated using the wpclmon() function.
Arguments..... none
Return..... Nothing
Comments..... Use this function to complete the use of newspaper columns in your document. All text entered after the wpclmoff() has been issued will assume the characteristics of the document prior to newspaper columns being activated. Newspaper columns separate your document into 2 evenly sized columns. This function is to be issued AFTER the newspaper columns have been turned on using the wpclmon() function.

CODE EXAMPLES:
This code example demonstrates how newspaper columns can be terminated by using the wpclmoff() function. Any text added to the document after the wpclmoff() function has been made, will NOT be in the format of newspaper columns.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen(filespec)
     ?"Successfully created and opened document"
     wpclmon() && activated newspaper columns
     wpadd("This text and all text before a wpclmoff() will be in columns")
     ...
     wpclmoff()     && de-activated newspaper columns
     wpadd("This text will no longer appear in newspaper columns")
     wpclose()
Else
     ?"Unsuccessful"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
     {
          wpclmon();     /* Activate newspaper columns */
          wpadd("This text will now appear in newspaper columns");
          wpclmoff();    /* de-activates newspaper columns */
          wpclose();     }
```

Function..... wpclmon()
Description..... Activates the usage of newspaper columns in a document. Newspaper columns are dual columns of equal sizing.
Arguments..... none
Return..... Nothing
Comments..... Use this function to begin the use of newspaper columns in your document. All text entered after the wpclmon() has been issued will assume the characteristics of the newspaper columns. Newspaper columns separate your document into 2 evenly sized columns. To de-activate the usage of newspaper columns, use the wpclmoff() function.

CODE EXAMPLES:
This code example demonstrates how newspaper columns can be activated by using the wpclmon() function. Any text added to the document after the wpclmon() function will assume the characteristics of the newspaper columns.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen(filespec)
     ?"Successfully created and opened document"
     wpclmon() && activated newspaper columns
     wpadd("This text and all text before a wpclmoff() will be in columns")
     ...
     wpclmoff()     && de-activated newspaper columns
     wpadd("This text will no longer appear in newspaper columns")
     wpclose()
Else
     ?"Unsuccessful"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {
          wpclmon();     /* Activate newspaper columns */
          wpadd("This text will now appear in newspaper columns");
          wpclmoff();    /* de-activates newspaper columns */
          wpclose();
     }
```

Function.................. wpclose()
Description.............. Closes a WordPerfect v5.1 document file, one that was previously created/opened from the wpopen() function
Arguments................. None
Return.................... Nothing
Comments.............. This routine CLOSES the WordPerfect document file that had been previously opened by using the wpopen() function. After issuing a wpclose(), no more text, document, and/or graphic codes can be written to the document file.

Should you issue a wpclose() directly after a wpopen(), you will only create a WordPerfect document file that contains WordPerfect header information (similar to an empty document). If you intend to add any formatting, graphics or text commands. The document file must remain open to insert text and formatting codes.

CODE EXAMPLES:

In the following example, the code will issue a call to wpopen() to create and keep open a WordPerfect document file. When all document and text formatting commands have been completed, the file is then closed with a call to wpclose().

