
Subject: Re: Matrox Framegrabber interface

Posted by [peter.mason](#) on Mon, 11 Feb 2002 21:43:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

"Gert Van de Wouwer" <Gert.VandeWouwer@NOSPAMua.ac.be> wrote...

> I need to interface a camera via a Matrox framegrabber that comes with a
> c-library to grab images, and I want to use these functions through a dll.
> The documentation specifies that the memory in which the image is grabbed
> must be in non-paged memory. To do this, I see two possibilities:
> 1) allocate memory in IDL, go to my dll, and specify that the grabber should
> use the IDL-allocated memory. But how can I make sure that this memory is
> non-paged?
> 2) go to my dll, use the Matrox lib's memory allocation function, and use
> this memory in IDL. But how can I return this memory in a valid IDL member.
>
> Off course, the easy way is: allocate IDL memory, go to my dll, allocate the
> Matrox memory, grab image, copy image data to the IDL memory, destroy matrox
> mem, return.... But since the acquisition is a time critical step...

Hi Gert,

Are you sure that copying the frame to an IDL variable is a serious overhead compared to what you're doing with it on the IDL side? E.g., If you're displaying the frame or saving it to disk then I'd be surprised if a memory copy was all that noticeable.

But given that it is important for you to reduce run-time...

I haven't done frame-grabber work myself but I think that you should be able to "lock" IDL-allocated memory (prevent it from being paged) by using the win32 function VirtualLock(). This function has two parameters, viz. success = VirtualLock(start_byte_address, number_of_bytes).

Once you're done frame-grabbing you should call VirtualUnlock(start_byte_address, number_of_bytes).

VirtualLock() doesn't let you lock a large block of memory by default.

(The default limit is something pathetic like 120 KBytes.) If the region that you want to lock is too large for its liking - as it probably is here - then (before locking) you'll have to bump up your working-set size using the Win32 function success = SetProcessWorkingSetSize(process_handle, min_size, max_size). You'll need power-user or administrator privilege to use this function. You can use the win32 function GetCurrentProcess() to get that process handle, and you should use the function success = GetWorkingSetSize(process_handle, min_size_ptr, max_size_ptr) to get

the working-set size before you clobber it.

I expect that you'll be writing wrapper functions in C? This will give you the chance to set things up and wind them down. e.g., Write a setup call that gets the current working-set size (for restoring when you're done), boosts the working set and locks the IDL array that you'll be using, and write a wind-down call that unlocks and restores the working-set size.

Remember to be very careful not to change the locked IDL variable on the IDL side while it is locked. I'll repeat something that Mark Rivers wrote about this sort of thing: never use the variable on the left-hand side of an expression. Doing so will typically make IDL try to reallocate it.

HTH,
Cheers
Peter Mason
