WindowRecord structure

#include <Windows.h>

typedef struct **WindowRecord** {

		Size	Offset	Description
GrafPort	port;	108	0	portBits, portRect, pnSize,txFont, etc....
short	windowKind;	2	108	holds dialogKind, userKind, or negative refnum of DA.
Boolean	visible;	1	110	TRUE if window is visible **ShowWindow**
Boolean	hilited;	1	111	TRUE if window hilited **HiliteWindow**
Boolean	goAwayFlag;	1	112	TRUE if window has a close box in top left
Boolean	spareFlag;	1	113	TRUE=zoom is enabled **ZoomWindow**
RgnHandle	strucRgn;	4	114	Content region plus the frame (global coords)
RgnHandle	contRgn;	4	118	Content rgn, including scroll bars (global)
RgnHandle	updateRgn;	4	122	Area needing update,(local) **InvalRgn**
Handle	windowDefProc;	4	126	Code to draw the window ('WDEF')
Handle	dataHandle;	4	130	Additional info; may lead to a WStateData struct
StringHandle	titleHandle;	4	134	Leads to pstring of title **SetWTitle**
short	titleWidth;	2	138	Width, in pixels, of the window title text
ControlHandle	controlList;	4	140	Window's first ControlRecord **SetWTitle**
WindowPeek	nextWindow;	4	144	The window behind this one (0 if this is last)
PicHandle	windowPic;	4	148	Leads to Picture; 0=none **SetWindowPic**
long	refCon;	4	152	Anything you want **SetWRefCon**
} **WindowRecord**;		156		

typedef WindowRecord ***WindowPeek**; /* use WindowPeek to access these fields */

typedef GrafPtr **WindowPtr**;/* Note: **Not** a pointer to **WindowRecord** */

Notes: A WindowPeek (ie, the address of a WindowRecord) is used in nearly all Window Manager calls.

The port field is a GrafPort (all 108 bytes of it). It contains such important items as the size and location of the window, the text font and display attributes, etc.

The windowKind field identifies which of the standard or user-defined window definition routines will draw the window.

> **Note**: For desk accessories, windowKind contains the driver reference number (a negative value). This affects how DAs must handle calls to **IsDialogEvent**.

The dataHandle field may contain either four bytes of data (as used in rDocProc type windows), or a handle to additional data needed by the window definition procedure. In the case of zoomable window types, dataHandle is a handle to a WStateData structure.

The nextWindow field contains the address of the next WindowRecord in the Window Manager's list. The global variable WindowList (at 0x09D6) contains the address of the first (frontmost) window in that list.

Notice that a WindowRecord begins with a GrafPort. Similarly, a DialogRecord begins with a WindowRecord (and thus begins with a GrafPort). The data types GrafPtr, WindowPtr, and DialogPtr may be used interchangeably when you pass a pointer to a function which expects a subset:

```
short dlgRsrcID;

myDlg = GetNewDialog(dlgRsrcID, nil, (WindowPtr) -1);
SetPort( myDlg );                  // expects a GrafPtr
ShowWindow( myDlg );               // expects a WindowPtr
```

To access the additional fields of a WindowRecord structure, create a WindowPeek variable:

```
WindowPtr     myWin;
WindowPeek    myWinPeek = (WindowPeek) myWin;

myWin->txFont = geneva;                  // access GrafPort fields
myWinPeek->windowKind = dBoxProc;  // access WindowRecord fields
```

To query the contents of a field, you can use "quick-and-dirty" type coercion:

```
aLong = ((WindowPeek)myWin)->refCon;
```