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen(filespec)
     ?"Successfully created and opened document"
     ...
     wpclose()
Else
     ?"Unsuccessful"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
     {
          ...;
          wpclose();
     }
```

Function.................. wpcntr()
Description.............. Centers the next text string specified to the document.
Arguments................. None
Return..................... Nothing
Comments.................. This routine centers the next text on a line of the document. The string is passed using the wpadd() function after the wpcntr() has been issued.

Code Examples:

In this example, the wpcntr() call is used to center the proceeding line of text that may be added to the document. Prior to issuing the wpcntr() call, a wphrtn() is issued to ensure that we are on a new line of the document. After the text is added with the wpadd(), another wphrtn() call is issued to place the next add position on a new line.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          ?"Successfully created and opened document"
          *Adds some centered text to the document, once on a new line*
          wphrtn(1) && position to a new line in the document
          wpcntr()  && Activate centering
          wpadd("Centering this text")
          wphrtn()  && position to a new line in the document
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               /* Adds some centered text to the document */;
               wphrtn(1);
               wpcntr();
               wpadd("Centering this text");
               wphrtn(1);
               ...
               wpclose();
          }
```

```
Function..................    wpdate(<arg1c>)
Description..............    Adds the system date to the document.
Arguments................    <arg1c>= character string representing the combination of arguments chosen
                            for the date format.
Return.....................  Nothing
Comments................. The system date and time is represented by your computer's date and time.
                         This feature is typically used in letters or documents where it is desirable to use
                         "Today's" date and time information.  This function adds the system date
                         (and/or time) to the document based on any of the following date formats:
                    1...Day of the month              6...Day of the week (word)
                    2...Month (represented as a number)     7...Hour (on a 24 hr. clock)
                    3...Month (represented as a word)  8...Hour (on a 12 hr. clock)
                    4...Year (all four digits)              9...Minute
                    5...Year (last two digits)         10.am/pm
               When the symbols '%' or '$' are used before one of the above numerically
               represented formats, will pad the number less than two digits with a leading
               zero or space:
                    "2/1/5"......       appears as 1/1/93
                    "%2/%1/%5"...       appears as 01/01/93
                    "8:90".......       appears as 9:15am
                    "%8:90"......       appears as 09:15am
               When the symbols '%' or '$' are used before a month or day of the week your
               results will be abbreviated:
                    "3 1,4".........appears as September 27, 1991
                    "%6 %3 1,4"....appears as Tue Oct 22, 1991
Code Examples:
In the following examples the wpdate() function is used to add the system date, day of the week and
time information.
Clipper:
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          wpdate("6 3 1,4")   && adds day of week and date
          wpdate("8:90")      && adds the time
          wpclose()
     Endif
C:
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {    wpdate("6 3 1,4");  /* adds day of week */
               wpdate("8:90");          /* adds the time */
               wpclose();     }
```

```
Function.....................  wpfig(<arg1c>,<arg2n>,<arg3n>,<arg4n>,<arg5n>,<arg6n>)
Description.................   Retrieves a figure graphic into the document.
Arguments..................    <arg1c>=the filename of the image to be retrieved into the document
               <arg2n>= Horizontal position (alignment)
                         0 =       left
                         1 =       right
                         2 =       centered
                         3 =  left and right justified
                         4 =       absolute position
               if <arg2n>  = 4  then
                    <arg3n> = x coordinate position (in wordperfect units)
                    <arg4n> = y coordinate position (in wordperfect units)
               otherwise
                    <arg3n> = 0
                    <arg4n> = 0
               <arg5n> =      the width of the graphic -  This value is measured in
                              WordPerfect Units (wpu's)
               <arg6n> = the height of the graphic - This value is measured in
                         WordPerfect Units (wpu's)
Return.........................    Nothing
Comments...................   Figure graphics are retrieved and positioned on the page per the user's
                              specification.  User's can specify specific x-y coordinate placement - or use left,
                              right or centered options -, line length, line thickness and shading.

               Arguments 3,4,5,6 are represented as a WordPerfect Unit (wpu).  To calculate
               the correct wpu,  multiply the desired value in inches by 1200.  For example to
               obtain a 8" width:    8 * 1200 = 9600.

               When using absolute positioning, where argument2 = 4, you will need to identify
               the x and y coordinate values for argument3 and argument4.  Argument 2 is
               calculated as a WordPerfect Unit (wpu).  For all other alignment values for
               argument 1, you will need to pass 0 for argument3 and argument4.

               *FILENAME for FIGURE GRAPHIC:  The filename provided as the first
               argument to this function must exist EITHER in the DIRECTORY that the
               wpfig() function call was issued from OR the default directory for figure graphics
               as defined within WordPerfect.
```

```
 wpfig("fig1.wpg",           4, 400, 800, 2400, 1200 )
      |                      |   |    |     |    |
      |                      |   |    |     |    |
      |                      |   |    |     |    height of graphic
      |                      |   |    |     width of graphic (2")
      |                      |   |    y pos 800 wp units
      |                      |   x pos 400 wp units
      |                  Absolute Position
     Filespec for graphic


 wpfig("fig1.wpg",           0,  0,   0, 2400, 1200 )
      |                      |   |    |     |    |
      |                      |   |    |     |    |
      |                      |   |    |     |    height of graphic
      |                      |   |    |     width of graphic (2")
      |                      |   |    y pos - not used
      |                      |   x pos - not used
      |                  Left Margin Position
     Filespec for graphic
```

Code Examples:

This example demonstrates creating/retrieving a figure graphic that is positioned starting at the left margin of the page, and extends a distance of 2 inches in width and height. Setting argument #2 to 0, identifies that the horizontal line starts on the left margin of the page. Arguments #3 and #4 are not used because we are not utilizing absolute positioning. Therefore, pass 0 as argument #3 and #4. The 2 inches (argument #5) is represented in WordPerfect units as 2400. The desired height of the figure graphic (argument #6) is 2 inches which is also represented as 2400.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          wpfig("FIG1.WPG",0,0,0,2400,2400)  && Adds figure
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {
               /* Adds figure graphic       */;
               wpfig("FIG1.WPG",0,0,0,2400,2400);
               wpclose();     }
```

Function.................. wpfont(<arg1n>)
Description.............. Inserts a font change code into the document.
Arguments................. <arg1n>=the number corresponding to the desired font
Return..................... Nothing
Comments.................. This routine appends the desired font change into the document. The following fonts may be used with the wpfont() function:

0 Line Printer 16.67 cpi
1 Courier 10 cpi
2 Courier Bold 10 cpi
3 Courier Italic 10 cpi
4 Courier 12 cpi
5 Courier Bold 12 cpi
6 Courier Italic 12 cpi
7 Helvetica 18 pt
8 Helvetica 24 pt
9 Roman 10 cpi
10 Roman 12 cpi
11 Century 10 cpi
12 Roman 20 cpi

NOTE: While any of the above fonts may be inserted into the WordPerfect document that you are creating, not all fonts may be available to your printer. Therefore, when opening your LumiNet Tools-1(TM) created document from WordPerfect - the fonts may be altered to the 'closest' available font.
All text that follows this font change will assume the attributes of the new font until another wpfont() call has been issued.

Code Examples:
In the following example, , through the use of the wpfont() function, font changes can be made throughout the document being created.

Clipper:

