

Re: update OS on custom device

Source:

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2008-05/msg00650.html>

- *From:* PaoloC <PaoloC@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 28 May 2008 01:54:00 -0700
-

Hi Paul,
thanks a lot for your very clear answer.
My question was not so clear as my image is running from RAM (not XIP).
Anyway I still have the problem to write raw blocks of flash while system is running.
How can avoid user processes accessing the flash through PSM driver?
Currently I boost my thread priority during a whole flash block erase/write operation, but I would like to find something more reliable.
So my question about interrupt disabling.

Best regards,
PaoloC

"Paul G. Tobey [eMVP]" wrote:

Those are *all* device-specific questions. Usually, the bootloader is responsible for doing flash image upgrades, so it doesn't have to "allocate" memory; it's the only possible code using the memory so it just says, hey, I'm going to use the X MB starting at address Y to store the image, so get used to it! Same goes for the interrupts. The bootloader generally doesn't enable interrupts at all, so no need to disable them.

You really can't partially write to flash if you're executing XIP. Think about it. When you are executing code, you're constantly grabbing the next few instructions from the flash to execute them. When interrupts are enabled, you might be grabbing some code from application X, then driver Y, then the kernel, etc., etc. Since writing to a flash region has to be done a block at a time, you have to be able to assure that *NO CODE WILL BE EXECUTED FROM FLASH* while that is happening. NONE. You can't do that when using XIP and running with interrupts enabled.

Paul T.

"PaoloC" <PaoloC@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message <news:6286C649-FC1B-4B9F-9D27-9ABB46DD715C@xxxxxxxxxxxxxxxxxxxx>

Good morning,

Re: update OS on custom device

I have already seen a thread about OS update where the suggested operation is :

"Easy to do if your OS is running from RAM. In case its XIP, you can write an app which is loaded completely into RAM. This app should read the update, turns off interrupts and erases/writes to flash directly."

I have to questions about this approach:

1) How can I allocate in my application enough memory to hold the new image

(> 27 MB)

2) How can I disable interrupts from application layer (may be an IOCTL to a custom driver ?)

Currently I am loading new image from external flash in chunks and write each chunk to the internal flash with interrupt enabled (just high priority thread).

Anyway this method is not reliable :-)

Best regards,
PaoloC