Summary of the Window Manager

Pascal Summary

Constants

```
CONST
   {window types}
   documentProc      = 0;  {movable, sizable window, no zoom box}
   dBoxProc          = 1;  {alert box or modal dialog box}
   plainDBox         = 2;  {plain box}
   altDBoxProc       = 3;  {plain box with shadow}
   noGrowDocProc     = 4;  {movable window, no size box or }
                           { zoom box}
   movableDBoxProc   = 5;  {movable modal dialog box}
   zoomDocProc       = 8;  {standard document window}
   zoomNoGrow        = 12; {zoomable, nonresizable window}
   rDocProc          = 16; {rounded-corner window}

   {window kinds}
   dialogKind        = 2;  {dialog or alert box window}
   userKind          = 8;  {window created by the application}

   {part codes returned by FindWindow}
   inDesk            = 0;  {none of the following}
   inMenuBar         = 1;  {in menu bar}
   inSysWindow       = 2;  {in desk accessory window}
   inContent         = 3;  {anywhere in content region except size }
                           { box if window is active, }
                           { anywhere including size box if window }
                           { is inactive}
   inDrag            = 4;  {in drag (title bar) region}
   inGrow            = 5;  {in size box (active window only)}
   inGoAway          = 6;  {in close box}
   inZoomIn          = 7;  {in zoom box (window in standard state)}
   inZoomOut         = 8;  {in zoom box (window in user state)}

   {axis constraints on DragGrayRgn}
   noConstraint      = 0;  {no constraints}
   hAxisOnly         = 1;  {move on horizontal axis only}
   vAxisOnly         = 2;  {move on vertical axis only}
```

```
{window definition function task codes}
wDraw       = 0;  {draw window frame}
wHit        = 1;  {report where mouse-down occurred}
wCalcRgns   = 2;  {calculate strucRgn and contRgn}
wNew        = 3;  {perform additional initialization}
wDispose    = 4;  {perform additional disposal tasks}
wGrow       = 5;  {draw grow image during resizing}
wDrawGIcon  = 6;  {draw size box and scroll bar outline}

{window definition function wHit return codes}
wNoHit      = 0;  {none of the following}
wInContent  = 1;  {anywhere in content region except size }
                  { box if window is active, }
                  { anywhere including size box if window }
                  { is inactive}
wInDrag     = 2;  {in drag (title bar) region}
wInGrow     = 3;  {in size box (active window only)}
wInGoAway   = 4;  {in close box}
wInZoomIn   = 5;  {in zoom box (window in standard state)}
wInZoomOut  = 6;  {in zoom box (window in user state)}

{window color information table part codes}
wContentColor     = 0;     {content region background}
wFrameColor       = 1;     {window outline}
wTextColor        = 2;     {window title and button text}
wHiliteColor      = 3;     {reserved}
wTitleBarColor    = 4;     {reserved}
wHiliteColorLight = 5;     {lightest stripes in title bar }
                           { and lightest dimmed text}
wHiliteColorDark  = 6;     {darkest stripes in title bar }
                           { and darkest dimmed text}
wTitleBarLight    = 7;     {lightest parts of title bar background}
wTitleBarDark     = 8;     {darkest parts of title bar background}
wDialogLight      = 9;     {lightest element of dialog box frame}
wDialogDark       = 10;    {darkest element of dialog box frame}
wTingeLight       = 11;    {lightest window tinging}
wTingeDark        = 12;    {darkest window tinging}

{resource ID of desktop pattern}
deskPatID         = 16;
```

