WINDOWSLIB

Conversion program

COLLABORATORS

	TITLE : WINDOWSLIB	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Conversion program	October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

WINDOWSLIB

1.1 Overview of WINDOWSLIB

Overview

An Acid Software Library

Converted to AmigaGuide by

Red When Excited Ltd

Used with the permission of Acid Software

Edited, fixed and cleaned by Toby Zuijdveld 27/02/1999.
mailto:hotcakes@abacus.net.au

1.2 WINDOWSLIB

```
Statement/Function: Window
------------------------------------------------------------------------------
Modes  :
Syntax : Window Window#,X,Y,Width,Height,Flags,Title$,Dpen,Bpen[,GadgetList#[, ↩
    BitMap#]]
      suc=Window(Window#,X,Y,Width,Height,Flags,Title$,Dpen,Bpen[,GadgetList#[, ↩
         BitMap#]])

Window opens an Intuition window on the currently used screen. Window#
is a unique object number for the new window. X and Y refer to the
offset from the top left of the screen the window is to appear at.
Width and Height are the size of the window in pixels.

Flags are the special window flags that a window can have when opened.
These flags allow for the inclusion of a sizing gadget, dragbar and
many other things. The flags are listed as followed, with their
corresponding values. To select more than one of these flags, they must
```

be logically Or'd together using the '|' operator.

For example, to open a window with dragbar and sizing gadget which is active once opened, you would specify a Flags parameter of $1|$2|$1000.

Title$ is a BASIC string, either a constant or a variable, that you want to be the title of the window.
Dpen is the colour of the detail pen of the window. This colour is used for the window title.

BPen is the block pen of the window. This pen is used for things like the border around the edge of the window.

The optional GadgetList# is the number of a gadgetlist object you have may want attached to the window.

After the window has opened, it will become the currently used window.

***** IMPORTANT NOTE *****
Before attempting to use the window you must check that it has opened properly. This can be done by either using the Window command as a function:

```
success=Window(...)
```

or by doing

```
success=peek.l(addr window(0))
```

either way will allow you to check that the window opened successfully.

The Window library has been extended to handle super bitmap windows. SuperBitMap windows allow the window to have it's own bitmap which can actually be larger than the window. The two main benefits of this feature are the window's ability to refresh itself and the ability to scroll around a large area "inside" the bitmap.

To attach a BitMap to a Window set the SuperBitMap flag in the flags field and include the BitMap# to be attached.

Window Value Description
WINDOWSIZING $0001 Attaches sizing gadget to bottom right corner of window and allows it to be sized.
WINDOWDRAG $0002 Allows window to be dragged with the mouse by it's title bar.
WINDOWDEPTH $0004 Lets windows be pushed behind or pulled in front of other windows.
WINDOWCLOSE $0008 Attaches a closegadget to the upper left corner of the window.
SIZEBRIGHT $0010 With GIMMEZEROZERO and WINDOWSIZING set, this will leave the right hand margin, the width of the sizing gadget, clear, and any drawing to the window will not extend over this right margin.
SIZEBBOTTOM $0020 Same as SIZEBRIGHT except it leaves a margin at the bottom of the window, the width of the sizing gadget.
BACKDROP $0100 This opens the window behind any other window that is already

opened. It cannot have the WINDOWDEPTH flag set also, as the window is intended to stay behind all others.
GIMMEZEROZERO $0400 This flag keeps the windows border separate from the rest of the windows area. Any drawing on the window, extending to the borders, will not overwrite the border. NOTE: Although convevient, this does take up more memory than usual.
BORDERLESS $0800 Opens a window without any border on it at all.

1.3 WINDOWSLIB

Statement: WaitEvent

Modes :
Syntax : WaitEvent

WaitEvent will halt program excution until an Intuition event has been received. This event must be one that satisfies the IDCMP flags of any open windows. If used as a function, WaitEvent returns the IDCMP flag of the event (please refer to DefaultIDCMP for a table of possible IDCMP flags). If used as a statement, you have no way of telling what event occured.

