

Re: Help in getting application to access I/O space

Source:

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.embedded/2006-08/msg00012.html>

- *From:* "Andy Purcell" <Andy_Purcell@xxxxxxxxxxx>
 - *Date:* Tue, 1 Aug 2006 13:28:28 -0600
-

Paul,

You talked about this code working for 16-bit quantities instead of 8-bit.

I need to do 32-bit.

Can this same technique be extended to 32-bits?

I am not an x86 assembly language expert, so if you could provide some more pointers for how to do 32-bit I/O space accesses, that would be great.

"Paul G. Tobey [eMVP]" <p space tobey no spam AT no instrument no spam DOT com> wrote in message news:%239IPKsZtGHA.4748@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

You can do it with some simple inline assembly. Again, remember the restrictions on this and its lack of portability. No one is later going to look at this code and think how smart you are...

```
BYTE inpb( USHORT addr )
{
    BYTE val;
    __asm
    {
        xor eax, eax
        mov dx, addr
        in al, dx
        mov val, al
    }

    return val;
}

void outpb( USHORT addr, BYTE val )
{
    __asm
    {
        mov dx, addr
        mov al, val
        out dx, al
    }
}
```

Re: Help in getting application to access I/O space

}
}

Similar things would be done to do word-wide, rather than byte-wide, outputs, if that's what your hardware requires.

Paul T.

"Andy Purcell" <Andy_Purcell@xxxxxxxxxxx> wrote in message
news:uCwISIZtGHA.4748@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

The processor is x86.
What code would I start to modify to create my own _inp() and _outp() for r/w?

"Paul G. Tobey [eMVP]" <p space tobey no spam AT no instrument no spam DOT com> wrote in message
news:OrOFFnYtGHA.1504@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

What sort of a processor is it? If it's x86, there's a separate set of instructions that access it and you could write your own versions of _inp() and _outp() to read and write. If it's not an x86, then I/O = memory, so you'll want to use MmMapIoSpace() with appropriate parameters. Based on that address, I'm guessing x86.

A twist is that it's also possible that your platform uses bus-relative addresses, in which case BusTransBusAddrToVirtual() would be more appropriate. Are you sure you shouldn't be doing this in a driver and not the application?

Paul T.

"Andy Purcell" <Andy_Purcell@xxxxxxxxxxx> wrote in message
news:%23dbEJiYtGHA.1512@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

I need to write CE app to manipulate my CPU GPIO outputs. The documentation for the CPU registers that control GPIO signals says that the registers are mapped as offsets into "I/O

Re: Help in getting application to access I/O space

Space". The base address
is 0xF0.

So the question is – how can my application
access these registers?

- can I use `WRITE_PORT_ULONG()`?
- must I map this space using some API?