

## Re: Polling loop good here???

---

*Source:* <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2005-08/msg02227.html>

---

- *From:* "Beth" <BethStone21@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Mon, 29 Aug 2005 06:24:34 GMT
- 

Frankie say:

> At the risk of beating a dead horse... This question just popped up on  
> the linux-assembly list. Hasn't gotten any answers yet, and I don't  
know  
> a good answer...  
>  
> -----  
> Any ideas how I might sync to bit 3 of port 0x3DA, the video card's  
> vertical sync signal?  
>  
> Just watching the bit in a loop doesn't seem like a good idea, since  
the  
> CPU could be off doing something else, but as far as I know any  
timer  
> function I might use will only be accurate to 1/100 of a second,  
which  
> wouldn't be much better than watching the bit in a loop.  
>  
> Since things like DOOM run in Linux, I think there must be some way to  
> do it. I've discovered that simultaneously changing the screen data  
and  
> the color palette without synchronization creates a flicker of a mess  
> on the screen, and DOOM doesn't do that in Linux, so I guess it must  
be  
> synchronizing somehow.  
> -----  
>  
> I suspect we're all agreed (?) that it would be really cool if there  
> were an interrupt on vertical retrace, but there isn't. There may be a  
> way to "block" waiting for this, but I don't know how to do it. I fear  
> we may be "stuck with" a polling loop here...  
>  
> Anybody know the "right" way to do this?

Basically, it goes something like this:

The "vertical retrace interrupt" was IRQ 2...

But then along came the second "slave 8259" PIC controller...and IBM

Re: Polling loop good here???

decided to hook on the second controller at IRQ 2, to "cascade" the second controller from the first controller (IRQ 9–15 triggering IRQ 2 and then being "passed on" to the second controller via the "cascade mode" that these controllers supported for hooking up more controllers together)...

As such, IBM didn't seem to restore a new "vertical retrace interrupt" in the new IRQ 9–15 range...and on their VGA adapters, the line for the interrupt can actually be physically seen to have been CUT (not just wolfgang's observation tells us that this was common, Ralf's list also mentions that the lines in IBM's adapter were "cut or removed")...

Hence, due to IBM's rather weird ad-hoc architecture, the "vertical retrace interrupt" was effectively "killed off"...I'm not sure if, in fact, you technically have to "surrender" IRQ 2 to "cascading" (is it still possible with the controllers hooked up in "cascade" to still get a plain old IRQ 2 that isn't destined for the second slave 8259 controller? I've not checked up the exact details)...but, even if you don't, IBM seem to have decided not to take any chances of it getting mixed up..."compatibility" possibly...

And, so, even though the VGA cards could provide a "vertical retrace" signal, IBM cut the line...

Nice, eh?

Hence, DOS games using plain old VGA graphics actually tended to just "poll" for it..."ugly" but with IBM's "nice" decision to go cutting IRQ lines (and many "clones" copied...also, because you "can't be sure" if a particular card does or doesn't have the line cut or not, the assumption, unfortunately, has to be that it doesn't have the line, even on a "clone" or two that didn't copy IBM's "cut the line" policy)...there was no other choice given...anyway, on a "single-tasking" OS, it's not as big a deal to "poll" as it is on the multi-tasking OS (though, if you could do it, then either system would "benefit" from such...the "multi-tasking" merely "magnifies" the problem...it would still be an excellent idea to work off "interrupts", even under DOS, if IBM had made it possible)...

In today's hardware – as has been mentioned – they've almost certainly now "corrected" IBM's little "defect" (indeed, easy way to check this...right, "Control Panel" -> "System" -> "Hardware" -> "Device Manager"...find my graphics card, click on "Resources"...there we have it: "IRQ 11" is reserved for the card...yeah, they HAVE corrected the problem ;)...

But, this is post-VGA stuff – "SVGA" – and, hence, none of it is "standard" between cards...each card manufacturer has probably done it a different way...

So, in order to make use of this – as has been mentioned – you basically

Re: Polling loop good here???

Re: Polling loop good here???

need to use one of the "API" (DirectX, OpenGL...DirectFB, if that supports this kind of thing properly??)...and these'll use the appropriate "drivers", which know all about the "IRQ line" and should make use of it on your behalf...