You may find the window object number that caused the event using the EventWindow function.

In the case of events concerning gadgets or menus, further functions are available to detect which gadget or menu was played with.

In the case of mouse button events, the MButtons function may be used to discover exactly which mouse button has been hit.

IMPORTANT NOTE: If you are assigning the result of WaitEvent to a variable, MAKE SURE that the variable is a long type variable. For example: MyEvent.l=WaitEvent

1.4 WINDOWSLIB

Statement: Event

Modes :
Syntax : Event

Event works similarly to WaitEvent in that it returns the IDCMP flag of any outstanding windows events. However, Event will NOT cause program flow to halt. Instead, if no event has occured, Event will return 0.

1.5 WINDOWSLIB

```
Statement: GadgetHit
------------------------------------------------------------------------------
Modes  :
Syntax : GadgetHit

GadgetHit returns the identification number of the gadget that caused
the most recent 'gadget pushed' or 'gadget released' event.

As gadgets in different windows may possibly posess the same
identification numbers, you may also need to use EventWindow to tell
exactly which gadget was hit.
```

1.6 WINDOWSLIB

```
Statement: EventWindow
------------------------------------------------------------------------------
Modes  :
Syntax : EventWindow

EventWindow may be used to determine in which window the most recent
window event occured. Window events are detected by use of either the
WaitEvent or Event commands.

EventWindow return the window object number in which the most recent
window event occured.
```

1.7 WINDOWSLIB

```
Statement: DefaultIDCMP
------------------------------------------------------------------------------
Modes  :
Syntax : DefaultIDCMP IDCMP_Flags

DefaultIDCMP allows you to set the IDCMP flags used when opening
further windows. You can change the flags as often as you like, causing
all of your windows to have their own set of IDCMP flags if you
wish.

A window's IDCMP flags will affect the types of 'events' reportable by
the window. Events are reported to a program by means of either the
WaitEvent or Event functions.

To select more than one IDCMP Flag when using DefaultIDCMP, combine the
separate flags together using the OR operator ('|').

Any windows opened before any DefaultIDCMP command is executed will be
opened using an IDCMP flags setting of:
$2|$4|$8|$20|$40|$100|$200|$400|$40000|$80000. This should be
```

sufficient for most programs.

If you do use DefaultIDCMP for some reason, it is important to remember
to include all flags necessary for the functioning of the program. For
example, if you open a window which is to have menus attached to it,
you MUST set the $100 (menu selected) IDCMP flag, or else you will have
no way of telling when a menu has been selected.

Here is a table of possible events and their IDCMP flags:

IDCMP FlagEvent
$2 Reported when a window has it's size changed.
$4 Reported when a windows contents have been corrupted. This may mean a windows ↩
   contents may need to be re-drawn.
$8 Reported when either mouse button has been hit.
$10 Reported when the mouse has been moved.
$20 Reported when a gadget within a window has been pushed 'down'.
$40 Reported when a gadget within a window has been 'released'.
$100 Reported when a menu operation within a window has occured.
$200 Reported when the 'close' gadget of a window has been selected.
$400 Reported when a keypress has been detected.
$8000 Reported when a disk is inserted into a disk drive.
$10000 Reported when a disk is removed from a disk drive.
$40000 Reported when a window has been 'activated'.
$80000 Reported when a window has been 'de-activated'.

1.8 WINDOWSLIB

Statement: MenuHit

Modes :
Syntax : MenuHit

MenuHit returns the identification number of the menu that caused the
last menu event. As with gadgets, you can have different menus for
different windows with the same identification number. Therefore you
may also need to use EventWindow to find which window caused the
event.

If no menus have yet been selected, Menuhit will return -1.

1.9 WINDOWSLIB

Statement: ItemHit

Modes :
Syntax : ItemHit

ItemHit returns the identification number of the menu item that caused
the last menu event.

