Macro Index By Statement Type

WinComm Command

Statements used to control WinComms operation

WinComm DLL

Statement to access a WinComm Dynamically Linked Library

WinComm File Control

Statements to control session files

WinComm File Transfer

Statements to send and receive files not using a protocol

WinComm Information

Statements to obtain information about the status of WinComm

WinComm Packet Data Transfer

Statements used to transmit and receive error free packets

WinComm Protocol File Transfer

Statements to send and receive files using a protocol

WinComm/WinLink File Control

Statement to send and receive files and provide remote control of the disks on a computer running WinLink

Windows Application Control

Statements to control and obtain information from other Windows applications

Windows Information/File Control

Statements to obtain information about the Windows environment

Clipboard Control

CLIPTOFILE(clipboard_type,filename)

Copies the Clipboard Text, Bitmap or Metafile to a disk file.

FILETOCLIP(clipbrd_type,file_name)

Copies a Windows metafile, bitmap or text file to the Clipboard.

PASTETEXT

Transmits and displays the text in the clipboard.

Conversion

INTTIME(hour,day,month,year)
: Converts times and dates to the WinComm integer time/day value.

STRBIN(number)
: Changes an integer number into a binary string representation of the number.

STRHEX(number)
: Changes an integer number into a Hexedecimal string representation of the number.

STRINT(number)
: Changes an integer number into a base 10 string representation of the number.

STROCT(number)
: Changes an integer number into an Octal string representation of the number.

STRTIME(time)
: Converts an integer time date value to a text string.

VALBIN(bin_text)
: Converts a string of 1's and 0's in binary form to an integer value.

VALHEX(hex_text)
: Converts a string representing a hex decimal number to an integer value.

VALINT(dec_text)
: Converts a string representing a decimal number to an integer value.

VALOCT(oct_text)
: Converts a string representing an octal number to an integer value.

Customizing

ALARM
: Beeps the speaker.

Dialog Box Customizing

DIALOGBOX(left,top,width,height) and DEND
: Used with the following items to create custom dialog boxes.

BEGINGROUP_RADIOBUTTON
: The first radio button in a radiobutton group.

CANCELBUTTON
: Defines the button that will be selected when the Esc key is pressed.

CHECKBOX
: Displays a standard check box.

COMBOBOX
: Displays a drop down list box for selection of an item from a list.

DEFCANCELBUTTON
: Defines a Cancel button that will be selected when the Enter key is pressed.

DEFPUSHBUTTON
: Defines the button that will be selected when the Enter key is pressed.

DIRLISTBOX
: Displays a list for selection of files, directories and drives.

DIRPATH
: Displays the current path when used with the DIRLISTBOX.

EDITBOX
: Used to create an edit box for text entry.

ENDGROUP_RADIOBUTTON
: The last radiobutton in a radiobutton group.

FLISTBOX
: Displays a list box for selection of an item from a large list.

GROUPBOX
: Draws a black frame in a dialog box.

HELPID integer_id
: Specifies the help topic to use with this dialog box.

LISTBOX
: Displays a list box for selection of an item from a list.

LTEXT, CTEXT, RTEXT
: Used to display static text in a dialog box.

PUSHBUTTON
: Creates additional pushbuttons.

RADIOBUTTON
: To create additional buttons in a radiobutton group.

SEDIT
: Creates a file linked edit box with a vertical scroll bar for text entry.

STEXT
: Creates a file linked text box with a vertical scroll bar.

UPDATEPB
: Updates variables in a dialog box.

VTEXT

Used to display a string variable in a dialog box.

Related Statements

DESTROYDLG

Used to remove or destroy a Modeless Dialog Box

DIALOG?

Indicates the status of a dialog box.

ENABLECTL(ctrl)

Enables and disables Push Button and List Box styles

UPDATEDLG(update_items)

Defines which variables are to be updated in a dialog box.

DOS File Control

ADD(text)
: Adds text to the capture file.

CHGDIR(directory_name)
: Changes the current directory.

CHGDRIVE(drive_letter)
: Changes the current drive.

COPYFILE(source_file,dest_file)
: Copies a disk file.

DELETEFILE(file_name)
: Deletes a disk file.

FCLOSE(file_no)
: Closes a disk file.

FOPEN(open_mode,file_name)
: Opens a disk file for information or for read/write operation.

FREAD(open_file,num_chars)
: Returns a specified number of characters from a file.

FREADLN(open_file)
: Returns a line of text from an open file.

FSEEK(open_file,offset,seek_type)
: Positions a pointer in an open file for reading or writing.

FWRITE(open_file,chars,text)
: Writes a string of text to an open file.

FWRITELN(open_file,text)
: Writes a line of text to an open file.

MKDIR(dir_name)
: Used to create a new disk directory.

RENAME(old_name,new_name)
: Renames a file.

RMDIR(dir_name)
: Removes a directory.

SETFILEATTR(attribute,file_name)
: Sets a file attribute.

SETFILEDATE(date_time,file_name)
: Sets a files date and time.

DOS File Information

CURDIR?
: Returns the name of the current directory.

CURDRIVE?
: Returns the name of the current disk drive.

FILEATTRIBUTE(file_name)
: Used to obtain the attributes of a file.

FILEFIND(file_name,attributes)
: Finds the first file meeting name and attribute requirements.

FILESIZE(open_file)
: Returns the size of a file.

FILETIME(open_file)
: Returns the creation or last modification date of a file.

FINDNEXT
: Finds the next file that meets the criteria set up by FILEFIND.

DOS Information

DOSVER?

Returns the version of DOS.

ENVIRON

Returns information about the current DOS environment.

Dynamic Data Exchange

DDEADVISE(chan_no,my_var$,server_var)

Requests the server application to send the value of a variable to WinComm each time it changes.

DDEEXEC(chan_no,"execute_text")

Executes commands in the server application.

DDEINIT(app_name,doc_name)

Requests the server application to establish a DDE channel for data exchanges or control.

DDEPOKE(chan_no,server_var,my_var$)

Sends data to the server application.

DDEREQ(chan_no,my_var,server_var)

Obtains the current value of a variable from the server application.

DDETERM(chan_no)

Terminates the DDE link or channel established with the server using DDEINIT.

DDEUNADVISE(chan_no,server_var)

Terminates an advise which was established by using DDEADVISE.

Graphics Customizing

BITBKG(h_pos,v_pos,from,file_name)

Displays a bitmap background in the user window.

DELOBJECT(type,start_id,end_id)

Used to remove different items of a graphic display.

METABKG(h_pos,v_pos,mapping,file_name)

Displays a metafile background in the user window.

USERWINDOW(pos,size_ref,size,bkg_color)

Used to define an area of the WinComm window for display of graphics.

Graphics Menus

BITMAP(left,top,id,text,file_name)

Places a bitmap icon on the background graphic.

BUTTON(left,top,width,height,"id,text")

Places a Windows push button on the background graphic.

HOTSPOT(left,top,right,bot,id)

Places a mouse selectable spot on the background graphic.

METAFILE(left,top,right,bot,file_name)METAFILEPlaces a metafile on the graphic.

OBJECT?

Returns information about selection of objects on a graphic display screen.

Macro Control

CHAIN(macro_name)

Transfers macro execution to another macro.

CHAINRETURN

Transfers macro execution back to the calling macro.

END

Indicates the logical end of a macro.

GOSUB label

Transfers execution of the macro to a subroutine.

GOTO label

Transfers execution of the macro to a label.

HALT

Stops the in-line execution of the macro.

MACROHALT(greyed)

Used to disable the Macro check box on the WinComm command bar.

MACROTRAP(enabled)

Routes all characters received on the communication port to the macro.

PROMPT and PEND and PROMPT?

Allows testing of the communication port and input strings.

The following statements are used in a prompt statement group:

PCOUNT(id,#char)

Tests for the number of characters received on communication port.

PDCD(id)

Test the status of the Data Carrier Detect modem signal.

PKEY(id,char_code)

Tests for a character typed on the keyboard.

PQUIET(id,10ths_sec)

Tests for quiet time on communication port.

PSTR(type,id,char_str)

Tests for a string received on the communication port.

PWAIT(id,10ths_sec)

Waits for a specified period of time.

RETURN

Used to denote the logical end of a subroutine.

RSTACK

Used to reset the subroutine stack pointer.

SEND(text)

Transmits text to the communication port.

STEP

Used to break execution in a running macro to allow debugging.

Macro Evaluation

IF(logical_experession)

Used to make branching decisions based on values of input variables.

ELSE

Used with the IF statement to group a number of statements

DO

Used with UNTIL to begin a looping operation.

UNTIL(logical_expression)

Ends a DO loop statement group.

WHILE(logical_expression)

Performs a statement or statement group while a condition is true.

WEND

Ends a WHILE statement group.

Macro Variable Control

GETGLOBALINT(integer_number)

Used to pass strings between chained macros.

GETGLOBALSTR(string_number)

Used to pass integers between chained macros.

PUTGLOBALINT(integer_number,value)

Passes numbers to macros that will be chained to.

PUTGLOBALSTR(string_number,text)

Passes strings to macros that will be chained to.

RESTOREVARS

Restores a macro's variables after a CHAINRETURN.

SAVEVARS

Saves macro variables for a CHAINRETURN from another macro.

Math

ABS(value)
: Gives the absolute value of the integer argument.

CHKSUM
: Calculates the checksum of a string.

CRC
: Calculates the CRC of a string.

Menu Customizing

ADDBAR
Adds a menu bar to the WinComm menu bar set.

ADDCOMMAND(menu_number,menu_id,menu_text)
Adds a command to a pop-up menu item.

ADDMENU(bar_number,menu_text)
Adds a menu to the Menu Bar.

ENABLEMENU(menu_bar,id,greyed)
Used to enable and disable individual menu items.

HELP(help_file)
Identifies the custom context sensitive help file.

MENU?
Returns the value of the selected menu item.

MENUBAR?
Returns an identifier for the current menu bar.

SHOWBAR(bar_number)
Displays a new or changed menu bar.

Session Variable Control

ACTIVATESESS

Activates changes made to session variables.

GETSESSINT(int_id)

Used to return the currently loaded integer session variable.

GETSESSSTR(string_id)

Used to return the currently loaded string session variable.

PUTSESSINT(int_id,int_value)

Used to assign values to any session integer.

PUTSESSSTR(string_id,string_value)

Used to assign values to any session string.

Stock Dialog Box

ALERT(text)

Displays a stock dialog box with text and an OK button.

ALERTCANCEL(text)

Displays text in a stock dialog box with OK and Cancel buttons.

FILEOPENDLG(head_txt,ext,return_txt)

Displays a stock dialog box for selecting a file.

INPUTDLG(input_txt,heading_txt)

Displays a stock dialog box for input.

SAVEASDLG(heading_text,def_filename)

Displays a stock dialog box used to name a file before it is saved.

System Information

TIME?

Returns the current system time in integer format.

Text

ASC(text)
Returns the decimal value of an ASCII character.

BITSTRIP(text)
Sets the high order bit in text to 0.

CHAR(ascii_value)
Returns an ASCII character from a decimal value.

CLEAN(text)
Removes non-printing ASCII characters from a string.

CMP(str1,str2)
Compares 2 text strings.

DELETE(text,beg_pos,num_char)
Removes text from a string

EXTRACT(bracket_text,text,inst_no)
Extracts a substring from a text string.

LEFT(text,num_chars)
Returns a number of characters from the left portion of a string.

LEN(text)
Returns the length of a string.

LOWERCASE(text)
Converts text to lower case.

LPRINT(text)
Sends a line of text to the printer.

MID(text,first_char,num_chars)
Returns characters from the middle portion of a string.

NEXTCHAR?
Returns the next character in the receive buffer.

NULL(text)
Tests to see if a string is null.

PRINT(text)
Displays characters in the WinComm window text area.

QUOTE(text)
Puts a string in quotation marks.

RIGHT(text,num_chars)
Returns a number of characters from the right end of a string.

SEARCH(substr,str)
Finds the occurrence of a string in another string.

SUBST(orig_txt,find_txt,replace_txt,times)
Substitutes text for other text in a string.

UPPERCASE(text)
Returns a string as all upper case characters.

WinComm Command

ACTIVATE
: Activates the instance of WinComm running this macro.

CAPTURE(cont)
: Controls the capture feature of WinComm.

CLEAR
: Clears the WinComm screen and display buffer.

DIAL(tele_number)
: Dials a telephone number.

PRINTER(cont)
: Turns the printer on and off.

SHOW(change)
: Repaints the WinComm window for graphics display.

START
: Starts the currently loaded session.

STATUSLINE(status_text)
: Places text in the status line.

STOP
: Stops the currently loaded session.

WINMOVE(left,top,width,height)
: Sizes and positions the WinComm Window.

WinComm DLL

CALL(dll.fil|process,str1,str2,int1,int2)

Calls a dynamically linked library from within a WinComm macro.

WinComm File Control

NEW

Loads default WinComm session settings.

OPEN(session_file_name,pswrd,mode)

Opens a WinComm session file.

SAVEAS(file_name)

Saves the current session file using file_name.

WinComm File Transfer

RECEIVEASCII(diag_disp,file_name)

Sets WinComm up to receive an ASCII file.

SENDASCII(diag_disp,file_name)

Sends an ASCII file using the settings assigned in the current session.

XFER?

Variable used to determine the status of a file transfer.

WinComm Information

ACTIVE?

Indicates if WinComm is active.

ONLINE?

Tests to see if a WinComm session has been started.

STATUSLINE?

Returns the text displayed on the status line.

WINCOMMVER?

Gives the version of WinComm.

WinComm Packet Data Transfer

PKTIME(trans_time,rec_time)

Used to set the timing for packet transmission and reception.

RXPKT(cmd,rec_str)

Receives an error-free string of data from another macro using the TXPKT command.

TXPKT(cmd,xmit_str)

Sends an error-free string to another macro using RXPKT statement.

TXPKTSTAT?

The variable that indicates the status of a packet protocol transfer.

WinComm Protocol File Transfer

RECEIVEFILE(diag_disp,file_name)

Sets WinComm up to receive a file using an error-correcting protocol.

SENDFILE(diag_disp,file_name)

Sends a file using an error-free protocol assigned in the current session.

