

## Re: display memory extra

**Source:** <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2003-11/0150.html>

---

**From:** Beth (*BethStone21\_at\_hotmail.NOSPICEDHAM.com*)

**Date:** 11/07/03

Date: Fri, 7 Nov 2003 14:28:07 -0000

Peter wrote:

> *Hi All*

Hi :)

> *In windows "device manager", i saw this of my display card (intel 810), Memory addresses are 44000000-47FFFFFF(64M), 40500000-4057FFFF(512K) and A0000-BFFFFFF.*

Well, the last one is the standard VGA "window" as found in DOS but written as a "linear address" rather than the 16-bit segment:address addresses are usually written in when using DOS...

Note, by the way, that it's not just the sizes...addresses 4050000 and 44000000 are just passed the `_1GB_` mark and, I presume, you don't actually have 1GB of physical RAM installed, right? This is a big clue as to what's going on...

> *My question is , does these two address 44000000-47FFFFFF(64M), 40500000-4057FFFF(512K) are mapped to the display card memory directly*  
> ?

Basically, with that big 64MB one, yes, it's mapped directly to the card's video framebuffer as one big "flat linear framebuffer"...your card has 64MB video RAM, yes? The smaller 512KB is probably some "administration" thing (for mapping some non-video RAM thing like maybe a set of registers or something like that...stuff the card also needs mapped into RAM to operate with the device driver or whatever but not actual display memory itself...but that's just a guess there, mind you :)...and the A0000-BFFFFFF memory window is the "compatibility" thing for working with DOS, as that's the standard VGA memory window there...

> *If yes, we don't need to do the blanking?*

Ummm, the term "blanking" is where the monitor has to switch off its beam at the right-hand and bottom-edges of the screen to move the electron beam back to the next line of the left-side edge (horizontal blanking) or right back to the top-left for the next frame (vertical blanking)...as this is something the monitor does itself and doesn't make much sense in the context, then I'm guessing you probably mean "banking" rather than "blanking", right?

If you mean "banking" – where "banks" of video memory are mapped into memory – then, actually, no, no banking will happen on a "flat linear framebuffer" which it appears Windows is using here (well, 64MB of VRAM sounds like a normal typical configuration and that the whole of the video memory is laid out "flat" here :)...also, if you notice, the mapped addresses are beyond your actual physical RAM, yes? So, actually, this video memory is NOT actually anywhere in main memory but all on the card itself...using an address that's beyond your actual physical RAM is the trick used to map this to a memory address without actually "covering over" any of your physical system RAM...the x86 CPU using 32-bit addressing can address up to 4GB so it's capable of sending data to addresses beyond your physical RAM and then the card is configured to respond to these addresses which are mapped there and get routed over the bus to the card...

> *and how can we get these address from BIOS?*

Well, it's highly likely that you've got a "plug and play" configuration here and when you switch on your computer, your hardware cards are not mapped to any particular resources at all...then, depending on your BIOS settings, either the BIOS or the OS using the PCI configuration stuff to negotiate between the various hardware cards and things a set of hardware settings where there won't be any "conflicts"...in older pre-"plug and play" machines, so-called "legacy" hardware would come bundled in a single possible hardware configuration and, thus, it was possible that two different card manufacturers might "conflict" in the resources they wanted from the machine and all hell would break loose...it was to avoid this sort of problem that the whole "plug and play" hardware thing was invented for...by simply not allocating any resources to the hardware cards and, instead, having it that either the BIOS or OS *tells* the card what resources it has been allocated (with the BIOS or OS selecting settings which won't conflict with all the cards installed :) then this particular problem is solved...plus, as the BIOS or OS is responsible for this allocation and talking to the PCI configuration thingies to allocate the resources, then the OS (either directly or by talking with the BIOS :) has an accurate map of what hardware is installed too...might seem mad in this day and age, but prior to "plug and play" this capability wasn't actually there...cards just sat at whatever hard-wired configuration they were given at the factory and if an OS wanted to know what hardware was installed, it would have to just "probe" for it by trying to use it and seeing what response it got...sending data bytes completely blindly to I/O port

addresses...this had a habit of locking up the machine if the hardware actually wasn't as expected...and is why Windows still – on the off-chance there's some "legacy" hardware and a bad choice of "probe" – warns that the machine may potentially "lock up" when it's scanning for "legacy" hardware...

