



# *smxFile*<sup>TM</sup>

## **DOS-Compatible Reentrant File Manager**

### **Product Description**

*smxFile*<sup>TM</sup> is a robust, DOS-compatible file system for hard real-time embedded systems. It supports all device types commonly used in embedded systems.

*smxFile* features high-performance file i/o and also provides disk directory management. *smxFile* is written in ANSI C. Source code is provided. The API is similar to POSIX and DOS. The device driver interface is similar to UNIX, but simpler. *smxFile* operates in x86 real mode and in x86 16-bit and 32-bit protected modes. It also operates with PowerPC, ColdFire, ARM, and other processors. A portable, standalone version is available.

This package is the latest in a series of DOS file system software packages that have been marketed since 1987. It and its predecessors have been included in hundreds of commercial embedded applications.

*smxFile* is integrated with SMX<sup>®</sup>. Multiple tasks may perform simultaneous file i/o. Low-level i/o calls to the same disk are serialized via semaphores since actual disk operations are inherently non-reentrant.

### **File Allocation Table (FAT)**

The FAT is a structure on a disk that maps the blocks of disk space occupied by individual files on that disk. These blocks are called clusters. A cluster

### **Features**

- FAT 12/16/32
- DOS/Win9x compatible
- Extensive drivers:
  - Floppy
  - IDE Hard Disk
  - DiskOnChip<sup>®</sup>
  - LS-120
  - PCMCIA-ATA
  - PCMCIA-linear flash
  - Ultra DMA
  - SCSI
  - Zip<sup>®</sup>
  - RAM disk
  - ROM disk
  - BIOS devices
- 20 MByte/sec sustained transfer rate ( IDE Ultra-DMA )
- Up to 2 terrabyte disks
- 55 KB typical code footprint
- 20 KB typical data footprint
- Multitasking support
- Contiguous file support
- Source code included
- Fully integrated with SMX

is a group of sectors, which is the smallest addressable unit of a disk. Sectors are 512 bytes in size. The number of sectors per cluster is

constant for the entire disk or partition<sup>1</sup>. Files typically contain many clusters.

FAT entries are in the same order as physical clusters on the disk and correspond one for one with them. Each FAT entry points to the next FAT entry for the same file or directory (Note: Directories are treated as files). A special value indicates the end of the file. The size of FAT entries dictates the size of the FAT and the maximum size of the disk. There are 3 common FAT entry sizes in use:

FAT12, FAT16, and FAT32. On a FAT12 disk, for example, each FAT entry is a 12-bit number. All FAT's have 18 reserved values (end of file, bad cluster, etc), so the number of FAT entries is given by  $2^n - 18$  (where  $n = 12, 16, \text{ or } 32$ ).

FAT12 is used for small media such as floppy disks. FAT12 has up to  $2^{12} - 18 = 4078$  entries. Floppies are formatted for 1 sector per cluster which means the largest floppy disk supported is almost 2 megabytes ( $512 * 4078$ ).

As the number of sectors per cluster is increased, so is the maximum size of the disk which can be supported.

FAT16 is used for larger media such as hard disks. FAT16 has up to  $64K - 18$  entries. Cluster size varies from 4 to 64 sectors (2KB to 32KB), allowing a maximum disk size of almost 2 *gigabytes*. *smxFile* allows caching (i.e. storing in memory) of any number of FAT sectors of 256 entries, each. To cache the entire FAT requires 128KB of RAM. Caching more sectors of the FAT improves performance by reducing disk accesses for the FAT, itself. (Since the FAT is stored at the beginning of the disk, the drive head must sweep back and forth from the data area to the FAT area. This may cause unacceptable performance if there is insufficient FAT caching.) FAT32 is used for very large

media, such as multi-gigabyte hard disks. FAT32 cluster size varies from 8 to 64 sectors (4KB to 32KB), and allows a maximum disk size of 2 *terabytes*. *smxFile* allows caching any number of FAT blocks for 32-bit memory models<sup>2</sup>. Since each entry is 32 bits, each sector of the FAT stores only half as many entries as FAT16. For hard drives, commonly in use today, the FAT occupies thousands of sectors, so it is often impractical to cache the entire FAT. A high performance, streaming application might cache as many as 500 sectors of the FAT. The best cache size for an application is determined by experimentation.

## Contiguous Files

*smxFile* permits preassigning contiguous clusters to a file, if available. This permits high transfer rates because track motion is reduced for accessing successive file clusters. Since FAT entries for successive file clusters are adjacent, less FAT accesses are also required.

## Long File Names

*smxFile* has long file name (VFAT) support. It will read and write Microsoft Windows 9x-compatible, long file names. Unfortunately, Microsoft has a patent on this technology so we ship with this feature disabled. It can be enabled by a single setting.

## *smxFile* Drivers

*smxFile* supports most storage devices commonly used in embedded systems. Devices supported are as follows:

---

<sup>1</sup> A disk can be divided into separate *partitions*, each with its own cluster size.

---

<sup>2</sup> 16-bit memory models, such as real mode, are limited to a 128KB cache.

**Floppy driver:** Supports 3.5" and 5.25" floppy disks of all capacities (360K, 1.2MB, 720K, and 1.44MB). It supports NEC 765 compatible controllers. Dual drives are supported.