The alternative approach, of course, is a little "ugly": Find out the "specifications" for your card (\_IF\_ you can find out...nVidia don't reveal any of their "secrets")...and then "direct access"...but this, of course, will only work for that particular card...so, you'd be faced with a problem of effectively duplicating every single "driver" out there...and some kind of "detect which card" routine to select it...

Basically, unless you fancy writing your own OS simply to make your game work (or you only care that your game works on your machine and screw all the rest of them), then this option isn't at all feasible...and, well, you know the drill: The OS has already got the drivers to do it...make use of the drivers...

[ Certain OSes – \*cough\* – of course, only let you "officially" access those third-party drivers through their proprietary "API" in order to "lock in" your game's graphics routines to their "monopoly of the entire universe"...this "indirection library", which has no requirement to even be there at all (because device drivers, by design and purpose, must present "portable" interfaces to the OS that the "API" is doing absolutely nothing useful but standing in the way, to force everything to be "tainted" into a "proprietary" lock-in monopoly), they ironically label "Direct"!!

Kind of like they steal everyone's ideas and have the motto "innovate!"...it's all just a little ironic "joke" or two that they like to laugh about at the office Christmas party:

"Innovate!!"...they all fall about laughing..."wait, wait...I've got a better one: Direct!!"...they're now splitting their sides with laughter...then in walks Master Gates: "Guys, guys...you're all such amateurs: Portable!!"...

Balmer pisses himself with uncontrollable laughter at that, as he tries to spit out his own joke: ".NET is state of the art!!"...the Longhorn programmers laugh hardest...they who couldn't manage to get the fabled ".NET" to actually work for the new OS and gave up even trying (you know, the thing that Microsoft encourages everyone else to use – because it's "easy" and "speeds up development" – that they themselves, as its inventors, couldn't manage to get to work...on software that wins the award for being "most behind deadline" in the entire history of software...yeah, that's perfect "quod erat demonstrandum" for you...so "easy", its inventors can't even fathom out how to make it work...it "speeds up development" so much that it's conspired to make Longhorn the "most behind schedule" software in the whole history of software...despite this unquestionable Aristotelean proof that Microsoft are talking through their arses about ".NET", just watch how many bosses

Re: Polling loop good here???

Re: Polling loop good here???

insist on their programmers using ".NET" just because there was, you know, this "pretty" TV advert where people were smiling and the "pretty colours" flashing in the background were "nice"...oh, and there was a "free T-shirt" involved...technical merit? What are you talking about? I'm the "boss"...what the frack do I know about "technical"? Microsoft are rich...I want to be rich...whatever Microsoft say, then I'm doing!! Then I'll be rich too!!! No, don't try and confuse me with any of that "as if Microsoft are telling you how to defeat them and become richer than them" logic!! I believe in the eternal sainthood of Sir Bill – against all evidence to the contrary for decades upon decades – to be incapable of lying for competitive advantage...and, yeah, folks...that's the \_REAL\_ joke that Bill and Balmer laugh over at the office Christmas party: What gullible idiots there are in this world that you can become the richest people on the planet, selling absolutely crap garnished with complete lies...and even those who should know better often don't even notice what's going on)... ]

Your standard "post-VGA" problem, Frank...the VGA had its IRQ line physically cut by IBM (probably to do with IRQ 2 becoming the "cascade IRQ" for the second 8259 controller...but why it had to be cut, rather than moved to one of the new IRQs, I don't know...ask IBM...although, they probably don't know either...just "felt like it" at the time...or it was the "easiest" way to get their "slightly backwards" PC invention out of the door fast...

And anything "post-VGA" is completely "non-standard" that you either program "only for my own personal videocard" (\_IF\_ you're lucky enough to even be able to get the "specs"...I have an nVidia card myself, so that's straight out the window as an option for me, thanks to their "Mamma's secret pasta sauce recipe" policy ;)...so, you've got to go via the "drivers"...which usually involves going via an "API"...

And, yeah, if the "API" does not supply any "trigger my code when the vertical blank happens" mechanism, then you're still not guaranteed to be able to do much of anything, anyway...unless you fire up multiple "threads"...then deal with the nightmare of "synchronising" them...because these "API", you know, they make it all "easier" for you by complicating it all unnecessarily...how thoughtful!!

