K-Windows

BGFX

Basic09 Interface Library

Edition 4

IMPORTANT NOTES

Unless noted otherwise, all numeric parameters should of type INTEGER. It appears that Basic will do this automatically for parameters being passed to the BGFX subroutine. However, it is critical that parameter variables in which a value is expected to be returned (eg: WINfo) should be INTEGER typed.

THE FUTURE

As much as possible, BGFX should make it easy (and has) to port many CoCo3 applications to OS9/68K. However, because it's not totally compatible (eg: the Palette function), I believe that a version called GFX2 will be done soon.

Additional Basic libraries in the works include:

BSND - allows loading, recording, playing sound files

BGUI - interfaces to the coming menus and controls

BIFF - easy IFF file format load and save interface

THANKS!

Thank you for purchasing this copy of the BGFX Interface Library!

If you have any ideas for additions or have found any bugs, please feel free to drop me a line. I can be contacted most easily via email:

Compuserve: 76703.4227

Internet: 76703.4227@compuserve.com
kdarling@catt.ncsu.edu

Updates will be made available to everyone who writes me at:

Kevin Darling
3000-79 Stonybrook Dr
Raleigh, NC 27604

Again, thank you very much.

OSK BGFx Library

Windowing

- DWSet
- DWEnd
- OWSet
- OWEnd
- Select
- CWArea
- WInfo

Get/Put Buffers

- GCSet
- GPLoad
- Get
- Put
- KillBuff

Colors, Logic, Pattern

- Color
- DefCol
- GetPals
- Logic
- Palette
- Pattern
- SetPals

Drawing Commands

- Arc
- Bar
- Bezier
- Box
- Circle
- CBox
- Ellipse
- Fill
- FCircle
- FEllipse
- GetPnt
- Line
- Point
- RBar
- RBox
- SBox
- ScaleSw
- RSetDptr
- SetDPtr

OSK BGFX Library

FUNCTION

Arc - Draw an arc (portion of ellipse)

CODE

```
1B 52 Xr xr Yr yr Xo1 xo1 Yo1 yo1 Xo2 xo2 Yo2 yo2
```

BGFX

```
RUN bgfx("Arc",xr,yr,xo1,yo1,xo2,yo2)
```

PARAMETERS

xr, yr	- the x and y radii of the basic ellipse
xo1,yo1	- offsets to the start of a bisecting line
xo2,yo2	- offsets to the endpoint of a bisecting line

DESCRIPTION

Draw an arc, made up of the visible section of an ellipse as determined by a bisecting line (the last parameters).

For example, to show the upper right quarter of a 80x40 ellipse centered at 320,100 (middle of screen), you could use:

```
RUN bgfx("arc",320,100,80,40,0,-40,80,0)

     X1,Y1                The line from X1,Y1 to X2,Y2 chops
       |...               off all but the upper right quarter.
     YR|.....
-------+-------X2,Y2      XR = 80     X1,Y1 = 0,-40
       |   XR             YR = 40     X2,Y2 = 80,0
       |                   + = origin = 320,100
```

The bisecting line coords are offsets from the ellipse origin. If you reverse its slope by swapping X1,Y1 and X2,Y2, then all BUT the upper right quarter would be seen.

Note: The bisecting line coords simply give the slope... they don't actually have to mark out the a line which crosses the basic ellipse... but instead a line drawn through the two points. In other words, take any ellipse which would be drawn of XR,YR shape, and then draw a line through any (offset from ellipse origin) X1,Y1-X2,Y2 point pair.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

Bar, RBar, - Draw a bar (filled rectangle)

CODE

```
1B 4A X2 x2 Y2 y2  (absolute)
1B 4B Xo xo Yo Yo  (relative)
```

BGFX

```
RUN bgfx([path,] "Bar", x2, y2)
RUN bgfx([path,] "Bar", x1, y1, x2, y2)
RUN bgfx([path,] "RBar",xo, yo)
```

PARAMETERS

x1,y1	- optional first move-to coordinates
x2,y2	- end coordinates
xo,yo	- end offsets from current draw pointer

DESCRIPTION

Both functions draw a filled rectangle beginning at the draw pointer's current position, using the current foreground color, logic and pattern.

Bar draws a bar ending at coordinates _x2_,_y2_.

RBar draws a bar to the _relative_ _offset_ coordinates _xo_,_yo_.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

Bell - Ring terminal bell

CODE

07

BGFX

RUN bgfx([path.] "Bell")

DESCRIPTION

On the MM/1, this currently rings the simple timer-based bell (usually attached to the small speaker inside your case).