WinComm/WinLink File Control

PCTOPC(cmd,opt,text)

Used to control file operations of another computer via the serial communication link.

Windows Application Control

APPACTIVATE(title_bar_text)

Activates a loaded Windows application.

RUN(app_name,cmd_line,size)

Used to start another Windows application.

SENDKEY(key_text)

Sends normal ASCII characters to the active Windows application.

SENDSPECKEY(ctrl,key1_val,key2_val)

Sends non-printing and combinations of non-printing and printing characters to the active Windows application.

SETFOCUS(id)

Sets the focus to a given window or control within a Windows application.

Windows Information/File Control

SYSTEM
Saves system information in the WIN.INI file.

FREEMEM?
Returns the number of bytes of Windows free memory.

GETFOCUS?
Returns an identifier of the window that has the input focus.

GETSCRAP(value_id)
Returns information about the Windows display screen.

WINVER?
Gives the version of Windows.

DDEADVISE(chan_no,my_var$,server_var) DDE

This statement requests the server application to send the value of a variable to WinComm every time the variable changes.

Returns

Integer	1 if the advise was successful, 0 if it was not.

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link over which to receive data.
my_var$ = String	The string variable in this macro that will be changed each time the server sends a new value.
server_var = String	The name of the server's variable or reference to which the "ADVISE" is established.

Acting as a server, WinComm will establish an ADVISE to any macro string variable.

DDEEXEC(chan_no,"execute_text") DDE

This statement executes commands in the server application.

Returns

Integer	1 if the execute was successful, 0 if it was not.

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link (over which to execute commands in the server application).
execute_text = String	Any text the server application recognizes in the execute command.

DDEINIT(app_name,doc_name) DDE

This statement requests the server application to establish a DDE channel for data exchanges or control.

Returns

Integer	0 if the channel was not established, or a number greater than 0 if the channel is open. Use this channel number for any control of, or data transfer with, the server application.

Arguments

app_name = String	The name the server application recognizes for its DDE exchanges. Refer to the server application documentation to obtain this information.
doc_name = String	The name of the document or "Topic" that the server application recognizes. For Excel, this might be a spread sheet or macro sheet or system. For WinComm, it will be a session or a macro file name or the Topic "System".

WinComm recognizes the app_name string "WINCOMM". If more than one instance of the server application is running, all instances will respond to the DDEINIT. To refer to a specific instance of the application, append the Windows module instance number as text to the the app_name string. If the application was launched using the RUN statement, the module instance number of that instance is returned by the statement. See the RUN statement description in Chapter 10.

DDEPOKE(chan_no,server_var,my_var$) DDE

This statement sends data to the server application.

Returns

Integer	1 if the poke was successful, 0 if it was not.

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link (over which to send data).
server_var = String	The name of the server's variable or reference to a variable to change.
my_var$ = String	The string variable in this macro that will be sent to the server application.

NOTE

WinComm will accecpt a POKE to any session integer or string variable (if a session is open) or to any macro variable name (if a macro is running). Session variables are identified by their ID number. These ID numbers can be found in Chapter 10 under the descriptions of the GETSESSINT and GETSESSSTR statements. For session integer variables, use the number (converted to a string) as shown in the table above, to identify the integer variable to POKE. For session string variables, use the number as shown in the table above plus 1000 (converted to a string) to identify the string variable to POKE.

DDEREQ(chan_no,my_var,server_var) DDE

This statement obtains the current value of a variable from the server application.

Returns

Integer	1 if the request was successful, 0 if it was not.

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link over which to receive the data.
my_var$ = String	The string variable in this macro that will be assigned the value of server_var.
server_var = String	The name of the server's variable or reference.

NOTE

WinComm will respond to SysItems, Topics, Status, a session integer or string variable (if a session is open) or to any macro variable name (if a macro is running). Session variables are identified by their ID number. These ID numbers can be found in Chapter 10 under the descriptions of the GETSESSINT and GETSESSSTR statements. For session integer variables, use the number (converted to a string) as shown in the table to identify the integer variable to REQUEST. For session string variables, use the number as shown in the table plus 1000 (converted to a string) to identify the string variable to REQUEST.

DDETERM(chan_no) DDE

This statement terminates the DDE link or channel established with the server using DDEINIT.

Returns

Nothing

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link to terminate.

DDEUNADVISE(chan_no,server_var) DDE

This statement terminates an ADVISE established using DDEADVISE.

Returns

Nothing

Arguments

chan_no = Integer	The number returned from the DDEINIT statement identifying the DDE channel or link from which the advise is to be removed.
server_var = String	The name of the server's variable from which to remove the advise.

ABS(number) Math

Gives the absolute value of the integer argument.

Returns

Integer	The absolute value of the integer argument.

Arguments

number = Integer

Example

```
absolute = ABS(-2)   ;absolute is assigned the value of 2.
```

ACTIVATE WinComm Command

Activates the instance of WinComm running this macro.

Returns

Nothing

Arguments

None

ACTIVATESESS Session Variable Control

Activates any changes made to session variables.

Returns

Nothing

Arguments

None

NOTES

This statement will activate changes made to the session variables that control the fonts, screen colors, function keys, the keyboard and communication port.

ACTIVE? WinComm Information

Indicates if WinComm is active.

Returns

Logical	TRUE if the instance of WinComm running this macro is the active application, otherwise returns FALSE.

Arguments

None

ADD(text) DOS File Control

Adds text to the capture file.

Returns

error_code	-1 if an error occurred, otherwise returns 0.

Arguments

text = string	The text to be added to the capture file.

ADDBAR Menu Customizing

Provides an identifier for a new WinComm menu bar.

Returns

Integer A value that identifies this menu bar for use by the ADDMENU command.

Arguments

None

NOTES

The integer ID returned by the ADDBAR statement. This is used by the SHOWBAR statement to display the menu bar after menus and commands have been added to it by the ADDCOMMAND and ADDMENU statements.

Example

```
new_bar = ADDBAR
new_menu = ADDMENUE(new_bar,"&Mail")
ADDCOMMAND(new_menu,1,"&Read Mail")
ADDCOMMAND(new_menu,2,"&Send Mail")
SHOWBAR(new_bar)
```

This macro code fragment would obtain a bar identifier; modify it by adding the menu item "Mail"; add two commands to the Mail menu item; (e.g., "Read Mail" and "Send Mail") and then display it. The statement MENU? would return a 1 if Read Mail was selected or a 2 if Send Mail was selected.

ADDCOMMAND(menu_number,menu_id,menu_text,) Menu Customizing

Adds a command to a pop-up menu item.

Returns

Nothing

Arguments

menu_number = Integer — The value returned from the ADDMENU statement identifying the pop-up menu to which this command will be added.

menu_id = Integer — This value is returned by the MENU? statement when this command has been selected. Each command that is added to any menu bar must have an exclusive menu_id number.

menu_text = String — The actual text that will appear in the pop-up menu.

NOTES

As each command is added to a menu, it is positioned below the preceding command. An ampersand (&) in front of a character causes that character to be underlined and designated as the shortcut key for that command. There must not be more than 1 of the same character assigned as the shortcut key per menu. The variable MENU? is used to identify which menu item has been selected. It will return the menu_id assigned by the ADDCOMMAND statement for custom menu item commands. See the notes and example under the ADDBAR statement.

ADDMENU(bar_number,menu_text) Menu Customizing

Adds a menu to the Menu Bar.

Returns

Integer	A value that identifies this menu for use by the ADDCOMMAND statement.

Arguments

bar_number = Integer	A value returned from the BAR or MENUBAR? statements identifying the menu bar to which this menu item will be added.
menu_text = String	The actual text that will appear in the menu bar.

NOTES

As each menu is added to a bar, it is positioned to the right of the preceding menu. An ampersand (&) in front of a character causes that character to be underlined and designated as the shortcut key for that bar. There must not be more than 1 of the same character assigned as the shortcut key per bar. The menu can be added to the standard WinComm menu bar using MENUBAR? to get the WinComm ID number; or it can be added to a new bar using the ID returned by the BAR statement. See the notes and example under the ADDBAR statement.

ALARM

Makes the speaker beep.

Returns

Nothing

Arguments

None

ALERT(text) Stock Dialog Box

Displays a stock dialog box with text and an OK button.

Returns

Nothing

Arguments

text = String	The text string displayed in the alert box.

NOTES

Used to notify the operator that something has occurred or action needs to be taken. The dialog box has only an [OK] button. If a choice needs to be made between two selections, use the ALERTCANCEL dialog box.

ALERTCANCEL(text) Stock Dialog Box

Displays text in a stock dialog box with OK and Cancel buttons.

Returns

Logical	TRUE (1) if the [OK] button was pressed, FALSE (0) if the [Cancel] button was pressed.

Arguments

text = String	The text string displayed in the alert box.

NOTES

Used to notify the operator and allows a choice to be made before proceeding with the macro.

APPACTIVATE(title_bar_text) Windows Application Control

Activates a loaded Windows application.

Returns

Nothing

Arguments

title_bar_text = String	This string is the applications title bar text just as it appears when the application is running.

NOTES

Used with the SENDKEY and SENDSPECKEY statements to control other Windows applications. See also the RUN command which will load and run the application. The application must be activated using this statement before keys can be sent to it.

ASC(text) Text

Returns the decimal value of an ASCII character.

Returns

String	The decimal value of the first character of text.

Arguments

text = String	The string which will be evaluated by the statement.

NOTES

The opposite of the CHAR statement.

BITBKG(horz_poz,vert_poz,from,file_name)Graphics Customizing

Displays a bitmap background in the user window.

Returns

Nothing

Arguments

horz_pos = Integer	Specifies the horizontal position of the bitmap within the user window. 0 = Centered, 1 = left, 2 = right.
vert_pos = Integer	Specifies the vertical position of the bitmap within the user window. 0 = Centered, 1 = top, 2 = bottom.
from = Integer	Specifies whether the bitmap will be loaded directly from disk or from memory. 0 = memory, 1 = disk. Displaying from disk takes less memory but takes more time.
file_name = String	The name of the .BMP file to display.

NOTES

A user window must first be created using the USERWINDOW statement before a background graphic can be displayed. The statements BITMAP, BUTTON, METAFILE and HOTSPOT create objects that can be positioned on the background graphic within the user window. The BITMAP, BUTTON and HOTSPOT objects can be given an ID value to test for their selection using the event status statement OBJECT?. BITMAP and BUTTON can also have accelerator keys assigned by placing an ampersand [&] in front of the character in the "text" field to be used as the accelerator. That object can then be selected by pressing Alt+the accelerator key.

Use the WCUTIL WinComm Utility program to create the Graphic Customizing and Graphic Menu statements.

BITMAP(left,top,id,"text",file_name) Graphics Menus

Places a bitmap icon on the background graphic.

Returns

Nothing	The variable OBJECT? is assigned the ID value of the selected bitmap.

Arguments

left = Integer	The left side of the bitmap rectangle.
top = Integer	The top of the bitmap rectangle.
id = Integer	The exclusive ID value assigned to each object displayed in the user window to be used for identification when the object is selected.
text = String	The text that will be centered below the bitmap object. An [&] can be placed in front of the character to use as the keyboard shortcut key.
file_name = String	The name of the *.BMP file to use.

NOTES

Bitmaps should be displayed after a graphic has been displayed and destroyed after the graphic has been removed, see DELOBJECT. Use the WCUTIL WinComm Utility program to create the Graphics Customizing and Graphics Menu statements.

BITSTRIP(text) Text

Sets the high order bit of a byte to 0, thus converting graphics characters to normal printing ASCII characters.

Returns

String	A new string with all bit eights set to 0.

Arguments

text = String	The string which will be processed by the statement.

BUTTON(left,top,width,height,"id,text") Graphics Menus

Places a Windows push button on the background graphic.

Returns

Nothing	The variable OBJECT? is assigned the ID value of the selected push button.

Arguments

left = Integer	The left side of the push button rectangle.
top = Integer	The top of the push button rectangle.
width = Integer	The width of the push button rectangle.
height = Integer	The height of the push button rectangle.
id,text = String	A comma separated string consisting of the ID value assigned to this button and the text that will appear on the button face. OBJECT? will be assigned the value of ID when this button is selected. An & can be placed in front of the character to use as the keyboard shortcut key.
file_name = String	The name of the *.WMF file to use.

NOTES

Buttons should be displayed after a graphic has been displayed and destroyed after the graphic has been removed, see DELOBJECT. Use the WCUTIL WinComm Utility program to create the Graphics Customizing and Graphics Menu statements.

CALL(dll.fil|process,str1,str2,int1,int2) WinComm DLL

Calls a dynamically linked library from within a WinComm macro.

Returns

Integer	From the DLL if successful or 0 if not.

Arguments

dll.fil|process = String	dll.fil = the name of the dynamically linked library file. | = Separator. process = the function within the library to call.
str1 = String Variable	The name of the string variable to pass to the DLL.
str2 = String Variable	The name of the string variable to pass to the DLL.
int1 = Integer Variable	The name of the integer variable to pass to the DLL.
int2 = Integer Variable	The name of the integer variable to pass to the DLL.

NOTES

When a CALL statement is executed, WinComm will load the library and pass long far pointers to the variables in the argument list. For more information on use of the CALL statement, contact Synappsys.

CAPTURE(cont) WinComm Command

Controls the capture feature of WinComm.

Returns

Nothing

Arguments

cont = Logical TRUE = Turn capture on, FALSE = Turn capture off.

CHAIN(macro_name) Macro Control

Transfers macro execution to another macro.

Returns

Nothing	Windows will display a dialog box if the macro_name cannot be found.

Arguments

macro_name = String	The name of a compiled macro which is to pass execution.

NOTES

If you plan to return to the current macro and want to save the status of all variables in it, use the SAVEVARS statement before the chain command and RESTOREVARS immediately upon returning. To pass variables to and from the chained macro, use PUTGLOBALINT, PUTGLOBALSTR, GETGLOBALINT and GETGLOBALSTR statements.

CHAINRETURN Macro Control

Transfers macro execution back to a calling macro.

Returns

Nothing

Arguments

None

NOTES

A CHAIN with a CHAINRETURN can be only one level deep. See the notes under CHAIN.