Data Types

```
TYPE  CWindowPtr  = CGrafPtr;
      CWindowPeek = ^CWindowRecord;

CWindowRecord =
RECORD
   port:          CGrafPort;     {window's graphics port}
   windowKind:    Integer;       {class of window}
   visible:       Boolean;       {visibility}
   hilited:       Boolean;       {highlighting}
   goAwayFlag:    Boolean;       {presence of close box}
   spareFlag:     Boolean;       {presence of zoom box}
   strucRgn:      RgnHandle;     {handle to structure region}
   contRgn:       RgnHandle;     {handle to content region}
   updateRgn:     RgnHandle;     {handle to update region}
   windowDefProc: Handle;        {handle to window definition function}
   dataHandle:    Handle;        {handle to window state data record}
   titleHandle:   StringHandle;  {handle to window title}
   titleWidth:    Integer;       {title width in pixels}
   controlList:   ControlHandle; {handle to control list}
   nextWindow:    CWindowPeek;   {pointer to next window record in }
                                 { window list}
   windowPic:     PicHandle;     {handle to optional picture}
   refCon:        LongInt;       {storage available to your application}
END;

WindowPtr   = GrafPtr;
WindowPeek  = ^WindowRecord;

WindowRecord =
RECORD                           {all fields have same use as }
                                 { in color window record}
   port:          GrafPort;      {window's graphics port}
   windowKind:    Integer;       {class of window}
   visible:       Boolean;       {visibility}
   hilited:       Boolean;       {highlighting}
   goAwayFlag:    Boolean;       {presence of close box}
   spareFlag:     Boolean;       {presence of zoom box}
   strucRgn:      RgnHandle;     {handle to structure region}
   contRgn:       RgnHandle;     {handle to content region}
   updateRgn:     RgnHandle;     {handle to update region}
   windowDefProc: Handle;        {handle to window definition function}
   dataHandle:    Handle;        {handle to window state data record}
```

```
   titleHandle:   StringHandle;  {handle to window title}
   titleWidth:    Integer;       {title width in pixels}
   controlList:   ControlHandle; {handle to control list}
   nextWindow:    WindowPeek;    {pointer to next window record in }
                                 { window list}
   windowPic:     PicHandle;     {handle to optional picture}
   refCon:        LongInt;       {storage available to your application}
END;

WStateDataPtr = ^WStateData;
WStateDataHandle = ^WStateDataPtr;

WStateData =            {zoom state data record}
RECORD
   userState:  Rect;    {size and location established by user}
   stdState:   Rect;    {size and location established by application}
END;

WCTabPtr = ^WinCTab;
WCTabHandle = ^WCTabPtr;

WinCTab =                       {window color information table}
RECORD
   wCSeed:     LongInt;         {reserved}
   wCReserved: Integer;         {reserved}
   ctSize:     Integer;         {number of entries in table -1}
   ctTable:    ARRAY [0..4] OF ColorSpec;
                                {array of color specification records}
END;

ColorSpec   =
RECORD
   value:      Integer;         {part identifier}
   rgb:        RGBColor;        {RGB value}
END;

AuxWinHandle= ^AuxWinPtr;
AuxWinPtr   = ^AuxWinRec;

AuxWinRec   =                   {auxiliary window record}
RECORD
   awNext:        AuxWinHandle;  {handle to next record}
   awOwner:       WindowPtr;     {pointer to window}
   awCTable:      CTabHandle;    {handle to color table}
   dialogCItem:   Handle;        {storage used by Dialog Manager}
```

```
   awFlags:       LongInt;       {reserved}
   awReserved:    CTabHandle;    {reserved}
   awRefCon:      LongInt;       {reference constant, for }
                                 { use by application}
END;
```

Window Manager Routines

Initializing the Window Manager

```
PROCEDURE InitWindows;
```

Creating Windows

```
FUNCTION GetNewCWindow      (windowID: Integer; wStorage: Ptr;
                             behind: WindowPtr): WindowPtr;
FUNCTION GetNewWindow       (windowID: Integer; wStorage: Ptr;
                             behind: WindowPtr): WindowPtr;
FUNCTION NewCWindow         (wStorage: Ptr; boundsRect: Rect;
                             title: Str255; visible: Boolean;
                             procID: Integer; behind: WindowPtr;
                             goAwayFlag: Boolean;
                             refCon: LongInt): WindowPtr;
FUNCTION NewWindow          (wStorage: Ptr; boundsRect: Rect;
                             title: Str255; visible: Boolean;
                             theProc: Integer; behind: WindowPtr;
                             goAwayFlag: Boolean;
                             refCon: LongInt): WindowPtr;
```

