back

384 Byte Democompo

The Mission

Time for another competition ? Right - here we go: Your task is to create a good looking demo (effect) in just **384** (0x180) bytes. To aid you a little and to prevent too system specific hacks a init-routine with basic output functions is given. If course just the code-size is counted, not the size of the resulting object-file. The usage of BSS-Sections is also free.

Since it is quite impossible to judge the demos cleary, we will have a public vote. The submission deadline for the demos is **sunday, february 22nd 1998 at 21.00 CET**. The executables of all demos will be released some days later. After that the public vote will begin. The voting deadline will be two weeks later.

Btw: you dont have to do any super-supreme-overoptimized code to win this compo - innovation is everything. Even beginners could win it, with a nice new idea.

All contributions must be mailed to Azure.

The Rules

- You have to submit both a source and an executable of your demo (assembled with the given init)
- All sources will be published here, after the competition is over. In case you dont yours to be published, please state it clearly in your contribution.
- Your demos **must** use the given init. It contains everything you need. Take a look at the section below for further information on it. Your demo is not allowed to access any other resources (library-functions, rom data etc).
- Direct hardware access is only allowed for sound-playing, nowhere else. (in case you really want to squeeze sound in your demo :))
- Selfmodifying code (SMC) is only allowed in connection with a cache-clear, which is provided as a function of the init. No 030-specific hacks.
- No FPU-Usage.
- Your demo must not have an own exit routine. The init contains a build-in one within the refresh-function.
- The three "RTS" are included in the 384 bytes ! (so you may only add 378 bytes, unless you

trick around with the labels)
- The startup-functions may be called with "BSR.W"

The Startup

The startup - code for this competition was provideded by **Blueberry**. Many many thanks to him for doing it! Please download the full package, also including this docs, here. You might have to set your assembler to "Case sensitive labels" to assemblee it !

Place your code between the labels Code_Start and Code_End and your BSS definitions at the bottom of the source. You have to supply 3 functions:

Init:

Called once in the beginning. Registers are undefined.

VBlank:

Called every vblank after Init has finished. On first call, registers are as left by Init. On subsequent calls, registers are as left by last call.

Main:

Called once when Init has finished. Registers are as left by Init. If it terminates, the demo will exit.

Your code may only reference memory inside the code or in the BSS section and may only call the 6 supplied functions: (The functions preserve all registers)

Update128x128:

Update the screen in 2x2 resolution using the supplied 128x128 linear chunky buffer. The sides will get color 0. Callable from Init and Main.
Input: A0 = Chunky buffer

Update160x128:

Update the screen in 2x2 resolution using the supplied 160x128 linear chunky buffer. Callable from Init and Main.
Input: A0 = Chunky buffer

Update256x256:

Update the screen in 1x1 resolution using the supplied 256x256 linear chunky buffer. The sides will get color 0. Callable from Init and Main.
Input: A0 = Chunky buffer

Update320x256:

Update the screen in 1x1 resolution using the supplied 320x256 linear chunky buffer. Callable from Init and Main.
Input: A0 = Chunky buffer

CacheClear:

Clear caches so written SMC will be valid. Callable from Init, VBlank and Main.

SetPalette:

Set the screen palette to the one supplied. Format is 256 longwords in $xxrrggbb. The upper byte of each longword is ignored. Callable from Init, VBlank and Main.
Input: A0 = Palette

Last change: 16.01.2001
