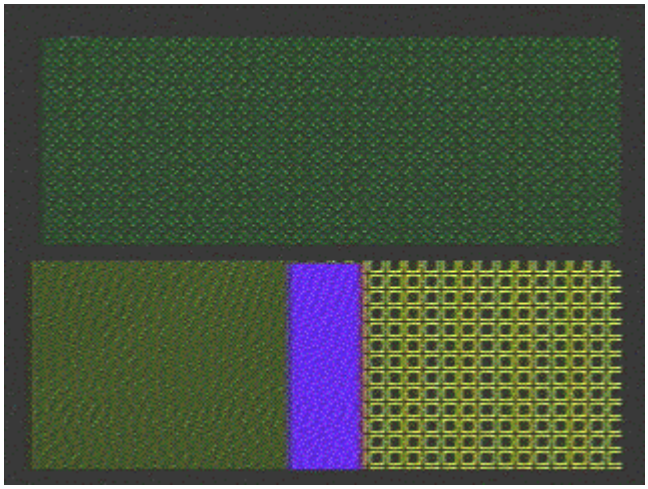
Charles Mac Donald ‘s technical notes

(2000/11/10)

I've been doing a lot of Genesis programming recently, because it's a really cool machine to program on, and it's an interesting challenge to push the hardware as much as possible. Here's the latest thing I've whipped up:



It may not look like much - but of course there is more 'behind the scenes' then a single low quality snapshot can show. The top half of the screen is in SMS mode, the bottom is the native Genesis display mode, which is then split again; the left half has shadow / hilight mode enabled, the right does not, and the blue column is made of high priority sprites which mask any flicker between the shadow/hilight display transition. So it's something of a 3-way split. :)

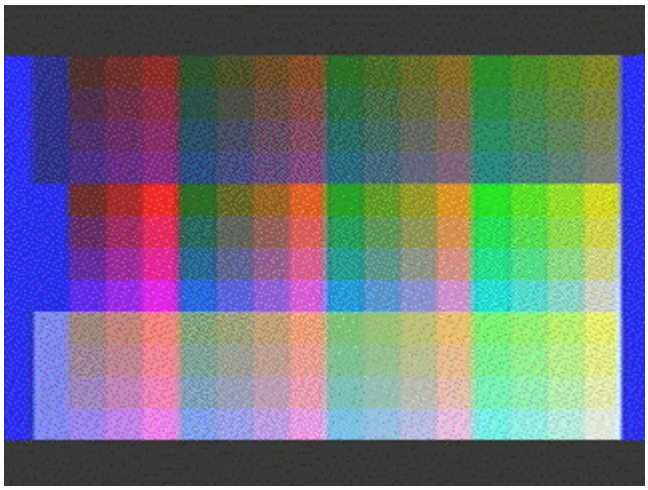
One thing that made this rather difficult to do was that the SMS section of the screen doesn't react at all like it should. For some reason the first and next group of four bytes both define pixels for the same line. And the Y scroll register doesn't work (X does), instead it offsets the actual screen a bit. It's the exact same result as messing around with the vertical hold - now that's weird. To actually do vertical scrolling, it seems that any VSRAM entry beyond 32 will work. Why VSRAM would have any affect at all is also a mystery!

I think this is probably some side-effect of when the SMS/Genesis mode switches are done. I originally thought that at a certain point, the VDP grabbed the register contents for whatever mode it thought it would display, and then ignore everything else. Not so. It's more like the VDP grabs settings at various points, so switching things at the wrong time gives unpredictable results. Case in point: messing with the interlace bits forced the display into a noninterlaced state, with only the odd frame being shown, and gap between every other line.

Some other interesting things I've noticed (about mode 5): sprites are processed on the previous line than the one they are shown on, and are not processed when the display is blanked. This is why in the picture above, there's a one line gap between the middle border and the sprite column. Also, when the display is un-blanked, the last sprite setting the VDP looked at will be displayed - quite noticable in games that do sprite tricks mid-frame. (like Castlevania Bloodlines) Before you point out that this would limit sprites from being shown on the first scanline, I'll mention that the NES has a similiar behavior, and solves the problem by parsing sprite data on the *last* line (262), so everything is ready for the first. Neat, huh. :)

(2000/10/29)

Well, I finally got around to taking a screenshot of the Genesis shadow/hilight colors from a real machine running my test program. This was something I was meaning to do, since the old 183 colors demo I wrote actually had some bugs that didn't show up on an emulator.



The colors make up a 256x192 block. I wanted to center the image as the resolution is 256x224, and came up with an interesting solution. What the test program does is enable the regular display mode (mode 5) at the top of the screen, wait a bit for everything to stabilize, then turn on the display. At the end of the frame (line $C0), the VDP is switched into mode 4 (the SMS display mode). This makes the VDP automatically center the first 192 lines in the middle of the display since it thinks it's going to show a mode 4 frame, but I just switch back before any of the graphics are actually displayed. It's a pretty neat trick, and all it took was a little clever thinking. :)

Maybe I'll release the test program in the future - needless to say it does not work with any emulator. :)

(2000/07/21)

Turns out there's a minor bug in the Genesis VDP documentation. After doing some debugging with Golden Axe II, I found that clearing the address/code registers after a VDP register write screws up the 'SEGA' logo name table data. Kind of annoying, since some tests I had done earlier seemed to confirm that both registers are indeed cleared, but it's quite hard to get accurate results when you have to patch games two bytes at a time to run your own tests.