**IDE/ATA driver:** Supports all IDE hard disks and Compact Flash. Interrupt or polled mode. Has timeout protection.

**IDE Ultra DMA driver:** Supports all IDE hard disks and offers Ultra DMA/33 support for drives and controllers with this feature. DMA/44 and DMA/66 are also supported. Code is included to probe the PCI BIOS for the configuration of the bus-mastering DMA controller.

**PCMCIA ATA driver:** Supports PCMCIA ATA disks and PD67xx- compatible PCMCIA controllers. Also permits hot swapping and supports Compact Flash.

**DiskOnChip<sup>®</sup> driver:** Supports the M-Systems DiskOnChip 2000 and Millennium. Based upon M-Systems proprietary TrueFFS<sup>®</sup> technology. Co-developed by M-Systems and Micro Digital. Source code is not included but may be available by special arrangement with M-Systems.

**LS-120 driver:** Supports all LS-120 drives. Also known as SuperDisk<sup>®</sup>.

**Zip<sup>®</sup> driver:** Supports 100MB and 250MB Iomega Zip disks.

**SCSI driver:** Supports SYMBIOS 53C8xx SCSI controllers for SCSI hard disks.

**RAMdisk and ROMdisk drivers:** Included with *smxFile*. A ROMdisk builder utility is also included.

**BIOS driver:** Available in x86 real-mode, only. Allows using any BIOS (int 13H) disk driver such as a resident flash disk driver.

## CD-ROM

CD-ROM requires a separate file system. See the *smxCD* brochure. *smxFile* and *smxCD* can run simultaneously in the same system.

If the storage device you require is not in the above list, contact us — it may be under development.

---

## API Functions

The following functions constitute the *smxFile* API. These functions have been categorized for convenience. All functions are "task-safe" — i.e. can be accessed simultaneously from different tasks.

### File I/O Functions

<code>po_chsize</code>	Extend or truncate a file
<code>po_close</code>	Close a file
<code>po_flush</code>	Flush a file to disk
<code>po_lseek</code>	Move file pointer
<code>po_open</code>	Open a file
<code>po_read</code>	Read from a file

As is apparent from the following list, the *smxFile* API is powerful and complete. However, for those preferring the DOS or Win32 API's, *unDOS* or *unWin* can be used in combination with *smxFile*.

po_truncate	Truncate a file
po_write	Write to a file

### **File Management**

pc_fstat	Obtain statistics on an open file
pc_get_attributes	Get file attributes
pc_mv	Rename a file or directory
pc_set_attributes	Set file attributes
pc_unlink	Delete a file

### **Directory Management**

pc_deltree	Delete a directory tree
pc_gdone	Free pc_gnext and pc_gfirst resources
pc_get_cwd	Get current working directory
pc_gfirst	Return the first entry in a directory matching specified pattern
pc_gnex	Return next entry in a directory matching specified pattern
pc_isdir	Test if a path is a directory
pc_mkdir	Create a subdirectory
pc_pwd	Return the current working directory
pc_rmdir	Delete a directory
pc_set_cwd	Set current working directory

### **Disk Management**

pc_cluster_size	Return a disk's cluster size
pc_diskabort	Abort operations on a disk without modifying disk.
pc_diskflush	Flush the FAT and all files to a disk
pc_format	Format a disk
pc_free	Return count of free bytes on disk
pc_getdfldrvo	Return the current default drive number
pc_isvol	Test if a path is a volume name
pc_set_default_drive	Set the default drive
pc_stat	Obtain statistics on a path
pc_write_protect	Indicates whether media is write protected (floppy only, currently)

### **Contiguous File Functions**

pc_get_file_extents	Get the list of clusters that make up a file
pc_get_free_list	Get a list of free clusters on the drive
pc_raw_read	Read blocks directly from a disk

<code>pc_raw_write</code>	Write blocks directly to a disk
<code>po_extend_file</code>	Extend a file with contiguous clusters

## Additional Support

In addition to the preceding API functions, low-level functions are provided for:

- (1) initialization
- (2) directory management
- (3) helper functions
- (4) block buffering
- (5) string processing

These are normally not needed at the application level, but are available, if needed. The string functions may be useful for other purposes. The package also includes a simplified `sprintf()`.

## Drives

The number of drives of each type is specified in `pcconf.h`. The order of definition and the number of each type determine drive lettering.

## Code Sizes

*smxFile*, with IDE and floppy drivers is approximately 55 KB for a 32-bit x86 processor.

## Performance

*smxFile*, with the Ultra-DMA driver, will support sustained data transfer rates of 20 MByte/sec (measured on a 20 GB 7200 rpm drive). Performance depends on the density and rotational speed of the drive.

## RAM Usage

RAM usage is tunable. It is determined by configuration parameters in `pcconf.h`. Memory is allocated statically from the BSS segment or section. The configuration parameters are:

<code>FAT_CACHE_SIZE</code>	Number of sectors of FAT that are cached in memory
<code>NBLKBUFFS</code>	Number of blocks in the buffer pool. (Which stores disk sectors during directory traversals.)
<code>NUM_USERS</code>	Maximum number of tasks that may access the file system
<code>NUSERFILES</code>	Maximum number of open files

In addition, tasks using *smxFile* require stacks of about 1500 bytes for 16-bit memory models or 2500 for 32-bit memory models.

## Portable *smxFile* (psf)

This version of *smxFile* does not require *smx*. It can be ported to any RTOS or used in a standalone mode. It can also be ported to other processors.

## What is Delivered

- Full source code, including purchased drivers
- Manual
- Make file to build the *smxFile* and driver library
- Demo