[ On the old C64, the interrupt wasn't only connected, it had the "Commodore touch" of knowing exactly what's \_USEFUL\_ to programmers...the C64 (and subsequently Amiga) video IRQ could be programmed to trigger at a specific \_scanline\_...and this was used (abused) by many programmers to create graphics that appeared "technically impossible" (the classic example is that the C64 video hardware had "hardware sprites"...the video chip could hold information for up to 8 "sprites", which were rendered as "overlays" onto the background...this meant that they could be moved around over the background without worrying about the problems of "clearing the background" and stuff...the hardware dealt with it as it rendered the screen itself, so they genuinely "hovered" – like the ghostly "sprite"

Re: Polling loop good here???

Re: Polling loop good here???

apparitions they are named after – over the screen background...anyway, the "trick" that games often employed was to use the "scanline interrupt" in order to have more than 8 hardware sprites on the screen...an apparent "technical impossibility", going by the "tech specs" of the video chip itself...the basic idea is rather simple and clever...the video chip would render the first 8 sprites at the top quarter of the screen...then the "scanline interrupt" would be triggered at this point...the sprites are moved to the next quarter of the screen...the "scanline interrupt" programmed to go off at the half-way mark...so, it renders another 8 sprites on the screen...and this is continued for each quarter (allowing what looks like 32 sprites on the screen simultaneously...it's actually the same 8 being "re-used")...and, as this was on the "scanline interrupt" then it repeated this for each "refresh" of the screen...60 times a second...there were also similar "tricks" with altering colours and graphics as a screen was rendered to, again, seemingly "do the impossible" when the machine is showing far more colours than it's "technical specifications" suggest is even possible)...

I repeat, in many, many ways, the PC was genuinely a "backwards step"...video hardware on the PC wasn't exactly "cutting edge", even at the time...then they go cut IRQ lines, just to make it worse still...while everything else had gone GUI, the PC was living in the '70s still...even the 8-bit C64 had a sophisticated musical synthesiser chip in it, the PC was silent but for the "beep" of the PC speaker as it booted up...the PC won because the PC was "cloneable"...and simple, basic "capitalist competition" took over...every PC manufacturer trying to "outdo" the other, lead to it evolving fast...so fast, in the end, even IBM – who invent it and were hardly "small fry" as a company – got left behind and "ditched" as the "standard" to follow...

Which is why it's "VGA" and "SVGA", rather than "XGA" and "SXGA"...which would have been nicer, in fact...I looked up the XGA's "specs" and it had some basic support for, yeah, a "hardware sprite" (only the one, it seems...but that's at least useful for a "mouse pointer" :)...also, decent resolution for "safe mode" than the frankly ugly 640 x 480 x 16 colours crap (that's literally 16 colours, not 16-bit colours...no mistake or omission has been made there ;)...shame IBM didn't stay "standard" just that little bit longer, for XGA to have been the "base", not VGA...they probably even "fixed" the IRQ issue too...remember, "deep irony" rules everything, in the end...one merely needs a "wicked" enough sense of humour to notice this...

If programming VGA and programming it "direct access", then it is a stupid idea to sit around "polling" when the machine could be doing so many other useful things...but you don't really have much of a choice in the matter...no reliable "interrupt", no reliable method to solve the problem (timers would simply not work: First, how do you "synch" them up in the first place? Second, "chaos" rules and they'd go "out of synch" – different timer sources – after a while, anyway...yes, however "accurate" they claim to be...butterflies, wings, tornadoes...you know

Re: Polling loop good here???

Re: Polling loop good here???

the drill ;)...

Also, how "healthy" is it to be "direct accessing" VGA hardware under Linux, anyhow? Being "multi-tasking" and the video a "shared resource", if you launch more than one program that tries the "direct access" to the video hardware, then does this cause some pretty nasty "conflicts"?

The "drivers" aren't only there for "portability"...they deal with such "sharing" in a more intelligent manner (e.g. just ignore the requests of the program that's "in the background" ...indeed, take it slightly further – reserved "areas" of the screen for each application – and that's the "base" of a GUI interface for you ;)...