1.10 WINDOWSLIB

```
Statement: SubHit
------------------------------------------------------------------------------
Modes  :
Syntax : SubHit

SubHit returns the identification number of the the menu subitem that
caused the last menu event. If no subitem was selected, SubHit will
return -1.
```

1.11 WINDOWSLIB

```
Statement: WindowFont
------------------------------------------------------------------------------
Modes  :
Syntax : WindowFont IntuiFont# [,SoftStyle]

WindowFont sets the font for the currently used window. Any further
printing to this window will be in the specified font. IntuiFont#
specifies a previously initialized intuifont object created using
LoadFont.
```

1.12 WINDOWSLIB

```
Statement: WPlot
------------------------------------------------------------------------------
Modes  :
Syntax : WPlot X,Y,Colour

WPlot plots a pixel in the currently used window at the coordinates X,Y
in the colour specified by Colour.
```

1.13 WINDOWSLIB

```
Statement: WBox
------------------------------------------------------------------------------
Modes  :
Syntax : WBox X1,Y1,X2,Y2,Colour

WBox draws a solid rectangle in the currently used window. The upper
left hand coordinates of the box are specified with the X1 and Y1
```

values, and the bottom right hand corner of the box is specified by the values X2 and Y2.

1.14 WINDOWSLIB

```
Statement: WCircle
------------------------------------------------------------------------------
Modes  :
Syntax : WCircle X,Y,Radius,Colour

WCircle allows you to draw a circle in the currently used window. You
specify the centre of the circle with the coordinates X,Y. The Radius
value specifies the radius of the circle you want to draw. The last
value, Colour specifies what colour the circle will be drawn in.
```

1.15 WINDOWSLIB

```
Statement: WEllipse
------------------------------------------------------------------------------
Modes  :
Syntax : WEllipse X,Y,X Radius,Y Radius,Colour

WEllipse draws an ellipse in the currently used window. You specify the
centre of the ellipse with the coordinates X,Y. X Radius specifies the
horizontal radius of the ellipse, Y Radius the vertical radius.

Colour refers to the colour in which to draw the ellipse.
```

1.16 WINDOWSLIB

```
Statement: WCls
------------------------------------------------------------------------------
Modes  :
Syntax : WCls [Colour]

WCls will clear the currently used window to clour 0, or colour is
specified, then it will be cleared to this colour. If the current
window was not opened with the GIMMEZEROZERO flag set, then this
statement will clear any border or title bar that the window has. The
InnerCls statement should be used to avoid these side effects..
```

1.17 WINDOWSLIB

```
Statement: WLocate
------------------------------------------------------------------------------
Modes  :
Syntax : WLocate Cursor x,Cursor y

WLocate is used to set the text cursor position within the currently
used window. X and Y are both specified in pixels as offsets from the
top left of the window. Each window has it's own text cursor position,
therefore changing the text cursor position of one window will not
affect any other window's
text cursor position.
```

1.18 WINDOWSLIB

```
Statement: WindowInput
------------------------------------------------------------------------------
Modes  :
Syntax : WindowInput Window#

WindowInput will cause any future executions of the Inkey$, Edit$ or
Edit functions to receive their input as keystrokes from the specified
window object.

WindowInput is automatically executed when either a window is opened,
or Use Window is executed.

After a window is closed (using Free Window), remember to tell Blitz 2
to get it's input from somewhere else useful (for example, using
another WindowInput command) before executing another Inkey$, Edit$ or
Edit function.
```

1.19 WINDOWSLIB

```
Statement: WindowOutput
------------------------------------------------------------------------------
Modes  :
Syntax : WindowOutput Window#

WindowOutput will cause any future executions of either the Print or
NPrint statements to send their output as text to the specified window
object.

WindowOutput is automatically executed when either a window is opened,
or Use Window is executed.

After a window is closed (using Free Window), remember to send output
somewhere else useful (for example, using another WindowOutput command)
before executing another Print or NPrint statement.
```

1.20 WINDOWSLIB

```
Statement: Qualifier
------------------------------------------------------------------------------
Modes  :
Syntax : Qualifier
```

Qualifier will return the qualifier of the last key that caused a 'key press' event to occur. A qualifier is a key which alters the meaning of other keys; for example the 'shift' keys. Here is a table of qualifier values and their equivalent keys.

```
Key          Left  Right
UnQualified $8000 $8000
Shift         $8001 $8002
Caps Lock Down  $8004 $8004
Control         $8008 $8008
Alternate $8010 $8020
Amiga         $8040 $8080
```

A combination of values may occur, if more that one qualifier key is being held down. The way to filter out the qualifiers that you want is by using the logical AND operator.