CHAR(ascii_value) Text

Returns an ASCII character from a decimal value.

Returns

String	A one-character string of the ASCII value.

Arguments

ascii_value = Integer	The decimal value to be returned as an ASCII character.

NOTES

The opposite of the ASC statement.

CHGDIR(directory_name) DOS File Control

Changes the current directory.

Returns

Integer	-1 if error, 0 if OK.

Arguments

directory_name = String	The directory name or path of the existing directory to which to change.

CHGDRIVE(drive_letter) DOS File Control

Changes the current drive.

Returns

Integer	-1 if error, 0 if OK.

Arguments

drive_letter = String	A single character string indicating the drive letter to which to change.

CHKSUM(text) Math

Generates the checksum of a string.

Returns

Integer	The checksum of the string argument.

Arguments

text = String	The string to evaluate.

NOTES

The checksum is an integer that is the result of a simple add with no carry of each character in the string.

CLEAN(text) Text

Removes non-printing ASCII characters from a string.

Returns

String	A duplicate of the argument string with all non- printing ASCII characters removed.

Arguments

text = String	The string from which to remove non-printing characters.

CLEAR WinComm Command

Clears the WinComm screen and display buffer.

Returns

Nothing

Arguments

None

CLIPTOFILE(clipboard_type,filename) Clipboard Control

Copies the Clipboard Text, Bitmap or Metafile to a disk file.

Returns

Logical	TRUE if ok, FALSE if not ok.

Arguments

clipboard_type = Integer	The clipboard format. 3 = Text, 2 = Windows metafile/picture, 1 = Bitmap.
file_name = String	The file name in which to store the clipboard contents.

NOTE

The inverse of the FILETOCLIP statement. These files can be used to display graphics in the WinComm display area with the BITBKG, METABKG, METAFILE and BITMAP statements. The graphics can be created by any Windows application that can copy bitmaps or metafiles to the clipboard. You can see which formats are in the clipboard by running CLIPBRD.EXE. Bit maps will be identified as Bitmap. Metafiles will be identified as Picture.

CMP(str1,str2) Text

Compares 2 text strings to see if they are exactly the same.

Returns

Logical	TRUE if the strings match, otherwise FALSE

Arguments

str1 = String	The first string to compare.
str2 = String	The second string to compare.

COPYFILE(orig_filename,dest_filename)DOS File Control

Copies a disk file

Returns

Integer	1 if the file was copied OK -1 if not.

Arguments

orig_filename = String	The file you want to copy.
dest_filename= String	The disk, path and file name you want the file copied to .

NOTES

Used just as the DOS COPY command.

CRC(initialization,text) Math

Generates a cyclic redundancy check character.

Returns

Integer	The CRC value of the string argument.

Arguments

text = String	The string to evaluate.
initialization = integer	The starting value of the polynomial used in the evaluation.

NOTES

The CRC statement returns an integer value generated using the standard CRC polynomial.

CURDIR? DOS File Information

Returns the name of the current directory.

Returns

String The complete path name including the drive of the current directory.

Arguments

None

Notes

The path returned when in a directory or sub-directory is of the form C:\TEMP\WINCOMM. In the root, C:\ is returned.

CURDRIVE? DOS File Information

Returns the name of the current disk drive.

Returns

String	A single character string indicating the current drive.

Arguments

None

DELETE(text,beg_position,num_characters) Text

Removes text from a string.

Returns

String A copy of text with num_characters removed.

Arguments

text = String The input string.

beg_position = integer The position of the first character in the text to be removed.

num_characters = integer The number of characters to remove from the string.

DELETEFILE(file_name) DOS File Control

Deletes a disk file.

Returns

Integer	-1 if error, 0 if OK.

Arguments

file_name = String	The name of the file to be deleted.

DELOBJECT(type,start_id,end_id) Graphics Customizing

Used to remove different items of a graphic display.

Returns

Nothing

Arguments

type = Integer	The type of object to delete: 0 = User window 1 = Background graphic 2 = Button 3 = Metafile 4 = Bitmap 5 = Hot Spot
start_id = Integer	The ID number of the first object to delete.
end_id = Integer	The ID number of the last object to delete.

NOTE

Deleting the user window will destroy the background and all objects. The arguments start_id and end_id apply only to objects and should be set to 0 for all other types of deletes.

DIAL(tele_number)WinComm Command

Dials a telephone number.

Returns

Nothing

Arguments

tele_number = String The telephone number to dial.

DIALOGBOX lft,top,wdth,hght,type,title
DEND Dialog Box Customizing

The first and last statements in a Dialog Box statement group used for creating custom dialog boxes.

Returns

Nothing	The variable DIALOG? is assigned a sequence number that identifies the dialog event that occurred. A dialog event is the pressing of a push-button or the selection of an item in a list box. See the descriptions under the push button style, LISTBOX, DIRLISTBOX and FLISTBOX.

Arguments

lft = Integer	Defines the left position of the dialog box item.
top = Integer	Defines the top position of the dialog box item.
wdth = integer	Defines the width of the dialog box item.
hght = Integer	Defines the height of the dialog box item.
type = Integer	Argument is optional. If used, it is a byte wise OR of the following table that determines the appearance and operation of the dialog box.
title = String	Argument is optional. The text will be displayed in the title bar of the dialog box if the Moveable With Title Text option is selected.

Type Table

Binary Representation				Operation/Appearance
23	22	21	20	
0	0	0	0	Normal Modal Dialog Box
0	0	0	1	Center the Dialog Box Horizontally
0	0	1	0	Not Used
0	1	0	0	Moveable With Title Text
1	0	0	0	Modeless

NOTES

The integer values used for position and size of both the dialog box and all controls within the dialog box are "dialog units". These are based on the size of the system font. Each system font character is 4 horizontal dialog units wide and 8 vertical dialog units tall. Since each screen resolution has a different system font, the size of the dialog box and its items should be approximately the same on any screen resolution.

The dialog box is positioned in reference to the WinComm client area with the upper left corner defined as "left" 0, "top" 0. If the Center Horizontally option is selected, the vertical position will be defined by "top", but the dialog box will be centered in the WinComm client area. Dialog box items are referenced from the upper left corner of the dialog box. The upper left corner of the dialog box defined as "left" 0, "top" 0.

A modal dialog box is one that displays and prompts for operator input and disables any input or control of WinComm until it is removed by selecting a push button. Most dialog boxes WinComm uses are modal. A Modeless dialog box also displays and prompts for operator input but does not disable input or control of WinComm. Modeless dialog boxes are removed using the DESTROYDLG statement. The dialog box that displays when the System|Monitor menu item is selected is a modeless dialog box. The Windows style guide recommends modeless dialog boxes should have the title bar, and modal should not. However, any combination of options

listed under "Operation/Appearance" can be used in WinComm dialog boxes.

When the title bar option is used, the dialog box will have a control menu that contains a Close menu item. When this item is selected, DIALOG? will return a 0 just as if a Cancel push button were selected.

The statements below are used within a DIALOGBOX statement group to create custom dialog boxes for many uses. Use the WinComm Dialog Editor to quickly and easily generate the statements necessary for displaying custom dialog boxes.

BEGINGROUP_RADIOBUTTON
Designates the first radiobutton in a radiobutton group.

RADIOBUTTON **Creates additional radiobuttons in a radiobutton group.**

ENDGROUP_RADIOBUTTON
Designates the last radiobutton in a radiobutton group.

CANCELBUTTON Defines the push button in a dialog box that will return zero [0] in the DIALOG? statement when it is selected or the Esc key is pressed.

DEFCANCELBUTTON
Defines a cancel button in a dialog box that will be selected when the enter key is pressed.

DEFPUSHBUTTON Defines the push button in a dialog box that will be selected when the Enter key is pressed.

PUSHBUTTON Creates additional pushbuttons in a dialog box.

CHECKBOX Displays a standard check box within a dialog box.

EDITBOX Used for creating an edit box for text entry in a dialog box.

SEDIT Used for creating an edit box with vertical scroll bar for multi-line text entry in a dialog box.

GROUPBOX Draws a black rectangle around items in a dialog box.

COMBOBOX **Displays a drop down list box that allows selection of a single item from a list of items in a dialog box. This control can be used when space is limited.**

DIRLISTBOX Displays a list box for selection of a file from any directory on the computer.

FLISTBOX Displays a list box that allows selection of a single item from a large list of items in a dialog box. The list of items comes from a disk file.

LISTBOX Displays a list box that allows selection of a single item from a list of items in a dialog box.

LTEXT,CTEXT,RTEXT
Used to display text in a custom dialog box.

VTEXT

Related Statements

DESTROYDLG Used to remove or destroy a modeless dialog box.

DIALOG? The variable used to indicate the status of the dialog box.

DISABLE **Used to enable and disable WinComm's menu items and dialog box controls.**

ENABLECTL Enables and disables Push Button and List Box styles

HELPID	Identifies the help topic for this dialog box, which will display when the F1 key is pressed.
UPDATEDLG	Used to change any variables displayed in a dialog box after a dialog box event occurs.

HELPID integer_id

Identifies the help topic for this dialog box, which will display when the F1 key is pressed.

NOTES

HELPID follows the DIALOGBOX statement in a Dialog Box statement group. It is used to identify the context sensitive help topic for this dialog box when it is displayed. The text that appears in the Help dialog box comes from a special help file that is created with a Text Editor and compiled using the WCUTIL Program menu item (Create Help!). Help text can also be created using the help compiler provided with the Windows version 3 SDK. The integer_id is the sequential number of the help item for a dialog box in the help file. See Help in the WinComm Utility Program section of this manual and the HELP macro statement.

Contact Synappsys for an application note if you would like to use the Windows 3 help program with your WinComm macro programs.

BEGINGROUP_RADIOBUTTON lft,top,wdth,hght,text,var

Designates the first radiobutton in a radiobutton group.

RADIOBUTTON lft,top,wdth,hght,text,var

Creates additional radiobuttons in a radiobutton group.

ENDGROUP_RADIOBUTTON lft,top,wdth,hght,text,var

Designates the last radiobutton in a radiobutton group.

Arguments

text = Literal String	The text that will appear to the right of the radio- button.
var = Integer	The variable name used to identify this radiobutton.

Notes

Radiobuttons are used in a dialog box to allow selection of one item from a small group of items. One button is used to represent each item in the group. The first item in the group is created with a BEGINGROUP_ button and the last with an ENDGROUP_ button. All buttons between are created with RADIOBUTTON. Only one radiobutton can be selected in a group at one time. One radiobutton in the group can be shown selected by assigning its variable to 1 before the dialog box statement group is executed. The variable set to 1 when a dialog event occurs, indicates the radiobutton that was selected. There can be 4 groups of 16 radio- buttons per dialog box.

CANCELBUTTON lft,top,wdth,hght,button_text

Defines the push button in a dialog box that will return zero [0] in the DIALOG? statement when it is selected or the Esc key is pressed. None of the variables in the dialog box will be updated. There can be only one CANCELBUTTON per dialog box.

DEFCANCELBUTTON lft,top,wdth,hght,button_text

Defines a cancel button in a dialog box that will be selected when the enter key is pressed. When the dialog box is first displayed, this push- button will have a blackened border. It will return a zero (0) in the DIALOG? statement when it is selected. A zero (0) will also be returned when the Enter or the Esc key is pressed. There can be only one DEF push button (DEFPUSHBUTTON or DEFCANCELBUTTON) per dialog box.

DEFPUSHBUTTON lft,top,wdth,hght,button_text

Defines the push button in a dialog box that will be selected when the Enter key is pressed. When the dialog box is first displayed, this push- button will have a blackened border. There can be only one DEF push button (DEFPUSHBUTTON or DEFCANCELBUTTON) per dialog box.

PUSHBUTTON lft,top,wdth,hght,button_text

Creates additional pushbuttons in a dialog box. There can be a maximum of 8 PUSHBUTTONs or 7 PUSHBUTTONs plus one DEFPUSHBUTTON per dialog box.

UPDATEPB lft,top,wdth,hght,button_text

UPDATEPB is a special case of the push button style that allows updating of modal dialog box variables. This occurs without destroying and re-creating the dialog box display. When an UPDATEPB is selected, DIALOG? will be assigned a series of values. These values begin with 100 for the first UPDATEPB in a sequence and up to 103 for the fourth (the maximum number).

The processing statement group that is executed when an UPDATEPB is selected will reassign the variables as required. The macro should then execute the UPDATEDLG statement to show the changes in the dialog box and return to the dialog box event processing procedure. UPDATEPB should be used only with modal dialog boxes since all pushbuttons operate like UPDATEPB's in a modeless dialog box.

Arguments

button_text = Literal StringThe text that will appear in the push button.

DESTROYDLG Modeless Dialog Box Control

Used to remove or destroy a modeless dialog box.

UPDATEDLG(update_items) Update Dialog Box Control

Used to change any variables displayed in a dialog box after a dialog box event occurs. Also must be used to update variables in a modeless dialog box.

Arguments

update_items = integer A number that represents the types of controls to be updated. The number is a byte wise or of the following table.

Binary Representation							Update items
26	25	24	23	22	21	20	
0	0	0	0	0	0	0	Nothing
0	0	0	0	0	0	1	File List Box (FLIST)
0	0	0	0	0	1	0	File Text Box (STEXT)
0	0	0	0	1	0	0	File Edit Box (SEDIT)
0	0	0	1	0	0	0	Edit Box (EDIT)
0	0	1	0	0	0	0	Vtext (VTEXT)
0	1	0	0	0	0	0	List Box (LISTBOX & COMBOBOX)
1	0	0	0	0	0	0	Directory List Box (DIRLISTBOX)
1	1	1	1	1	1	1	Everything

DIALOG? Event Identifier

The variable used to indicate the status of the dialog box.