FUNCTION

Bezier - draw a bezier curve

CODE

```
1B 55 X1x1 Y1y1 X2x2 Y2y2 X3x3 Y3y3 X4x4 Y4y4
```

BGFX

```
RUN bgfx("Bezier",x1,y1,x2,y2,x3,y3,x4,y4)
```

PARAMETERS

x1,y1	- start point
x2,y2	- second control point
x3,y3	- third control point
x4,y4	- end point

DESCRIPTION

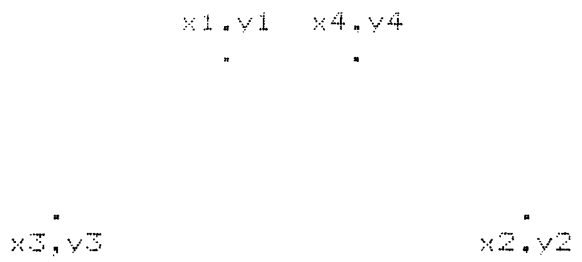
Draws a four-point Bezier line, from (x1,y1) to (x4,y4), curved by the two middle control points.

CAVEATS

Note that the coordinates are absolute. I hope to add a relative coordinate version soon, which should be more useful for characters.

EXAMPLE

See the "testbez" program in your DEMOS disk directory.



FUNCTION

Box, RBox, - Draw a box (rectangle)

CODE

```
1B 48 X2 x2 Y2 y2  (absolute)
1B 49 Xo xo Yo Yo  (relative)
```

BGFX

```
RUN bgfx([path,] "Box", x2, y2)
RUN bgfx([path,] "Box", x1, y1, x2, y2)
RUN bgfx([path,] "RBox",xo, yo)
```

PARAMETERS

x1,y1	- optional first move-to coordinates
x2,y2	- end coordinates
xo,yo	- end offsets from current draw pointer

DESCRIPTION

Both functions draw a rectangle beginning at the draw pointer's current position.

Box draws a box ending at coordinates x2,y2.

RBox draws a line to the relative offset coordinates xo,yo.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

CBox, RCBox, - Draw a box with curved corners

CODE

```
1B 4C X2 x2 Y2 y2  Xr xr Yr yr   (absolute)
1B 4D Xo xo Yo Yo  Xr xr Yr yr   (relative)
```

BGFX

```
RUN bgfx([path,] "CBox", x2, y2)
RUN bgfx([path,] "CBox", x1, y1, x2, y2)
```

PARAMETERS

x1,y1	- optional first move-to coordinates
x2,y2	- end coordinates
xr,yr	- radii of curved corners

DESCRIPTION

Draw a rectangle with curved corners (such as you might see depicted as buttons on some computers).

CBox draws a box ending at coordinates x2,y2.

CRBox draws a line to the relative offset coordinates xo,yo.

CAVEATS

Because of screen aspects, you usually want to make the X radius about twice that of the Y radius.

SEE ALSO

SetDPtr, RSetDPtr, Box

FUNCTION

Circle - Draw a circle
FCircle - Draw a filled circle

CODE

```
1B 50 r r  (normal)
1B 53 r r  (filled)
```

BGFX

```
RUN bgfx([path,] "Circle", x, y, r)
RUN bgfx([path,] "Circle", r)

RUN bgfx([path,] "FCircle", x, y, r)
RUN bgfx([path,] "FCircle", r)
```

PARAMETERS

x,y	= optional first move-to coordinate
r	= radius

DESCRIPTION

Draws a circle centered on the passed coordinates or the current draw pointer. A filled circle will use the current pattern, logic and colors inside.

CAVEATS

If a coordinate is passed, the draw pointer is updated to it.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

Clear - clear a window

CODE

0C

BGFX

```
RUN bgfx([path,] "Clear)
```

DESCRIPTION

Clears the current working area of a window to the background color.

SEE ALSO

Color, CWArea

FUNCTION

Set Text/Drawing Colors to specified palette register

CODES

```
FColor - 1B 32 prn
BColor - 1B 33 prn
Border - 1B 34 prn
```

BGFX

```
RUN bgfx([path,] "Color",fore_prn [,back_prn] [,border_prn])
RUN bgfx([path,] "Border",border_prn)
```

PARAMETERS

path	= optional OS-9 path number for the window
fore_prn	= palette register number for foreground drawing
back_prn	= optional register number for background
border_prn	= optional register number for screen border

DESCRIPTION

Color changes the foreground (and optionally, background and border) colors for a window. Most graphics use only the foreground color. Text uses both fore/background unless in transparent mode.