Naming Windows

```
PROCEDURE SetWTitle         (theWindow: WindowPtr; title: Str255);
PROCEDURE GetWTitle         (theWindow: WindowPtr; VAR title: Str255);
```

Displaying Windows

```
PROCEDURE DrawGrowIcon      (theWindow: WindowPtr);
PROCEDURE SelectWindow      (theWindow: WindowPtr);
PROCEDURE ShowWindow        (theWindow: WindowPtr);
PROCEDURE HideWindow        (theWindow: WindowPtr);
PROCEDURE ShowHide          (theWindow: WindowPtr; showFlag: Boolean);
PROCEDURE HiliteWindow      (theWindow: WindowPtr; fHilite: Boolean);
PROCEDURE BringToFront      (theWindow: WindowPtr);
PROCEDURE SendBehind        (theWindow, behindWindow: WindowPtr);
```

Retrieving Window Information

```
FUNCTION FindWindow         (thePoint: Point;
                             VAR theWindow: WindowPtr): Integer;
FUNCTION FrontWindow        : WindowPtr;
```

Moving Windows

```
PROCEDURE DragWindow        (theWindow: WindowPtr;
                             startPt: Point; boundsRect: Rect);
PROCEDURE MoveWindow        (theWindow: WindowPtr;
                             hGlobal, vGlobal: Integer; front: Boolean);
FUNCTION DragGrayRgn        (theRgn: RgnHandle; startPt: Point;
                             limitRect, slopRect: Rect; axis: Integer;
                             actionProc: ProcPtr): LongInt;
FUNCTION PinRect            (theRect: Rect; thePt: Point): LongInt;
```

Resizing Windows

```
FUNCTION GrowWindow         (theWindow: WindowPtr;
                             startPt: Point; sizeRect: Rect): LongInt;
PROCEDURE SizeWindow        (theWindow: WindowPtr; w, h: Integer;
                             fUpdate: Boolean);
```

Zooming Windows

```
FUNCTION TrackBox           (theWindow: WindowPtr; thePt: Point;
                             partCode: Integer): Boolean;
PROCEDURE ZoomWindow        (theWindow: WindowPtr;
                             partCode: Integer; front: Boolean);
```

Closing and Deallocating Windows

```
FUNCTION TrackGoAway        (theWindow: WindowPtr; thePt: Point): Boolean;
PROCEDURE CloseWindow       (theWindow: WindowPtr);
PROCEDURE DisposeWindow     (theWindow: WindowPtr);
```

Maintaining the Update Region

```
PROCEDURE BeginUpdate       (theWindow: WindowPtr);
PROCEDURE EndUpdate         (theWindow: WindowPtr);
PROCEDURE InvalRect         (badRect: Rect);
PROCEDURE InvalRgn          (badRgn: RgnHandle);
PROCEDURE ValidRect         (goodRect: Rect);
PROCEDURE ValidRgn          (goodRgn: RgnHandle);
```

Setting and Retrieving Other Window Characteristics

```
PROCEDURE SetWindowPic      (theWindow: WindowPtr; Pic: PicHandle);
FUNCTION GetWindowPic       (theWindow: WindowPtr): PicHandle;
PROCEDURE SetWRefCon        (theWindow: WindowPtr; data: LongInt);
FUNCTION GetWRefCon         (theWindow: WindowPtr): LongInt;
FUNCTION GetWVariant        (theWindow: WindowPtr): Integer;
```

Manipulating the Desktop

```
PROCEDURE SetDeskCPat       (deskPixPat: PixPatHandle);
FUNCTION GetGrayRgn         : RgnHandle;
PROCEDURE GetCWMgrPort      (VAR wMgrCPort: CGrafPtr);
PROCEDURE GetWMgrPort       (VAR wPort: GrafPtr);
```

Manipulating Window Color Information

```
PROCEDURE SetWinColor       (theWindow: WindowPtr;
                             newColorTable: WCTabHandle);
FUNCTION GetAuxWin          (theWindow: WindowPtr;
                             VAR awHndl: AuxWinHandle): Boolean;
```