1.21 WINDOWSLIB

```
Statement: RawKey
------------------------------------------------------------------------------
Modes  :
Syntax : RawKey
```

RawKey returns the raw key code of a key that caused the most recent 'key press' event.

1.22 WINDOWSLIB

```
Statement: Cursor
------------------------------------------------------------------------------
Modes  :
Syntax : Cursor Thickness : <0 - block, >=0 - underline thickness
```

Cursor will set the style of cursor that appears when editing strings or numbers with the Edit$ or Edit functions. If Thickness is less than 0, then a block cursor will be used. If the Thickness is greater then 0, then an underline Thickness pixels high will be used.

1.23 WINDOWSLIB

```
Statement: Editat
------------------------------------------------------------------------------
Modes  :
Syntax : Editat

After executing an Edit$ or Edit function, Editat may be used to
determine the horizontal character position of the cursor at the time
the function was exited.

Through the use of Editat, EditExit, EditFrom and Edit$, simple full
screen editors may be put together.
```

1.24 WINDOWSLIB

```
Statement: EditFrom
------------------------------------------------------------------------------
Modes  :
Syntax : EditFrom [Characterpos]

EditFrom allows you to control how the Edit$ and Edit functions operate
when used within windows.

If a Characterpos parameter is specified, then the next time an edit
function is executed, editting will commence at the specified character
position (0 being the first character position).

Also, editting may be terminated not just by the use of the 'return'
key, but also by any non printable character (for example, 'up arrow'
or 'Esc') or a window event. When used in conjunction with Editat and
EditExit, this allows you to put together simple full screen editors.

If Characterpos is omitted, Edit$ and Edit return to normal - editting
always beginning at character postition 0, and 'return' being the only
way to exit.
```

1.25 WINDOWSLIB

```
Statement: EditExit
------------------------------------------------------------------------------
Modes  :
Syntax : EditExit

EditExit returns the ASCII value of the character that was used to exit
a window based Edit$ or Edit function. You can only exit the edit
functions with keypresses other than 'return' if EditFrom has been
executed prior to the edit call.
```

1.26 WINDOWSLIB

```
Statement: WScroll
------------------------------------------------------------------------------
Modes  :
Syntax : WScroll X1,Y1,X2,Y2,Delta X,Delta Y

WScroll will cause a rectangular area of the currently used window to
be moved or 'scrolled'. X1 and Y1 specify the top left location of
the rectangle, X2 and Y2 the bottom right. The Delta parameters determine
how far to move the area. Positive values move the area right/down,
while negative values move the area left/up.
```

1.27 WINDOWSLIB

```
Statement: WMouseX
------------------------------------------------------------------------------
Modes  :
Syntax : WMouseX

WMouseX returns the horizontal x coordinate of the mouse relative to
the left edge of the current window. If the current window was opened
without the GIMMEZEROZERO flag set, then the left edge is taken as the
left edge of the border around the window, otherwise, if GIMMEZEROZERO
was set, then the left edge is the taken from inside the window
border.
```

1.28 WINDOWSLIB

```
Statement: WMouseY
------------------------------------------------------------------------------
Modes  :
Syntax : WMouseY

WMouseY returns the vertical y coordinate of the mouse relative to the
top of the current window. If the current window was opened without the
GIMMEZEROZERO flag set, then the top is taken as the top of the border
around the window, otherwise, if GIMMEZEROZERO was set, then the top is
taken from inside the window border.
```