SEE ALSO

Palette, Clear

CURSOR COMMANDS

```
CurHome - move cursor to col and row 0
CurLft  - move cursor back one position
CurRgt  - move cursor right one position
CurUp   - move cursor up one row
CurDwn  - move cursor down one row
CrRtn   - send carriage return
CurXY   - position cursor
CurOn   - turn text cursor on
CurOff  - turn text cursor off
```

BGFX

```
RUN bgfx([path,] "CurXY",col,row)
RUN bgfx([path,] "CurHome")
RUN bgfx([path,] "CurLft")
RUN bgfx([path,] "CurRgt")
RUN bgfx([path,] "CurUp")
RUN bgfx([path,] "CurDwn")
RUN bgfx([path,] "CrRtn")
RUN bgfx([path,] "CurOn")
RUN bgfx([path,] "CurOff")
```

PARAMETERS

```
col,row = for absolute cursor positioning
             column and row begin at ZERO (0)
```

DESCRIPTION

These commands allow manipulation of the text cursor.

CAVEATS

These are Window specific calls and may not work properly on a remote dumb terminal.

SEE ALSO

Clear, CWArea, and Erase functions

FUNCTION

CWArea - Change window working area

CODE

```
1B 25 cpx cpy szx szy
```

BGFX

```
RUN bgfx([path,] "CWArea",cpx,cpy,szx,szy)
```

PARAMETERS

```
path  = optional OS-9 path number for the window
cpx   = horizontal character position for the upper
        left corner of the working area
cpy   = vertical position for the upper left corner
        of the working area
szx   = size (width) of the working area in characters
szy   = size (height) of the working area in characters
```

DESCRIPTION

CWAREA changes the working area of a window to the new location given by <u>cpx,cpy</u> , which is an offset from the position of the original window area. The size is <u>szx,szy</u>.

Text and graphics output will be confined to this area until changed. If scaling is turned out, graphics output will be scaled down.

CAVEATS

This call cannot make a window larger than originally defined. It can only change the working area inside the window.

FUNCTION

DefCol - Reset all palettes to default colors

CODE

1B 30

BGFX

```
RUN bgfx([path,] "DefCol")
```

PARAMETERS

path = optional OS-9 path number for the window

DESCRIPTION

DefCol resets the colors associated with the window specified by <u>path</u> to the default (Startup) colors.

CAVEATS

DefCol will restore any changes which have occurred to the palettes on the specified window since bootup.

SEE ALSO

Palette

FUNCTION

DWSet - Device Window Set

CODE

```
1B 20 sty cpx cpy szx szy fprn prn
```

BGFX

```
RUN bgfx([path,] "DWSet",sty,cpx,cpy,szx,szy,fprn,bprn)
```

PARAMETERS

path	= OS-9 path number for the window
sty	= window type
cpx	= horizontal character position for the upper left corner of the window
cpy	= vertical position for the upper left corner of the window
szx	= size (width) of the window in characters
szy	= size (height) of the window in characters
fprn	= foreground palette register number
bprn	= background palette register number

DESCRIPTION

Creates a device in a window of type sty. If sty=ff , the system opens the window on the current screen.

The window has its upper left corner located at cpx,cpy , and its size set to szx,szy. Note that the coordinates and size values are in standard character (8x8) coordinates.

The window uses fprn as the foreground palette and bprn as the background palette.

(continued on next page)

The following list describes the supported screen types:

Code	Char Size	Pixels Size	Colors	Type
00	80 x 26	640 x 208	16	non-interlaced
01	80 x 26	640 x 416	16	interlaced repeat
02	80 x 52	640 x 416	16	interlaced
03	40 x 26	320 x 208	256	non-interlaced
04	40 x 26	320 x 416	256	interlaced repeat
05	40 x 52	320 x 416	256	interlaced
06	90 x 30	720 x 240	16	non-interlaces overscan
07	90 x 60	720 x 240	16	interlaced overscan
08	48 x 30	384 x 240	256	overscan
09	48 x 60	284 x 480	256	interlaced overscan
FF	Currently Displayed Screen			
FE	Currently Selected Screen			

CAVEATS

These modes are the modes for the MM/1 and may not be supported on other hardware.

FUNCTION

DWEnd - Device Window End

CODE

1B 24

BGFX

RUN bgfx([path,] "DWEnd")

DESCRIPTION

Closes the device window associated with the specified path. If the closed window is the last device window on the screen, DWEND deallocates the screen.

