Data Logging with Flash Memory File Systems

A common use for Micro-Robotics products is data logging, made possible by Venom's file system support for either RAM disk, the VM2's on board flash memory or a memory card interface, like the one on Application Board 2 (product 5902). However the nature of flash memory and filing systems needs to be taken into account when designing systems for maximum reliability and lifetime. This application note sets out best practices for each media type and also offers an approach for systems where a memory card is likely to be manually removed.

Summary of File System Capabilities

Media	*Capacity*	*Write wear*	*Cache*	*Speed R/W*
RAM	900KB max	None	n/a	fast/fast
On Board Flash	7MB	yes, no wear levelling	2x64K non-volatile	fast/slow
Memory Card	32GB max	yes, but mitigated by wear levelling	5K-100K volatile	slow/slow

Write Wear in Flash Memory

All flash memory devices have a limited number of write cycles to each page in memory. For example a San Disk MMC datasheet specifies 300,000 write cycles, which implies that non-stop periodic updates would give approximate card lifetimes as follows:

Update period	*Card Lifetime*	*Update period*	*Card Lifetime*
1 second	3 days	2 minutes	1 year
10 seconds	1 month	10 minutes	5 years
1 minute	6 months		

However, memory cards use a technique called wear levelling to maximise the lifetime of the card by remapping blocks and spreading rewrites evenly throughout the memory. The performance of the wear levelling algorithms is difficult to predict as their detailed operation is made public and they probably depend on file system usage patterns and the internal structure of the memory card. However, it seems likely that using a small fraction of a large capacity card would enable wear to be spread widely and thus improve the lifetime of the card.

The internal flash memory used by the VM2 is also subject to write wear, and of course is more expensive and cumbersome to replace (you have to replace the whole VM2 card). It also doesn't have any wear levelling, but Venom uses a non-volatile RAM cacheing scheme that can

The Old Maltings
135 Ditton Walk
Cambridge
CB5 8QB

tel: +44 (0) 1223 523100
fax: +44 (0) 1223 524242
sales@microrobotics.co.uk
www.microrobotics.co.uk

dramatically reduce the amount of writing to the flash memory.

The Flush Message

Both memory card and on board flash use cache memory in RAM to hold recently accessed blocks of data. When writing, data is written to the cache memory first, and later the cache block is written to the actual media (memory card or flash memory). A `Flush` message sent to the file system object forces all such data to be written immediately, bringing the storage media up to date. However there are conflicting requirements: to minimize the loss of data if the power fails, and to minimise write wear on the storage media.The criteria for when and how often to use the flush message are different for memory cards and the on board flash.

Memory Cards – Best Practice

1. Writing Infrequently to a Card

If you are writing a batch of data to a file once per minute or less frequently:

1. Open file
2. Write updates to file
3. Close File
4. Send a `Flush` message to File System

2. Writing Frequently or Continuously To a File or Files

Start a separate task to flush the file system periodically:

```
MAKE fs FileSystem("MMC")
START EVERY 60000 fs.Flush  ; update memory card once per minute
```

Open files, leave them open and simply write to them as required.

3. Keeping Potential Data Loss to an Absolute Minimum, and Enabling Memory card to be Temorarily Removed during Operation.

Create a RAM disk and write data to a file (or files) on that in the first place.

When the RAM disk file exceeds a certain size or when RAM disk free space is lower that a set limit, and if the memory card is present:

1. Open a file on the memory card
2. Append the RAMdisk file contents to the Memory card file.
3. Close the memory card file.
4. Flush the memory card file system
5. Empty the Ram disk file.

See example code at the end of this document.

You can also use the flash file system instead of RAM disk, with the advantage of higher temporary storage capacity.

Memory Cards – Technical Details

The cache memory for memory cards is volatile i.e. it will be lost if the power is cut. Fortunately, a large capacity memory card will tolerate relatively frequent writes because of its built in wear levelling, and the card itself is relatively cheap and replaceable.

The FileSystem `Flush` message ensures that all file directory entries are brought up to date with the current file size, and then writes all cached write data to the card.

If power is cut and re-applied, the file system will attempt to come up in the same state it was after the last `Flush` message was sent to it. Any writes after the `Flush` message and before loss of power will be lost.

