

#1\$2 +3 **midibkeyb.c**

DESCRIPTION:

This is a custom control that implements a MIDI style keyboard. It's not exactly the best keyboard for a jam session; don't expect to hit a bunch of hemidemisemiquavers at correct tempo...

In any case, this custom control responds to and sends MIDI short messages. This could be expanded on, but this is a good start. The right mouse button is used for 'sticky keys' and will toggle the state appropriately. If you drag the pointer off of a key with the left mouse button pressed, you will notice that the key stays down. This is because I do not SetCapture--and I should.

Another enhancement might be to add CTRL key and SHIFT key modifiers to allow key selection like a multi-select list box.

Before you can use this control, you MUST first export the window procedure for the control:

EXPORTS midiKeyBProc

You then need initialize the class before you use it:

```
if ( !midiKeyBInit( hInst ) )
die a horrible death
else
you are good to go
```

The colors used by the control default to black and white (black for the accidental keys and white for the normal keys). This should work just dandy on all displays.

To select your own colors, you can send the KEYB_SETFGCOLOR and KEYB_SETBKCOLOR messages to set the foreground (accidental) and background (normal) key colors. The lParam is the RGB() value--wParam is a BOOL telling the control to repaint immediately if set to TRUE.

The layout of the keyboard is set using the MIDI key values with C0 being MIDI note number 0, C2 is 48 (middle C), etc. The default layout is: start = 48 (middle C), for 36 keys (3 octaves), ending on 83. I chose this by throwing darts at my MIDI poster...

You can set the layout to whatever you want by sending the KEYB_SETLAYOUT message to the control. The HIWORD() of lParam is the starting key, the LOWORD() of lParam is the number of keys. wParam is a BOOL telling the control to repaint immediately if set to TRUE (actually non-zero...). You can get the current layout by sending the KEYB_GETLAYOUT message--the LONG return value can be decoded as the lParam of KEYB_SETLAYOUT is encoded.

1#DEV_MIDIKEYBD

2\$Midi Keyboard Sample Application

3*SampleApps:010;Development

You also have the option of having labels printed on the keys for MIDI note referencing. Set the KEYBS_LABELS style flag to get labels.

The font used for the label text can be set using the standard WM_SETFONT message. You can get the current font at any time with the WM_GETFONT message.

For other messages and the documentation, see the header block for midiKeyBProc at the end of this source file.

End.