PERSISTOR
Instruments Inc.

254-J Shore Road, Bourne, MA 02532, USA
Tel: 508-759-6434 Fax: 508-759-6436
www.persistor.com info@persistor.com

Data Acquisition and Storage Solutions for Industry and Science

How fast can I write to the cards?

Introduction

Probably the most common questions we get is some variation of "How fast can I write to the cards?" Unfortunately, unless a ballpark "Faster than 1KByte/sec" is good enough for your application, there's just no simple answer. Actual rates can vary from below 1Byte/sec to over 300KBytes/sec depending on the following factors:

- System Speed
- CPU Loading
- Block Size
- C Buffer Size
- Writing Techniques
- Cache Mechanics
- Active Files
- File Commitment
- Card Technologies

System Speed

The operating speed for the microcontrollers on current Persistor products can vary over two orders of magnitude with a strong, but non-linear affect on card write-performance. Obviously, faster clock rates make for faster write rates. However, other system-wide considerations may make the fastest CPU speeds impractical and those compromises will have a negative effect on card rates. Memory and I/O wait states affect the efficiency of the raw CPU speed and slowed bus accesses will also result in reduced card rates. Only you have final control of these critical factors. All of the upper card rates discussed here assume the Persistor running at top speed with the factory default wait states.

CPU Loading

The flash card subsystem competes for CPU resources with other running subsystems and processes. A system that acquires data at high rates and performs complex processing or transformations on the data will have less time to service the card and the write rates will suffer proportionally. Again, only you have control of this critical factor. All of the upper card rates discussed here assume that at least 90% of the CPU clock cycles are devoted to the card subsystem during file operations.

Block Size

Within certain limits, the larger the chunk of data that you write, the faster the write rate. This ratio has always been a factor with flash cards, but with the switch from NOR to NAND technology cards in late 2001, its importance has grown literally by a factor of twenty. The flash cards exist to serve the huge digital camera market so it's no surprise that their characteristics evolved to meet the demands of 2MByte pictures in rapid succession. The technologies which make that possible exact a price at the opposite end – small blocks have very slow write rates. For Persistor products, blocks of

64KByte and up give the best write rates while blocks below the native sector size of 512Bytes yield horrendous write rates. The latter is disproportionately bad because sub-512Byte writes require pre-reads and lots of memory transfers. There is a downside to large buffers that only shows up during a catastrophic system failure – the data waiting in memory never makes it to the card and will not be recoverable. Again, only you have control of this critical tradeoff factor. All of the upper card rates discussed here assume that you are writing in blocks of at least 64KBytes.

C Buffer Size

Closely related to block size is the size of the buffer used by your compilers Standard C Library for the buffered I/O functions like fwrite and fprint. Aztec C supplies a default buffer size of 2046 Bytes and Metrowerks supplies a 4096Byte default. While acceptable for modest data rate applications with older NOR cards, these just won't cut it for high rate applications or the new NAND cards. Increasing the C buffer size with the setvbuf call indirectly changes the block size with the same enhancements and tradeoffs discussed above. This, too, is now under your control. All of the upper card rates discussed here assume that you are using a C buffer size of at least 64KBytes for fwrite operations.

Writing Techniques

How you write to the cards can also have an effect on write rates. The standard C library offers fputc, fputs, fwrite, and fprint to do the job. These could be classified as good, better, best, and absolute worst. Everything but fwrite carries a lot of overhead penalties and should only be considered for slow data rate systems. In particular, fprintf invokes an interpreter with each call that really grinds writes to a halt. For versions of PicoDOS that support the POSIX file functions, you can gain some small speed improvements by using the lower level write function instead of fwrite, but you must take care to make sure that all writes occur in blocks of 512 bytes and that all buffers are passed with starting locations that fall on even boundaries. Failing to do so will actually result in slower operations than fwrite, which automatically takes care of such things. All of the upper card rates discussed here assume that you are using fwrite to move data to the card.

Cache Mechanics

PicoDOS's FAT file system gives us the ability to simply toss a memory card into a PC for data recovery, but that requires a number of behind the scenes card operations that interfere with large block writes. In particular, the critical FAT tables that PCs use to index a card's contents frequently need to be updated several times during a large block write. The only way to eliminate the penalties incurred by interspersing FAT read/writes is to buffer these in a ram cache until the block write concludes. Just as with large blocks themselves, this carries the potential for lost data in the event of a system failure. If the FAT tables do not get written, the data they index is effectively lost. For maximum write security, all PicoDOS systems default to leaving the cache turned off, and you must explicitly enable it as a system wide preference or inside your programs. All of the upper card rates discussed here assume that you are enabling any and all of the PicoDOS cache optimizations.

Active Files

The number of open and active files effects the overall write rates. Interleaving files complicate the cache mechanisms and cause extra card operations decreasing speed. All of the upper card rates discussed here assume that you are writing to a single file.

File Commitment

PicoDOS performs a lot of housekeeping every time you open and close a file. In general, opening always forces a complete re-scan of the directory sectors and closing always results in flushing all the caches and resetting any optimization assumptions. You cannot expect to achieve streaming rates much higher than several KBytes/sec if you periodically need to close and reopen for maximum card write security. All of the upper card rates discussed here assume that you keep the file open for the duration of the write operations.

Card Technologies

As mentioned in the block size description, the underlying card technologies play a major role in card speed. Fortunately, the five year trend has all been towards faster, better, cheaper, and lower power. However, things like the NOR to NAND transition which help make the continuing trend possible, can also cause unexpected new behaviors that may require periodic updates to existing support software. That implies that some of the statements made above may become irrelevant or just plain wrong as the technology continues to evolve. That also implies that you will need to keep abreast of major changes if flash cards become a critical component in your research or business. We'll be here to try and help.