Software compatibility considerations

Discussion

Every new Amiga model and Amiga accelerator in history has had compatibility issues when running software that was written for older configurations. As a result of bringing modern technologies to the Amiga, the Vampire has similar compatibility problems. However, the Vampire was designed to be **more backward-compatible than any other Amiga model or accelerator**. For example, the Vampire is **more compatible** with Amiga 500 software than an Amiga 1200 is with Amiga 500 software!

To put this in perspective, we should list the software compatibility problems typically encountered with Amiga upgrades, and describe how the Vampire solves them at the hardware/firmware level:

1. Newer 68k CPUs & FPUs do not support all instructions of older 68k CPUs & FPUs. If a program is using those unsupported instructions, it fails on newer CPUs, unless an emulation library is provided.
    - The Apollo 68080 supports **every** CPU & FPU instruction of the whole 68k family (68000, 68010, 68020, 68030, 68881, 68882, 68040, 68060). [1] An emulation library is not needed.
2. Newer 68k FPUs do not support all data types and Effective Address (EA) modes of older 68k FPUs. If a program is using those unsupported data types or EA modes, it fails on newer FPUs, unless an emulation library is provided.
    - The Apollo 68080 supports **every** FPU data type and EA mode of the whole 68k family (68881, 68882, 68040, 68060). [2] An emulation library is not needed.
3. The instruction cache in newer 68k CPUs does not reflect runtime code modifications in RAM. If a program is running self-modifying code, it fails on newer CPUs.
    - The Apollo 68080's smart instruction cache snoops code modifications in RAM, and updates itself accordingly.
4. The data cache in newer 68k CPUs does not reflect DMA-based data writes done by the Amiga chipset (in Chip RAM). If a program is relying on chipset DMA, it encounters stale data in the cache and fails on newer CPUs.
    - If you have Core version < GOLD3: The Apollo 68080's data caching is disabled for the Chip RAM region. [3]
    - If you have Core version >= GOLD3 (as in the Vampire Standalone): The Apollo 68080's smart data cache snoops data writes done by the chipset (in Chip RAM), and updates itself accordingly.
5. A Blitter operation consumes most memory cycles of the Chip RAM. So, in Amigas with no Fast RAM and having an older 68k CPU without an instruction cache, the CPU can't make much progress while a Blitter operation is running. As a result, the subsequent Blitter operation is almost guaranteed to be delayed until the current Blitter operation has completed. But with the available memory cycles on Fast RAM, and due to the instruction cache and higher speed of newer 68k CPUs, the CPU can make much more progress while a Blitter operation is running. If a program is relying on the CPU being delayed during a Blitter operation, it fails on Amigas with Fast RAM and having a newer CPU, if the unimpeded CPU reaches the subsequent Blitter operation and re-programs the Blitter mid-operation, for example.
    - If you have Core version < GOLD3: If the Apollo 68080 tries to re-program the Blitter, the Vampire checks if a Blitter operation is already running. If so, then the 68080 is

delayed until the current Blitter operation is complete.
    - If you have Core version >= GOLD3 (as in the Vampire Standalone): If the Apollo 68080 starts a Blitter operation, the 68080 is delayed until that Blitter operation is complete.
6. Newer 68k CPUs simply run “too fast”. If a program is relying on the exact timing of CPU instructions, it fails on newer CPUs.
    - Most programs that rely on the legacy CPU behavior store their code in Chip RAM. If the Apollo 68080 starts executing program code from Chip RAM, then it puts itself into “turtle mode” automatically. (You can also trigger turtle mode manually.) In turtle mode, the CPU does not run too fast.
7. Amiga models other than the CD32 do not have the Akiko chip. If a program is using Akiko C2P routines, it fails on Amigas other than the CD32.
    - The Vampire supports all Akiko C2P routines.
8. Newer AmigaOS versions have different internal (undocumented) behavior than older AmigaOS versions. If a program is relying on the exact internal behavior of a particular OS version, it fails on newer OS versions.
    - The Vampire is compatible with all AmigaOS versions, and provides a variety of convenient ways to use older Kickstart versions.

These measures make the Vampire more backward-compatible than other Amiga upgrades. Rarely, you might encounter a legacy program whose incompatibility problems can't be solved at the hardware/firmware level. In order to run such a program on a modern Amiga / Vampire system, you would need to apply an appropriate software patch to it.

What to expect

Due to the reasons discussed above, you can expect a high degree of compatibility, compared to other kinds of Amiga upgrades.

- System-friendly software should run perfectly on the Vampire.
- RTG games and demos are usually written in a system-friendly way, so they are expected to run perfectly.
- Most non-system-friendly software should run well. Some badly-written software might fail. See the “Tips” section below for handling such software.