CAVEATS

This call is not necessarv if the device was not iniz'd, as the internal device termination will automaticallv call the DWEnd function.

SEE ALSO

DWSet()

FUNCTION

Ellipse - Draw an ellipse
FEllipse - Draw a filled ellipse

CODE

```
1B 51 rx rx  ry ry  (normal)
1B 54 rx rx  ry ry  (filled)
```

BGFX

```
RUN bgfx([path,] "Ellipse", x, y, xr, yr)
RUN bgfx([path,] "Ellipse", xr, yr)

RUN bgfx([path,] "FEllipse", x, y, xr, yr)
RUN bgfx([path,] "FEllipse", xr, yr)
```

PARAMETERS

x,y	= optional first move-to coordinate
rx,ry	= x and y radius

DESCRIPTION

Draws an ellipse centered on the passed coordinates or the current draw pointer. A filled ellipse will use the current pattern, logic and colors inside.

CAVEATS

If a coordinate is passed, the draw pointer is updated to it.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

Fill - Flood Fill Area

CODE

1B 4F

BGFX

```
RUN bgfx([path,] "Fill" [,x,y])
```

PARAMETERS

x,y = optional first move-to position

DESCRIPTION

Sets the pixels surrounding the current draw pointer to the foreground color. The Fill operation continues outward until it reaches either the edge of the screen or pixels that are a color other than the pixel at the draw pointer's current position.

CAVEATS

The current version of WindIO uses a smoother (but slower) fill method if no pattern is in use. The method used with a pattern can get "stuck" (because it tries to go back over the same holes left by the pattern) and may be delayed in returning. I'm working on it.

SEE ALSO

SetDPtr, RSetDPtr, Point, GetPnt

FUNCTION

Font - Select/Change the font used for Text.

CODE

```
1B 3A grp bfn
```

BGFX

```
RUN bgfx([path,] "Font", grp, bfn)
```

PARAMETERS

```
path    = optional path number for the window
grp     = group associated with the buffer
bfn     = buffer number
```

DESCRIPTION

Font specifies the Get/Put buffer to use for text. The font must be loaded into the buffer using GPLoad.

A font buffer is normally composed of on/off data bits, and, for now, must be 8 pixels wide.

At this time, groups $80 and $81 are reserved for vector fonts and Amiga fonts.

To return to the default font in the stdfonts module, use a group and buffer number of zero.

SEE ALSO

GPLoad

FUNCTION

GCSet - Set buffer to use for graphics cursor

CODE

```
1B 39 grp bfn
```

BGFX

```
RUN bgfx([path,] "GCSet", grp, bfn)
```

PARAMETERS

```
grp    = buffer group
bfn    = buffer number within group
```

DESCRIPTION

Sets the source buffer for the graphics cursor in a window when it is the current input device.

To revert to the default built-in arrow cursor, simply GCSet to group and buffer number zero.

Several predefined cursor shapes for your use are provided in the stdptrs files (group $CA).

CAVEATS

The preloaded buffer type (1,2,4,8-bit color) must match the screen type of the destination window.

SEE ALSO

GPLoad

FUNCTION

Get - Save an area of the screen to a Get/Put buffer.

CODE

```
1B 2C grp bfn xh xl yh yl xsizeh xsizel ysizeh ysizel
```

BGFX

```
RUN bgfx([path,] "Get",grp,bfn,x,y,xsize,ysize)
```

PARAMETERS

path	=	optional OS-9 path number for the window
grp	=	group number associated with the buffer
bfn	=	buffer number
x	=	starting horizontal screen position
y	=	starting vertical screen position
xsize	=	horizontal number of pixels to get
ysize	=	vertical number of pixels to get

DESCRIPTION

Copies a block of screen data from x,y to x+xsize,y+ysize. Stores the data in the buffer specified by group,buffer. Once the block is saved, you can put it back in its original location or in another on the screen, using the Put function.

Generally, it is considered a good idea to use your process id as the group number. It is also considered a good idea to first KillBuff the group when your program first starts up. This requirement may disappear soon.

CAVEATS

Get will be affected by Scaling. If the Get/Put buffer is not already defined, Get creates it. If the buffer is defined, its size must be greater than or equal to the desired new Get size.

