
# *clySmic Icon Bar*
# *Add-In Software Development Kit*

## Version 2.12

*Programs and Documentation are Copyright 1992 - 1994 by clySmic Software.*
*All rights reserved.*

## 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 2.12, and a suitable compiler that can create Windows DLLs. This SDK is provided free of charge as an adjunct to our Shareware product Clysbar. See Clysbar's documentation for complete ordering information.

## Included Files

| | |
|---|---|
| ADD-IN.INC | Pascal include file for the Clysbar Add-In SDK |
| ADD-IN.H | C header file for the Clysbar Add-In 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, ***AddInInit*** is called. If there is a version problem, AddInInit returns InitNotOk, otherwise it returns InitOk. ***AddInTimerNeeded*** is called to see what kind (if any) of timer is needed). If *any* Add-In has asked for a timer, ***AddInTimerTick*** 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 ***AddInPaint***. If the button is pressed and released, ***AddInPressed*** (as well as the paint routines) is called. As Clysbar exits, ***EAddInExit*** 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 ***AddInPaint*** procedure.

## API Call Name Changes

The function names have been changed since version 1.70, and three new functions have been added:

| **Old Name** | **New Name** | **Notes** |
|---|---|---|
| InitAddIn | AddInInit | |
| PaintAddIn | AddInPaint | |
| TimerNeeded | AddInTimerNeeded | |
| AddInPressed | AddInPressed | No change |
| ExitAddIn | AddInExit | |
| AddInAbout | AddInAbout | No change |
| | AddInAcceptDrops | *New function* |
| | AddInDrop | *New function* |
| | AddInGetInfoWinTx | *New function* |

## Add-In API Calls

All functions are exported by ordinals.

## *AddInInit*

**TPW Declaration:** Function AddInInit(CurVer : PChar; GTxBtn : Bool) : InitResult; Export;
**C Declaration:** InitResult FAR PASCAL AddInInit(char far *CurVer, BOOL GTxBtn)
**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. '2.1' / "2.1") and should be checked against an internal "CBVersion" var in the DLL. Note that only the x.x version numbers are passed, this is so a small (0.0x) change can be made to Clysbar and the Add-Ins will still be happy. So, an Add-In compiled for V2.12 will work under 2.13, 2.14, &c. It's only when a V2.20 is released that the Add-Ins will need to be updated.

If all is ok, return InitOk, otherwise return InitNotOk. InitResult, InitOk, and InitNotOk are defined in the include/header files.

## *AddInPaint*

**TPW Declaration:** Procedure AddInPaint(Wnd : HWnd; DC : HDC; Pressed : Boolean); Export;
**C Declaration:** VOID FAR PASCAL AddInPaint(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.

N.B.: The Wnd parameter **might be 0 (NULL)**. If it is, ***do not paint anything!*** This is because your AddIn is on a button bar that hasen't been "entered" for the first time since starting Clysbar.

## *AddInTimerNeeded*

**TPW Declaration:** Function AddInTimerNeeded : Integer; Export;
**C Declaration:** int FAR PASCAL AddInTimerNeeded()
**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.

## *AddInTimerTick*

**TPW Declaration:** Procedure AddInTimerTick(Wnd : HWnd; DC : HDC); Export;
**C Declaration:** VOID FAR PASCAL AddInTimerTick(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.

## *AddInExit*

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

Any exit processing, such as writing information to the add-in's 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.

## *AddInAcceptDrops*

**TPW Declaration:** Function AddInAcceptDrops : Boolean;
**C Declaration:** BOOL FAR PASCAL AddInAcceptDrops()
**Ordinal:** 8
**Purpose:** *Clysbar queries Add-In whether it will accept drops.*

An Add-In can accept drops from a drag 'n' drop server (such as File Manager). If you want your AddIn to accept drops, return True to this function. You will receive a file name via ***AddInDrop*** when a file is dropped on your AddIn's button. If you do not want to accept drops, return False. Then ***AddInDrop*** will never be called.

## *AddInDrop*

**TPW Declaration:** Procedure AddInDrop(hDrop : THandle);
**C Declaration:** VOID FAR PASCAL AddInDrop(HANDLE hDrop)
**Ordinal:** 9
**Purpose:** *Clysbar detects a file dropped onto the AddIn's button.*

You can only receive this call if you've returned True to ***AddInAcceptDrops***. hDrop is handle to an internal data structure describing the dropped files. See the Windows SDK documentation of the WM_DROPFILES message for more information. You may do anything you wish with the drop information: delete the file, send it somewhere, &c.

## *AddInGetInfoWinTx*

**TPW Declaration:** Procedure AddInGetInfoWinTx(Tx : PChar);
**C Declaration:** VOID FAR PASCAL AddInGetInfoWinTx(char far *Tx)
**Ordinal:** 10
**Purpose:** *Clysbar asks Add-In for alternate text for the AddIn's Info Window.*

Normally, an Add-In's button shows "<name> AddIn" when the right mouse button is clicked over it, where <name> is the name of the AddIn. The AddIn can substitute any text it wishes by returning a string to this call. **Important:** in order to keep the original, default text, the AddIn must return a ***zero-length string, not a Nil (NULL) pointer***.

## Example Add-In Source Code

The Borland 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.

For C programmers, Calend is provided in a Borland C version (C-CALEND.C and C-CALEND.DEF). The code has been compiled cleanly in BC++ 4.0, large model.

## 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 contact us 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. To fix this, either restart Windows or use a utility like REMDLL (from Windows Tech Journal) or WPS.

## 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.


**P.O. Box 2421**
**Empire State Plaza**
**Albany, N.Y. 12220**

***e-mail: 76156.164@compuserve.com***

***clySmic Software is a member of the***
***Association of Shareware Professionals***