Thus, you'd have to look up the PCI / ACPM specifications for how to ask the BIOS about this sort of thing...if operating under a Windows DOS box here then I wouldn't put any guarantees on Windows actually permitting you to do this without complaining...and, anyway, if you did get the addresses, then you're under a multi-tasking operating system so you'd be in "contention" with Windows' own display device driver for accessing this memory, stepping on each other's toes...to which end, I wouldn't be at all surprised to find that Windows has deliberately left out these memory addresses from the DOS box's available range and you'd GPF (General Protection Fault) or page fault from even trying to access these areas, as Windows can use protected mode to include / exclude what memory a particular process can access (and getting at this memory from a DOS box directly would create a "conflict of interests" between your poking around in that memory and Windows' own device driver)...

Anyway, the basic moral of the message is that, under Windows, you're supposed to – and it'll probably deliberately set up the process' memory map to enforce this and exclude the possibility of you even trying by GPFing any such "naughty" access (NT, surely...9x is notoriously buggy and it does "forget" to protect things here and there so MS might have "forgotten" to exclude things there...but they really shouldn't have and it's exploiting a "non-portable" OS bug so that won't travel from OS to OS too well ;) – get access to the video via the device driver...and via the device driver \_only\_...

In short, "direct access" is verboten under modern OSES that use device drivers...it's not designed to work that way and will probably blow a fuse for you even trying to do so, with GPFs everywhere...note that this is modern OS architecture design and not just a "Windows thing" too...under Linux – though it can be told to be "more slack", if you have "root" permissions – it'll \_also\_ GPF if you try poking around memory and you should also go via device drivers...these OSES basically \_insist\_ upon it because they are usually multi-tasking and if any and every process could start poking around wherever it liked without consulting any other process, they'd all start treading on each other's toes...as one process is trying to draw a box, another is trying to draw a series of lines and whichever gets there last – which will be a mostly "random" thing, depending on the code and the scheduler's behaviour – will "win" and claim the pixel...leading to the possibility of all these processes trying to draw completely different graphics at the exact same time to the screen...at the very least, for example, if you try to write to it and were to succeed then Windows' own display device driver might start overwriting your good work because it has no idea that you've just written this stuff and

writes its own stuff, believing it owns the entire screen...

To avoid this nonsense under multi-tasking OSes, they use the device driver architecture to insist that only the display device driver can access these things and that all other software should talk to the driver (usually via API like DirectX or GDI or XLib or whatever :) so that it can "serialise" things and make sure that the graphics don't all overwrite each other...

Under DOS, which is single tasking (and a tad bit simplistic...though that can be a good thing at times :), this stuff isn't in force...so, under pure DOS (not a DOS box...that's just a "faking it" under Windows and Windows is still actually in control but is allowing DOS programs to think that they are operating under DOS for "compatibility" reasons), you could use the PCI configuration API to find out where the BIOS set the hardware configuration (has to be the BIOS here because DOS is NOT a clever enough OS to be doing the "plug and play" itself...it comes from pre-PnP days and just uses the BIOS to do its stuff :) and then try writing to the hardware card yourself...

But this is where you'll discover a slightly horrible problem...you might know the video memory addresses to write the pixels but how do you set up the card to work in a 1024x768x32bpp mode? Actually, how do you tell it to use the "linear framebuffer" memory mapping that it was using under Windows even?

Unfortunately, you tell video cards how to do this by programming their hardware registers...which hardware registers? Ah, here's where our major little problem shows up...for anything up to VGA, the registers and hardware was all "standardised"...but from SVGA onwards, they stopped making standards and every video chipset does things in its own way and has its own register set and so forth...every single card is, therefore, completely different to every other one (well, a particular manufacturer may make a "family" of cards which all work the same way...but maybe they don't...you just can't tell short of looking up the specific details of each card yourself...there are no "standards" they follow at all about these things anymore...they all do whatever they like :)...what may work to change a video mode with "direct access" on my nVidia card will probably do no such thing on an Intel or ATI card, akin to sending random bytes to those cards...