SEE ALSO

Put

FUNCTION

GetPals - Read window palette settings

BGFX

```
RUN bgfx([path,] "GetPals",array,first,count)
```

PARAMETERS

```
path  = optional path to window
array = buffer for returned data
first = first paletter register to return
count = number of registers to return
```

DESCRIPTION

GetPals returns the values for the number of palette registers specified by _count_ starting at the CLUT (Color Look Up Table) offset passed in _first_. A copy of the color information is returned in the buffer pointed to by _array_. GetPals allows getting any or all the palette registers information at one time.

CAVEATS

Array should be large enough to hold 3 times the desired count. To reserve space for all 256 palettes, use something like:

```
DIM palettes(768):BYTE
```

Count is the number of triplets (R,G,B bytes) to copy, NOT the length of the palette data in bytes.

FUNCTION

GetPnt - Get color of pixel

BGFX

```
RUN bgfx([path,] "GetPnt".x,y,prn)
```

PARAMETERS

path	=	OS-9 path number for the window
x,y	=	window coordinate
prn	=	return palette register number

DESCRIPTION

GetPnt returns the color (or rather, the palette register number from 0-255) of a pixel within a window.

EXAMPLES

CAVEATS

SEE ALSO

Color, Point

FUNCTION

GPLoad - Preload a Get/Put Buffer

CODE

```
1B 2B grp bfn sty XSiz xsiz Ysiz ysiz Len len [..data..]
```

BGFX

```
RUN bgfx([path,] "GPLoad",group,buffer,type,xsize,ysize,length)
PUT #path,dataarray
```

PARAMETERS

```
group  = group number associated with the buffer
buffer = buffer number
type   = type for the data
           00 = non-gfx data (fonts, for example)
           01 = 2 bits/pixel
           02 = 4 bits/pixel
           03 = 8 bits/pixel
xsize  = horizontal size for the data
ysize  = vertical size for the data
length = length of data in bytes
```

DESCRIPTION

GPLoad allocates and prepares to load a Get/Put buffer with data. After receiving a GPLoad() call, the system loads the next bytes written to that path into the specified get/put buffer.

CAVEATS

If the Get/Put buffer is not already created, GPLoad creates it. If the buffer was previously created, it is deleted first.

If you accidentally specify a larger data length than you have, a CTRL-C will abort the GPLoad.

Note that the data types are different from screen types. This is a result of back compatibility with the very earliest WindIO versions. In the future, new calls will be added to fix this.

SEE ALSO

Get, Put, Font, KillBuff

FUNCTION

ID - Returns process id

BGFX

```
RUN bgfx("ID",id)
```

PARAMETERS

id - process id returned in integer

DESCRIPTION

Instead of using svscall, ID can be used as a quick method of getting the process id for use as the group number for buffers, etc.

FUNCTION

Killbuff - Deallocate a Get/Put Buffer

CODE

```
1B 2A grp bfn
```

BGFX

```
RUN bgfx([path,] "KillBuff", grp, bfn)
```

PARAMETERS

path	= optional OS-9 path number for the window
grp	= group number associated with the buffer
bfn	= buffer number

DESCRIPTION

Deallocates the specified get/put buffer. To deallocate an entire group of buffers, set the buffer number to 0.

CAVEATS

Most times, a program will use its own process id as the group number (this may be gotten with the BGFX "ID" command). It is also considered prudent to KillBuff that group when your program begins, and is considered polite for your program to KillBuff the group on exit (to save memory space).

FUNCTION

Line - Draw lines.

CODE

```
1B 44 X2 x2 Y2 v2  (Line)
1B 45 Xo xo Yo vo  (RLine)
1B 46 X2 x2 Y2 v2  (LineM)
1B 47 Xo xo Yo vo  (RLineM)
```

BGFX

```
RUN bgfx([path.] "Line", x1, v1, x2, v2)
RUN bgfx([path.] "Line", x2, v2)
```

PARAMETERS

x1,v1	= optional start move-to coordinate
xv,v2	= end coordinate

DESCRIPTION

All the Line escape codes draw from the current drawpointer position. The "R" versions use a relative offset to specific the end coordinate, and the "M" versions also update the draw pointer to the endpoint.

The BGFX Line command uses absolute coordinates. A "DRAW" command which uses the relative version will be added to BGFX soon.

SEE ALSO

SetDPtr, RSetDptr

FUNCTION

Logic - Set drawing logic mode

CODE

1B 2F mode

BGFX

RUN bgfx([path,] "Logic","mode")

MODES

Code	BGFX	Description
0	"off"	no logic code. store new data on screen
1	"and"	AND the new data with data on screen
2	"or"	OR new data with the data on screen
3	"xor"	XOR new data with the data on screen
4		Store all pattern
5		Brush

DESCRIPTION

Logic defines the combinatorial mode to be used in all drawing commands on the window specified by path. Logic allows creating special affects. The specified mode will be used until another Logic call changes it.