```
     if wpopen("c:\wp51\docs\mydoc.w51")
          wpfont(0) && courier 10
          wpadd("document created and using Courier 10")
          wpfont(7) && helvetica 18
          wpadd("now - using Helvetica 18")
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("c:\\wp51\\docs\\mydoc.w51"));
          {
          wpfont(0);          /* Courier 10 */
          wpadd("document created and using Courier 10");
          wpfont(7);
          wpadd("now - using Helvetica 18"); /*Helvetica 18 */
          wpclose();
          }
```

Function..... wphfend()
Description..... De-activates the usage of headers or footers in a document. Provides for deactivation of Header A, B and Footer A, B.
Arguments..... none
Return..... Nothing
Comments..... Used in conjunction with the wphfstrt() function, the wphfend() will deactivate any header or footer that was previously opened with the wphfstrt(). Any LumiNet Tools-1(TM) functions called after a wphfend() will appear in the body of the document (rather than the header or footer).

CODE EXAMPLES:
This code example demonstrates how headers and footers can be deactivated by using the wphfend() function. Any text added to the document after the wphfend() function will appear in the body of the document. The header or footer is terminated with the wphfend() function.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen(filespec)
          wphfstrt(0,1)  && activated Header A for all pages in document
          wpadd("This text and all text before a wphfend() will be in Header A")
          ...
          wphfend() && de-activated the use of Header A
          wpadd("This text will no longer appear in Header A")
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               wphfstrt(2,4); /* Activate Footer A on Even Pages */
               wpadd("This text will now appear in Footer A");
               wphfend();     /* de-activates Footer A */
               wpclose();
          }
```

```
Function..................  wphfstrt(<arg1n>, <arg2n>)
Description..............   Activates the usage of headers or footers in a document.  Provides for
                            activation of Header A, B and Footer A, B.
Arguments................    <arg1n> - defines the type of header or footer
                    0 = Header A        2 = Footer A
                    1 = Header B        3 = Footer B
               <arg2n> - defines the pages that the header or footer will appear on
                    1 = All Pages  2 = Odd Pages       3 = Even Pages
Return..................... Nothing
Comments.................   The wphfstrt() function activates the usage of headers or footers in your
                            WordPerfect document.  All text and graphic commands that are issued after a
                            wphfstrt() and before a wphfend() will appear in the header or footer as
                            selected.  The wphfstrt()  allows you to identify the type of header (A,B) or the
                            type of footer (A,B) that will be activated.  The next argument represents the
                            pages that the header or footer will appear on.
```

CODE EXAMPLES:
This code example demonstrates how headers and footers can be activated by using the wphfstrt() function. Any text added to the document after the wphfstrt() function will appear in the header or footer as selected. The header or footer is terminated with the wphfend() function.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen(filespec)
          wphfstrt(0,1)  && activated Header A for all pages in document
          wpadd("This text and all text before a wphfend() will be in Header A")
          ...
          wphfend() && de-activated the use of Header A
          wpadd("This text will no longer appear in Header A")
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {
               wphfstrt(2,4); /* Activate Footer A on Even Pages */
               wpadd("This text will now appear in Footer A");
               wphfend();     /* de-activates Footer A */
               wpclose();     }
```

Function..... wphline(<arg1n>,<arg2n>,<arg3n>,<arg4n>,<arg5n>,<arg6n>)
Description..... Creates a horizontal line on the document.
Arguments..... <arg1n>= Horizontal position.
0 = left
1 = right
2 = centered
3 = left and right justified (not used)
4 = absolute position
if <arg1n> = 4 then
<arg2n> = x coordinate position
<arg3n> = y coordinate position
<arg4n> = the horizontal length of the line - This value is measured in WordPerfect Units (wpu's)
<arg5n> = the height of the line (or line thickness) - This value is calculated by multiplying the desired thickness in inches by 100
<arg6n> = shading - an integer value from 1 to 100 which represents the percent of darkness or shade of the line.
Return..... Nothing
Comments..... Graphically created lines are positioned on the page per the user's specification. User's can specify specific x-y coordinate placement - or use left, right or centered options -, line length, line thickness and shading.

Argument 4 is represented as a WordPerfect Unit (wpu). To calculate the correct wpu, multiply the desired line-width (as inches) by 1200. For example to obtain a 8" line width: 8 * 1200 = 9600.

The line height (argument 5) represents the desired thickness of the horizontal line. In this case, to calculate the correct line height, multiply the desired line height in inches by 100. A 1/2" line height can be obtained by multiplying: .5" * 100 = 50. 50 is the value to be entered as argument 5.

When using absolute positioning, where argument1 = 4, you will need to identify the x and y coordinate values for argument2 and argument3. Argument 2 is calculated as a WordPerfect Unit (wpu

```
    wphline(4, 400, 800, 8000, 60, 100 )
          |   |    |     |    |   |
          |   |    |     |    |   Shading
          |   |    |     |    line-thickness
          |   |    |     horizontal length of line
          |   |    y pos 800 wp units
          |   x pos 400 wp units
          Absolute Position

    wphline(0,  0,   0,  8000, 60, 100 )
          |   |    |     |    |   |
          |   |    |     |    |   Shading
          |   |    |     |    line-height (thickness of the line)
          |   |    |     length of line horizontally
          |   |    y pos not used - must pass 0
          |   x pos not used - must pass 0
          left margin
```

Code Examples:

This example demonstrates creating a horizontal line that is positioned starting at the left margin of the page, and extends a distance of 8 inches. Setting argument #1 to 0, identifies that the horizontal line starts on the left margin of the page. Arguments #2 and #3 are not used because we are not utilizing absolute positioning. Therefore, pass 0 as argument #2 and #3. The 8 inches (argument #4) is represented in WordPerfect units as 9600. The desired thickness of the horizontal line (argument #5) is 6/10's of an inch, which is represented as 6/10*100 or 60. The shading of the line (argument #6) is 100% - or completely filled in.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          wphline(0,0,0,9600,60,100)    && Adds horizontal line
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               /* Adds horizontal line      */;
               wphline(0,0,0,9600,60,100);
               wpclose();     }
```

Function..... wphpage()
Description..... Appends a single WordPerfect Hard Page break code to the document.
Arguments..... None
Return..... Nothing

Comments..... This routine appends a single WordPerfect compatible hard page break code into the document file. This forces new pages to be created in the document file.