Thus, even under DOS, due to the massive amount of video cards that all work in completely different ways for this fundamental "mode setting" stuff you'll need, you either spend large amounts of time and money looking up the specifications for every single video card on the market and buy each one so that you can test your code and then write a set of routines for each particular card and so on and so forth (which is basically just duplicating the immense effort of re-creating every single "display device driver" out there :)...which will be stupidly impractical and expensive to actually do, even though

technically a possibility...then the only real option here is to either stick with boring VGA (which is "standardised" so it'll happily work the same way on all machines :) or you use the VESA VBE BIOS extensions (found at AH=4Fxx INT 10h on VESA-capable BIOSes)...

Basically, due to "OS architecture models" (device drivers) and all the manufacturers giving up on "standards" so that they can offer "unique" abilities in their cards (which is made possible by using these device drivers...it doesn't really matter how "different" a video card is to program with "direct access" when you have device drivers because they are responsible for the "direct access" but always present a nice "standardised" interface to the OS using them :) and so on and so forth, it's been made a practical impossibility to feasibly program video cards with "direct access"...

At least, you can learn about your own particular card and play around with that (look up the specifications with the manufacturer...although, increasingly, the manufacturers are actually getting less and less happy about giving out their specifications...nVidia simply don't whatsoever...providing "binary only" drivers for Linux so that you can't even reverse engineer the source code or anything...nVidia has started a trend amongst the manufacturers to keep these things "hidden", which may start spreading elsewhere, making even this option a practical impossibility, unfortunately)...but it will be different on another card that what you learn about your own card and the programs you write won't be "portable" to any other card...

> *If no, then i guess the OS need to do the blanking to map these two  
> address to the display card manually.*

Basically, yes, Windows is responsible for mapping these areas of memory...and, unfortunately, there's nothing much you can do about this...well, short of booting up "pure DOS" and then using the effective "device drivers" of the VESA VBE extensions...

The introduction of "device drivers" to the PC was a good and bad thing...it allowed these video card manufacturers the "freedom" to innovate and break loose from "standards" to implement whatever they liked (such as all the groovy 3D stuff :)...but, at the same time, they have made the use of "device drivers" somewhat non-optional, making "direct access" a thing of the past that only really can still be done with VGA or lower modes (which are just crap resolution and colour depth by modern standards)...this is why you find that so many DOS-based demos still work under blocky mode 13h...other than using the VESA VBE (which can have problems...XP has some "compatibility issues" with using these under a DOS box), you're left no real practical options that are "portable"...

And the essential problem here is that this is a practical thing...there are too many video cards out there that work nothing at

all alike...there's little you can do about this but either limit the cards you'll support or go down the "device driver" / DirectX / OpenGL / GDI / XLib route...that is, this isn't just a case of some theoretical "academic" person coming up with some "theory" that we should all use device drivers and insisting upon it...it's an actual \_physical\_ and \_practical\_ problem that's exceedingly difficult to overcome...

In fact, it's such a difficult problem, that this is probably "problemo numero uno" when it comes to why Microsoft have no effective rivals to their OS monopoly...after all, an OS is just software so why can't people write their own OSes? The problem is "hardware coverage" in a nutshell...Windows has been at this for two decades or so and all the hardware manufacturers themselves automatically write the Windows device drivers because, without them, they won't be able to sell many units if it can't work with Windows, the main PC operating system...but, for your own OS, you'd have to take on the "too many different hardware configurations" problem yourself...which is absurdly time-consuming and expensive to do alone...or, otherwise, you have to use things like the VESA VBE but that's a problem in itself as this stuff is 16-bit code and, under a 32-bit OS, you'd have to preserve the BIOS and work out some special stuff for transferring control to the VESA BIOS and so on and so forth...and it just won't work as well or as fast as having specially coded device drivers for each hardware configuration like Windows does that actually do do "direct access" to those cards (which is why device drivers, when you download new ones off the internet or something, are completely hardware specific and you have to match up your hardware exactly to the driver on the website :)...)