FUNCTION

Mouse - Get mouse status

BGFX

```
RUN bgfx("Mouse",valid,area,control,wx,wy,b1,b2)
```

PARAMETERS

```
valid    - validity flag; if non-zero, window is selected
area     - window area mouse is over
             0 = outside window
             1 = inside window, but outside working area
             2 = inside current working area
control  - two byte id of any hotspot under cursor
wx,wy    - window relative/scaled coordinates
b1       - main button status (0 if up)
b2       - secondary button status (not supported yet)
```

DESCRIPTION

Reads the current mouse position and button status, and returns the main mouse information required by most programs.

The X and Y coordinates are scaled to the window.

CAVEATS

The first value (valid flag) is important, as a zero value indicates that the other information should be ignored.

SEE ALSO

OnMouse

FUNCTION

OnKey - Set keyboard signal code

BGFX

```
RUN bgfx([path,] "OnKey",signal)
```

PARAMETERS

signal = signal on key. If 00, sleep until key.

DESCRIPTION

Calls SS_SSig to set up a signal code to be sent to your program when a key is hit. If a zero (00) signal value is passed, OnKey will change it to a 1 (S$Wake) and go to Sleep until woken by a signal from the keyboard or other device.

In packed programs, an ON ERROR statement may be used to vector program flow on signals.

CAVEATS

This call is a one-shot signal, and must be reset each time.

Generally, you should use signal values >= 32.

EXAMPLES

```
RUN bgfx("onkey",0)  \ (* Sleep until mouse click
RUN bgfx("onkey",32) \ (* Signal 32 sent on next click
```

SEE ALSO

Release, OnMouse

FUNCTION

OnMouse - Set mouse signal code

BGFX

```
RUN bgfx([path,] "OnMouse",signal)
```

PARAMETERS

signal = signal on click. If 00, sleep until click.

DESCRIPTION

Call SS_MsSig to set up a signal code to be sent to your program when the mouse button is pressed. If a zero (00) signal value is passed, OnMouse will change it to a 1 (S$Wake) and go to Sleep until woken by a signal from the mouse or other device.

In packed programs, an ON ERROR statement may be used to vector program flow on signals.

CAVEATS

This call is a one-shot signal, and must be reset each time.

Generallv, you should signal values >= 32.

EXAMPLES

```
RUN bgfx("onmouse",0)  \ (* Sleep until mouse click
RUN bgfx("onmouse",32) \ (* Signal 32 sent on next click
```

SEE ALSO

Release, OnKev

FUNCTION

OWSet - Overlav Window Set

CODE

```
1B 22 svx cpx cpv szx szv forn bprn
```

BGFX

```
RUN bgfx([path,] "OWSet", svs, cpx, cpy, szx, szy, forn, bprn)
```

PARAMETERS

svs	=	save switch
cpx	=	horizontal character position for the upper left corner of the window
cpv	=	vertical position for the upper left corner of the window
szx	=	size (width) of the window in characters
szy	=	size (height) of the window in characters
forn	=	foreground palette register number
born	=	background palette register number

DESCRIPTION

Creates an overlav window of size _cpx.xpv_ on the current device window.

If the save switch (_svs_) is 0, the system does not save the area under the overlav window. If (_svs_) is 1, the system saves the area under the window if possible and restores it when OWEnd is called.

SEE ALSO

CWArea, OWEnd

FUNCTION

OWEnd - Overlay Window End

CODE

1B 23

BGFX

```
RUN bgfx([path,] "OWEnd")
```

DESCRIPTION

Deallocates the top overlay window. If you created the window with a save switch of 1, the area under the screen is restored.

SEE ALSO

OWSet

FUNCTION

Palette - Change the color in a palette register

CODE

```
1B 31 prn red grn blu
```

BGFX

```
RUN bgfx([path,] "Palette", prn, red, grn, blu)
```

PARAMETERS

```
path = optional OS-9 path number for the window
prn  = palette register number to load (0-255)
red  = red value (0-255)
grn  = green value (0-255)
blu  = blue value (0-255)
```

DESCRIPTION

Palette allows changing the colors in the palette register specified by _prn_ to contain the color _red_, _grn_, _blu_. The color value may be any value between _0 and 255_, which gives more than _16 million color combinations_ to choose from.

CAVEATS

Only palettes 0-15 may be changed on a 16-color window.