Returns

Integer	0 = CANCELBUTTON or DEFCANCELBUTTON pressed 1 - 8 = PUSHBUTTON or DEFPUSHBUTTON pressed 100 - 103 = UPDATEPB pressed 150 - 153 = Item has been selected in a LISTBOX 200 - 203 = Item has been selected in a FLISTBOX 250 = File selected in the DIRLISTBOX

Arguments

None

NOTES

The value returned by DIALOG? indicates the status of a dialog event when it occurs in a dialog box. A dialog event occurs when a push button or list box item is selected. While a dialog box is displayed, DIALOG? will return 255 until something is selected. Regardless of its position in the push button sequence, the CANCELBUTTON or DEFCANCELBUTTON always returns a 0 in DIALOG?, and no variables in the Dialog box are changed when they are selected. Depending on their sequence in the dialog box statement group, PUSHBUTTON and DEFPUSHBUTTON return a value other than zero in DIALOG?. The first button in the statement group returns a 1, the second a 2, etc. If there are UPDATEPB pushbuttons, they will return numbers beginning with 100. LISTBOX controls return values beginning with 150. FLISTBOX controls return values beginning with 200. A selection in a DIRLISTBOX will return 250. See the notes under LISTBOX, DIRLISTBOX and FLISTBOX.

ENABLECTL(id,enabled) Dialog Box Control

Enables and disables Push Button and List Box styles

Arguments

id = Integer	The ID of the Push Button or List Box to be controlled.
enabled = Integer	0 = disable, 1 = enable the control.

Notes

The ENABLECTRL statement is used in a macro to "grey out" a control and prevent it from being selected. The ID of a Push Button or List Box is the same as would be returned in DIALOG? when the item is selected.

CHECKBOX lft,top,wdth,hght,text,variable Check Box Style

Displays a standard check box within a dialog box.

Arguments

text = Literal String	The label that appears to the right of the check box.
variable = integer	The variable that is assigned the value of the "check" when any dialog event occurs.

Notes

Check boxes are used in a dialog box to allow selection of options or items in a dialog box. If a check box variable is set to 1 before the dialog box statement group is executed, the box will be checked when the dialog displays. Clicking the check box will toggle the check on and off. Variable will be 1 if the box was checked when a dialog box event occurres. If not checked, variable will be 0.

EDITBOX lft,top,wdth,hght,variableSingle Line Edit Box Style

Used for creating an edit box for text entry in a dialog box.

Arguments

variable = String VariableContains the text that is present in this edit box when a dialog box event occurs. It may be initialized before the dialog box is displayed to provide default text for this variable.

NOTES

The EDITBOX is used in a dialog box to allow the entry of text and data. The edit box uses the automatic horizontal scroll style which will allow text strings longer than the box width to be typed -- up to the string variable maximum of 255 characters. The cursor keys can be used to position the text and review everything in the box. Edit boxes are usually 12 units high.

SEDIT lft,top,wdth,hght,file_name MultiLine Edit Box Style

Used for creating an edit box with vertical scroll bar for multi-line text entry in a dialog box.

Arguments

file_name= String VariableThe string variable assigned the name of the file to be used by the SEDIT statement.

NOTES

SEDIT is used in a dialog box to allow entry of text that is expected to be longer than 255 characters. It displays an edit box with a vertical scroll bar to allow editing of a text disk file. If file_name contains text when the dialog box containing SEDIT is opened, it will be displayed in the edit box. The text can be typed in and edited using the standard Windows Edit Box procedures. When a dialog box event occurs, all text in the SEDIT box (including any text that may not be visible because it is not in the scroll area) will be written to the file designated by file_name. See STEXT for multi-line text display without the editing capability.

GROUPBOX lft,top,wdth,hght,text_to_display Black Frame Style

Draws a black rectangle around items in a dialog box.

Arguments

text_to_display = String The text that will be displayed at the upper left corner of the group box.

NOTES

The GROUPBOX is used to draw a black frame around related items within a dialog box.

COMBOBOX lft,top,wdth,hght,text,sel_text List Box Style

Displays a drop down list box that allows selection of a single item from a list of items in a dialog box. This control can be used when space is limited.

Arguments

text = String	The comma-separated string of items that will be displayed in the list box.
sel_text = String	The text string that was selected in the list box.

DIRLISTBOX left,top,width,height,filter_var,sel_file
List Box Style

Displays a list box for selection of a file from any directory on the computer.

Arguments

fliter_var = String — A filter using the standard DOS convention of ? and * to display only selected files.

sel_file = String Variable — A string variable that is assigned the full path and file name selected in the list box.

NOTES

The list box displays the files, sub-directories, disk drives and the parent directory [..] of the current directory. The DIRLISTBOX control will change directories when the operator makes the selection by double clicking on the directory. When a file is selected in the list box, sel_file will = the selected file name, and DIALOG? will be assigned the value of 250 to indicate the selection. If the operator double clicks a file name, the control will operate as if DEFPUSHBUTTON has been selected.

FLISTBOX lft,top,wdth,hght,file_name,sel_text List Box Style

Displays a list box that allows selection of a single item from a large list of items in a dialog box. The list of items comes from a disk file.

Arguments

file_name = String Variable The string variable assigned the name of the file containing the list of items to display in the list box. The list is made up of CR LF separated strings.

The string file_name can have two forms:

1 - "file_name" only; display the complete file.

2 - "file_name,seek_position,number_bytes", display the file beginning at seek_position and display number_bytes. seek_position and number_bytes are numeric values.

sel_text = String Variable A string variable that is assigned the text that was selected in the list box.

NOTES

FLISTBOX is used in a dialog box to allow selection of one item from a large number of items in a list. The width of the FLISTBOX should be the width of the longest line expected to be displayed in the box. See the notes under LISTBOX for information regarding updating the listbox after a selection has been made.

LISTBOX lft,top,wdth,hght,text,sel_text List Box Style

Displays a list box that allows selection of a single item from a list of items in a dialog box.

Arguments

text = String	The comma-separated string of items that will be displayed in the list box.
sel_text = String	The text string that was selected in the list box.

NOTES

LISTBOX is used in a dialog box to allow selection of one item from a list. The list comes from a variable that is composed of comma-separated items that make up the list. If the list is longer than 255 characters, use FLISTBOX. Any time a selection is made within a LISTBOX or FLISTBO, the value of DIALOG? will change to indicate the selection. DIALOG will be assigned a value of 150 for the first LISTBOX in the dialog box statement group, 201 for the second, etc. FLISTBOX values begin at 200. The processing statement group executed when an item is selected in a list box should reassign the variables as required; execute the UPDATEDLG command to actually show the changes in the dialog box and return to the dialog box procedure.

LTEXT,CTEXT,RTEXT lft,top,wdth,hght,text Static Text Style

Used to display text in a custom dialog box.

Arguments

text = String	The text that will be displayed in the dialog box, up to 255 characters.

NOTES

LTEXT, CTEXT and RTEXT are used to display static text in a dialog box. To display the value of a string variable, use VTEXT. LTEXT, CTEXT and RTEXT position text at the left, center and right of the rectangle, respectively.

STEXT lft,top,wdth,hght,file_name

Used for creating a text box with vertical scroll bar for multi-line text viewing in a dialog box.

Arguments

file_name= String VariableThe string variable assigned the name of the file to be displayed by the STEXT statement.

file_name can have two forms:

1 - "file_name" only; display the complete file

2 - "file_name,file_length" where file_length is the number of characters in the file. This form is used to display a file that is increasing in length, such as one being received on the communication port. This STEXT operating mode allows the scroll bars to work properly even when displaying a partial file.

NOTES

STEXT is used to display multi-line text files in a dialog box. The text is automatically wrapped horizontally to fit inside the STEXT frame. The standard Windows cursor movement conventions apply within the STEXT frame. See SEDIT for multi-line text editing capability.

DIRPATH lft,top,wdth,hght Variable Text Box Style

Used to display the current disk drive and directory within a dialog box.

Arguments

None	Only position and size are used with this control.

NOTES

This is a special case of the variable text box used with the DIRLISTBOX to show drive and path selected from within the DIRLISTBOX.

VTEXT lft,top,wdth,hght,variable_name Variable Text Box Style

Used to display a string variable within a dialog box.

Arguments

variable_name = String The variable name to be displayed by the VTEXT statement.

NOTES

VTEXT can be used with UPDATEPB to change text displayed in the dialog box.

DO

Macro Evaluation

Used to begin a looping operation.

Returns

Nothing

Arguments

None

NOTES

Used in conjunction with the UNTIL statement to set up a looping statement group.

DOSVER? DOS Information

Gives the version of DOS.

Returns

String The version of DOS currently running.

Arguments

None

ELSE Macro Evaluation

Used in IF branching structures to group a number of statements..

Returns

Nothing

Arguments

None

NOTES

See the notes under the IF statement.

ENABLEMENU(menu_bar,id,greyed) Menu Customizing

Used to enable and disable individual menu items.

Returns

Nothing

Arguments

menubar = Integer	The ID of the menu bar which contains the menu to control. This value is obtained from the ADDBAR or MENUBAR? statements.
id = Integer	The value of the menu item to control. If the menu item is a standard WinComm menu item, this value comes from the table below. If the menu item is a custom menu item created with the ADDCOMMAND statement, use the value assigned to its menu_id argument.
greyed = Integer	0 = Enable, 1 = Disabled or Greyed

ID table for WinComms standard menu items:

File	
New Session	200
Open Session	201
Edit Session	202
Start Session	203
Send File	204
Receive File	205
Change Protocol	206
Send ASCII	207
Receive ASCII	208
Printer Setup	209
Playback File	210
Exit	211
Edit	
Copy Text	213
Paste	214
Buffer to Capture	215
Buffer to File	216
Buffer to Printer	217
Clear Screen	218
System	
Defaults	219
Monitor	220
Macro	
Run	221
Files	222
Start/Stop Recorder	223

END — Macro Control

Indicates the logical end of a macro.

Returns

Nothing

Arguments

None

NOTES

Must be included as the last statement in a macro. The first END statement encountered by the compiler stops compilation.

ENVIRON(environ_ident) DOS Information

Returns information about the current DOS environment.

Returns

String	The string representing the DOS environment variable requested.

Arguments

environ_ident = String	The DOS environment variable to return. See your DOS manual for the names of the environment variables.

EXTRACT(bracket_text,text,instance_number) Text

Returns part of a string that is bracketed by a sub-string.

Returns

String — The string that lies between the instance_number of bracket_text and the preceding bracket_text.

Arguments

bracket_text = String — The character or characters that "bracket" the sub-string to extract from text.

text = String — The string to evaluate.

instance_number = integer — The occurrence number of bracket_text in text that immediately follows the sub-string in text to return. The string returned by the statement consists of the characters that lay between this point and the preceding bracket_text.

NOTES

This statement is useful for returning text from a comma or tab separated string or a word from a sentence.

Example:

```
within_text$ = extract(",","no1, no2, no3, no4,",3)
;would assign the variable within_text$ the value " no3"
fifth_word$ = extract(" ","Now is the time for all",5)
;would assign the variable fifth_word$ the value "for"
```

FCLOSE(file_number) DOS File Control

Closes a disk file.

Returns

Nothing

Arguments

file_number = Integer The number of the file (returned from the FOPEN statement) to close.

FILEATTRIBUTE(file_name) DOS File Information

Used to obtain the attributes of a file.

Returns

Integer	The bit wide "or" of the attributes of the file.

Attribute Table

Binary Representation						Attribute Indicated
25	24	23	22	21	20	
0	0	0	0	0	0	Normal
0	0	0	0	0	1	Read Only
0	0	0	0	1	0	Hidden
0	0	0	1	0	0	System
0	0	1	0	0	0	Disk Volume Name
0	1	0	0	0	0	Subdirectory Name
1	0	0	0	0	0	Archive Bit Set

Arguments

file_name = String	The name of the file for which you want to know the attributes.

FILEFIND(file_name,attributes) DOS File Information

Finds the first file meeting name and attribute requirements.

Returns

String	The file name of the first file that meets the file name and attribute search criteria.

Arguments

file_name = String	The file name (which can include the path or wild card characters) of the file to be found.
attributes = Integer	A byte wide "or" of the attribute table specifying the type of file to find

See the Attribute table under FILEATTRIBUTE

NOTES

Can be used in looping statement groups with FINDNEXT to return all files that meet the search criteria. If no files are found, the statement returns a null string.

Example

```
file$ = FILEFIND("c:\*.*",VALBIN("00111111"))
PRINT(file$+"^m^j")
file$ = FINDNEXT
WHILE(NULL(file$) == FALSE)
        PRINT(file$+"^m^j")
        file$ = FINDNEXT
WEND
;this macro code will print all files including sub-directories
;and volume names in the root directory of drive C.
```

FILEOPENDLG(heading_text,file_extension,return_txt) Stock Dialog Box

Displays a stock dialog box for selecting a file.

Returns

Logical	TRUE if the OK button was pressed, FALSE if the Cancel button was pressed.

Arguments

heading_text = String	This is the text that will be displayed at the top of the stock dialog box.
file_extension = String	A valid DOS file name. This will be used by the dialog box as a "show filter" allowing the dialog box to show only the files you want to show. It can contain the [?] and [*] wild card characters.
return_text = string	The name of the file that was selected when the OK button was pressed.

FILESIZE(open_file) DOS File Information

Returns the size of a file.

Returns

Integer	The size of the file in bytes.

Arguments

open_file = integer	The number (returned from the FOPEN statement) of the file which to obtain the size.

FILETIME(open_file) DOS File Information

Returns the creation or last modification date of a file.

Returns

Integer	The number of seconds from midnight Jan 1, 1970, to the time the file was created or last modified.

Arguments

open_file = integer	The number (returned from the FOPEN statement) of the file from which to get the time.

NOTES

To convert this integer time to a string use the STRTIME statement.

FILETOCLIP(clipboard_type,file_name) Clipboard Control

Copies a Windows metafile, bitmap or text file to the Clipboard.

Returns

Integer	-1 if error, 0 if OK.

Arguments

clipboard_type = Integer	The Clipboard format. 3 = Text, 2 = Windows metafile (picture). 1 = Bitmap.
file_name = String	The file name of the file to be copied to the Clipboard.

FINDNEXT DOS File Information

Finds the next file that meets the search criteria set up by the FILEFIND statement.

Returns

String	The name of the file that meets the search criteria.

Arguments

None

NOTES

The search criteria must have first been established by the FILEFIND statement. If no other files are found that meet the search criteria, FINDNEXT returns a null string. See the example under FILEFIND.