Tips

- For using older AmigaOS versions, and for AmigaOS compatibility considerations, see this page.
- You can use the Amiga Early Startup Control, the Degrader tool, or the TUDE tool to degrade your Amiga / Vampire. Since the Vampire supports many backward-compatibility features natively, a lot of the degradation features in these programs are superfluous.
- We strongly recommend you to use WHDLoad to run legacy software. On the Vampire, a lot of WHDLoad's degradation features are superfluous, but still, WHDLoad provides a convenient way to perform degradation and use older Kickstarts. It also allows floppy-based programs to be installed on hard disk, and even applies the appropriate patches to programs whose incompatibility problems can't be solved at the hardware/firmware level.
- Make sure you are aware of the bugs in WHDLoad. Most importantly, on AmigaOS, you should disable the TCP/IP and USB stacks before running WHDLoad. This can be automated by

configuring startup and cleanup scripts in WHDLoad's global configuration file. Example files: `S:WHDLoad.prefs`, `S:WHDLoad-Startup`, `S:WHDLoad-Cleanup`. (Note: This problem does not exist on AROS.)
    - If you are using the demo version of Roadshow, you must manually disable it before starting WHDLoad, so that you get a chance to dismiss the demo version's shutdown message.
- If WHDLoad exits with an “NMI Autovector” error, enable the `NoAutoVec` option in `S:WHDLoad.prefs`.
- If WHDLoad suddenly exits, enable the “ChkInts” option in `S:WHDLoad.prefs`.
- If a program is running too fast, or it simply fails, then it might get fixed by putting the 68080 into “turtle mode”, so that it runs slower, like an older Amiga. Most legacy programs that exhibit this problem store their code in Chip RAM. If the 68080 starts executing program code from Chip RAM, then it puts itself into turtle mode automatically. But if the program keeps its code in Fast RAM, then you would need to trigger turtle mode manually. There are two ways to do this:
    1. **Set the TURTLE bit in the Processor Configuration Register**: This bit can be toggled on and off using `VControl TURTLE`, or by pressing `F12`. (If you are using WHDLoad, you can use the `ExecuteStartup` and `ExecuteCleanup` tooltypes to run `VControl` TURTLE.)
    2. **Force program code into Chip RAM**: This can be done by turning off Fast RAM completely. Fast RAM can be disabled by running `SYS:System/NoFastMem`, or by using a degrader tool. (If you are using WHDLoad, you don't have to turn off Fast RAM, because you can use the `ExpChip` tooltype to force program code into Chip RAM.) Once the 68080 starts executing program code from Chip RAM, it will put itself into turtle mode automatically, as expected.
- The 68080 is superscalar, meaning that it has two pipelines and can execute more than 1 instruction in parallel. If an instruction in one pipeline attempts to modify an instruction in RAM, but that instruction has already entered the second pipeline, then the CPU can't detect the modification. The second pipeline continues with the old version of the instruction, and the program fails. This rare form of self-modifying code is used in some demos. To get such programs to run correctly, you can use `VControl SUPERSCALAR` to turn off the superscalar features of the CPU. (If you are using turtle mode, then you do not need to do this, because superscalar features are automatically turned off in turtle mode.)
- If you want to run a CD32 program that uses Akiko C2P routines, you should use a Kickstart that contains the CD32 extended ROM, so that Akiko C2P routines get initialized by the OS automatically. If you can't do this, then you can still initialize these routines manually, using `VControl AKIKO`.
- If you have Core version >= GOLD3 (as in the Vampire Standalone), the default chipset configuration is PAL, not NTSC. But you can easily switch to NTSC mode from the Amiga Early Startup Control, or from a degrader tool. (If you are using WHDLoad, you can use the `NTSC` tooltype.)
- If you have Core version >= GOLD3 (as in the Vampire Standalone), you can press `F11` to toggle scanlines, in order to get a video experience that is similar to old CRT monitors.
- The Vampire patches `exec.library` to keep the lower 8 KB of Chip RAM reserved, which is required by Macintosh emulators. You should not use any other patches that do the same thing. For example, you should not use the `PrepareEmul` program that comes with ShapeShifter. To complete preparing your Vampire's memory for Macintosh emulators, you simply need to move the VBR to Fast RAM. You can do this with `VControl VBRMOVE` or with a tool like VBRControl.
- If you have a Vampire accelerator board connected to a classic Amiga, see “Precision” under 68080 FPU Core for some compatibility concerns regarding software requiring an FPU.

[1] All instructions are natively implemented, except for deprecated FPU instructions, which are emulated in the firmware using optimized mechanics.
[2] All FPU data types and EA modes are natively implemented, except for deprecated data types, which are emulated in the firmware using optimized mechanics.
[3] This is similar to how other Amiga models and accelerators solve the issue.