

Re: Writing and reading from a physical address

Source:

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.app.development/2008-08/msg00084>

- *From:* M. Essig <MEssig@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 19 Aug 2008 07:28:02 -0700
-

This will not work. If you just return the pointer to the memory you got from MmMapIoSpace and access this in your application you get an exception. Have you ever done something like this?

Martin Essig

"Chris Tacke, eMVP" wrote:

Can't see why it's "not possible". Write a driver, expose an IOCTL that does it, then create a shim API that converts your MmMapIoSpace calls into a call to the IOCTL (the original MmMapIoSpace is just a wrapper around VirtualAlloc anyway). What's so difficult about that?

Chris Tacke, Embedded MVP
OpenNETCF Consulting
Giving back to the embedded community
<http://community.OpenNETCF.com>

"M. Essig" <MEssig@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message <news:38DF074F-8C7A-4D14-94BB-70CBC49A1817@xxxxxxxxxxxxxxxxxxxx>

Thank you for your answer,

I was afraid to here something like this. It seems that it is not possible to port my application to CE6. It's regrettable that MS removed this feature from CE because it's a very common task in that field I'm working in. So it seems the best solution to keep CE5.

"Paul G. Tobey [eMVP]" wrote:

Re: Writing and reading from a physical address

The driver can have various IOCTL commands that would be sent to it using DeviceIoControl() by the application. That would be one way to command the driver to read or write some registers on the hardware. It could also implement suitable handlers for ReadFile() and WriteFile() and provide some abstraction of the underlying hardware in that way. There's no way to keep doing what it sounds like you're doing now, writing directly to the hardware device from application code; you need to change your way of thinking about accessing that hardware.

Paul T.

"M. Essig" <MEssig@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote in message
news:BDCE9C33-87A6-4137-871A-0327DEE8E1ED@xxxxxxxxxxxxxxxxxxxx

Thank you Dean,

But again the question. How can I do this through a driver. As I understood the Kernel driver and the application can't share the same memory which is mapped by the kernel. Or is there a way to do this?

Thanks
Martin

"Dean Ramsier" wrote:

You must provide it through a driver. Your application then calls the driver interface in order to do the work. It is impossible to run an "application" in kernel

Re: Writing and reading from a physical address

mode.

--

Dean Ramsier – eMVP
BSQUARE Corporation

"M. Essig"

<MEssig@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

wrote in message

news:064C2BF5-B578-4F70-9E31-CB2C7467DF17@xxxxxxxxxxxxxxxxxxxx

Thank you
for your
answer,
I know that
I need a
kernel
driver. But
this exactly
the
problem.
In the
application
this
memory is
accessed
like normal
memory.
And
it's
not
possible to
change this.
The
question is
can I
provide
such an
access
through an
driver or
can/must
I
run my
complete
application
in kernel
mode.

Thank you

Re: Writing and reading from a physical address

"vaisakh p
s" wrote:

yes
windows
ce
6.0
has
improved
memory
protection
mechanims.
So
you
have
to
be
kernel
mode
driver
to
access
the
registry.
Its
prety
easy.
You
just
need
to
create
a
driver
DLL
and
set
the
registry
entries
under
HKLM\Drivers\Builtin

On
Aug
4,
2:43
pm,
M.

Re: Writing and reading from a physical address

Essig
<MEs...@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

Hi
all,
I
currently
port
an
application
from
WindowsCE
5
to
CE
6.
In
this
application
we
need
access
to
several
hardware.
SRAM
and
dual
ported
RAM.
In
CE
5
I
do
this
by
mapping
the
physical
address
of
this
hardware
into
my
virtual
memory
with
MmMapIoSpace().

Re: Writing and reading from a physical address

But
this
is
not
possible
with
CE
6
anymore.
Is
there
another
way
to
do
something
like
this
or
have
I
to
build
my
complete
application
in
kernel
mode.

Thanks
for
help.