FOPEN(open_mode,file_name) DOS File Control

Opens a disk file for information, read or read/write operation.

Returns

Integer	The ID number of the open file for use by all other file statements that will operate on this file.

Arguments

open_mode = Integer	See open mode table below.

Open Mode Table

Decimal Value	Type of Open	Action
16384	Exists	FOPEN returns Non zero if the file exists
4096	Create	Creates new if not existing, overwrites if existing, pointer at beginning
512	Delete	Deletes the file
2	Read/Write	Opens for Read/Write, pointer at beginning
0	Read	Opens for Read, pointer at beginning

file_name = String	The name of the file to open. The name may include the drive and path.

NOTES

Use the integer number from the table to open the file in the mode required.

FREAD(open_file,number_of_characters) DOS File Control

Returns a specified number of characters from a file.

Returns

String	The number of characters specified.

Arguments

open_file = integer	The ID number of the file (returned from the FOPEN statement) from which to read the text.

NOTES

To read a "line" of text from the open file, use FREADLN. The point in the file from which the characters are read is determined by the "pointer". The pointer is initially positioned by the "open mode" of the FOPEN statement. It is repositioned with the FSEEK statement. The pointer is positioned at the character immediately following the last character read by the preceeding FREAD. FREAD returns a null if no characters remain to be read from the file.

FREADLN(open_file) DOS File Control

Reads a line of text from an open file.

Returns

String	The next line of text. Terminated by a CR/LF or a null (for a read error).

Arguments

open_file = integer	The ID number (returned from the FOPEN statement) of the file from which to read the text.

NOTES

Returns characters from the file up to and including the next LF (or 255 characters if a LF is not found). Including the LF the maximum number of characters returned is 255. To return a given number of characters from a file, use FREAD. The point in the file from which the next line will be read is determined by the "pointer". The pointer is initially positioned by the "open mode" of the FOPEN statement. It is repositioned with the FSEEK statement. The pointer is positioned at the beginning of the next line to be read after a FREADLN. FREADLN returns a null if no lines remain to be read from the file.

FREEMEM? Windows Information

Returns the number in bytes of Windows free memory.

Returns

Integer	The number in bytes of Windows free memory.

Arguments

None

FSEEK(open_file,pointer_offset,seek_type)DOS File Control

Positions a pointer in an open file for reading or writing.

Returns

Integer	-1 if error, or if OK. The position of the pointer in number of characters (bytes) from the beginning of file.

Arguments

open_file = integer	The ID number (returned from the FOPEN statement) of the file of which to position the pointer.
pointer_offset = Integer	The number of character positions to offset the pointer with reference to the seek_type position.
seek_type = Integer	Indicates the position in the file where the pointer should first be placed. 0 = Beginning, 1 = Current 2 = End.

NOTES

The current location of the pointer within an open file can be found by evaluating the integer returned using: position = FSEEK(open_file,0,1).

FWRITE(open_file,num_of_char,text) DOS File Control

Writes a string of text to an open disk file.

Returns

Integer	-1 if error, 0 if OK.

Arguments

open_file = integer	The number of the file (returned from the FOPEN statement) for which text is to be written.
num_of_char = Integer	The number of characters to be written to the file.
text = String	The text to be written into the file.

NOTES

If you want to write a line of text to the file, the FWRITELINE statement writes a string of text into the file terminated by a CR LF. The point within the file where the characters will be written is determined by the pointer. See the FOPEN mode action and the FSEEK descriptions.

FWRITELN(open_file,text) DOS File Control

Writes a line of text to an open disk file.

Returns

Integer	-1 if error, 0 if OK.

Arguments

open_file = integer	The number of the file (returned from the FOPEN statement) for which text is to be written.
text = String	The line of text to be written in the file.

NOTES

The FWRITELN statement writes the string of text into the file appended with a CR LF. To write a string without the CR LF appended, use FWRITE. The point within the file where the characters will be written is determined by the pointer. See the FOPEN mode action and the FSEEK descriptions.

GETFOCUS? Windows Information

Returns an identifier of the window that has the input focus.

Returns

Integer	The number identifying the window or control within a dialog box that currently has the input focus.

Arguments

None

NOTES

This statement is used with the SENDKEYS commands to determine when the focus has changed from one window to another. A typical use of this statement would be to determine if a dialog box has disappeared (indicating that an application is finished sending a document to the printer, or that the operator has made a selection). The value returned can also be used by the SETFOCUS statement to set the focus back to this control after the focus has been moved. The "focus" is defined as the control in a dialog box that will be selected when the return or space key is pressed. They would have a darkened border or a grey outline.

GETGLOBALINT(integer_number) Macro Variable Control

Used to pass integers between chained macros.

Returns

Integer	The number that was assigned to this global integer variable.

Arguments

integer_number = Integer0 through 7 identifying the ID of the global number.

NOTES

See CHAIN and CHAINRETURN statements.

GETGLOBALSTR(string_number) Macro Variable Control

Used to pass strings between chained macros.

Returns

String The string that was assigned to this global integer variable.

Arguments

string_number = Integer 0 or 1 identifying the ID of the global string.

NOTES

See CHAIN and CHAINRETURN statements.

GETSCRCAP(value_id) Windows Information

Used to obtain information about the display screen.

Returns

Integer	The value of the requested variable.

Arguments

value_id = Integer	See the table below.

Screen Capacity Table

Decimal Value	Information Returned
8	The screen width in pixels
10	The screen height in pixels
24	The number of colors the screen supports

GETSESSINT(int_id) Session Variable Control

Used to return the currently loaded integer session variables.

Returns

Integer The value of the requested session integer.

Arguments

int_id = Integer The ID number of the requested integer session variable.

Session Integer ID Number Table

ID	Variable	Notes
	Terminal Session Parameters	
0	Session type	0=Terminal 1=Phone Book 2=PC to PC
1	Password Protected	0=NO 1=Yes
2	Communication Port	0=Port 1 1=Port 2
3	Type of connection	0=Modem 1=Direct
4	Comm port baud rate	0=300 baud 1=600 baud 2=1200 baud 3=2400 Baud 4=4800 baud 5=9600 baud 6=19.2k baud 7=38.4k baud 8=57.6k baud 9=115k baud
5	Terminal emulation type	0=ANSI 1=ANSI-BBS 2=None 3=Vidtex 4=VT-102 5=VT-52
6	File transfer protocol	0=CIS-B+ 1=Kermit 2=XModem 3=XModem-1K 4=XModem-CRC 5=Ymodem 6=Ymodem=g 7=Zmodem
7	Modem type	Integer index of modem in listbox
8	Redial after unsuccessful	0=No 1=Yes
9	Time between redial attempts	The number of seconds between
10	The number of redial attempts	The number of attempts
11	Start capture on session start	0=No 1=Yes Capture file is String ID #7
12	Append the capture file	0=Start over 1=Append
13	Query for capture file Name	0=No 1=Yes
14	Run a macro on session start	0=No 1=Yes Macro file is string ID #8
15	Connect on session open	0=No 1+Yes
	ASCII Send File Parameters	
16	Send Carriage returns	0=No 1=Yes
17	Send line feeds	0=No 1=Yes
18	Wait between sending lines	0=No 1=Yes
19	Time between sending lines	Time given in tenths of seconds if ID 18 = Yes
20	Wait between characters	0=No 1=Yes
21	Time between characters	Time given in tenths of seconds if ID 19 = yes
	Display Parameters	
22	Screen Font Height	Number of Pixels in font as shown in list box
23	Screen Font Width	Number of Pixels in font as shown in list box
24	Character Set	0=ANSI 1=OEM As shown in Session Editor
25	Character Color	Value = 0 through the maximum number
26	Screen Color	of colors the system can
27	Blinking Color	display. Use GETSCRCAPS macro statement
28	Bold Color	to obtain the number of system colors available.
29	Hide Password on screen	0=No 1=Yes
30	Display the Function keys	0=No 1=Yes
31	Double click one character	0=No 1=Yes

32	Tab stops every nth position	Integer, character positions between tab stops				
33	Screen Scroll Buffer	0 = None	1 = 8K	2 = 16K	3 = 32K	
	Terminal Parameters					
34	Echo typed characters to screen	0=No	1=Yes			
35	Remove ANSI escapes	0=No	1=Yes			
36	Display graphic characters	0=No	1=Yes			
37	132 Columns	0=No	1=Yes			
38	Wrap line	0=No	1=Yes			
39	Force return on linefeed	0=No	1=Yes			
40	Force linefeed on return	0=No	1=Yes			
	Modem Parameters					
41	Allow comm port speed change	0=No	1=Yes			
42	Value for "OK"					
43	Value for "Busy"					
44	Value for "Error"					
45	Value for "No Dial Tone"					
46	Value for "Connect 300 Baud"					
47	Value for "Connect 600 Baud"					
48	Value for "Connect 1200 Baud"					
49	Value for "Connect 2400 Baud"					
50	Value for "Connect 2400 Baud Error Free"					
51	Value for "Connect 4800 Baud Error Free"					
52	Value for "Connect 9600 Baud Error Free"					
53	Value for "Connect 19200 Baud Error Free"					
	Communication Port Parameters					
54	Number of Data Bits					
55	Stop Bits	0 = 1	1 = 1.5	2 = 2		
56	Parity	0=No	1=Yes			
57	Comm Buffer Size	0=2K	1=4K	2=8K	3=12K	4=16k
58	Type of Handshake	0=none	1=Both	2=Software	3=Hardware	
59	Start Character	Decimal Integer Value				
60	Stop Character	Decimal Integer Value				
61	PC to PC Connect on Open	0=No	1=Yes			
62	Phone Book Entries	The number of entries in the autodialer				
63	Printer Font Height	The height as shown in the Printer Settings dialog box				
64	Printer Character Set	0=ANSI	1=OEM			
65	Protocol Timing	0=Tight	1=Normal	2=Loose	3=Sloppy	
66	Make Backup File	0=No	1=Yes			

NOTES

See PUTSESSINT to set these values.

GETSESSSTR(string_id) Session Variable Control

Used to return the currently loaded string session variables.

Returns

String — The value of the requested session string.

Arguments

string_id = Integer — The ID number of the requested string session variable.

Session String ID Number Table

ID	Variable	Notes
0	Notes	0 to 48 Characters
1	Session File Password	0 to 11 Characters
2	Terminal Telephone Number	0 to 30 Characters
3	Session Variable Password	0 to 30 Characters
4	User ID	0 to 30 Characters
5	Net ID	0 to 30 Characters
6	Capture File Name	Proper DOS Filename 12 characters max.
7	Macro File Name	Proper DOS Filename 12 characters max.
8	ASCII send "wait for" characters	0 to 8 characters
9	Display Font	The name of the font as shown in the list box
10	Capture Strip Filter	Comma separated string of 34 1's and/or 0's. The first 32 characters in the string represent the first 32 non-printing ASCII characters. Character 33 represents bit Eight of each ASCII code, Character 34 represents ANSI escapes. 1 means remove from the string before capturing 0 means leave in the string for capturing
11	Answer Back String	0 to 20Characters
12	Keyboard Remap	The name of the keyboard file 12 characters max.
13	Terminal Strip Filter	Comma separated string of 34 1's and/or 0's. The first 32 characters in the string represent the first 32 non-printing ASCII characters. Character 33 represents bit Eight of each ASCII code, Character 34 represents ANSI escapes. 1 means remove from the string before display, 0 means leave in the string for display.
14	Modem initialization string	0 to 32 characters
15	Modem active DCD string	0 to 8 characters
16	Modem ignore DTR string	0 to 8 characters
17	Modem disconnect string	0 to 20 characters
18	Modem error free initialization string	0 to 20 characters
19	Modem answer mode string	0 to 8 characters
20	Modem set speaker volume string	0 to 8 characters
21	Modem speaker control string	0 to 8 characters
22	Modem pre dial string	0 to 8 characters
23	Modem dial suffix string	0 to 8 characters
24	PC to PC Dial String	0 to 30 characters
25	WinComm.exe Directory path	0 to 64 characters

26	WinComm Session file directory path	0 to 64 characters
27	WinComm download file directory path	0 to 64 characters
28	WinComm macro source file path	0 to 64 characters
29	WinComm capture file directory path	0 to 64 characters
30	WinComm macro default text editor	0 to 64 characters
31	Function key F2	
32	Function key F3	The string information for the function
33	Function key F4	keys consists of comma separated
34	Function key F5	strings each consisting of 3 fields.
35	Function key F6	Field 1--One character = macro Identifier
36	Function key F7	1 = Field 3 is a macro name
37	Function key F8	0 = Field 3 is text to transmit
38	Function key F9	
39	Function key ctl F2	Field 2--0 to 10 Characters = Label
40	Function key ctl F3	The label that appears on the
41	Function key ctl F4	screen
42	Function key ctl F5	
43	Function key ctl F6	Field 3--0 to 30 characters = Text field
44	Function key ctl F7	The text to be transmitted or
45	Function key ctl F8	the macro name
46	Function key ctl F9	
47	Printer Font	The font name as it appears in the list box
48	Printer Header	0 to 65 Characters
49	Printer Footer	0 to 65 Characters
50	Selected Modem	The modem name as it appears in the list box

NOTES

See PUTSESSSTR to set these values.

GOSUB label — Macro Control

Transfers execution of the macro to a subroutine.

Returns

Nothing

Arguments

label = String	The label of the subroutine where execution is to proceed.

NOTES

The subroutine must contain a RETURN statement for macro execution to return to the statement immediately following the GOSUB statement.

GOTO label — Macro Control

Transfers execution of the macro to a label.

Returns

Nothing

Arguments

label = String	The label in the macro whereexecution is to proceed.

HALT

Macro Control

Stops the in-line execution of the macro.

Returns

Nothing

Arguments

None

HELP(help_file) WinComm Custom Help

Identifies the context sensitive help file for WinComm to use when running this macro.

Returns

Nothing

Arguments

help_file = String	The name of the compiled Help file WinComm will use when the F1 key is pressed or HELP is selected from the menu. The file must be located in the same directory as WINCOMM.EXE. If the string is null, WinComm will revert to the standard WinComm Help file.