The "typical" usage of this function is to append a single hard page return, forcing all additional text to be added to the new page. Should your document span multiple pages, you should utilize the wphpage() function to force page breaks at SPECIFIC points in your document.

Code Example:

In this example, the wphpage() function passes a value of 1, forcing a hard page return. Any text added to the document (via the wpadd() function) will now appear on the new page.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     ?"Successfully created and opened document"
     *Adds 1 hard page break  - forcing a new page into the document*
     wpadd("This text is on the current page")
     wphpage()
     wpadd("This text appears on the new page")
     wpclose()
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {
          /* Adds 1 hard page break to the document */;
          wpadd("This text is on the current page");
          wphpage();
          wpadd("This text appears on the new page");
          ...
          wpclose();     }
```

Function.................. wphrtn(<arg1n>)
Description.............. Appends a WordPerfect Hard LINE Return into the document, based on the number of times that is specified by <arg1n>.
Arguments................. <arg1n>=the number of times to insert a hard line return
Return..................... Nothing
Comments................. This routine appends a hard line return code into the document that was created and opened using the wpopen() function. By issuing a wphrtn() call, any text that is appended using the wpadd() function will appear on the new line.

Code Examples:

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     ?"Successfully created and opened document"
     *-Add 2 hard returns to the document-*
     wphrtn(2)
     wpadd("This text starts on a new line")
     ...
     wpclose()
Else
     ?"Unsuccessful in creating a WPC file"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {
          /* Add 2 hard returns to the document */;
          wphrtn(2);
          wpadd("This text starts on a new line");
          ...
          wpclose();
     }
```

Function.................. wpindent(<arg1n>)
Description.............. Appends the WordPerfect Indent code to the document.
Arguments................. <arg1n>=the number indents to be inserted
Return..................... Nothing
Comments.................. This routine appends 'n' number of indent codes as specified in the argument.
All text that follows this attribute will be indented until a Hard Return - wphrtn()
has been sent.

Code Examples:

In the following example, two indents were added to the document through the use of wpindent(2). Any
text added after the wpindent() will appear indented in the document. Once the wphrtn() call is issued,
the text will no longer appear indented.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          ?"Successfully created and opened document"
          *-- Adds 2 indents to the document --*
          wpindent(2)
          wpadd("This text is indented")
          wphrtn(1)
          wpadd("This text is no longer indented")
          ...
          wpclose()
     Else
          ?"Unsuccessful"
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {
               /* Adds 2 indents to the document */;
               wpindent(2);
               wpadd("This text is indented");
               wphrtn(1);
               wpadd("This text is no longer indented");
               ...
               wpclose();     }
```

Function..... wpjust(<arg1n>)
Description..... Defines the document justification as right, left, center, or full.
Arguments..... <arg1n>=the number representing the desired justification type (see below)
Return..... Nothing
Comments..... Use this function to create justification settings for your document. The justification setting is used to determine the overall placement of text on the document. Any text that is added to the document after the wpjust() call will take on the attributes of the justification setting. Use one of the four argument values defined below to identify the desired justification.

0 = left -any text added to the document will be positioned along the left margin
1 = full -both the left and right margins will be used so that any text on a line will not appear with 'jagged' edges.
2 = center -any text added to the document will be centered on each line
3 = right -the text will be positioned along the right margin of the page

CODE EXAMPLES:
This code example demonstrates how justification settings can be used to place the text to the left side of the page. After the file has been successfully created, the wpjust() passes a '0' - indicating that justification will be to the left.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen(filespec)
          wpjust(0) && Select left justification
          wpadd("This text is now justified to the left side of the page")
          wpclose()
     Else
          ?"Unsuccessful"
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {    wpjust(0);     /*Select left justification */
               wpadd("This text is now justified to the left side of the page");
               wpclose();     }
```

Function..... wplrmarg(<arg1n>,<arg2n>)
Description..... Creates a left and right margin for the document-defined by arguments 1 and 2.
Arguments..... <arg1n>=the number of wordperfect units from the left
<arg2n>=the number of wordperfect units from the right
Return..... Nothing
Comments..... Use this call to set a new left and right margin. The first value is 'n' number of wpu's (WordPerfect Units) from the left side of the document, and the second value is 'n' number of wpu's from the right side of the document.

The wplrmarg(#, #) function requires that you enter the values as WordPerfect Units. There are 1200 WordPerfect Units per inch. For example, to obtain a ONE INCH value, you would require 1200 WordPerfect Units.