Low-Level Routines

```
FUNCTION CheckUpdate        (VAR theEvent: EventRecord): Boolean;
PROCEDURE ClipAbove         (window: WindowPeek);
PROCEDURE SaveOld           (window: WindowPeek);
PROCEDURE DrawNew           (window: WindowPeek; update: Boolean);
PROCEDURE PaintOne          (window: WindowPeek; clobberedRgn: RgnHandle);
PROCEDURE PaintBehind       (startWindow: WindowPeek;
                             clobberedRgn: RgnHandle);
PROCEDURE CalcVis           (window: WindowPeek);
PROCEDURE CalcVisBehind     (startWindow: WindowPeek;
                             clobberedRgn: RgnHandle);
```

Application-Defined Routine

The Window Definition Function

```
FUNCTION MyWindow           (varCode: Integer; theWindow: WindowPtr;
                             message: Integer; param: LongInt): LongInt;
```

C Summary

Constants

```
enum {
   /*window types*/
   documentProc      = 0,  /*movable, sizable window, no zoom box*/
   dBoxProc          = 1,  /*alert box or modal dialog box*/
   plainDBox         = 2,  /*plain box*/
   altDBoxProc       = 3,  /*plain box with shadow*/
   noGrowDocProc     = 4,  /*movable window, no size box or zoom box*/
   movableDBoxProc   = 5,  /*movable modal dialog box*/
   zoomDocProc       = 8,  /*standard document window*/
   zoomNoGrow        = 9,  /*zoomable, nonresizable window*/
   rDocProc          = 16, /*rounded-corner window*/

   /*window kinds*/
   dialogKind        = 2,  /*dialog or alert box window*/
   userKind          = 8,  /*window created by the application*/

   /*part codes returned by FindWindow*/
   inDesk            = 0,  /*none of the following*/
   inMenuBar         = 1,  /*in menu bar*/
   inSysWindow       = 2,  /*in desk accessory window*/
   inContent         = 3,  /*anywhere in content region except size box if*/
                           /* window is active, anywhere including */
                           /* size box if window is inactive*/
   inDrag            = 4,  /*in drag (title bar) region*/
   inGrow            = 5,  /*in size box (active window only)*/
   inGoAway          = 6,  /*in close box*/
   inZoomIn          = 7,  /*in zoom box (window in standard state)*/
   inZoomOut         = 8   /*in zoom box (window in user state)*/
};

enum {
   /*axis constraints on DragGrayRgn*/
   noConstraint      = 0,  /*no constraints*/
   hAxisOnly         = 1,  /*move on horizontal axis only*/
   vAxisOnly         = 2   /*move on vertical axis only*/
};
```

```
enum {
   /*window definition function task codes*/
   wDraw       = 0,  /*draw window frame*/
   wHit        = 1,  /*report where mouse-down occurred*/
   wCalcRgns   = 2,  /*calculate strucRgn and contRgn*/
   wNew        = 3,  /*perform additional initialization*/
   wDispose    = 4,  /*perform additional disposal tasks*/
   wGrow       = 5,  /*draw grow image during resizing*/
   wDrawGIcon  = 6,  /*draw size box and scroll bar outline*/

   /*window definition function wHit return codes*/
   wNoHit      = 0,  /*none of the following*/
   wInContent  = 1,  /*in content region (except grow, if active)*/
   wInDrag     = 2,  /*in drag region*/
   wInGrow     = 3,  /*in grow region (active window only)*/
   wInGoAway   = 4,  /*in go-away region (active window only)*/
   wInZoomIn   = 5,  /*in zoom box for zooming in (active window */
                     /* only)*/
   wInZoomOut  = 6,  /*in zoom box for zooming out (active window */
                     /* only)*/
   deskPatID   = 16, /*resource ID of desktop pattern*/

   /*window color information table part codes*/
   wContentColor     = 0,     /*the background of the window's */
                              /* content region*/
   wFrameColor       = 1,     /*the window outline*/
   wTextColor        = 2,     /*window title and text in buttons*/
   wHiliteColor      = 3,     /*reserved*/
   wTitleBarColor    = 4,     /*reserved*/
   wHiliteColorLight = 5,     /*lightest stripes in title bar */
                              /* and lightest dimmed text*/
   wHiliteColorDark  = 6,     /*darkest stripes in title bar */
                              /* and darkest dimmed text*/
   wTitleBarLight    = 7,     /*lightest parts of title bar background*/
   wTitleBarDark     = 8,     /*darkest parts of title bar background*/
   wDialogLight      = 9,     /*lightest element of dialog box frame*/
   wDialogDark       = 10,    /*darkest element of dialog box frame*/
   wTingeLight       = 11,    /*lightest window tinging*/
   wTingeDark        = 12     /*darkest window tinging*/
};
```