NOTES

WinComm's macro language supports either custom context sensitive help as described here, or the Windows 3 help engine. If you use the help as described here, the Help file must be compiled with the WCUTIL program and have a .WCH extension. If you use the Windows 3 help engine, the Help file must be compiled with the help compiler provided with the Windows 3 SDK and must have a .HLP extension. WinComm will determine which type of help to provide based on this extension.

WinComm will use the following context codes when the macro programmer chooses to use the Windows 3 help program. Keyboard = 10, Commands = 20, Procedures = 30. Other context codes are established by the sequence of an item on a custom menu bar as described below and by the ID assigned using the dialog box statement HELPID. These context codes are related to help topics in the [MAP] section of the Windows 3 help job file.

The help topic displayed when the F1 key is pressed is determined by a highlighted custom menu or the currently displayed custom dialog box. The help "index" is based on the sequence of the help text in the help file. To use WinComm's internal help, the Help file is created using any text editor and is organized as a sequence of help "entries". Help "entries" consist of topics and the associated text that accompanies each topic. The topic consists of up to 40 characters terminated with a CR\LF. The text consists of up to 1024 characters terminated with a CR\LF. The first entry in the Help file will be displayed when the F1 key is pressed with no custom menu selected or no custom dialog box displayed, or when the Help item is selected on the custom menu bar. The topic of this first entry is usually an index or a help on the Help topic. The second entry in the Help file describes the top menu selection on the left most menu bar item of the custom menu bar; the next topic for the second selection on the left most menu bar item; etc. The help entry sequence follows for all menu selections top to bottom and then for each menu bar item left to right. Entries in the Help file following the menu item entries correspond to custom dialog boxes. The position number of the entry in the Help file is used as the argument in the HELPID dialog box group statement to identify the help text and topic to be displayed when help is requested for that dialog box.

After the text file has been created for the custom help, it is "compiled" using the Create Help! menu item of the WCUTIL program. Also, see information on the HELPID macro statement and the section of the manual describing the WCUTIL program. The compiled Help file must be in the same directory as WINCOMM.EXE

Example:

For this Custom Menu Bar . . .

File	**Edit**	**Options**
Open	Copy	Start

Close	Paste	Stop
About		Reset

the help file would be organized this way, followed by two help entries for dialog boxes:

```
General Help\                                   <<Topic for the first entry
General Help                                    <<Text for the first entry
Text for the general help topic.\               <<End of text for first entry
File Open\                                      <<Topic for the second entry
File Open                                          <<Text for the second entry
Text for the File Open topic.\                  <<End of text for second entry
File Close\
File Close
This is the text for the File Close topic.\
File About\
File About
This is the text for the File About topic.\
Edit Copy\
Edit Copy
This is the text for the Edit Copy topic.\
Edit Paste\
Edit Paste
This is the text for the Edit Paste topic.\
Options Start\
Options Start
This is the text for the Options Start topic.\
Options Stop\
Options Stop
This is the text for the Options Stop topic.\
Options Reset\
Options Reset
This is the text for the Options Reset topic.\
Dialog Box WUD\
Dialog Box WUD
This is the text for the help that describes the operation of the WUD dialog box.  The WUD
dialog box HELPID statement would use an ID value of 10 since this is the 10th entry in the
help file.\
Dialog Box BUD\
Dialog Box BUD
This is the text for the help that describes the operation of the BUD dialog box.  The BUD
dialog box HELPID statement would use an ID value of 11 since this is the 11th entry in the
help file.\
```

HOTSPOT(left,top,right,bottom,id) Graphics Menus

Places a mouse selectable "hot spot" on the user graphic.

Returns

Nothing	The variable OBJECT? is assigned the ID value of the selected hot spot.

Arguments

left = Integer	The left side of the hot spot rectangle.
top = Integer	The top of the hot spot rectangle.
right = Integer	The right side of the hot spot rectangle.
bot = Integer	The bottom of the hot spot rectangle.
id = integer	The exclusive ID value assigned to each hot spot for identification when the spot is selected.

NOTES

Hot spots should be created after a graphic has been displayed and destroyed after the graphic has been removed. See DELOBJECT.

If the USERWINDOW sizing option "Display Screen" is selected, the user window will be a fixed size and the units used for left, right, top and bottom are given as a percentage (%) of the total display screen. The ID value in this case must be less than 1000.

If the USERWINDOW sizing option "WinComm Screen" is selected, the user window will change sizes when WinComm changes sizes. In this case, the user window is divided into 1000 horizontal and 1000 vertical units, and the positions of the hot spot rectangle sides are calculated as a ratio in proportion to it. For example, if the hot spot were to be positioned 1/4 of the way down and 1/4 of the way over (to the right of the left edge), left and top would equal 250. If the hot spot were to be 10% of the user window size, the right and bottom values would be 350. The ID value to be assigned for a HOTSPOT placed on a scalable user window must be 1000 or greater. Use the WCUTIL WinComm Utility program to create the Graphic Customizing and Graphic Menu Statements.

IF(logical_expression) Macro Evaluation

Used to make branching decisions based on values of input variables.

Returns

Nothing

Arguments

logical_expression A valid logical expression that tests for the result on which to branch.

NOTES

If logical_expression is TRUE, the statement immediately to the right of the IF statement is executed, and execution proceeds with the line following the IF statement. If logical_expression is FALSE, execution proceeds with the line immediately following the IF statement. This operation can be modified by using {}, the ELSE statement and combinations of the two.

The {} can be used to form statement groups that allow more than one statement to be executed when the IF evaluation is TRUE.

Example:

```
IF(information == FALSE){
        PRINT("Cancel")
        GOTO main
        }
```

In the example above, if information != (meaning not equal to) is FALSE, execution would continue after the }.

If the ELSE statement immediately follows the IF statement, statements following the ELSE will not be executed when IF returns TRUE.

Example:

```
IF(i > 10) PRINT("OK")
ELSE PRINT("Not Enough")
i = i+1
```

This example would:
Print OK if i is greater than 10.
Print Not Enough if i is equal to or less than 10.
Add 1 to i in both cases.
The {} can also be used to group more than one statement following ELSE.

Example:

```
IF(i > 10) PRINT("OK")
ELSE{
        PRINT("Not Enough")
        i = i+1
}
```

This example would:
Print OK if i is greater than 10 and skip the bracketed statements following ELSE.
Print Not Enough if i is equal to or less than 10 and add one to i.
Complex statement groups can be created by nesting IF and ELSE statements using {}. IF statements can be nested up to 8 levels deep. The following example demonstrates a receive packet processing loop:

Example

```
IF(rec_active == TRUE){
        DO status = RXPKT(command,count,rec_str$) UNTIL(status > 0)
        IF(status == 1){
                        IF(command == 10) GOTO id_prosc
                        ELSE IF(command == 11) GOTO name_prosc
                        ELSE IF(command == 12) GOTO password_prosc
                        ELSE IF(command == 13 ) GOTO xfer_prosc
        }
        ELSE IF(status == 2) GOTO timeout
        ELSE GOTO canceled
}
skip:
```

In the preceding example:
If rec_active were FALSE, all statements would be skipped and execution would proceed after the lable skip.
If rec_active were TRUE, the DO UNTIL loop would execute until RXPKT returned a value greater than 0.
If RXPKT returns one, indicating a packet received, command is tested and execution branches to process the command.
If RXPKT returns greater than one, there is an error and the final ELSE IF and ELSE statements branch to handle the two error conditions.

INPUTDLG(input_text,heading_text) Stock Dialog Box

Displays a stock dialog box for text input.

Returns

Integer	0 if the Cancel button was pressed or 1 if the OK button was pressed.

Arguments

input_text = String	The text that was typed into the dialog box.
heading_text = String	A text string that is displayed at the top of the dialog box.

INTTIME(hour,day,month,year) Conversion

Converts times and dates to the WinComm integer time/day value.

Returns

Integer	The WinComm integer time/day value of the arguments.

Arguments

hour = Integer	The hour to convert in the range 0 to 23.
day = Integer	The day to convert in the range 1 to 31.
month = Integer	The month to convert in the range 1 to 12.
year = Integer	The number of years since 1900.

LEFT(text,num_characters) Text

Returns a number of characters from the left end of a string.

Returns

String The left most number of characters of the text string.

Arguments

text = String The text to evaluate.

num_characters = integerThe number of characters to be returned from the left end of the text.

LEN(text) Text

Returns the length in number of characters of a string.

Returns

Integer	The integer value that is the number of characters in the string.

Arguments

text = String	The string to evaluate.

LOWERCASE(text)Text

Converts text to lower case.

Returns

String	A copy of the argument string in all lower case characters.

Arguments

text = String	The string to convert to lower case.

LPRINT(text) Text

Sends a line of text to the printer.

Returns

Nothing

Arguments

Text = String	The text to be sent to the printer.

NOTES

To send text to the printer from a macro, turn the printer on using the PRINTER statement and use LPRINT to send the text as needed. When printing is complete, turn the printer off with the PRINTER statement.

MACROHALT(greyed) Macro Control

Used to disable the Macro check box on the WinComm command bar.

Returns

Nothing

Arguments

greyed = Logical TRUE enables, FALSE disables the control.

NOTES

This statement is used to disable the Macro control on the command bar to prevent the operator from halting the macro using this means. This should be used in cases where the macro might need to clean up, close files etc., before halting.

MACROTRAP(enabled) Macro Control

Guarantees that all characters received on the communication port will be processed by the macro.

Returns

Nothing

Arguments

enabled = Integer 1 for trap on, 0 for trap off.

NOTE

When MACROTRAP is enabled, terminal emulation is suspended and all characters received on the communication port can be processed by NEXTCHAR. MACROTRAP should also be enabled when the RXPKT packet processing statements are in use.

MENU? Menu Customizing

The event status statement that returns the value of a selected menu item.

Returns

Integer	The ID value of the menu selected from a custom menu bar. The ID value is assigned by the ADDCOMMAND statement.

NOTE

See ADDBAR, ADDCOMMAND, ADDMENU and SHOWBAR.

MENUBAR? Menu Customizing

System status statement that returns an identifier for the current menu bar.

Returns

Integer	The ID of the currently displayed menu bar.

Arguments

None

NOTES

Used to obtain an ID to identify the current menu bar. This ID can be used by the ADDCOMMAND, ADDMENU and SHOWBAR statements to modify and re-display the current menu bar, or to redisplay a previous menu bar.

METABKG(horz_poz,vert_poz,mapping,file_name)Graphics Customizing

Displays a metafile background in the user Window.

Returns

Nothing

Arguments

horz_pos = Integer	Specifies the horizontal position of the metafile within the user window. 0 = Centered, 1 = left 2 = right.
vert_pos = Integer	Specifies the vertical position of the metafile within the user window. 0 = Centered, 1 = top 2 = bottom.
mapping = Integer	Specifies whether the height to width ratio will be maintained when the metafile is displayed. 0 = no, the metafile will totally fill the user window; 1 = yes, the height to width ratio of the metafile will be maintained.
file_name = String	The proper DOS file name (.WMF) of the metafile to be displayed. If no path is given, WinComm will look for the file in the default directory.

NOTES

A user window must first be created using the USERWINDOW statement before a background graphic can be displayed. The statements BITMAP, BUTTON, METAFILE and HOTSPOT create objects that can be positioned on the background graphic within the user window. The BITMAP, BUTTON and HOTSPOT objects can be given an ID value to test for their selection using the event status statement OBJECT?. BITMAP and BUTTON can also have accelerator keys assigned by placing a ampersand (&) in front of the character in the "text" field to use as the accelerator. The object can then be selected by pressing Alt+the accelerator key.

Use the WCUTIL WinComm Utility program to create the Graphic Customizing and Graphic Menu Statements.

METAFILE(left,top,right,bottom,file_name) Graphics Menus

Places a Windows metafile on the background graphic within the user window.

Returns

Nothing

Arguments

left = Integer	The left side of the metafile rectangle.
top = Integer	The top of the metafile rectangle.
right = Integer	The right side of the metafile rectangle.
bot = Integer	The bottom of the metafile rectangle.
file_name = String	The proper DOS file name (.WMF) of the metafile to be displayed. If no path is given, WinComm will look for the file in the default directory.

MID(text,first_character,num_of_chars) Text

Returns characters from the middle portion of a string.

Returns

String	The middle number of characters from the text string.

Arguments

text = String	The text to evaluate.
first_character = Integer	The position beginning at the left end of the string of the first character to return.
num_of_chars = Integer	The number of characters to return from the string.

MKDIR(directory_name) DOS File Control

Creates a new disk directory.

Returns

Integer	-1 if error, 0 if OK.

Arguments

directory_name = String	The valid DOS directory name of the directory to be created.

NEW

Loads default WinComm session settings.

Returns

Nothing

Arguments

None

WinComm File Control

NEXTCHAR? Text

Returns the next character in the receive buffer.

Returns

String	The next character in the receive buffer. If the receive buffer is empty, returns a null.

Arguments

None

NOTE

For NEXTCHAR to operate, the MACROTRAP statement must be enabled.

NULL(text) Text

Tests to see if a string is null.

Returns

Logical	TRUE if argument is null, FALSE otherwise.

Arguments

text = String	The string to be evaluated.

OBJECT? Graphics Menus

Returns a value indicating the selection of an object on a graphic menu.

Returns

Integer	The ID of the selected object on a graphic menu. See HOTSPOT, BUTTON and BITMAP.

Arguments

None

ONLINE? WinComm Information

System status statement used to determine if a WinComm session has been started.

Returns

Integer	0 = Not connected - in process, 1 = Connected Local, 2 = Connected Modem, 3 = Error.

Arguments

None

OPEN(session_file_name,pswrd,mode)WinComm File Control

Opens a WinComm session file.

Returns

Integer	0 = OK, 1 = Session Already Running, 2 = Not Enough Memory, 3 = File Error, 4 = Incorrect Password.

Arguments

session_file_name = String	The name of the session file to open.
pswrd = String	The password string if the file is protected, otherwise null
mode = Integer	0 = Originate mode, 1 = Originate mode do not execute automatics, 2 = Answer mode, 3 = Answer mode do not execute automatics.

NOTE

Automatics include "Connect" and "Run Macro" on session open. "Originate" mode will send the modem initialization strings and dial the telephone number when the session is started. "Answer" mode will send the modem initialization strings and the answer mode string when the session is started.

