NOTE: MCIWnd is an unsupported API for the Video for Windows 1.0 Developer's Kit and is included as a sample only. Your application is free to use mciwnd.lib and the APIs for mciwnd but you do so at your own risk and PSS will not answer questions about this component if you ask.

Using MCIWND to Make Developing MCI Applications Easier

Overview

Using the MCI interface to allow your application to play MCI device files can sometimes be confusing and complicated, especially with all of the different commands that a device might support. Also, there is no user interface built in to MCI to let the user control the playback of files - an application must provide its own scrollbar and buttons for the user to play, pause, rewind, or seek through a file, or not provide this service at all.

MCIWND is a library that your application can link to that will create a new class of window. Your application needs only to create a window of this class, and then send it a message to open an MCI file. It can then send other messages to control the playback of the file, or give the user this control with the built in toolbar, scrollbar and menus.

MCIPLAY.C is sample code of an Multiple Document Interface (MDI) application that uses this window class to allow the playback and control of multiple MCI devices/files.

Services of the MCIWND window class

- A Toolbar with a PLAY, PAUSE and STOP button
- A Trackbar (scrollbar) to allow seeking within a file
- A Pop-Up Track Menu with some common commands when the right mouse button is clicked over the window
- Simple single-command macros for many of the common MCI commands which eliminates the need for longer, multi-line mciSendCommand or mciSendString calls.

Using MCIWND

- Include the file **VFW.H** in your application's source files to give you access to function prototypes and Macros and defines you will need.
- Link your application with **MCIWND.LIB** to get the new functionality.

- Call **MCIWndRegisterClass(HINSTANCE hInst)** to register the new Window Class **"MCIWND_WINDOW_CLASS"**. This function returns TRUE if successful.

- Use the standard windows function **CreateWindow()** to create a window to play an MCI file inside of.

  or...

  Use the **MCIWndCreate(HWND hwndParent, HINSTANCE hInst, DWORD dwStyle, LPSTR szFile)** function. This takes the place of CreateWindow() and has the advantage of being able to open an MCI device in the same call as the window is created in. NOTE: You do not need to call MCIWndRegisterClass() if you use this function, only if you use CreateWindow().

  For non-windowed devices, you will need a window to hold the toolbar and trackbar. If there will be no controls, and you are playing a non-windowed device (EG playing a Sound file) you may want to leave the window invisible.

  For both of these calls, you have some new styles you can choose from as well as the standard window styles. They are as follows:

  • **MCIWND_NOAUTOSIZE** Let the app size the window. The default is to automatically size the created window to a default size big enough for the window and the playbar (if used).

  • **MCIWND_NOPLAYBAR** Do not put a playbar (Toolbar and Trackbar) in the window. By default, a playbar appears. It provides a PLAY, PAUSE and a STOP button, as well as a Trackbar to seek through the file.

  • **MCIWND_NORESIZETOWINDOW** For MCI devices that can window (have video to display), MCIWND will ordinarily resize the image to any size you make the window. This flag inhibits that action. (The image is a constant size regardless of the size of the window containing it).

  • **MCIWND_NOTRACKMENU** Do not provide a pop-up menu. Normally, pressing the right mouse button over the window will bring up a track menu with commands for Play, Pause, Stop, Rewind, Volume, Speed, and for windowed devices, a Window (zoom) command.

  • **MCIWND_NOTIFYSTATE** Whenever the state of the device changes (eg. from Stop to Play) the parent window will receive a MCIWNDM_NOTIFYSTATE msg with an lParam of the new state of the device (eg. MCI_MODE_STOP).

  • **MCIWND_NOTIFYPOS** Whenever the position of the device changes (eg. as it's playing) the parent window will receive a MCIWNDM_NOTIFYPOS msg with an lParam of the new position in the media.

  • **MCIWND_SHOWNAME** Sets the window text of the window to the filename of any MCI file you open in this window.