Data Types

```
struct CWindowRecord {
   CGrafPort            port;          /*window's graphics port*/
   short                windowKind;    /*class of the window*/
   Boolean              visible;       /*visibility*/
   Boolean              hilited;       /*highlighting*/
   Boolean              goAwayFlag;    /*presence of close box*/
   Boolean              spareFlag;     /*presence of zoom box*/
   RgnHandle            strucRgn;      /*handle to structure region*/
   RgnHandle            contRgn;       /*handle to content region*/
   RgnHandle            updateRgn;     /*handle to update region*/
   Handle               windowDefProc; /*handle to window definition */
                                       /* function*/
   Handle               dataHandle;    /*handle to window state data record*/
   StringHandle         titleHandle;   /*handle to window title*/
   short                titleWidth;    /*title width in pixels*/
   ControlHandle        controlList;   /*handle to control list*/
   struct CWindowRecord *nextWindow;   /*next window in window list*/
   PicHandle            windowPic;     /*handle to optional picture*/
   long                 refCon;        /*storage available to your */
                                       /* application*/
};

typedef struct CWindowRecord CWindowRecord;
typedef CWindowRecord *CWindowPeek;

struct WindowRecord {
   GrafPort             port;          /*window's graphics port*/
   short                windowKind;    /*class of the window*/
   Boolean              visible;       /*visibility*/
   Boolean              hilited;       /*highlighting*/
   Boolean              goAwayFlag;    /*presence of close box*/
   Boolean              spareFlag;     /*presence of zoom box*/
   RgnHandle            strucRgn;      /*handle to structure region*/
   RgnHandle            contRgn;       /*handle to content region*/
   RgnHandle            updateRgn;     /*handle to update region*/
   Handle               windowDefProc; /*handle to window definition */
                                       /* function*/
   Handle               dataHandle;    /*handle to window state data record*/
   StringHandle         titleHandle;   /*handle to window title*/
   short                titleWidth;    /*title width in pixels*/
   ControlHandle        controlList;   /*handle to window's control list*/
   struct WindowRecord  *nextWindow;   /*next window in window list*/
```

```
   PicHandle            windowPic;     /*handle to optional picture*/
   long                 refCon;        /*reference constant*/
};

typedef struct WindowRecord WindowRecord;
typedef WindowRecord *WindowPeek;

struct WStateData {
   Rect  userState;  /*user state*/
   Rect  stdState;   /*standard state*/
};

typedef struct WStateData WStateData;
typedef WStateData *WStateDataPtr, **WStateDataHandle;

struct AuxWinRec {
    struct AuxWinRec **awNext;      /*handle to next record*/
    WindowPtr        awOwner;       /*pointer to window */
    CTabHandle       awCTable;      /*handle to color table*/
    Handle           dialogCItem;   /*storage used by Dialog Manager*/
    long             awFlags;       /*reserved*/
    CTabHandle       awReserved;    /*reserved*/
    long             awRefCon;      /*reference constant, for use by */
                                    /* application*/
};

typedef struct AuxWinRec AuxWinRec;
typedef AuxWinRec *AuxWinPtr, **AuxWinHandle;

struct WinCTab {
   long        wCSeed;           /*reserved*/
   short       wCReserved;       /*reserved*/
   short       ctSize;           /*number of entries in table –1*/
   ColorSpec   ctTable[5];       /*array of color specification records*/
};

typedef struct WinCTab WinCTab;
typedef WinCTab *WCTabPtr, **WCTabHandle;
```

