Q&A

Q: The DOS Library function FGets() seems to read too many bytes if it doesn't come across a newline character. What's happening?

A: This is an excerpt from a recent Autodoc for FGets:

In V36 and V37, it copies one more byte than it should if it doesn't hit an EOF or newline. In the example above, it would copy 50 bytes and put a null in the 51st. This is fixed in dos V39. Workaround for V36/V37: pass in buffersize-1.

Q: Is there an interaction between DMA devices (such as hard drive controllers) and the 68040?

A: Yes. This interaction is only an issue when the 68040 is in copyback data cache mode and the timing is just right. Note that the chance of this interaction affecting any particular application is rather small. The interaction is as follows:

The 68040 data cache has a mode called copyback. In this mode, when the CPU modifies memory, it does not write the data to memory right away. It waits until the cache is needed to cache some other memory or someone explicitly flushes the cache. This creates a problem for DMA devices because they can read directly from memory without using the CPU. If a DMA device tries to read memory that happens to be in the CPUs data cache, the DMA device will read the wrong data.

To prevent problems with DMA, a DMA device has to call CachePreDMA() before accessing memory. This function does both address translations (for future MMU issues) and cache flushing as needed. When the transfer is complete, the device has to call CachePostDMA() which performs some cache flushes and whatever cleanup it needed.

Unfortunately, there is a nasty interaction between the copyback mode of the 68040 and flushed data areas.

In copyback mode, the 68040 uses complete cache lines (which are four long words in length) when copying data back and forth from the cache. There are good reasons for this, including that the CPU can do a full burst transfer (which is four long words). However, the 68040 does this even if only 2 of the long words in the cache line are changed. So, what does this mean?

Imagine a program that has two memory buffers right next to each other. The program is running on a 68040 in copyback mode. These buffers meet in the middle of a 128 bit boundary, so a cache line would overlap the end of the first buffer and the beginning of the second buffer. If the timing is just right, the following situation can occur:

The program calls CachePreDMA() and starts an asynchronous DMA write to the second buffer.

The program uses the CPU to write to the last two long words of the first buffer which causes the CPU to load the end of first buffer and the beginning of the second buffer as they are part of the same cache line.

The DMA write writes to the beginning of the second buffer and finishes the transfer.

The program calls CachePostDMA(), which causes the CPU to write the cache line that overlaps the first and second buffer.

Because the overlapping cache line still contains the data that was in the second buffer before the DMA transfer took place, the CPU overwrites the the first two long words of the second buffer.

The current version of *68040.library* (37.4) has the code needed to work around this problem. Note that if any developer used SetFunction() to patch some of the cache control functions, it is important to patch all of these functions with SetFunction() as needed to work around this problem. *68040.library* already does SetFunction() all of the cache functions to deal with copyback issues and this one just happens to be the most complex of them. Version 37.4 of the 68040.library is available through the CATS closed developer confoerences on BIX.

Q: I'm using ExAll() (dos.library 37.44) with `ED_COMMENT`. Everything works fine except that the string ead->ed_Comment points to isn't `NULL`-terminated.

Is this a bug or am I doing something wrong? If it's a bug, is there a more simple workaround than looking at the FileInfoBlock?

A: This is a bug. It turns out that the V37 FFS does the comment wrong when in ExAll(). It does it as a BSTR rather than a standard C-String. Other filesystems (such as RAM) do it right. Also, filesystems that do not directly support ExAll() but have DOS simulate ExAll() do it right, too.

Q: The 37.4 DOS library Autodoc says that the FPutC() function looks like this:

```
LONG FPutC(BPTR, UBYTE)
```

while version 37.4 of the <clib/dos_protos.h> include file looks like this:

```
LONG FPutC( BPTR fh, unsigned long ch)
```

Which one is right?

A: Would you believe neither? It's really:

```
LONG FPutC(BPTR, LONG ch)
```

There's no particularily good reason for it, other than that's how BCPL did it, and this is a transliterated (directly translated) version of the BCPL. BCPL has only one real type: LONG.

It wouldn't make a difference except that the value returned for success is the longword passed in, and longword comparisons are done against things like '\n', etc. So you must pass in a longword value 0-255.

Q: How do I determine the frequency of the CIA B time of day (TOD) counter?

A: The CIA B time of day counter counts horizontal sync pulses. The counter rate varies depending on the video mode the Amiga is currently displaying. Instead of using the TOD counter, you want use one of the more conventioanl CIA timers, and scale the time constants based on whether the system is PAL or NTSC.