1 INCH = 1200/1200 WPU's
The following example shows how to obtain a 1 inch left margin, and a 1/2 inch right margin: wplrmarg(1200, 600)
1200/1200 = 1" from left
1200/600 = 0.5" from right

CODE EXAMPLES:
In this example, the wplrmarg() command is used to create a 2 inch left margin, and a 1 inch right margin at the beginning of the document. Note: If the wplrmarg() command is issued within the middle of a page, the margin settings will take effect on the next page of the document.
Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          wplrmarg(2400,1200) &&  Adds 2" left margin, 1" right margin
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {    wplrmarg(2400,1200);     /* 2" left margin, 1" right margin */
               ...
               wpclose();     }
```

Function..... wpopen(<arg1c>)
Description..... Creates a WordPerfect v5.1 document file, and then Opens the file that was created to allow writing to the file
Arguments..... <arg1c>=file specification for created file
Return..... logical
<TRUE>=successful
<FALSE>=unsuccessful
Comments..... This routine CREATES the WordPerfect 5.1 compatible document file - that is, a new document file will exist after this command has successfully completed. Once the document file is created, it remains open during the session until a wpclose() call is made. By leaving the file open, you have access to all document formatting, graphics and text commands. These commands will be placed into the document that is created and opened by using the wpopen() call. Once the wpclose() command has been issued - you may no longer make additions to the document file. You may issue a wpclose() command directly after a wpopen() when you intend to create an empty WordPerfect document file.

CODE EXAMPLES:

In the following example, the code will issue a call to wpopen() using the default printer. The function wpopen() will return a (true) value if the file is successfully created and opened. A (false) value is returned if the creation of the file was unsuccessful.

Clipper:

```
     If wpopen ("C:\WP51\DOCS\DOC00001.w51")
          ?"Successfully created and opened document"
     Else
          ?"Unsuccessful"
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("c:\\wp51\\docs\\mydoc.w51"))
          {
          wpadd("document created");
          wpclose();
          }
```

Function..... wporient(<arg1n>)
Description..... Sets Portrait or Landscape page orientation
Arguments..... <arg1>=0-Portrait, 1-Landscape
Return..... Nothing
Comments..... Use this function to establish the page orientation. Page orientation can be set to one of two modes: Portrait and Landscape.

Portrait orientation will assume 8 1/2" page width and 11" page length.
Landscape orientation will assume 11" page width and 8 1/2" page length.
Without using the wporient() function, your page orientation will assume the WordPerfect setup defaults for your document.

NOTE: This function can only provide SUCCESSFUL PAGE ORIENTATION CHANGES when the selected orientation is available with the associated WordPerfect printer being used. Landscape Orientation will only be successful with Printers that CAN print in landscape orientation modes.

CODE EXAMPLES:
By passing a 1 value to wporient(), we can initially set up the page orientation to landscape. Then, sending a 0 value, we can change the orientation to portrait.

Clipper:

```
     If wpopen("c:\\wp51\\docs\\mydoc.w51")
          wporient(1)    && landscape
          wpadd("document created and using landscape")
          wphpage()
          wporient(0)    && portrait
          wpclose()
     Endif
```

C:

```
     If wpopen("c:\\wp51\\docs\\mydoc.w51")
          {
          wporient(1);   /* landscape */
          wpadd("document created and using landscape");
          wphpage();
          wporient(0);   /* portrait */
          wpclose();
     }
```

Function..... wppagen(<arg1n>)
Description..... Creates page numbering on your WordPerfect document file.
Arguments..... <arg1n>=the page positioning value (see below)
Return..... Nothing
Comments..... Use this function when the document you are creating has multiple pages and require page numbers. This function allows you to identify the location on the page for the page number. Any one of the following values should be passed depending on which position of the page the number should appear:

0 = none	5 = bottom left
1 = top left	6 = bottom center
2 = top center	7 = bottom right
3 = top right	8 = alternating bottom left & right
4 = alternating top left and right	9 = insert page code

CODE EXAMPLES:
By passing a parameter of a value from 0 through 8, the type of page numbering can be selected for the WordPerfect document. In this example, bottom of page and centered was selected by passing parameter 6 with the wppagen() call.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen(filespec)
          wppagen(6)     && Select bottom centering for page numbers
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {
               /*Select bottom centering for page numbers*/
               wppagen(6);
               ...
               wpclose();
          }
```

Function..... wpsettab(<arg1n>, <arg2n>,<arg3n>,<arg4n>,<arg5n>,<arg6n>,
<arg7n>,<arg8n>,<arg9n>,<arg10n>)
Description..... Sets tab positions on your WordPerfect document file.
Arguments..... <arg1-10n>=the tab positioning value in WordPerfect Units
Return..... Nothing
Comments..... Use this function to establish the tab positioning. The tab positioning will affect the placement of your text after issuing the wptab() function. Without using the wpsettab() function, your tabs will assume the default value for tab sets.

Up to 10 tab positions may be set with this function. However, if you require less tab positions than the full 10, you must pass an 0 value for each tab position that you are not setting up. All arguments are in WordPerfect Units. To obtain the wordperfect unit value, multiply the desired value in inches by 1200. For example a 1 inch tab set will be 1200 WordPerfect Units. The position of a tab setting is relative to the left margin of the document.