Window Manager Routines

Initializing the Window Manager

```
pascal void InitWindows(void);
```

Creating Windows

```
pascal WindowPtr GetNewCWindow
                            (short windowID, void *wStorage,
                             WindowPtr behind);
pascal WindowPtr GetNewWindow
                            (short windowID, void *wStorage,
                             WindowPtr behind);
pascal WindowPtr NewCWindow (void *wStorage, const Rect *boundsRect,
                             ConstStr255Param title, Boolean visible,
                             short procID, WindowPtr behind,
                             Boolean goAwayFlag, long refCon);
pascal WindowPtr NewWindow  (void *wStorage, const Rect *boundsRect,
                             ConstStr255Param title, Boolean visible,
                             short theProc, WindowPtr behind,
                             Boolean goAwayFlag, long refCon);
```

Naming Windows

```
pascal void SetWTitle       (WindowPtr theWindow, ConstStr255Param title);
pascal void GetWTitle       (WindowPtr theWindow, Str255 title);
```

Displaying Windows

```
pascal void DrawGrowIcon    (WindowPtr theWindow);
pascal void SelectWindow    (WindowPtr theWindow);
pascal void ShowWindow      (WindowPtr theWindow);
pascal void HideWindow      (WindowPtr theWindow);
pascal void ShowHide        (WindowPtr theWindow, Boolean showFlag);
pascal void HiliteWindow    (WindowPtr theWindow, Boolean fHilite);
pascal void BringToFront    (WindowPtr theWindow);
pascal void SendBehind      (WindowPtr theWindow, WindowPtr behindWindow);
```

Retrieving Mouse Information

```
pascal short FindWindow     (Point thePoint, WindowPtr *theWindow);
pascal WindowPtr FrontWindow(void);
```

Moving Windows

```
pascal void DragWindow      (WindowPtr theWindow, Point startPt,
                             const Rect *boundsRect);
pascal void MoveWindow      (WindowPtr theWindow, short hGlobal,
                             short vGlobal, Boolean front);
```

```
pascal long DragGrayRgn     (RgnHandle theRgn, Point startPt,
                             const Rect *boundsRect,
                             const Rect *slopRect,
                             short axis, DragGrayRgnProcPtr actionProc);
pascal long PinRect         (const Rect *theRect, Point *thePt);
```

Resizing Windows

```
pascal long GrowWindow      (WindowPtr theWindow, Point startPt,
                             const Rect *bBox);
pascal void SizeWindow      (WindowPtr theWindow, short w, short h,
                             Boolean fUpdate);
```

Zooming Windows

```
pascal Boolean TrackBox     (WindowPtr theWindow, Point thePt,
                             short partCode);
pascal void ZoomWindow      (WindowPtr theWindow, short partCode,
                             Boolean front);
```

Closing and Deallocating Windows

```
pascal Boolean TrackGoAway  (WindowPtr theWindow, Point thePt);
pascal void CloseWindow     (WindowPtr theWindow);
pascal void DisposeWindow   (WindowPtr theWindow);
```

Maintaining the Update Region

```
pascal void BeginUpdate     (WindowPtr theWindow);
pascal void EndUpdate       (WindowPtr theWindow);
pascal void InvalRect       (const Rect *badRect);
pascal void InvalRgn        (RgnHandle badRgn);
pascal void ValidRect       (const Rect *goodRect);
pascal void ValidRgn        (RgnHandle goodRgn);
```

Setting and Retrieving Other Window Characteristics

```
pascal void SetWindowPic    (WindowPtr theWindow, PicHandle pic);
pascal PicHandle GetWindowPic
                            (WindowPtr theWindow);
pascal void SetWRefCon      (WindowPtr theWindow, long data);
pascal long GetWRefCon      (WindowPtr theWindow);
pascal short GetWVariant    (WindowPtr theWindow);
```