SEE ALSO

Color, DefCol

FUNCTION

Pattern - Set Get/Put buffer as graphics pattern

CODE

```
1B 2E grp bfn
```

BGFX

```
RUN bgfx([path,] "Pattern", grp, bfn)
```

PARAMETERS

grp	= group number associated with the buffer
bfn	= buffer number

DESCRIPTION

Sets a Get/Put Buffer, previously copied from the screen with GetBlk or loaded with GPLoad as the current working graphics pattern. This pattern will be used with any graphics command (eg: Point, Line, Put, etc) until turned off by setting to a group and buffer number of zero.

Several fancy patterns are supplied in the stdpats files.

CAVEATS

The buffer used must match in color type. In addition, the X size of the buffer must be of a power of two (2,4,8,16,32,64, and so forth, up to full screen).

SEE ALSO

Get, GPLoad

FUNCTION

Point, RPoint - Draw a Point

CODE

```
1B 42 X  x  Y  y
1B 43 Xo xo Yo yo
```

BGFX

```
RUN bgfx([path,] "Point", x, y)
RUN bgfx([path,] "Point")
```

PARAMETERS

x,y = optional move-to absolute coordinate

DESCRIPTION

Point draws a point either at the draw pointer's current position, or at the coordinates specified by x,y.

CAVEATS

Point will update the draw pointer to x,y.

SEE ALSO

SetDPtr, RSetDPtr

FUNCTION

PropSw - Set/reset proportional character attribute

CODE

```
1B 3F switch   (00=off, 01=on)
```

BGFX

```
RUN bgfx([path,] "PropSw", "on")
RUN bgfx([path,] "PropSw", "off")
```

DESCRIPTION

PropSw turns proportional text output on or off.

CAVEATS

At this time, **PropSw** does not affect normal text. However, turning it on is important for best results when using vector or Amiga fonts.

SEE ALSO

TChrSw, BoldSw

FUNCTION

Put - Put a Get/Put Buffer to the screen

CODE

```
1B 2D grp bfn xh xl yh yl
```

BGFX

```
RUN bgfx([path,] "Put", grp, bfn, x, y)
```

PARAMETERS

path	= optional OS-9 path number for the window
grp	= group number associated with the buffer
bfn	= buffer number
x	= horizontal (x) coordinate to put the upper left hand corner of the buffer.
y	= vertical (y) coordinate to put the upper left hand corner of the buffer.

DESCRIPTION

Moves a Get/Put Buffer, previously copied from the screen with Get or loaded with GPLoad to an area of the screen. The dimensions of the buffer were saved in the Get/Put buffer when it was created. Windio uses these dimensions when restoring the buffer.

CAVEATS

Get/Put buffers cannot be scaled. The image will be clipped if it does not fit within the current working area.

SEE ALSO

Get, GPLoad

FUNCTION

Release - releases any device signals

BGFX

```
RUN bgfx([path,] "Release")
```

DESCRIPTION

Calls SS_Relea to turn off any signals set up by OnMouse or OnKey. You might use this call if you don't with to be interrupted within part of your program.

SEE ALSO

OnKey, OnMouse

FUNCTION

```
RevOn   - turn reversed text on
RevOff  - turn reversed text off
```

CODE

```
1F 20  (on)
1F 21  (off)
```

BGFX

```
RUN bgfx([path.] "RevOn")
RUN bgfx([path.] "RevOff")
```

DESCRIPTION

CoCo compatible codes to turn reverse-characters on or off.

In the reverse mode, text is written in the background color, with backfill (if enabled by non-transparency) in the foreground color.

SEE ALSO

TChrSw, BoldSw, Underline functions

FUNCTION

SBox - Draw a two color shadowed box

CODE

None: BGFX internally does Color and Line calls

BGFX

```
RUN bgfx([path,] "SBox",color1,color2,x1,y1,x2,y2)
```

PARAMETERS

color1	- top and left color
color2	- bottom and right color
x1,y1	- upper left coordinates
x2,y2	- lower right coordinates

DESCRIPTION

Draw a 3D-looking rectangle. For example, if color1 is a light grey, and color2 is black, the resulting box will appear as a "button" protruding towards you. The same colors in reverse would look like a depressed area.

CAVEATS

The current desired foreground color should be reset after using this command.

FUNCTION

ScaleSw - Turn scaling on or off.