*NOTE: The format for using the wpsettab() function from a C program is DIFFERENT than the format for Clipper programs. Please review the C program example:

CODE EXAMPLES:
By passing 10 parameters to the wpsettab() function, we can reset the default tab setting for the document. In this example, only the first 6 tab settings are utilized.
Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen(filespec)
          wpsettab(1200, 2400, 3600, 4800, 6000, 7200, 0,0,0,0) && tab settings
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     /* void   wpsettab( int *tab ); */
     int tab[10] = {600, 600*2, 600*3, 600*4, 600*5, 600*6, 600*7,00,00,00};
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))
          {
               /*Select tab settings for 7 positions in the document*/
               wpsettab(&tab[0]);
               ...
               wpclose();   }
```

Function..... wptab(<arg1n>)
Description..... Appends the WPC Tab code to the document.
Arguments..... <arg1n>=the number tabs to be inserted
Return..... Nothing
Comments..... This routine appends 'n' number of tab codes as specified in the argument. All text that follows this attribute will be positioned after the tab.

Code Examples:

In the following example, a single tab was created by the use of the wptab() function. The text that appears immediately following the tab, is 'tabbed over' one tab position.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     ?"Successfully created and opened document"
     *-- Adds 1 tab to the document --*
     wptab(1)
     wpadd("This text appears after the tab")
     ...
     wpclose()
Else
     ?"Unsuccessful"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {
          /* Adds 1 tab to the document */;
          wptab(1);
          wpadd("This text appears after the tab");
          ...
          wpclose();
     }
```

Function..... wptbmarg(<arg1n>,<arg2n>)
Description..... Creates a top and bottom margin for the document as defined by arguments 1 and 2.
Arguments..... <arg1n>=the number of wordperfect units from the top of the page
<arg2n>=the number of wordperfect units from the bottom of the page
Return..... Nothing
Comments..... Use this call to set a top and bottom margin for the WordPerfect document file. The values passed represent WordPerfect Units (wpu). The first value passed is 'n' number of wpu's from the top of the document, and the second value is 'n' number of wpu's from the bottom. If the wptbmarg() function is not placed at the top of the page, the margin settings will become active on the next page of the document. NOTE: Keep in mind that the wptbmarg(#, #) function requires that you enter the parameter values as WordPerfect Units. WordPerfect Units are 1200 per inch. wptbmarg(1200, 600)
1200/1200 = 1" from top
1200/600 = 0.5" from bottom

Code Example:
In the following example, a 2 inch top margin is created by passing a WordPerfect Unit of 2400. A 1 inch bottom margin is created using the WordPerfect Unit of 1200. The wptbmarg() function call is issued directly after the WordPerfect document has been created and opened. In this case, the top and bottom margin settings will be effective for this current page.

Clipper:

```
     filespec = "C:\WP51\DOCS\DOC00001.W51"
     If wpopen (filespec)
          wptbmarg(2400,1200) && Adds 2" top margin, 1" bottom margin
          ...
          wpclose()
     Endif
```

C:

```
     #include "luminet1.h"
     if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

          {    wptbmarg(2400,1200);          /* 2" top margin, 1" bottom margin */
               ...
               wpclose();     }
```

```
Function..................    wpvline(<arg1n>,<arg2n>,<arg3n>,<arg4n>,<arg5n>,<arg6n>)
Description...............    Creates a vertical line on the document.
Arguments.................    <arg1n>= Vertical position.
                    0 =       left
                    1 =       right
                    2 =       centered
                    3 =       absolute position
               if <arg1n> = 3  then
                    <arg2n> = x coordinate position
                    <arg3n> = y coordinate position
               <arg4n> =      the vertical length of the line -  This value is measured in
                              WordPerfect Units (wpu's)
               <arg5n> = the line thickness -  This value is calculated by multiplying the
                         desired length in inches by 100
               <arg6n> = shading - an integer value from 1 to 100 which represents the
                         percent of darkness or shade of the line.
Return.......................    Nothing
Comments..................   Graphically created lines are positioned on the page per the user's specification.
                             User's can specify specific x-y coordinate placement - or use left, right or
                             centered options -, line length, line thickness and shading.

               Argument 4 is represented as a WordPerfect Unit (wpu).  To calculate the
               correct wpu,  multiply the desired line-width (as inches) by 1200.  For example
               to obtain a 8" line width:    8 * 1200 = 9600.

                   wpvline(3, 400, 800, 8000, 60, 100 )
                            |   |    |     |    |   |
                            |   |    |     |    |   Shading
                            |   |    |     |    line thickness
                            |   |    |     line height (vertical length)
                            |   |    y pos 800 wp units
                            |   x pos 400 wp units
                           Absolute Position

                    wphline(0,  0,   0,  8000, 60, 100 )
                            |   |    |     |    |   |
                            |   |    |     |    |   Shading
                            |   |    |     |    line thickness
                            |   |    |     line height (vertical length)
                            |   |    y pos not used - must pass 0
                            |   x pos not used - must pass 0
                           left margin
```

Code Examples:

This example demonstrates creating a vertical line that is positioned starting at the right margin of the page, and extends a distance of 10 inches. Setting argument #1 to 1, identifies that the vertical line starts on the right margin of the page. Arguments #2 and #3 are not used because we are not utilizing absolute positioning. Therefore, pass 0 as argument #2 and #3. The 8 inches (argument #4) is represented in WordPerfect units as 12000. The desired thickness of the vertical line (argument #5) is .15 of an inch, which is represented as .15*100 or 15. The shading of the line (argument #6) is 100% - or completely filled in.