1.29 WINDOWSLIB

```
Statement: WColour
------------------------------------------------------------------------------
Modes  :
Syntax : WColour Foreground Colour[,Background Colour]

WColour sets the foreground and background colour of printed text for
the currently used window. Any further text printed on this window will
be in these colours.
```

1.30 WINDOWSLIB

```
Statement: WJam
------------------------------------------------------------------------------
Modes  :
Syntax : WJam Jammode

WJam sets the text drawing mode of the currently used window. These
drawing modes allow you to do inverted, complemented and other types of
gaphics. The drawing modes can be OR'ed together to create a
combination of them. Here are the different modes.


Mode          Value Description
Jam1          0 This draws only the foreground colour and leaves the background ↩
   transparent. Eg For the letter O, any empty space (inside and outside the ↩
   letter) will be transparent.
Jam2          1 This draws both the foreground and background to the window. Eg ↩
   With the letter O again, the O will be drawn, but any clear area (inside and ↩
   outside) will be drawn in the current background colour.
Complement  2 This will exlusive or (XOR) the bits of the graphics. Eg Drawing on ↩
   the same place with the same graphics will cause the original display to return ↩
   .
Inversvid 4 This allows the display of inverse video characters. If used in ↩
   conjunction with Jam2, it behaves like Jam2, but the foreground and background ↩
   colours are exchanged.
```

1.31 WINDOWSLIB

```
Statement: WLine
------------------------------------------------------------------------------
Modes  :
Syntax : WLine X1,Y1,X2,Y3[,Xn,Yn...],Colour

Wline allows you to draw a line or a series of lines into the currently
used window. The first two sets of coordinates X1,Y1,X2,Y2, specify the
start and end points of the initial line. Any coordinates specified
after these initial two, will be the end points of another line going
from the last set of end points, to this set. Colour is the colour of
the line(s) that are to be drawn.
```

1.32 WINDOWSLIB

```
Statement: Activate
------------------------------------------------------------------------------
Modes  :
Syntax : Activate Window#

Activate will active the window specified by Window#.
```

1.33 WINDOWSLIB

```
Statement: WindowX
------------------------------------------------------------------------------
Modes  :
Syntax : WindowX

WindowX returns the horizontal pixel location of the top left corner of
the currently used window, relative to the screen the window appears
in.
```

1.34 WINDOWSLIB

```
Statement: WindowY
------------------------------------------------------------------------------
Modes  :
Syntax : WindowY

WindowY returns the vertical pixel location of the top left corner of
the currently used window, relative to the screen the window appears
in.
```

1.35 WINDOWSLIB

```
Statement: MButtons
------------------------------------------------------------------------------
Modes  :
Syntax : MButtons

MButtons returns the codes for the mouse buttons that caused the most
recent 'mouse buttons' event. If menus have been turned off using
Menus Off, then the right mouse button will also register an event and can be
read with MButtons.

The following are the values returned for the buttons by MButtons.
```

```
Button  Down  Up

Left  1 5
Right 2 6
```

1.36 WINDOWSLIB

```
Statement: Menus
------------------------------------------------------------------------------
Modes  :
Syntax : Menus On|Off

The Menus command may be used to turn ALL menus either on or off.
Turning menus off may be useful if you wish to read the right mouse
button.
```

1.37 WINDOWSLIB

```
Statement: WCursX
------------------------------------------------------------------------------
Modes  :
Syntax : WCursX

WCursX returns the horizontal location of the text cursor of the
currently used window. The text cursor position may be set using
WLocate.
```

1.38 WINDOWSLIB

```
Statement: WCursY
------------------------------------------------------------------------------
Modes  :
Syntax : WCursY

WCursY returns the vertical location of the text cursor of the
currently used window. The text cursor position may be set using
WLocate.
```

1.39 WINDOWSLIB

```
Statement: WPointer
------------------------------------------------------------------------------
Modes  :
Syntax : WPointer Shape#

WPointer allows you to determine the mouse pointer imagery used in the
currently used window. Shape# specifies an initialized shape object the
pointer is to take it's appearance from, and must be of 2 bitplanes
depth (4 colours).
```

