clySmic

S o f t w a r e

clySmic Icon Bar (Clysbar) Add-In Software Development Kit

Version 1.00 / 1-Nov-92

❃

Introduction

This document is designed to help you create your own Add-Ins (.CLB files) for Clysbar. Examples are in Turbo Pascal for Windows and Borland "C". This document assumes you are a programmer and know how to write a simple Windows DLL (Dynamic Link Library). Requirements are Windows 3.1, Clysbar version 1.70, and a suitable compiler that can create Windows DLLs.

Included Files

ADD-IN.INC	Pascal include file for the Clysbar SDK
ADD-IN.H	"C"header file for the Clysbar SDK
CALEND.PAS	Turbo Pascal for Windows source code for the **Calend** Add-In
LINES.PAS	Turbo Pascal for Windows source code for the **Lines** Add-In
C-CALEND.C	Borland "C" source code for the **Calend** Add-In
CALEND.RES	Resource file for the **Calend** Add-In
LINES.RES	Resource file for the **Calend** Add-In
CBSDK.WRI	This document

Using Add-Ins

Add-Ins are DLLs that are loaded and called by Clysbar. An Add-In button is created by placing the Add-In file in Clysbar's directory and specifying a program entry line in CLYSBAR.INI like:

```
*MYADDIN=
```

This would load the Add-In MYADDIN.CLB. Clysbar can have up to five Add-Ins running at once.

Creating Add-Ins

Any language that can create a Windows DLL can create a Clysbar Add-In. The Add-In is a DLL that has its entry points called by Clysbar when information is needed or somethjing needs to be done, like displaying information.

First decide whether the Add-In needs a timer or not. Then get your code ready that calculates whatever information you wish to display. Then, using the example programs and the API below, create a Windows DLL that exports the API calls. Compile to a DLL and rename .DLL -> .CLB. Then create an entry in your CLYSBAR.INI file for the Add-In, run Clysbar and test.

Flow of Control

After the DLL is loaded, ***InitAddIn*** is called. If there is a version problem, InitAddIn returns InitNotOk, otherwise it returns InitOk. ***TimerNeeded*** is called to see what kind (if any) of timer is needed). If *any* Add-In has asked for a timer, ***AddInTimer*** will be called whenever that timer expires. If more than one Add-In is loaded, the Add-In that asked for the *shortest* timer will cause Clysbar to call *all* Add-Ins at that rate.

Whenever the Add-In button needs to be painted, Clysbar draws the blank button background and then calls ***PaintAddIn***. If the button is pressed and released, ***AddInPressed*** (as well as the paint routines) is called. As Clysbar exits, ***ExitAddIn*** is called.

If you do something in your Add-In that requires the button to be redrawn, you may call ***InvalidateRect(Wnd,Nil,True)*** to cause Clysbar to a) redraw the button background, and b) call the ***PaintAddIn*** procedure.

Add-In API Calls

All functions are exported by ordinals.

InitAddIn

TPW Declaration: Function InitAddIn(CurVer : PChar) : InitResult; Export;
C Declaration: InitResult FAR PASCAL InitAddIn(char far *CurVer)
Ordinal: 1
Purpose: *Perform the Add-In's initialization and version check.*

This proc is called right after the DLL is loaded. You may perform any initialization tasks you need, such as reading an INI file or setting global variables. The Add-In should perform a sanity check that the version of Clysbar it was written for is the same one that's calling it. Clysbar's version is passed in CurVer (in ASCIIZ format, e.g. '1.70' / "1.70") and should be checked against an interal "CBVersion" var in the DLL. If all is ok, return InitOk, otherwise return InitNotOk.

InitResult, InitOk, and InitNotOk are defined in the include/header files.

PaintAddIn

TPW Declaration: Procedure PaintAddIn(Wnd : HWnd; DC : HDC; Pressed : Boolean); Export;
C Declaration: VOID FAR PASCAL PaintAddIn(HWND Wnd, HDC DC, BOOL Pressed)
Ordinal: 2
Purpose: *Paint on the Add-In's button.*

Called whenever the Add-In needs to update its display area. Wnd is a handle to the Add-In button's window, DC is an hDC to use when painting, and Pressed tells you whether the button is *currently* pressed (down). You treat this call as the Add-In's Paint method (WM_PAINT message).