- Open an MCI file or device using the macro **MCIWndOpen(HWND hwnd, LPSTR sz, UINT f)**. Hwnd is the window you have created, and sz is the name of the file or device to open. F is currently unused and should be set to 0.

- Send the window a command using one of the following macros. Unless otherwise specified, the return code is the same as you would get from mciSendString() using the same command.

  - **MCIWndClose(hwnd)** Close the MCI file. You can then re-open another file in the same window, or just call open again and the currrent file will close automatically.

  - **MCIWndPlay(hwnd)** Play the file from the current position.

  - **MCIWndStop(hwnd)** Stops the device.

  - **MCIWndPause(hwnd)** Pauses the device.

  - **MCIWndResume(hwnd)** Resumes playing (after a pause).

  - **MCIWndSeek(hwnd, lPos)** Seeks to a specified position in the file. If lpos == MCIWND_SEEKSTART it will seek to the beginning. If lpos == MCIWND_SEEKEND it will seek to the end.

  - **MCIWndPlayReverse(hwnd)** Play the file backwards starting at the current position.

  - **MCIWndGetMode(hwnd)** Returns the current mode of the device (eg. MCI_MODE_PLAY.

  - **MCIWndGetDeviceID(hwnd)** Gets the deviceID of the open file which you will need if you wish to call mciSendCommand or mciSendString to do any commands that are not supported by this interface.

  - **MCIWndGetStart(hwnd)** Returns the starting position of the file. Seeking here will place the file at the beginning

  - **MCIWndGetLength(hwnd)** Returns the length of the file. The start plus the length will give you the end of the media.

  - **MCIWndGetEnd(hwnd)** Returns the end position of the file.

  - **MCIWndStep(hwnd, n)** Steps n frames or milliseconds, depending on the time format of the device. A positive value is a step forward. A negative value is a step backwards.

  - **MCIWndDestroy(hwnd)** Destroys the window. No return code.

  - **MCIWndSetZoom(hwnd, n)** For windowed devices, sets the size of the window to n percent of the original size of the window.

  - **MCIWndSetVolume(hwnd, n)** Sets the volume of audio playback (if supported) to n.

1000 is normal volume. Higher numbers are louder. Lower numbers are quieter.

- **MCIWndGetVolume(hwnd)** Gets the current volume.

- **MCIWndSetSpeed(hwnd, n)** Sets the playback speed of the device (if supported) to n. 1000 is normal speed. Higher numbers are faster. Lower numbers are slower.

- **MCIWndGetSpeed(hwnd)** Gets the playback speed of the device.

- **MCIWndRealize(hwnd, f)** Tells MCI to realize the palette of the image it is displaying in the window. f is TRUE if the window is in a background application. You should call this function in your app's WM_PALETTECHANGED and WM_QUERYNEWPALETTE code, instead of using the standard windows function RealizePalette. This MCIWnd function will use the palette of the MCI device and call RealizePalette for you. On the other hand, you could just pass the WM_PALETTECHANGED or WM_QUERYNEWPALETTE msg on to the MCI window and this will happen automatically.

- **MCIWndSendString(hwnd, sz)** Takes the place of mciSendString(sz, NULL, 0, NULL). Simply give the string to send to the window. Leave out the alias after the first word. (eg. sz = "set time format frames" is a valid command).

- **MCIWndUseTime(hwnd)**
- **MCIWndUseFrames(hwnd)** Sets the time format of the device to either milliseconds or frames mode. This determines how to interpret a position in the file. By default, when you open a file in an MCI Window, the device will be set to frames mode. If that fails, it will try millisecond mode.

- **MCIWndValidateMedia(hwnd)** If you ever do anything to a device that changes the time format of the media (like changing time formats in some other way than by using an MCIWnd macro) the starting and ending position of the media, as well as the trackbar will still be using the old values, and need to be updated. Send this message to update these values. Normally, you should not need to use this macro.