1.40 WINDOWSLIB

```
Statement: MenusOn
------------------------------------------------------------------------------
Modes  :
Syntax : MenusOn
```

1.41 WINDOWSLIB

```
Statement: MenusOff
------------------------------------------------------------------------------
Modes  :
Syntax : MenusOff
```

1.42 WINDOWSLIB

```
Statement: WMove
------------------------------------------------------------------------------
Modes  :
Syntax : WMove X,Y

WMove will move the current window to a screen position specified by X
and Y.
```

1.43 WINDOWSLIB

```
Statement: WSize
------------------------------------------------------------------------------
Modes  :
Syntax : WSize Width,Height
```

```
WSize will alter the width and height of the current window to the
values specified by Width and Height.
```

1.44 WINDOWSLIB

```
Statement: WindowWidth
------------------------------------------------------------------------------
Modes  :
Syntax : WindowWidth

WindowWidth returns the pixel width of the currently used
window.
```

1.45 WINDOWSLIB

```
Statement: WindowHeight
------------------------------------------------------------------------------
Modes  :
Syntax : WindowHeight

WindowHeight returns the pixel height of the currently used
window.
```

1.46 WINDOWSLIB

```
Statement: InnerWidth
------------------------------------------------------------------------------
Modes  :
Syntax : InnerWidth

InnerWidth returns the pixel width of the area inside the border of the
currently used window.
```

1.47 WINDOWSLIB

```
Statement: InnerHeight
------------------------------------------------------------------------------
Modes  :
Syntax : InnerHeight

InnerHeight returns the pixel height of the area inside the border of
the currently used window.
```

1.48 WINDOWSLIB

```
Statement: InnerCls
------------------------------------------------------------------------------
Modes  :
Syntax : InnerCls [Colour]

InnerCls will clear only the inner portion of the currently used
window. It will not clear the titlebar or borders as Cls would do if
your window was not opened with the GIMMEZEROZERO flag set. If colour
is specfied, then that colour will be used to clear the window.
```

1.49 WINDOWSLIB

```
Statement: WTopOff
------------------------------------------------------------------------------
Modes  :
Syntax : WTopOff

WTopOff returns the number of pixels between the top of the current
window border and the inside of the window.
```

1.50 WINDOWSLIB

```
Statement: WLeftOff
------------------------------------------------------------------------------
Modes  :
Syntax : WLeftOff

WLeftOff returns the number of pixels between the left edge of the
current window border and the inside of the window.
```

1.51 WINDOWSLIB

```
Statement: SizeLimits
------------------------------------------------------------------------------
Modes  :
Syntax : SizeLimits Min Width,Min Height,Max Width,Max Height

SizeLimits sets the limits that any new windows can be sized to with
the sizing gadget. After calling this statement, any new windows will
have these limits imposed on them.
```

1.52 WINDOWSLIB

```
Statement: EMouseX
------------------------------------------------------------------------------
Modes  :
Syntax : EMouseX

EMouseX will return the horizontal position of the mouse pointer at the
time the most recent window event occured. Window events are detected
using the WaitEvent or Event commands.
```

1.53 WINDOWSLIB

```
Statement: EMouseY
------------------------------------------------------------------------------
Modes  :
Syntax : EMouseY

EMouseY will return the vertical position of the mouse pointer at the
time the most recent window event occured. Window events are detected
using the WaitEvent or Event commands.
```

1.54 WINDOWSLIB

```
Statement: AddIDCMP
------------------------------------------------------------------------------
Modes  :
Syntax : AddIDCMP IDCMP_Flags

AddIDCMP allows you to 'add in' IDCMP flags to the IDCMP flags selected
by DefaultIDCMP. Please refer to DefaultIDCMP for a thorough discussion
of IDCMP flags.
```

1.55 WINDOWSLIB

```
Statement: SubIDCMP
------------------------------------------------------------------------------
Modes  :
Syntax : SubIDCMP IDCMP_Flags

SubIDCMP allows you to 'subtract out' IDCMP flags from the IDCMP flags
selected by DefaultIDCMP. Please refer to DefaultIDCMP for a thorough
discussion of IDCMP flags.
```

1.56 WINDOWSLIB

```
Statement: FlushEvents
------------------------------------------------------------------------------
Modes  :
Syntax : FlushEvents [IDCMP_Flag]
```

When window events occur in Blitz 2, they are automatically 'queued' for you. This means that if your program is tied up processing one window event while others are being created, you wont miss out on anything. Any events which may have occured between executions of WaitEvent or Event will be stored in a queue for later use. However, there may be situations where you want to ignore this backlog of events. This is what FlushEvents is for.