Linux partially overcomes this problem because it's "open source" and, therefore, there are thousands of developers all with different configurations...if each developer just works out a device driver for their own hardware (not an unreasonable or impractical thing to do :) and then contributes that to Linux, it slowly builds up wide "hardware coverage" by "sheer force of numbers" methods...even then, Linux often lags behind Windows in what it can deal with and there are some "gaps" in the hardware coverage that Windows and its monopoly doesn't suffer because, basically, hardware manufacturers are \_literally\_ "designing for Windows XP" and sticking on badges to say just how "designed for Windows" their hardware is...

The growing popularity of Linux has started to "crack" this underlying problem, as many manufacturers do now also consider providing Linux drivers too because it's popular enough to make it worth their while to try to catch those "other people" who use Linux and not Windows...but that was won the hard and long way by many, many developers all working on Linux together following the "open source" method (which, in a sense, is a literal "sheer force of numbers" method :), still lagging behind Windows in hardware support in many places...

If working alone or as a commercial non-open source company or something then this problem is a very tough nut to crack...it's not the only thing that gives MS "monopoly" with Windows but it's a large chunk of the \_practical\_ reasons why other OSes can't easily crack this nut...convincing people over to some other OS would be a difficult enough task \_without\_ having the addition problem that this OS simply will be "lesser" than Windows always because it can't manage anywhere near the "hardware coverage" Windows has (which is basically nigh-on total...when hardware people make PC hardware, they make sure to get it working for Windows for obvious reasons...in fact, some hardware is even Windows-specific - "WinModems" which actually use software rather than hardware to most things that the hardware is simply just a "conduit" by which the software can do everything that needs to be done...so, without the accompanying software, these "WinModems" are almost practically useless - so there's no Hope for these things at all...

> *Another thing i found is, windows allocated a huge memory address*  
> *for Raedon display card (8M). Don't know why. Memory address are*  
> *E0000000-E7FFFFFF (128M), 2000-2FFFF AND A0000-BFFFF.*

Hmmm, that's interesting...it could, mind you, just be "lazy programming"...that is, they wrote the device drivers for their 128MB model of that particular video card and then just re-use it for the versions with less video RAM, adding some minor variable somewhere...oh, yes, I know...this sort of practice sounds distinctly horrible and kludgy but it, unfortunately, happens all the time...the whole "just buy more RAM" lazy attitude to programming...it might be that in this particular case (though, like you, I can't really see why it would need so much more RAM than 8MB otherwise) but this sort of thing does sometimes happen...because, in actual fact, as these memory addresses are actually "virtual" and not necessarily "physical" and that these addresses are beyond the physical RAM in "virtual space", anyway, then despite being "lazy" and "naughty" programming, then it wouldn't actually cause any particular problems or issues to be "lazy" like this...as the only thing that could possibly be accessing this memory range is just the display device driver, anyway, then as long as you're sure it won't try to access the "unavailable" bits, this is a "logical" rather than "physical" thing...it won't actually make any difference to the physical operation or whatever to be "over-allocating" virtual memory space...it shouldn't be done do casually, mind you, as it's "reserving" space that could actually be useful when there really is 1 or 2GB of RAM installed in the machine...but I guess the manufacturers are taking the typical "lazy programming" attitude of "Why worry? They'll have upgraded their machine and cards and things by the time this becomes a problem...and if they haven't, well, then it's 'their own fault' for trying to deny us our constant income by enforcing the 'upgrade cycle' as aggressively as we do" (remembering that behind the smiles to the customers to keep us happy, it \_is\_ merely "all about profits" for

them that we shouldn't be at all surprised at a sudden "lack of support" when we allow ourselves to fall too far behind the constant "upgrade drive"...for instance, Windows 95 is officially "dead", Windows 98 will be "buried" this year, Windows ME and 2K "die" support-wise in two years' time...because, yup, Microsoft have a strict "we kill an OS after five years" policy...do they kill it because it's no longer used? Nope...half the universe could still be using Windows 98 and they'll still bury it and force an "upgrade" because you are being so "nasty" as not to give them their constant source of "perpetual income", selling you what you already own over and over again ;)...