If **Pressed** is True, you can offset any drawing coordinates by (+1,+1) to have the drawing "sink in" just like the button background. If you don't do this, the drawing appears to "float" when the button is pressed. **Pressed** only tells you if the button is currently pressed - to toggle flags because of a press and release, use the ***AddInPressed*** procedure, below.

TimerNeeded

TPW Declaration: Function TimerNeeded : Integer; Export;
C Declaration: int FAR PASCAL TimerNeeded()
Ordinal: 3
Purpose: *Tell Clysbar what kind of timer we need.*

Clysbar calls this proc to "ask" the DLL what kind of timer it needs. Return a value from the Timer Constants list below. Clysbar then sets its own Windows timer to the *fastest* value returned from all Add-Ins. The values are:

Timer Constant	Value	Meaning	Timer Length
ait_None	0	No timer is needed	N/A
ait_Slow	1	Slow timer	30 seconds
ait_Med	2	Medium timer	2 seconds
ait_Fast	3	Fast timer	250 milliseconds (1/4 second)

The ait_xxx constants are declared in the example programs.

AddInTimer

TPW Declaration: Procedure AddInTimer(Wnd : HWnd; DC : HDC); Export;
C Declaration: VOID FAR PASCAL AddInTimer(HWND Wnd, HDC DC)
Ordinal: 4
Purpose: *Proc called when timer expires, perform timed duties.*

This proc is called whenever the timer expires. If you are using animation, here is the place to draw it (see LINES.PAS for an example). Wnd is a handle to the Add-In button's window, DC is an hDC to use when drawing.

AddInPressed

TPW Declaration: Procedure AddInPressed(Wnd : HWnd; DC : HDC); Export;
C Declaration: VOID FAR PASCAL AddInPressed(HWND Wnd, HDC DC)
Ordinal: 5
Purpose: *Proc called when button pressed and released.*

This is called when a button is pressed and released, and is usually used to toggle states or increment variables. Wnd is a handle to the Add-In button's window, DC is an hDC to use when drawing.

ExitAddIn

TPW Declaration: Procedure ExitAddIn; Export;
C Declaration: VOID FAR PASCAL ExitAddIn()
Ordinal: 6
Purpose: *Exit processing for Add-In.*

Any exit processing, such as writing states to an INI file, is done here.

AddInAbout

TPW Declaration: Procedure AddInAbout(Str1,Str2 : PChar;
Var TheIcon : HIcon;
Var TitleCol,TxCol,BkCol : TColorRef); Export;

C Declaration: VOID FAR PASCAL AddInAbout(char far *Str1, char far *Str2,
HICON far *TheIcon,
COLORREF far *TitleCol,
COLORREF far *TxCol,
COLORREF far *BkCol)

Ordinal: 7

Purpose: *Clysbar queries Add-In about itself for About Add-Ins box.*

The Add-In is expected to return two identifying strings (Str1 and Str2). Str1 is usually the title of the Add-In and Str2 is usually a copyright notice. A handle to an icon is also expected, as well as title, foreground and background colors. These are all used in the ***About Add-Ins*** Box that Clysbar will display for each Add-In that's loaded.

Included Example Add-In Source Code

The Turbo Pascal for Windows source code for the two Clysbar Add-Ins **Calend** and **Lines** are provided in CALEND.PAS, CALEND.RES, LINES.PAS and LINES.RES. Calend is also provided in a Borland "C" version (C-CAL.C).

Debugging Your Add-In

A couple of notes on debugging and crashes are in order. I haven't created vast Add-Ins that open windows and dialog boxes when the button is clicked. I'm sure its possible, but Add-Ins were designed mostly for calculating information that is then displayed in the button. But, hey, have fun.

Its fairly easy to crash your Add-In and Clysbar if there's an error in the Add-In. So please don't write me the first time you get a GP fault - keep trying. Also, when they GPF, the Add-In, being a DLL, stays in memory, and when you recompile and rerun, you are still running the *old* Add-In. The only way I know of to fix this is to restart Windows. If anyone knows of a utility to remove "stuck" DLLs from memory, please let me know!

Caveats & Information

These programs and documentation are provided *AS IS* without any warranty, expressed or implied, including but not limited to fitness for a particular purpose. So there.

clySmic Software is not responsible for anything that may happen when you use these products, including hardware damage or information loss.

Ralph B Smith Jr
clySmic Software
P. O. Box 2421
Empire State Plaza
Albany, NY 12220

CompuServe 76156,164