Executing FlushEvents with no parameters will completely clear Blitz 2's internal event queue, leaving you with no outstanding events. Supplyng an IDCMP_Flag parameter will only clear events of the specified type from the event queue.

1.57 WINDOWSLIB

```
Statement: CatchDosErrs
------------------------------------------------------------------------------
Modes  :
Syntax : CatchDosErrs
```

Whenever you are executing AmigaDos I/O (for example, reading or writing a file), there is always the possibility of something going wrong (for example, disk not inserted... read/write error etc.). Normally, when such problems occur, AmigaDos displays a suitable requester on the WorkBench window. However, by executing CatchDosErrs you can force such requesters to open on a Blitz 2 window.

The window you wish dos error requesters to open on should be the currently used window at the time CatchDosErrs is executed.

1.58 WINDOWSLIB

```
Function: RastPort
------------------------------------------------------------------------------
Modes  :
Syntax : RastPort (Window#)
```

RastPort returns the specified Window's RastPort address. Many commands in the graphics.library and the like require a RastPort as a parameter.

1.59 WINDOWSLIB

```
Statement: SetEventFilter
--------------------------------------------------------------------------------
Modes  :
Syntax : SetEventFilter preprocess,postprocess
```

1.60 WINDOWSLIB

```
Statement: WTitle
--------------------------------------------------------------------------------
Modes  :
Syntax : WTitle windowtitle$[,screentitle$]

WTitle is used to alter both the current window's title bar and it's
screens title bar. Useful for displaying important stats such as
program status etc.

The screen title does not have to be specified when using the command.
Specifying only a window title will cause the screen title to be unchanged.
```

1.61 WINDOWSLIB

```
Statement: CloseWindow
--------------------------------------------------------------------------------
Modes  :
Syntax : CloseWindow Window#

CloseWindow has been added for convenience. Same as Free Window but a
little more intuitive (added for those that have complained about such
matters).
```

1.62 WINDOWSLIB

```
Statement: WPrintScroll
--------------------------------------------------------------------------------
Modes  :
Syntax : WPrintScroll

WPrintScroll will scroll the current window upwards if the text cursor
is below the bottom of the window and adjust the cursor accordingly.
Presently WPrintScroll only works with windows opened with the gimme00
flag set (#gimmezerozero=$400).
```

1.63 WINDOWSLIB

```
Statement: WBlit
------------------------------------------------------------------------------
Modes  :
Syntax : WBlit Shape#,x,y

WBlit can be used to blit any shape to the current window. Completely
system friendly this command will completely clip the shape to fit
inside the visible part of the window. Use GimmeZeroZero windows for
clean clipping when the window has title/sizing gadgets.
```

1.64 WINDOWSLIB

```
Statement: BitMaptoWindow
------------------------------------------------------------------------------
Modes  :
Syntax : BitMaptoWindow Bitmap#,Window#[,srcx,srcy,destx,desty,wid,height]

BitMaptoWindow will copy a bitmap to a window in an operating system
friendly manner (what do you expect). The main use of such a command is
for programs which use the raw bitmap commands such as the 2D and Blit
libraries for rendering bitmaps quickly but require a windowing
environment for the user inyerface.
```

1.65 WINDOWSLIB

```
Statement: EventCode
------------------------------------------------------------------------------
Modes  :
Syntax : EventCode

EventCode returns the actual code of the last Event received by your
program, EventQualifier returns the contents of the Qualifier field. Of
use with the new GadTools library and some other low level event
handling requirements.
```

1.66 WINDOWSLIB

```
Statement: EventQualifier
------------------------------------------------------------------------------
Modes  :
Syntax : EventQualifier

EventCode returns the actual code of the last Event received by your
program, EventQualifier returns the contents of the Qualifier field. Of
```

use with the new GadTools library and some other low level event
handling requirements.