(Sending `Flush` to individual files in a memory card file system does nothing useful)

On Board Flash Memory – Best Practice

1. If there are frequent log file updates, it's best not to interleave many small writes to different files.
2. Keep files for writing in the root directory.
3. After writing to a file, send a `Flush` message ***to the file*** (not to the file system).
4. Ensure the VM2 is fitted with a Lithium backup battery.

On Board Flash Memory – Technical Details

Updates are written in the first instance to a non-volatile RAM area, and if the system is powered off and on again the data is retained. The Flash File System has two page-cache buffers of size 64k (the flash memory's erase block size). One is fixed at the first page of the memory and holds the FAT (File allocation table) and root directory which are usually the most frequently accessed parts of the file system. The other is movable and is only written to the flash when a new page is needed for writing, e.g. when 64k of data has been written to a file or when switching from one file to another and, as is likely, the data is on different pages.

Because the cache pages are non-volatile (held in battery backed RAM), sending the Flush message to the Flash FileSystem is not recommended for most applications and may actually cause unnecessary wear.

Sending a `Flush` message to a file *is* necessary as it copies the file length information from a volatile data structure in the running program to the nonvolatile cached block corresponding to the file's directory entry. If you fail to do this, the data written to the file since the last Flush of the file will be lost. Usually the file `Flush` message will only update a cache block, not result in writing to the flash.

Because there are only two 64K pages of cache, it is recommended to keep log files in the root directory so that all directory updates remain in Page 0. Note there is a limit of 512 files in the root directory

Frequent writes to multiple files may encourage thrashing of the moveable cache page, so it's best not to interleave many small writes to different files

Note that use of the flash file system takes away 128K from the available system RAM for use as cache memory.

RAM disk – Best Practice

1. Choose RAM disk size carefully – it's a compromise between system memory and file system capacity.
2. Check file system space and be prepared to discard data if the limited space becomes full. There is a feature that will do this automatically, discarding the oldest data.
3. Ensure the VM2 is fitted with a Lithium backup battery.

RAM Disk – Technical Details

RAM disk is very fast for both reading and writing, has no wear problems at all and is non-volatile as long as the battery is fitted and working. It does, of course, have limited data capacity compared with the other media, and it the memory reserved for it is taken away from the pool of system memory. RAM disk files have their directory entry length updated every time the file is written, so don't need a file `Flush` message unless you also want to set the time and date stamp, and of course the there is never a need to send a `Flush` message to the RAM filesystem.

Example Code for Dealing With Memory Card Removal

The code below shows how to use a memory card that can be removed while a program is running, and also to minimise the possibility of loss of data due to power failure.

It is not safe to remove the card while a file is open or write data is in the cache (this is indicated by the LED on the memory card interface) so the file should be kept closed most of the time and the filesystem should be flushed after closing the file.

A good way to deal with this is to log data to a file on RAM disk, and then copy the RAM disk file contents to the memory card file when a certain file size has been reached.

The update can be preceded by a check for media present (The file system `Valid` message suffices for this) and must consist of opening the file, writing the data and immediately closing the file. An example logging function might look like this:

```
TO init
  MAKE ramfs Filesystem ("RAM", 50*1024)
  ; all logging data is printed to this file
  logtemp := ramfs.open("logtemp.txt", "")
  MAKE logtimer Stopwatch
  MAKE cardfs FileSystem("MMC")
  START copy_logfile
END

TO copy_logfile
  local f

  EVERY 10000 ; check every 10 seconds
```

```
    [
      IF (logtimer.Time > 600000 OrElse logtemp.Length > 20000)
      AndAlso logtemp.Length > 0
      AndAlso cardfs.Valid    ; media must be present
      [
        f := cardfs.Open("log.txt", "")
        WAIT 1000         ; give operator time to notice LED
        IF cardfs.Valid
        [
          logtemp.lock
          logtemp.Reset
          PRINT TO f, logtemp
          f.Close
          logtemp.Empty
          logtimer.Reset
          logtemp.Unlock
        ]
      ]
    ]
  END
```

The application itself can simply `PRINT TO logtemp` and need take no special other precautions. Note that while the card is not present the data will simply accumulate in the RAM disk file until the card is replaced or the RAM disk capacity is exceeded.

Flash File instead of RAM File

Similarly the flash file system could be used for temporary storage. If the temporary file size is limited to less than 64k while the memory card is present, there will be no flash memory wear at all, while in the absence of a memory card it would allow accumulation of a larger volume of data up to the capacity of the flash memory, which is 7MB.

Updated for later versions of Venom – Anahata 2011-06-16