PASTETEXT Clipboard Control

Transmits and displays the text in the Clipboard.

Returns

Integer	-1 if there is no text in the Clipboard, 0 if OK.

Arguments

None

PCTOPC(cmd,opt,text) WinComm/WinLink Remote Control

Command used to control file operations of another computer running WinLink using a serial comm link.

Returns

Integer	0 if OK, greater than 1 = error.

Arguments

cmd = 0	**Command to receive a file**
opt = Integer	A value greater than 0 displays the file transfer progress dialog box that allows canceling the transfer, otherwise it will not display.
text = String	The name of the file to receive.
cmd = 1	**Command to send a file**
opt = Integer	A value greater than 0 displays the file transfer progress dialog box that allows canceling the transfer, otherwise it will not display.
text = String	The name of the file to send.
cmd = 2	**Command to return a file date/time code**
opt = Integer	Assigned the value of the creation or last modified date/time code.
text = String	The name of the file in the remote computer.
cmd =3	**Command to return a file size**
opt = Integer	Assigned the value of the file size.
text = String	The name of the file in the remote computer.
cmd = 4	**Command to change drive or directory**
opt = null	Not used in this command.
text = String	The name of the drive or directory (in the remote computer) to which to change.
cmd = 5	**Command to get the current path**
opt = null	Not used in this command.
text = String	Assigned the full DOS path name of the current directory in the remote computer.
cmd = 6	**Command to rename a file**
opt = null	Not used in this command.
text = String	A comma-separated string of the form "old_file_name,new_file_name".
cmd = 7	**Command to delete a file**
opt = null	Not used in this command.

text = String	The name of the file to be deleted in the remote computer.
cmd = 8	**Command to create a directory**
opt = null	Not used in this command.
text = String	The directory name to be created in the remote computer. The directory will be created as a sub-directory of the current directory.
cmd = 9	**Command to delete a directory**
opt = null	Not used in this command.
text = String	The name of the directory to be deleted in the remote computer. The directory must be in the current directory.
cmd = 10	**Command to establish a file search filter**
opt = null	Not used in this command.
text = String	A file name or wild card search filter to be used with command 13, find next file.
cmd = 11	**Command to get the next drive letter**
opt = null	Not used in this command.
text = String	The drive letter of the next disk drive in the remote computer. To determine all drives in the remote computer, repeat this command until the drive letters repeat.
cmd = 12	**Command to get the next sub-directory**
opt = null	Not used in this command.
text = String	The name of the next sub-directory in the current directory of the remote computer. To determine all sub-directories in this directory, repeat this command until the names repeat.
cmd = 13	**Command to get a file name**
opt = null	Not used in this command.
text = String	Returns the name of the next file that meets the search criteria set up with command 10.

PEND — Macro Control

Ends a PROMPT branching statement group. See PROMPT statement.

Returns

Nothing

Arguments

None

PKTIME(trans_time,rec_time)WinComm Packet Data Transfer

Used to put WinComm in Packet Mode and to set timings for packet transmission and reception.

Returns

Integer	1 if OK, 0 if error. An error is usually due to insufficient memory for setting up packet transfer.

Arguments

trans_time = Integer	The number of milliseconds allowed for the TXPKT statement to attempt to deliver a packet to the receiver before returning to the macro with an error (timeout). During this time, TXPKT will attempt to transmit the packet 3 times.
rec_time = Integer	The number of milliseconds allowed the RXPKT statement to wait before notifying the macro of not receiving a good packet (timeout).

NOTE

If trans_time or rec_time are set to 0, packet transfers are terminated and communication processing returns to normal. See also TXPKT, RXPKT, TXPKTSTAT?.

PRINT(text) Text

Places the characters in the WinComm screen text area at the current cursor position.

Returns

Nothing

Arguments

text = String	The text that will be displayed in the WinComm display text area at the current cursor position.

PRINTER(cont) WinComm Command

Turns the printer off and on.

Returns

Nothing

Arguments

cont = Logical TRUE = On, FALSE = Off

NOTES

To send text to the printer from a macro, turn the printer on using the PRINTER statement and use LPRINT to send the text as needed. When printing is complete, turn the printer off with the PRINTER statement.

PROMPT and PEND and PROMPT? Macro Control

Allows testing of communication input strings, keyboard characters and time used to control the flow of a macro.

Returns

Nothing

The event status statement PROMPT? is set to the ID Value assigned by one of the Prompt Condition statements within a PROMPT statement group.

Arguments

None

PROMPT and PEND enclose a group of Prompt Condition statements which are described below. These statements are used to alter the execution of a macro based on the following: 1) receipt of a matching string, 2) receipt of a certain number of characters, 3) quiet time on the communication line, 4) the status of the modem data carrier detect line, 5) a character typed on the keyboard, and 6) the passing of a given amount of time. When a PROMPT statement group is encountered in a macro, the tests established by the prompt condition statements are begun. The execution of the macro continues with the statement following the PEND. When a condition of one of the statements is met, the value of PROMPT? is assigned the ID value of that statement, and the tests established by the PROMPT statement group terminate. It is then up to the macro code to test the value of PROMPT? to determine which condition has been met within the statement group. To reestablish the test, the PROMPT statement group must be re-executed.

Prompt Condition Statements

Sixteen prompt condition statements can be in used one group. The following statements are used in a PROMPT statement group:

PCOUNT(id,#char)

Tests for a given number of characters received on the communication port.

id = Integer	The exclusive value to give this prompt condition statement.
#char = Integer	The number of characters to be received on the communication port before being notified.

PDCD(id)

Notifies when Data Carrier Detect goes FALSE.

id = Integer	The exclusive value to give this prompt condition statement.

PKEY(id,char_code)

Tests for a character typed on the keyboard.

id = Integer	The exclusive value to give this prompt condition statement.
char_code = Integer	The ASCII decimal character code of the key you designate to be notified when it is typed.

PQUIET(id,10ths_sec)

Tests for a quiet time on the communication port.

id = Integer	The exclusive value to give this prompt condition statement.
10ths_sec = Integer	The amount of time, in tenths of seconds, during which no activity is to take place on the communication port before being notified.

PSTR(match_type,id,char_string)

Tests for a given string received on the communication port.

match_type = 0	Case insensitive, Strip white space
match_type = 1	Case sensitive, Strip white space
match_type = 2	Case insensitive, Don't strip white space
match_type = 3	Case sensitive, Don't strip white space
id = Integer	The exclusive value to give this prompt condition statement.
char_string = String	The string for which to test.

Note: White Space is defined as -- Vertical Tab, Horizontal Tab, Line Feed, Form Feed, Carriage Return and Space. The following conventions can be used within the char_string for "wild card" character matching:

^"character" = "character" with the Ctrl key pressed

"~_" Matches any white space character

"~A" Matches any upper case character A through Z

"~a" Matches any lower case character a through z

"~#" Matches any number 0 through 9

"~X" Matches any number 0 through 9 or any character a through z case insensitive

"~?" Matches any character a through z case insensitive

"~~" Matches a ~ character

PWAIT(id,10ths_sec)

Waits for a given amount of time.

id = Integer	The exclusive value to give this prompt condition statement.
10ths_sec = Integer	The amount, of time in tenths of seconds, to pass before being notified.

PUTGLOBALINT(integer_number,value)Macro Variable Control

Passes numbers to macros that are run using the CHAIN statement.

Returns

Nothing

Arguments

integer_number = Integer0 through 7, the ID assigned each global number.

value = Integer The variable to be passed to the chained macro.

NOTES

See CHAIN, CHAINRETURN, GETGOLBALSTR and GETGLOBALINT statements.

PUTGLOBALSTR(string_number,text)Macro Variable Control

Passes strings to macros that are run using the CHAIN statement.

Returns

Nothing

Arguments

string_number = Integer 0 or 1, the ID assigned each global string.

text = String The variable to pass to the chained macro.

NOTES

See CHAIN, CHAINRETURN, GETGOLBALSTR and GETGLOBALINT statements.

PUTSESSINT(int_id,int_value) Session Variable Control

Assigns values to any session integers.

Returns

Nothing

Arguments

int_id = Integer	The ID number of the integer session variable.
int_value = Integer	The value to assign to this session integer.

NOTES

To retrieve the values assigned the session integers, use GETSESSINT. For a list of all session integer ID numbers see GETSESSINT.

PUTSESSSTR(string_id,string_value)Session Variable Control

Assigns values to any session strings.

Returns

Nothing

Arguments

string_id = Integer	The ID number of the string session variable.
string_value = String	The value to assign to this session string.

NOTES

To retrieve the values assigned the session strings, use GETSESSTR. For a list of all session strings ID numbers, see GETSESSSTR.

QUOTE(text) Text

Puts a string in quotation marks.

Returns

String	The argument string enclosed in double quotation marks.

Arguments

text = String	The string to have quoted.

RECEIVEASCII(diag_disp,file_name)WinComm File Transfer

Sets WinComm up to receive an ASCII file.

Returns

Nothing	The variable TRANSFER? is set to 1 while the transfer is in process and 0 when it is finished.

Arguments

diag_disp		If TRUE, displays the File Transfer Progress dialog box.
file_name	String	The name to give the file received over the communication port.

NOTES

The file is stored in the session receive file directory.

RECEIVEFILE(diag_disp,file_name)WinComm Protocol File Transfer

Sets WinComm up to receive a file using an error-correcting protocol.

Returns

Nothing	The event status statement XFER? is set to indicate the status of the transfer.

Arguments

diag_disp		If TRUE, displays the File Transfer Progress dialog box.
file_name	String	The name to give the file received over the communication port.

NOTES

The protocol used is the one designated in the current session. The file is stored in the session receive file directory.

RENAME(old_file_name,new_file_name) DOS File Control

Renames a file.

Returns

Integer	0 if OK, 1 if Error.

Arguments

old_file_name = String	The file to be renamed.
new_file_name = String	The new file name.

NOTES

Error can be caused by an existing file with the same name as new_file_name, not being able to find old_file_name or an invalid path.

RESTOREVARS Macro Variable Control

Used to restore the currently running macro variables after returning from another macro.

Returns

Nothing

Arguments

None

NOTES

Re-establishes all variables just as though the chain had never occurred. The SAVEVARS command must have been the last statement executed before the CHAIN command that transferred to another macro. See SAVEVARS, CHAIN, CHAINRETURN, GETGOLBALSTR and GETGLOBALINT statements.

RETURN — Macro Control

Used to denote the logical end of a subroutine.

Returns

Nothing

Arguments

None

NOTES

The execution of the macro returns to the point immediately following the GOSUB statement that called the subroutine.

RIGHT(text,num_of_chars) Text

Returns a number of characters from the right end of the string.

Returns

String The right most number of characters of the text string.

Arguments

text = String The text to evaluate.

num_of_chars = integer The number of characters to be returned from the right end of the text.

RMDIR(dir_name) DOS File Control

Removes a directory.

Returns

Integer	-1 if error, 0 if OK.

Arguments

dir_name = String	The name of the directory to be removed.

RSTACK — Macro Control

Used to reset the subroutine stack pointer.

Returns

Nothing

Arguments

None

NOTES

Each time a GOSUB statement is encountered in macro execution, the address is "pushed" on the subroutine stack. This address is where execution proceeds when a RETURN is encountered in the subroutine. Execution then proceeds to the label in the GOSUB statement. This stack is 8 levels deep. RSTACK resets the stack as if there had been no GOSUB statements. The RSTACK statement should be followed by a GOTO statement that would in essence start the macro over. It may be used in cases where the operator might want to "recycle" a macro after an error has occured. Returning the error back through several levels of subroutine nesting is unnessarily complicated.

RUN(app_name,command_line,size)Windows Application Control

Used to start and send command line information to another Windows application.

Returns

Integer	A value less than 32 indicates an error. A Value of 32 or greater indicates the application is running. This number is the Windows module instance number which can be used in the DDEINIT statment to initiate DDE activity with the specific instance of an application.

Arguments

app_name = String	The name of the Windows application which is to be run. If the application is not in the WinComm macro directory, a complete path should be given.
command_line = String	Text that will be sent to the application when it is started, generally the name of the document the application is to load.
size = Integer	Howthe application is to appear upon loading. 3 = Maximized, 1 = Restored or Normal,2 = Minimized, 0 = Hidden.

NOTES

May be used in conjunction with the SENDKEYS and SENDSPECIALKEYS as well as the DDE commands.

RXPKT(command,count,rec_str)WinComm Packet Data Transfer

Receives an error-free string of data from another system which is running a macro using the TXPKT statements.

Returns

Integer	A value that represents the status of the packet reception. The values are 0 = waiting, 1 = have received an error free packet, 2 = time out and 3 = cancelled by remote.

Arguments

command = Integer	The integer value that was given to this packet when the packet was sent, 0 - 255.
count = Integer Variable	The variable is assigned the number of characters in the packet.
rec_str = String	The text that was sent by TXPKT, 0 - 255 characters.

NOTES

The time out value for RXPKT is set using PKTIME. See also TXPKT.

SAVEAS(file_name) WinComm File Control

Saves the current session file as file_name.

Returns

Integer	-1 indicates error, 0 is ok.

Arguments

file_name = String	The name used for saving the session file.

SAVEASDLG(heading_text,def_filename) Stock Dialog Box

Displays a stock dialog box which is used to name files before they are saved.

Returns

Logical	FALSE if the Cancel button was pressed or if the file could not be saved (due to an existing file of the same name or not enough disk space), TRUE if the OK button was pressed and the file was sucessfully saved.

Arguments

def_filename = string	A string that is displayed in the edit box giving the default filename. This name can be changed by the normal Windows edit procedures when the dialog box is shown.

SAVEVARS Macro Variable Control

Used to save all currently running macro variables for chaining to and returning from another macro.

Returns

Nothing

Arguments

None

NOTES

These variables are restored with the RESTOREVARS statement to re-establish the environment upon returning from the chained macro. To pass variables to and from the chained macro use PUTGLOBALINT, PUTGLOBALSTR, GETGLOBALINT and GETGLOBALSTR statements.

SEARCH(substr,str) Text

Used to find the number of occurrences of a substring within a string.