1.67 WINDOWSLIB

```
Statement: PositionSuperBitMap
------------------------------------------------------------------------------
Modes  :
Syntax : PositionSuperBitMap x,y
```

PositionSuperBitMap is used to display a certain area of the bitmap in
a super bitmap window.

1.68 WINDOWSLIB

```
Statement: GetSuperBitMap
------------------------------------------------------------------------------
Modes  :
Syntax : GetSuperBitMap
```

After rendering changes to a superbitmap window the bitmap attached can
also be updated with the GetSuperBitMap. After rendering changes to a
bitmap the superbitmap window can be refreshed with the PutSuperBitMap
command. Both commands work with the currently used window.

1.69 WINDOWSLIB

```
Statement: PutSuperBitMap
------------------------------------------------------------------------------
Modes  :
Syntax : PutSuperBitMap
```

See GetSuperBitmap description.

1.70 WINDOWSLIB

```
Statement: WindowTags
------------------------------------------------------------------------------
Modes  :
Syntax : WindowTags Window#,Flags,Title$ [,&TagList] or [[,Tag,Data]...]
```

Similar to ScreenTags, WindowTags allows the advanced user to open a
Blitz window with a list of OS Tags as described in documentation for
the operating system prior to 2.0.

1.71 WINDOWSLIB

```
Statement: AddWaitEvent
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : AddWaitEvent bitnumber,returncode

This command allows you to make the waitevent command wait for different
signal bits - allowing your program to 'wake up' when other events happen,
like an ARexx message for example.  You must provide the bitnumber to check
for, plus the desired return code you want back when the bit gets set.

  An example using the RIARexxLib would be:

    port.l=RXCreatePort("MYPORT")
    addwaitevent PortSigBit(port),$8000000

When using this code, all arexx messages to your program will cause
WaitEvent to exit and return a value of $8000000.
```

1.72 WINDOWSLIB

```
Statement: DelWaitEvent
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : DelWaitEvent bitnumber

This command clears a bitnumber from the current list of bits that
WaitEvent will wait for.  It should be used after you have finished with a
particular bit and do not want to here any more messages from it.
```

1.73 WINDOWSLIB

```
Statement: EventIAddress
------------------------------------------------------------------------------
Modes  : Amiga
Syntax : ie.l=EventIAddress

This returns the iAddress of the last event code.  E.g. after a
IDCMP_GADGETUP event, this command will return the address of the gadget
that was hit.
```

1.74 WINDOWSLIB

```
Statement: WaitSigBits
------------------------------------------------------------------------------
```

```
Modes  : Amiga
Syntax : sigbits.l=WaitSigBits

Returns the current signal bits that WaitEvent will wait for when used.
```

1.75 WINDOWSLIB

```
                .-------------------------------------------------------------------------
|                                  WINDOWSLIB                                  |
`------------------------------------------------------------------------------'


                    Overview
                                          Command Index


                Activate

                Cursor

                DefaultIDCMP

                Editat

                EditExit

                EditFrom

                Event

                EventWindow

                GadgetHit

                InnerCls

                InnerHeight

                InnerWidth

                ItemHit

                MButtons

                MenuHit

                Menus

                MenusOff

                MenusOn

                Qualifier
```

RawKey

SubHit

WaitEvent

WaitSigBits

WBox

WCircle

WCls

WColour

WCursX

WCursY

WEllipse

Window

WindowFont

WindowHeight

WindowInput

WindowOutput

WindowWidth

WindowX

WindowY

WJam

WLine

WLocate

WMouseX

WMouseY

WMove

WPlot

WPointer

WScroll

WSize

```
WTopOff
```