CODE

```
1B 35 switch   (00=off, 01=on)
```

BGFX

```
RUN bgfx([path,] "ScaleSw", "on")
RUN bgfx([path,] "ScaleSw", "off")
```

DESCRIPTION

ScaleSw turns graphics coordinate scaling on or off.

SEE ALSO

CWArea

FUNCTION

Select - Select interactive window

CODE

1B 21

BGFX

RUN bgfx([path,] "Select")

DESCRIPTION

Defines the window associated with the specified path as the interactive (input) device window for that process. If the previous interactive window for that program was currently displayed, the display will change to the newly selected window.

Note that graphics and text output can continue to any other path.

NOTES

The display change will also not take place if the previous path is no longer valid (open). This allows neat (and commonly used) tricks when forking other programs from yours...

To change to the other window, and then come back on exit:

1) Open, dwset and select new window path.
2) Fork the subprogram to that new path.
3) Wait for subprogram to exit.
4) Reselect original window (display will flip).
5) Close new window path.

To change to the other window, but leave it displayed:

1) Open, dwset and select new window path.
2) Fork the subprogram to that new path.
3) Close new window path
4) Reselect original window (display will not change).

FUNCTION

SetDPtr, RSetDPtr - Set Draw Pointer Position

BGFX

```
RUN bgfx([path,] "SetDPtr",x,y)
RUN bgfx([path,] "RSetDPtr",xo,yo)
```

PARAMETERS

```
x,y    = direct window coordinate addressing
xo,yo = relative addressing (from last dptr x,y)
```

DESCRIPTION

SetDPtr positions the draw pointer at position _x,y_ , in relation to the upper left corner of the working region of the window. RSetDPtr positions the draw pointer at offset _xo,yo_ , from the draw pointer's current position.

SEE ALSO

ScaleSw, CWArea

FUNCTION

SetPals - Set multiple palette values

BGFX

```
RUN bgfx([path,] "SetPals",array,first,count)
```

PARAMETERS

path	=	optional path to window
array	=	buffer containing palette data
first	=	first paletter register to set
count	=	number of registers to set

DESCRIPTION

SetPals updates the values for the number of palette registers specified by _count_ starting at the CLUT (Color Look Up Table) offset passed in _first_. A copy of the color information is passed to the hardware from the buffer pointed to by _array_. SetPals allows setting any or all the palette registers information at one time.

CAVEATS

Array should be large enough to hold 3 (R,G,B) times the count.

To reserve space for all 256 palettes, use something like:

```
  DIM palettes(768):BYTE
```

Count is the number of triplets (R,G,B bytes) to copy, NOT the length of the palette data in bytes.

FUNCTION

Sleep - Put calling process to sleep

BGFX

```
RUN bgfx("Sleep",ticks)
```

PARAMETERS

ticks	- time in 1/100ths of a second
	- or zero to sleep until signal

DESCRIPTION

This function calls F$Sleep. There are two main uses:

First, in place of "busy loops" for program delays such as pausing between Puts during an animation.

Second, when used with a parameter of zero and in conjunction with OnKey and/or OnMouse signal calls, the process can easily be suspended until the user hits a key or mouse button.

SEE ALSO

OnKey, OnMouse

FUNCTION

TChrSw - Set/reset transparent character attribute

CODE

```
1B 3C switch    (00=off, 01=on)
```

BGFX

```
RUN bgfx([path,] "TChrSw", "on")
RUN bgfx([path,] "TChrSw", "off")
```

DESCRIPTION

TChrSw turns text transparency on or off for anv following text output.

When set to "on", only the printable text pixels are set in the current foreground color... the emptv pixels are left alone.

This is handv for printing labels within graphic drawings, or for not affecting predrawn backgrounds.

SEE ALSO

BoldSw, PropSw

FUNCTION

UndLnOn - turn text underlining on
UndLnOff - turn text underlining off

CODE

```
1F 22  (on)
1F 23  (off)
```

BGFX

```
RUN bgfx([path,] "UndLnOn")
RUN bgfx([path,] "UndLnOff")
```

DESCRIPTION

CoCo compatible codes to turn underlining on or off.

CAVEATS

Backspacing does not delete underline.

SEE ALSO

TChrSw, BoldSw, Reverse functions

FUNCTION

WInfo - Get information about window

BGFX

```
RUN bgfx([path,] wtype,xsize,ysize,fore,back,border)
```

PARAMETERS

wtype	= returned window type
xsize	= returned size in pixels
ysize	= returned size in pixels
fore	= foregnd palette number
back	= backgnd palette number
border	= border palette number

DESCRIPTION

WInfo returns various information about the window associated with path.