Did you ever try running multiple copies of your "PC speaker" program, Frank? You get a rather strange "tremolo" effect, as each instance reprograms the timer, blissfully unaware that another instance is also trying to program the thing at the same time...

Unfortunately, though, if the hardware itself provides no form of "notification" – and only the video hardware knows when the "vertical retrace" happens – then I can't see any way to "fake up" any kind of "block"...

Basically, the video hardware is the only thing that knows when "vertical retrace" happens and it's "keeping mum" about it...and the CPU is not in any way "locked out" from access because of the "vertical retrace" (so, no "side effects" that could be "blocked" on either...which is an alternative and perhaps what you're asking when you ask "is there anything else to block on?"...if there were some "side effect" then it might have been just possible – if you're lucky – but no such luck here, as far as I'm aware...as the videocard happily lets the CPU access anything, whatever "state" it's in...and just "snows" or "tears" – as it famously does on a CGA – rather than "prohibiting" it in any way...or a "block" on the "prohibition" could have been an "alternative" way to "sense" it ;)...

As has been said, the options seem to be to just "poll" and, yes, just accept that's crap...or find out how to access the "drivers" to take advantage of the "fixed" IRQ line stuff in the modern hardware (but has to go via "drivers" or something similar because none of this stuff is "standard VGA" in any way, so each one'll probably do it vastly differently to all the others)...

Though, via "drivers", it's no longer truly "direct access" but then you're pretty much screwed with "direct access" because of the "sharing" problem, anyway...the "drivers" are as equally responsible for ensuring two programs "multi-tasking" at the same time don't screw up each other's display (typically, only one can be "full screen" at a time or something along those lines)...indeed, now that even Apple sound like they are going x86 and "PC", it's arguable that this is the `_primary_`, not secondary purpose of device drivers these days...

Re: Polling loop good here???

Re: Polling loop good here???

Acting as what's called a "monitor" in concurrent circles...one program that has "exclusive access" to a resource that all others must make "requests" to do things for them...and, thus, it means that the actual "direct access" is only done by the one "driver" program...so, no issues about "sharing": Only the "driver" makes access to the actual device, all other programs make "requests" of the "driver" which then sensibly "sequentialises" those requests (or ignores them or sends them elsewhere or simply sends back a "video is currently in use" error code to the program or whatever makes sense for the particular situation)...

I've only "heard" of DirectFB and "svgalib" but never actually used them...don't know if they are at all useful...as for X, it's a totally different structure altogether...GUI stuff...OpenGL certainly "comprehends" the vertical blank stuff...goodness knows if that's a "good match"...

Unfortunately, Linux's graphics capabilities are a lot confusing...because Linus never bothered with anything beyond "text mode"...XFree86 actually does "direct access" (of no use to anything that doesn't want to run inside X)... "svgalib" is out-of-date and "not supported"...and "DirectFB" is brand new, so how far that "extends" at the moment, I'm not sure (though, it is supposed to basically be the "plug" for the "gap" Linus left in not really putting anything "graphical" directly in there...this is the proposed "cure" but I don't know how far the "driver support" and so forth currently goes)...

Really, Linus should have – even if he wasn't himself going to do it – left a "gap" in there for "graphics drivers"...you know, a "placeholder" – some "specifications" of what should go in the "gap" to be a "standard" for it – and let others actually put something useful in there...then XFree86 could have been based on using those "drivers"...and they could also be useful outside of X too...a single set of "video drivers" created...programs using them "straight" or X using them "on behalf" of other programs...and it all would have been nice and Lovely...

Didn't happen that way...things like "DirectFB" trying to "correct the fault" after-the-fact...you can get versions of X where the source has been changed to go via "DirectFB"...this should be the "cure", if all goes well...but how far along this currently is, I don't know...

Beth :)

.

---

• *References:*

◆ *Polling loop good here???*

Re: Polling loop good here???

Re: Polling loop good here???

◇ *From:* Frank Kotler

- Prev by Date: *Symbol Scope documentation*
- Next by Date: *Re: model of Z*
- Previous by thread: *Re: Polling loop good here???*
- Next by thread: *Re: Polling loop good here???*
- Index(es):
  - ◆ *Date*
  - ◆ *Thread*