Returns

Integer	The number of times substring occurs in str.

Arguments

substr = String	The sub-string to find in str.
str = String	The string to search.

SEND(text) — Macro Control

Transmits text to the communication port.

Returns

Nothing

Arguments

text = String	The string to transmit.

SENDASCII(diag_disp,file_name) WinComm File Transfer

Sends an ASCII file using the settings assigned in the current session.

Returns

Nothing	The variable XFER? is assigned the status of the SENDASCII command.

Arguments

diag_disp = Logical	If TRUE, displays the file transfer progress dialog box.
file_name = String	The name of the text file to be transmitted.

SENDFILE(diag_disp,file_name)WinComm Protocol File Transfer

Sends a file using an error-correcting protocol assigned in the loaded session.

Returns

Nothing	The status statement XFER? is assigned the status of the SENDFILE command.

Arguments

diag_disp	If TRUE, displays the File Transfer Progress dialog box.
file_name = String	The name of the file to be transmitted.

NOTES

The protocol is the one assigned in the current session variable. If no path is given with the file name, WinComm will look for the file in the DOWNLOAD directory.

SENDKEY(key_text) Windows Application Control

Sends normal ASCII character keystrokes to the active Windows application.

Returns

Nothing

Arguments

key_text = String	The text that is sent to the active application just as if it had been typed on the keyboard.

NOTES

SENDKEYS and SENDSPECKEYS are two powerful statements that can be used to create batch files for controlling other Windows applications.

SENDSPECKEY(ctrl,key1_code,key2_code)Windows Application Control

Sends non-printing Alt, Ctl, Shift and combinations of non-printing and printing characters to the active Windows application.

Returns

Nothing

Arguments

ctrl = Integer — The value of the special key used to modify the other keys in the argument list. Use a code from the following list to duplicate sending key_code with the following keys pressed:

None	0
Alt Key	1
Ctl Key	2
Shift Key	3
Ctrl and Shift Keys	4

key1_code = Integer — The decimal value of the key to send to the active application.

key2_code = Integer — The decimal value of the key to send to the active application.

NOTES

SENDKEYS and SENDSPECKEYS are two powerful statements that can be used to create batch files used for controlling other Windows applications.

The table below lists the values to use for the key_code arguments:

Key	Value	Key	Value	Key	Value	Key	Value
LBUTTON	1	DOWN	40	E	69	Z	90
RBUTTON	2	SELECT	41	F	70	MULTIPLY	106
CANCEL	3	PRINT	42	G	71	ADD	107
BACK	8	EXECUTE	43	H	72	SEPARATOR	108
TAB	9	INSERT	44	I	73	SUBTRACT	109
CLEAR	12	DELETE	45	J	74	DECIMAL	110
RETURN	13	HELP	46	K	75	DIVIDE	111
SHIFT	16	0	48	L	76	F1	112
CONTROL	17	1	49	M	77	F2	113
MENU	18	2	50	N	78	F3	114
PAUSE	19	3	51	O	79	F4	115
CAPITAL	20	4	52	P	80	F5	116
ESCAPE	27	5	53	Q	81	F6	117
SPACE	32	6	54	R	82	F7	118
PRIOR	33	7	55	S	83	F8	119
NEXT	34	8	56	T	84	F9	120
END	35	9	57	U	85	F10	121
HOME	36	A	65	V	86	F11	122
LEFT	37	B	66	W	87	F12	123
UP	38	C	67	X	88		
RIGHT	39	D	68	Y	89		

SETFILEATTR(attribute,file_name) DOS File Control

Sets a file attribute.

Returns

Integer	-1 if error, 0 if OK.

Arguments

attribute = Integer	The byte wise "or" of the DOS file attribute bits to be set for this file.
file_name = String	The valid file name of an existing file.

NOTES

See the table under FILEFIND for the attribute bit assignments.

SETFILEDATE(date_time,file_name) DOS File Control

Sets a file date and time.

Returns

Integer	-1 if error, 0 if ok.

Arguments

date_time = Integer	The WinComm integer date time code to set for the file.
file_name = String	The valid file name of an existing file.

SETFOCUS(id) Windows Application Control

Sets the focus to a given control within a dialog box.

Returns

Nothing

Arguments

id = Integer	The ID of the control within the dialog box to which the focus will be set. This value must be obtained using the GETFOCUS statement.

NOTES

This statement is generally used when "Sending Keys" to WinComm or to another application. The ID value is obtained by "tabbing" to a given control then getting its ID value using GETFOCUS. The value returned can then be used to set the focus back to this particular control using this statement.

SHOW(change) WinComm Command

Used to adjust the WinComm window and "repaint" after changes are made to graphic display areas.

Returns

Nothing

Arguments

change = Integer

The following table lists the change values used with the SHOW statement and the effect they have on the WinComm window:

Change Value	Effect
1	Maximizes the WinComm window
2	Minimizes the WinComm window
3	Restores the WinComm window
4	Re-display WinComm window after hidden
5	Hide the WinComm window
6	Repaint the User Window

NOTE

To move or change the size of the WinComm window, use the MOVE statement.

SHOWBAR(bar_number) Menu Customizing

Displays a new or changed menu bar.

Returns

Nothing

Arguments

bar_number = integer The ID of the menu bar for WinComm to display.

NOTES

The ID is returned from the ADDBAR or MENUBAR? statements.

START WinComm Command

Starts the currently loaded session.

Returns

Nothing	CONNECT? is set to a value indicating the status of the connection.

Arguments

None

STATUSLINE(status_text) WinComm Command

Places text in the status line.

Returns

Nothing

Arguments

status_text = String The text to display in the status line.

STATUSLINE? WinComm Information

Returns the text that is displayed on the WinComm status line.

Returns

String	The text displayed on the status line.

Arguments

None

STEP Macro Control

Used by the compiler in debug mode to break execution at run time and enter single step mode.

Returns

Nothing

Arguments

None

NOTE

If a macro is compiled with the ^Compile With Debug option set, information is added to the macro code that will allow "single stepping" of the macro at run time, when a STEP statement is encountered.

STOP WinComm Command

Stops the current session, terminates the phone call or connection.

Returns

Nothing

Arguments

None

STRBIN(number) Conversion

Changes an integer number into a binary string representation of the number.

Returns

String	ASCII 0 and 1 binary representation of the argument.

Arguments

number = Integer	The number to be returned as a binary string.

STRHEX(number) Conversion

Changes an integer number into a hex decimal string representation of the number.

Returns

String	The ASCII hex decimal representation of the argument.

Arguments

number = Integer	The number to be returned as a hex decimal string.

STRINT(number) Conversion

Changes an integer number into a base 10 string representation of the number.

Returns

String	The ASCII base 10 representation of the argument.

Arguments

number = Integer	The number to be returned as a integer string.

STROCT(number) Conversion

Changes an integer number into an octal string representation of the number.

Returns

String	The ASCII octal representation of the argument.

Arguments

number = Integer	The number to be returned as a octal string.

STRTIME(time) Conversion

Converts an integer system time value into a Mon Sep 25 17:23:16 1989 type string.

Returns

String	Of the form Mon Sep 25 17:23:16 1989.

Arguments

time = Integer	Time in the system integer format.

NOTES

The integer time/date value is the number of seconds between midnight Jan. 1, 1980 and the time to be represented.

SUBST(find_text,orig_text,replace_text,times) Text

Substitutes text for other text in a string.

Returns

Integer	The number of times find_text was replaced

Arguments

find_text = String	The sub-string to replace.
orig_text = String	The original text string.
replace_text = String	The sub-string to replace find_text.
times = Integer	The number of find_text instances to be replaced, beginning at the left end of the string.

NOTES

After execution of the SUBST statement, orig_text is changed to reflect the substitution(s).

SYSTEM Windows Information/File Control

Saves the system information in the WIN.INI file.

Returns

Nothing

Arguments

None

NOTE

This statement is used to save the changes made in the session string variable list for ID numbers 25 through 30. These changes will be written to the WIN.INI file when the SYSTEM statement is executed.

TEXTAREA(cols,rows) Graphics Customizing

Creates a text area in the WinComm window to prevent graphics from covering displayed text.

Returns

Nothing

Arguments

cols = Integer	The minimum number of columns to be visible in the text area.
rows = Integer	The minimum number of rows to be visible in the text area.

NOTE

By default, the text area is zero rows and zero columns.

TIME? System Information

Gets the current system time in the system integer format.

Returns

Integer	The number of seconds that have elapsed since midnight Jan 1, 1970 and the current system time.

Arguments

None

TXPKT(command,count,xmit_str)WinComm Packet Data Transfer

Sends an error-free string of data to another computer running a macro using the RXPKT statement.

Returns

Nothing	The event status statement TXPKTSTAT? is assigned a value indicating the status of the packet transfer.

Arguments

command = Integer	The integer value identifying this packet of data.
count = Integer	Must be set to the number of characters in xmit_str
xmit_str = String	The text that will be sent by TXPKT.

NOTES

TXPKT will attempt to send the data 3 times equally spaced during the time out period established by PKTIME. Also see RXPKT. These statements allow the macro programer to send error free commands from one computer to another, as well as sending binary data. For TXPKT to operate properly, the communication port must be set to 8 bits, no parity and no handshaking.

TXPKTSTAT? WinComm Packet Data Transfer

The event status statement that indicates the status of a TXPKT data transfer.

Returns

Integer	A number that represents the status of the packet transfer: 0 = waiting, 1 = have successfully sent an error-free packet, 2 = time out, 3 = cancelled by remote.

Arguments

None

UNTIL(logical_expression) Macro Evaluation

Ends a DO loop statement group.

Returns

Nothing

Arguments

logical_expression Any valid logical expression that is FALSE while the macro is in the loop.

NOTES

If the logical_expression is TRUE, macro execution passes to the next statement. If logical_expression is FALSE, execution returns to the preceding DO statement.

UPPERCASE(text) Text

Returns a string as all uppercase characters.

Returns

String	A copy of the argument string converted to all upper case characters.

Arguments

text = String	The string to be converted to upper case characters.

USERWINDOW(pos,size_ref,size,bkg_color)Graphics Customizing

Used to define an area of the WinComm window for display of graphics.

Returns

Nothing

Arguments

pos = Integer — The position of the user window within the WinComm window:

0 = destroy or remove the window
1 = position the window at WinComm's left side
2 = position the window at WinComm's right side
3 = position the window at WinComm's top
4 = position the window at WinComm's bottom
5 = fill the WinComm window

size_ref = Integer — Specifies how the user window will be sized.

0 = size the window relative to WinComm's window. The user window size will change when WinComm's window size changes.

1 = size the window relative to the screen. The user window size will be fixed and may extend outside WinComm's window, if it does the window will be "clipped".

size = Integer — Specifies the percentage of the WinComm window (size_ref = 0) or the screen (size_ref = 1) the graphic will occupy.

bkg_color = Integer — The decimal integer value of the red, green and blue intensities for the background color. The background color fills the user window. Each color has a range of 0 (no color) to 255 (maximum intensity). The decimal color = (red_intensity * 65536) + (green_intensity * 256) + blue_intensity.

NOTE

Use the WinComm utility program to generate the source code for graphics display.

VALBIN(bin_text) Conversion

Converts a string of 1's and 0's in binary form to an integer.

Returns

Integer	The integer value of the binary text string.

Arguments

bin_text = String	A string of 1's and 0's that represent the binary number to convert to an integer.

VALHEX(hex_text) Conversion

Converts a string representing a hexadecimal number to an integer.

Returns

Integer	The integer value of the hexadecimal text string.

Arguments

hex_text = string	A hexadecimal representation of the number to convert to an integer.

VALINT(dec_text) Conversion

Converts a string representing a decimal number to an integer value.

Returns

Integer	The integer value of the decimal text string.

Arguments

dec_text = string	The decimal text representation of the number to convert to an integer.

VALOCT(oct_text) Conversion

Converts a string representing a octal number to an integer value.

Returns

Integer	The integer value of the octal text string.

Arguments

oct_text = string	The octal text representation of the number to convert to an integer.

Arguments

WEND — Macro Evaluation

Used to define the end of a WHILE looping statement group.

Returns

Nothing

Arguments

None

WHILE(logical_expression) Macro Evaluation

Performs a statement or statement group while a condition is true.

Returns

Nothing

Arguments

logical_expression	A valid logical expression that tests for the condition you want to exist while in the WHILE/WEND loop.

NOTE

Statements between WHILE and WEND are executed until the logical_expression in the WHILE statement turns FALSE. When this occurs macro execution proceeds with the statement immediately following the WEND statement.

WINCOMMVER? WinComm Information

Gives the version of WinComm.

Returns

String	The version of WinComm currently running.

Arguments

None

WINMOVE(left,top,width,height) WinComm Command

Sizes and positions the WinComm window.

Returns

Nothing

Arguments

left = Integer	The left position of the WinComm window.
top = Integer	The top position of the WinComm window.
width = integer	The width of the WinComm window.
height = Integer	The height of the WinComm window.

NOTES

The units used for positioning and size are in pixels and are referenced to the upper left corner or the screen. The size of the screen can be obtained using the GETSCRCAP statement.

WINVER? — Windows Information

Gives the version of Windows.

Returns

String	The version of Windows currently running.

Arguments

None

XFER? WinComm File Transfer

Variable used to determine the status of a file transfer.

Returns

Integer	A code from the following table that represents the status of the file transfer:

XFER? Error Code Table:

0	Transfer Complete
1	Transfer in Process
2	Transfer Aborted by Remote
3	Transfer Aborted Timed Out
5	Transfer Aborted Loss of Carrier
6	Transfer Aborted by User
7	Transfer File Already Exists
8	Transfer Aborted Bad Block Number Received
9	Transfer Aborted File Write Error
10	File Creation Error
12	Transfer Aborted Error Count Exceeded
13	Host Not Using True YMODEM
14	Transfer Aborted File Read Error
15	Remote Skipped File
16	Transfer Aborted File Creation Error
18	Transfer Aborted Packet Type Unknown
19	Transfer Aborted File Not Found

Arguments

None

NOTES

This variable is valid during SENDFILE, RECEIVEFILE, SENDASCII and RECEIVEASCII.