WHAT SCSI-ACCELERATOR IS ALL ABOUT.

The accelerator works only on a MacPlus! It enhances the throughput of I/O operations for so called blind read and write operations. Nothing else is affected.

The reason that the performance of these operations can be enhanced is that Apple's code to do these operations is (and must be) suited to handle a variety of disk types. Some of these are slower than others. In the following discussion we will talk about read operations only, but the discussion applies just as well to write operations.

When transfering data to or from a SCSI device, there is no support for hardware handshake on the MacPlus (there is some on the other Macintoshes). Because of this, the only really safe way to do the I/O is to poll the SCSI chip for the arrival of a new byte each time you need one. Of course this is slow. Therefore Apple introduced the "blind" operations. For blind operations, the system only waits for the arrival of the first byte, the rest of them are read in a small loop which looks like this:

```
@1      move.b      (a1),(a2)+
        dbra        d6,@1
```

This means that after every transfer of a single byte, the Macintosh waits at least the time to execute the DBRA (about 10 cycles) before fetching the next byte. This is long enough for even the slowest hard disk that Apple anticipated to have the next byte ready. In fact, most hard disk can deal with a lot less! Reduction of this "dead" time can be achieved by unfolding the loop, i.e. reducing the loop trip count and putting more than a single move.b instruction in the loop body. Of course, if we put two moves right next to each other, we get the fastest transfer that is possible (knowing that we do not have a DMA controller). This might be too fast for some hard disks, so it may well be that we have to put one or more NOP instructions between each two move.b instructions. The execution time of a NOP is only 4 cycles however, much less than for the DBRA and thus, throughput can be increased even if we need 2 NOPs per move.b. By increasing the number of move.b instructions in the loop body, we further decrease the looping overhead, leading to increased performance, but of course, also to more use of memory for the code.

Since the loop illustrated is 6 code bytes long, there is just enough place to replace the loop with a JSR instruction to a patched version of the loop that applies these techniques. This is exactly what the SCSI-Accelerator does. The reason that this does not improve performance on the Mac SE or II/IIx is that those machines support a sort of pseudo DMA transfer mode that already takes care of getting in the bytes as soon as they arrive. For this reason, the accelerator init refuses to install itself on anything other than a Mac Plus.

HOW TO USE IT.

As said before, if you do not operate a Mac Plus, forget it, it will not install! If you are using a MacPlus, the thing to do is to find out what kind of unfolded loop will still work for your configuration. This depends mainly on two things: disk type(s) and processor (in case you operate an accelerator board). For this reason, several variants of the init code have been provided. They are all named SCSI-Accel-r<x>w<y>s<z>. Here the <x>, <y> and <z> are

single digits meaning:

<x> The number of NOPs inserted after each move.b in the loop body for reading.
<y> The number of NOPs inserted after each move.b in the loop body for writing.
<z> The number of move.b instructions in the loop body is equal to 2^<z>

This means that in general you should use the init with the lowest <x> and <y> that still works with your configuration. Once you know which one to use, the next thing to do is to decide how you want to trade of memory and speed by choosing the <z> that you want to use. In general z=5 works quite well. You find out which <x> and <y> version to use by putting one of the INITs in your system folder and rebooting. If the boot works it is quite likely that that version works for you. Test this by duplicating a file with the finder. You should start to test this with a variant with <x> and <y> large. If it works you can progressively try out lower numbers. Don't worry, during boot the disk is only read and even if the Mac crashed (which is the usual symptom of a <x>,<y> which is too low), no real harm will be done.

The code in this patch should work with all disks (providing you choose the right <x> and <y>. Disks with block sizes that are a multiple of 2^<z> work fastest, but any other block size will be handled correctly (for those of you that have disks that operate with tags). The original accelerator worked only with 512 bytes/block disks.

HOW GOOD DOES IT WORK?

Since the SCSI manager read and write blind operations are patched, you will not get improved performance if your disk's driver does not use the SCSI manager. In that case it is quite likely that the driver will already do the same kind of optimization as suggested here, so there won't be much to gain here. Supposing your driver *does* use the SCSI manager, performance will improve, depending on the type of disk you have. An example:
Using a HD SC80 (Quantum Q280) disk with my own custom driver that *does* use the SCSI manager we get the following DiskTimer II results:

Variant	Reads	Writes	Seeks
none	106	105	18
r0w0s5	63	67	18
r1w1s5	83	82	18

The SC 80 is a disk which is formatted with a 1:1 interleave. Its controller caches a complete disk track though, hence the almost twofold improvement. For a Rodime RO632 (Some Apple HD 20's) and the same driver we get:

Variant	Reads	Writes	Seeks
none	160	161	52
r0w0s5	109	107	52
r1w1s5	110	108	52

This disk is quite a bit slower, but still there is improvement. What I have not yet tested is what happens if, in addition to using r0w0s5, I also reformat the disk with a lower interleave factor. It might very well be that because of the improved data transfer rate, the Mac Plus

can keep up with an interleave that is one lower. In that case, performance would improve even more.

Apple's drivers also use the SCSI manager, so the INIT should at least work with that software. It is known that some Rodime drivers bypass the SCSI manager. You will have to try and measure to see wether or not the INIT works for you. In general: just try and see.