Clipper:

```
filespec = "C:\WP51\DOCS\DOC00001.W51"
If wpopen (filespec)
     ?"Successfully created/opened document"
     *-- Adds vertical line         --*
     wpvline(1,0,0,12000,15,100)
     ...
     wpclose()
Else
     ?"Unsuccessful"
Endif
```

C:

```
#include "luminet1.h"
if (wpopen("C:\\wp51\\docs\\doc00001.w51"))

     {
          /* Adds vertical line        */;
          wpvline(1,0,0,12000,15,100);
          wpclose();
     }
```

Appendix A - Attribute Values

Attributes are used in WordPerfect to adjust the size, positioning and/or intensity of text. The values for attributes used in LumiNet Tools-1(TM) as follows:

ExtraLarge.....	0
VeryLarge.....	1
Large.....	2
Small.....	3
Fine.....	4
SuperScript...	5
SubScript.....	6
Outline.....	7
Italic.....	8
Shadow.....	9
RedLine.....	10
Double Und....	11
Bold.....	12
StrikeOut.....	13
UnderLine.....	14
Small Caps.....	15

Appendix B - Linking

FILES WE SUPPLY FOR LINKING

```
CLP2WPC.LIB    REQUIRED FOR ALL CLIPPER LINKING

BCC2WPCS.OBJ   BORLAND SMALL MODEL
BCC2WPCM.OBJ   BORLAND MEDIUM MODEL
BCC2WPCL.OBJ   BORLAND LARGE MODEL

MSC2WPCS.OBJ   MICROSOFT SMALL MODEL
MSC2WPCM.OBJ   MICROSOFT MEDIUM MODEL
MSC2WPCL.OBJ   MICROSOFT LARGE MODEL
```

CLIPPER 87 BLINKER EXAMPLE

```
# wpclp000.LNK
NOBELL
BLINKER EXECUTABLE CLIPPER E0;F10;R25
BLINKER INCREMENTAL OFF
BLINKER MEMORY PACK 10
OUTPUT wpclp000
FILE wpclp000
LIBRARY CLP2WPC
library \CLIPPER\EXTEND
search \clipper\clipper
```

CLIPPER 87 MICROSOFT LINKER EXAMPLE

```
LINK MYFILE+CLP2WPC+CLPSUP,MYFILE,,EXTEND+CLIPPER;
```

MICROSOFT "C" v6.0 LINK EXAMPLE

```
LINK @MYFILE.LNK

This is myfile.lnk

myfile+        <- your obj file
msc2wpcs  <- our interface routines
myfile         <- your exe name
,
slibce;        <- microsoft small model library
```

BORLAND C++ v2.0

Refer to your Borland manual on the Integrated Development Environment as that would be the perfered method. If you need to link from the command line, the following example is for large model.

```
tlink c:\bC\LIB c0l MYFILE bcc2wpcl, MYFILE, nul,emu mathl cl
```

Appendix C - Program Examples

The following example is for a Microsoft C or Borland C program.

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <process.h>
#include <conio.h>
#include <fcntl.h>
#include <errno.h>
#include <bios.h>
#include <direct.h>
#include <string.h>
#include <luminet1.h>
#define   BOLD 12

main(int argc, char *argv[])
{
char junk[1024];
int     a;
if(argc >1)
     wpopen(argv[1]);  /* open the document per command line argument */
     else
     {
     printf("\nusage msctest[size option l,m,s] <filename>");
     exit(1);
     }
```

```
strcpy(junk,"this is added from msctest"); /*create variable of text*/
wpadd( junk );      /*add the text to the document*/
wphrtn( 3 );        /*add 3 hard returns*/
strcpy(junk,"this is added after the hard returns");
wpadd( junk );
wphrtn( 3 );        /*add 3 hard returns*/
wpcntr();           /*set centering*/
strcpy(junk,"this is added and centered");/*centering this text*/
wpadd( junk );      /*add the text to the document*/
wplrmarg( 2400,1200);    /*set left margin at 2 inches set right 1 inches*/
wptbmarg( 2400,1000);    /*set left margin at 2 inches set right 1 inches*/
wppagen(6);              /*set page numbering for bottom center*/
wphrtn(3);               /*add 3 hard returns*/
wpvline(3, 100, 100, 10, 6000, 100 );   /*set absolute position */
                                 /*at x position 100 wpu 0.08 inches*/
                                 /*at y position 100 wpu 0.08 inches*/
                                 /*horizontal width 10 wpu */
                                 /*vertical length 6000 wpu 5 inches*/
                                 /*100% shading*/

wphrtn(3);                   /*add 3 hard returns*/

wphline(3, 400, 800, 8000, 60, 100 );   /*set absolute position */
                                 /*at x position 400 wpu 0.33 inches*/
                                 /*at y position 800 wpu 0.66 inches*/
                            /*horizontal length 8000 wpu 6.66 inches*/
                            /*vertical length 60 wpu 0.05 inches*/
                            /*100% shading*/

wphrtn(3);                   /*add 3 hard returns*/

wpdate("$2/$1/5 ($6)");      /*set date as nn/nn/nn (<day>)*/
wpatton(BOLD);          /*set BOLD attribute on */

wpdate("3 1, 4");
/*set date as month <word> day <nmb>, year <century/year ccyy> */

wpattoff(BOLD);              /*set BOLD attribute off */

wphrtn(1);                   /*add 1 hard return*/

wpclose();                   /*close the document*/
exit(0);
}
```

The following example is for Clipper '87.