Manipulating the Desktop

```
pascal void SetDeskCPat     (PixPatHandle deskPixPat);
#define GetGrayRgn()        (* (RgnHandle* 0X09EE))
pascal void GetCWMgrPort    (CGrafPtr *wMgrCPort);
pascal void GetWMgrPort     (GrafPtr *wPort);
```

Manipulating Window Color Information

```
pascal void SetWinColor     (WindowPtr theWindow,
                             WCTabHandle newColorTable);
pascal Boolean GetAuxWin    (WindowPtr theWindow, AuxWinHandle *awHndl);
```

Low-Level Routines

```
pascal Boolean CheckUpdate  (EventRecord *theEvent);
pascal void ClipAbove       (WindowPeek window;)
pascal void SaveOld         (WindowPeek window);
pascal void DrawNew         (WindowPeek window, Boolean update);
pascal void PaintOne        (WindowPeek window, RgnHandle clobberedRgn);
pascal void PaintBehind     (WindowPeek startWindow,
                             RgnHandle clobberedRgn);
pascal void CalcVis         (WindowPeek window);
pascal void CalcVisBehind   (WindowPeek startWindow,
                             RgnHandle clobberedRgn);
```

Application-Defined Routine

The Window Definition Function

```
pascal long MyWindow        (short varCode, WindowPtr theWindow,
                             short message, long param);
```

Assembly-Language Summary

Data Types

Window Record and Color Window Record Data Structure

0	`windowPort`	108 bytes	window's graphics port
108	`windowKind`	word	how window was created
110	`wVisible`	byte	visibility status
111	`wHilited`	byte	highlighted status
112	`wGoAway`	byte	presence of close box
113	`wZoom`	byte	presence of zoom box
114	`structRgn`	long	handle to structure region
118	`contRgn`	long	handle to content region
122	`updateRgn`	long	handle to update region
126	`windowDef`	long	handle to window definition function
130	`wDataHandle`	long	handle to window state data record
134	`wTitleHandle`	long	handle to window's title
138	`wTitleWidth`	word	title width in pixels
140	`wControlList`	long	handle to window's control list
144	`nextWindow`	long	pointer to next window in window list
148	`windowPic`	long	handle to picture for updates
152	`wRefCon`	long	reference constant field

Window State Data Structure

0	`userState`	8 bytes	user state rectangle
8	`stdState`	8 bytes	standard state rectangle

Window Color Information Table Data Structure

0	`ctSeed`	long	ID number for table
4	`ctFlags`	word	flags word
6	`ctSize`	word	number of entries minus 1
8	`ctTable`	variable	a series of color specification records (8 bytes each)

Auxiliary Window Record Data Structure

0	`awNext`	long	handle to next window in chain
4	`awOwner`	long	pointer to associated window record
8	`awCTable`	long	handle to window color information table
12	`dialogCItem`	long	handle to dialog color structures
16	`awFlags`	long	handle for QuickDraw
20	`awResrv`	long	reserved
24	`awRefCon`	long	user constant

Global Variables

AuxWinHead	Handle to beginning of auxiliary window list.
CurActivate	Pointer to window to receive activate event.
CurDeactive	Pointer to window to receive deactivate event.
DeskHook	Address of procedure for painting desktop.
DeskPattern	Pattern in which desktop is painted (8 bytes).
DragHook	Address of optional procedure to execute during TrackGoAway, TrackBox, DragWindow, GrowWindow, and DragGrayRgn.
DragPattern	Pattern of dragged region's outline (8 bytes).
GrayRgn	Handle to desktop region.
OldContent	Handle to saved content region.
OldStructure	Handle to saved structure region.
PaintWhite	Flag indicating whether to paint window white before update event (2 bytes).
SaveUpdate	Flag indicating whether to generate update events (2 bytes).
SaveVisRgn	Handle to saved visible region.
WindowList	Pointer to first window in window list.
WMgrPort	Pointer to Window Manager port.