It's the blessing and also the curse of the PC architecture...it was the first desktop architecture to allow any old third-party hardware to just be "slotted in and out" at will, which is a great little feature of the PC that lead to it becoming so popular (not just the feature itself but the fact that it lead to PC "clones" made by other manufacturers other than IBM – who actually made the first one...which we probably actually need to start mentioning in this day and age, now that IBM have been effectively completely forced out of their own market that they originally created and that many people are arriving at the PC long after IBM has been completely dropped as the prefix (for those who're somewhat new to PCs, they used to be referred to as "IBM PC and compatibles" but, eventually, the "clone" manufacturers pushed IBM to one side, taking over the PC, leaving its creator far behind as just another PC manufacturer...and though strictly "PC" means "personal computer" so it could actually mean an iMac or something too, they've so taken over the market that "PC" automatically means "IBM PC compatible" without anyone asking for clarification whether what you mean :)...)

Want to break the Microsoft monopoly? A single masterstroke could potentially do it...if the hardware manufacturers came up with a method to offer full "hardware coverage" automatically to all potential OSes – some "standard" device driver specification that all agree on, supplied directly in the ROM, using something like the old "BIOS"–like way of providing this support – then anyone who has the time can write an OS and doesn't have to worry about "hardware configurations"...a sudden "flood" of rivals and "clones" would do to Microsoft in the OS field exactly what it did to IBM with the PC...no-one would have believed that "Big Blue" – the original "evil empire" – could have been felled by something so seemingly small and trivial...but they were and the same could potentially happen to Microsoft were things to head in that direction...

Note that in the anti-trust case, Microsoft were found guilty of specifically bribing hardware manufacturers to show unfair "bias" Windows for money...Microsoft themselves know only too well just where their "Achilles heel" lies...I mean, look at Microsoft's monopoly and you'd have to wonder why on Earth they'd bother spending money (illegally) in this way...I mean, they "own" the whole thing as it

stands, right? Ah, but being a software company, of course, they depend in a very big way on the trends of hardware manufacturers...currently, those manufacturers are "bribed" into favouring Windows...either literally as in the illegal crimes that Microsoft committed as discovered by the anti-trust case or by the implicit "bribe" that if you don't support Windows then you just aren't going to get anywhere near the sort of sales you would do supporting it...

Look carefully at Microsoft's actions and there's a pattern to be seen...in how they step in and try to lead the way on the ACPM specifications to tell hardware manufacturers how to implement this stuff in a Windows-friendly way, not wanting to let them think for themselves, lest they think the unthinkable...in how, despite having a monopoly that's quite frankly absurd by anyone's standards, they still risk taking illegal actions in bribing those hardware manufacturers threatening "mutiny" (I mean, if the original sentence had stood, this would have forced Microsoft to be split in two...and their "punishment" could potentially have been far worse because the potential power of the courts extends way beyond just that sort of thing...you'd only really take such risky actions – especially when you're sooo ahead of the game in "owning" everything, you'd have to wonder what they specifically "win" by doing this sort of illegal action in the overall picture – when there's something worth the risk)...

There's your "Achilles Heel" for you...if the hardware people "mutiny" the software people at Microsoft, software is useless unless the hardware is "playing ball"...there's a natural "symbiosis" there that can't be broken...maybe, in fact, Bill's "paranoia" that "things in this industry can change in an instant" is partially justified...if the hardware people all get together and conspire to "mutiny" – even in small part – to let in "rivals" then it comes down to who can code the best and Microsoft's track record doesn't suggest they'd necessarily "win" any such battle, especially not against an "open source" project as those things are already biting at their heels as it is, despite a clear, clear advantage to Microsoft...it's only a slimmest sliver of a chance but it's there...if a Linux-like "open source" project could somehow "tempt" the hardware manufacturers "on side" so that they happily equally cater for it as Windows then the "clones" may do to MS what was done to IBM by those very same hardware manufacturers themselves...

"Begun this clone war has"

[ Yoda ]

Beth ;)