```
* test for clipper '87
* accepts filename from the command line
parameter thefile
XL =     0
VL =     1
LRG =    2
SML =    3
FINE =   4
SPRS =   5
SUBS =   6
OLINE =  7
ITLC =   8
SHDW =   9
RDLN =  10
DUNDRL =11
BOLD =  12
STRKO = 13
UNDRL = 14
SCAPS = 15
LINEC=1
hdr=0
hdrsize =0
use sample     *create a dbf file called sample dbf
          *                   fieldname type
          *it should contain       FNAME     C
          *                   LNAME     C
          *                   ADDR1     C
          *                   ADDR2     C
          *                   CITY      C
          *                   STATE           C
          *                   ZIP       C

@2,3 say  "Creating Word Perfect version 5.1 document."
@3,3 say  "Document filename -> "+thefile
if wpopen(thefile)
     wphrtn(1)                *add 1 hard return
     wphpage()                *add hard page
     wpatton(BOLD)            *set BOLD attribute on
     DO WHILE .NOT. EOF()
          wpadd(FNAME)             *add text from FNAME field
          wptab(1)                 *add 1 tabs
          wpadd(LNAME)             *add text from FNAME field
```



```
          wptab(1)            *add 1 tabs
          wphrtn(1)           *add 1 hard return
          wpadd(ADDR1)        *add text from ADDR1 field
          wptab(1)            *add 1 tabs
          wpadd(ADDR2)        *add text from ADDR2 field
          wphrtn(1)           *add 1 hard return
          wpadd(CITY)         *add text from CITY field
          wptab(1)            *add 1 tabs
          wpadd(STATE)        *add text from STATE field
          wphrtn(1)           *add 1 hard return
          wpadd(ZIP)               *add text from ZIP field
          wphrtn(2)           *add 2 hard return
          skip
          LINEC = LINEC + 5
          IF LINEC > 50
               LINEC = 1
               wphpage()      *add 1 hard page
          endif
     ENDDO
     wpattoff(BOLD)      *set BOLD attribute off
     wpclose()           *close the document
```

luminet1.h - - for Borland C/C++ compilers

```
/*************************************************************************/
/*******    Function prototypes, DEFINES, and required structures   *********/
/*************************************************************************/
#define   HDFT_ALL_PAGES      (char)1
#define   HDFT_ODD_PAGES      (char)2
#define   HDFT_EVEN_PAGES          (char)4
#define   HEADER_A       (char)0
#define   HEADER_B       (char)1
#define   FOOTER_A       (char)2
#define   FOOTER_B       (char)3

#define   LINE_PRINTER        (int)0
#define   COURIER_10          (int)1
#define   COURIER_10_BOLD          (int)2
#define   COURIER_10_ITALIC   (int)3
#define   COURIER_12               (int)4
#define   COURIER_12_BOLD          (int)5
#define   COURIER_12_ITALIC   (int)6
#define   HELVETICA_18PT      (int)7
#define   HELVETICA_24PT      (int)8
#define   ROMAN_10       (int)9
#define   ROMAN_12       (int)10
#define   CENTURY_10          (int)11
#define   ROMAN_20            (int)12
#define   MAX_WP_PRINTERS          20
#define   MAX_WP_FONTS        50
#define  DM_EXTRALARGE        0
#define  DM_VERYLARGE         1
#define  DM_LARGE             2
#define  DM_SMALL             3
#define  DM_FINE              4
#define  DM_SUPERSCRIPT       5
#define  DM_SUBSCRIPT         6
#define  DM_OUTLINE           7
#define  DM_ITALIC            8
#define  DM_SHADOW            9
#define  DM_REDLINE           10
#define  DM_DOUBLEUND         11
#define  DM_BOLD              12
#define  DM_STRIKEOUT         13
#define  DM_UNDERLINE         14
#define  DM_SMALLCAPS         15
```

```
struct    printer
          {
          int  status;   // 0xffff not used 0x00 selected 0x01 default printer
          unsigned char  LongName[37];
          unsigned char  PRSName[13];
          unsigned char  Descriptor[27];
          unsigned int   MinTM;
          unsigned int   MinBM;
          unsigned int   MinLM;
          unsigned int   MinRM;
          unsigned char  flags;
          unsigned int   PRSDate;
          unsigned int   PRSTime;
          char fpath[80];
          };

struct    fonts
          {
          int  status;   // 0xffff not used 0x00 selected 0x01 default printer
          char name[40];
          };


int       wpopen(char *ptr );                    // open document using
int       wpadd( char *ptr );           // add text to document
void      wpclose( void );              // close document
void      wphrtn( int count);           // add n hard returns
void      wpindent( int count );             // add n indents
void      wpcntr(void);                 // center line
void      wptab(int count);             // add n tabs
int       wpatton( unsigned char  att_type );     // set attribute on
int       wpattoff( unsigned char  att_type );    // set attribute off
void      wphpage( void );              // insert hard page break
void      wpdate( char *ptr );               // insert system date
int       wppagen( unsigned char  np );      // insert page number
void      wphline(int hpf, int xpos, int ypos, int line_width,
                      int line_height, int shading);  //draw horizontal line
void      wpvline(int hpf, int xpos, int ypos, int line_width,
                      int line_height, int shading);  //draw vertical line
void      wplrmarg( int newlmarg, int newrmarg ); // left and right margins
void      wptbmarg( int newtmarg, int newbmarg ); // top and bottom margins

// header and footer start
int  wphfstrt( char hf_type, char oe_page ); // header and footer start/end
int       wphfend(void);

// insert graphic
void   wpfig(char *fname, int alignment, int x_pos,
                   int y_pos, int width, int height);
// justification
void      wpjust( unsigned char just);

// columns on
void      wpclmon( void );

// columns off
void      wpclmoff( void );

void wpsettab( int *tab );

// change font
// int    wpfont( int font_type, int point_size );
int  wpfont( int f_nmb );


// set orientation
void wporient( int orient );

//   convert document s_ptr to     ascii text file d_ptr
int  wpascii( char *s_ptr, char *d_ptr );
```