GPL32XXX File System Library User Manual

V1.0 - Jul. 16, 2009

For 立奕企業股份有限公司 Confidential Use Only

3F, No.8, Dusing RD., Hsinchu Science Park, Hsinchu City 30078, Taiwan, R.O.C.
Tel: 886-3-666-2118 Fax: 886-3-666-2117 www.generalplus.com

Important Notice

Generalplus Technology reserves the right to change this documentation without prior notice. Information provided by Generalplus Technology is believed to be accurate and reliable. However, Generalplus Technology makes no warranty for any errors which may appear in this document. Contact Generalplus Technology to obtain the latest version of device specifications before placing your order. No responsibility is assumed by Generalplus Technology for any infringement of patent or other rights of third parties which may result from its use. In addition, Generalplus products are not authorized for use as critical components in life support devices/ systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Generalplus.

For 立奕企業股份有限公司 Use Only
Generalplus Confidential

Table of Content

For Generalplus Confidential 立奇企業股份有限公司 Use Only

Generalplus Confidential
For 立奕企業股份有限公司 Use Only

Revision History

Revision	Date	By	Remark
V1.00	2009-7-16	Jacky Lin YaoZurong	Original Version
V1.01	2007-8-2	zhangzha	Add program example

For Generalplus Confidential 立奕企業股份有限公司 Use Only

1 Introduction

1.1 General Description

This guide describes the functionality and user API of DOS FAT/FAT32 File System for GPL32 system.

For 立奕企業股份有限公司 Use Only
Generalplus Confidential

2 Function List

Name	Function	Input Parameter	Return	Description
C language call				
open	Open a file	LPSTR pathname int flags	int	Open the specified file with the specified mode.
close	Close a file	int filedes	int	The function close the file specified by the file node index and flush the buffers associates to the file.
read	Read data from a file	int filedes unsigned long buffer unsigned int size	int	The read function reads up to size bytes from the file with descriptor filedes, storing the results in the buffer.
write	Write data to a file	int filedes unsigned long buffer unsigned int size	int	The write function writes up to size bytes from buffer to the file with descriptor filedes.
lseek	Sets the read-write file pointer	int fd long offset int whence	long	The lseek sets the read-write file pointer for the open file specified by the fd.
mkdir	Create a directory	LPSTR pathname	int	Directories are created with the mkdir function.
rmdir	Remove a directory	LPSTR pathname	int	The rmdir function deletes a directory.
chdir	Change the directory	LPSTR pathname	int	This function is used to set the process's working directory to *filename*.
getcwd	Get current directory	LPSTR buffer int size	LPSTR	Get current directory if success return buffer address else return NULL.
unlink	Delete a file	LPSTR pathname	int	Delete the specified file.
rename	Rename a file for directory	LPSTR oldname, LPSTR newname	int	The function can be used to move or rename a file or a directory.
stat	Get a file`s status	LPSTR filename struct stat *buf Int	int	The stat function fills the specified structure with the information about the specified file.
fs_init	Initialize file system.	void	void	Initialize file system.
fs_uninit	uninitialize file system.	void	void	uninitialize file system.
_ getfserrcode	Get the last error code	int	void	Get the last error code of the file system.

For Generalplus Confidential 立奕企業股份有限公司 Use Only

Name	Function	Input Parameter	Return	Description
C language call				
_clsfserrcode	Clear the error code	void	void	Set the global error code value to zero
_findfirst	Find the first file	LPSTR pathname struct f_info *f_info, unsigned int attrib	int	Find the first appointed name and attribute's file.
_findnext	Find the next file	struct f_info *f_info	int	Find next appointed name and attribute's file.
_copy	Make a copy of a file	LPSTR srcfile, LPSTR destfile	int	The function can be used to make a copy for a file.
_format	Format the driver	unsigned char drv unsigned char fstype	int	Create a file system with the specified driver.
_deleteall	Delete all files and folders in the specified directory.	LPSTR pathname	int	Delete all files and folders in the specified directory.
_devicemount	Mount a disk	unsigned char diskid	int	Mount a disk, load the information about the device and the file system information on the device.
_deviceunmount	Umount the specified device	unsigned char diskid	int	Umount the specified device, flush all cached data associates to the device.
_getdiskfree	Get information about the space	short driver, struct _diskfree_t * dfreep	int	Get information about the space of the specified device.
Assembler call				

Generalplus Confidential Use Only
For 立奕企業股份有限公司

3 Global Variable List

Name	Description	Setting Function	Getting Function	Condition
None				

For Generalplus Confidential 立奕企業股份有限公司 Use Only

4 Resource List

4.1 RAM Size

	IRAM	ISRAM	RAM	SRAM	ORAM	OSRAM
File system	0	0	2194	0	0	0

4.2 ROM Size

	TEXT	CODE	DATA
File system	0	40916	0

4.3 Peripheral

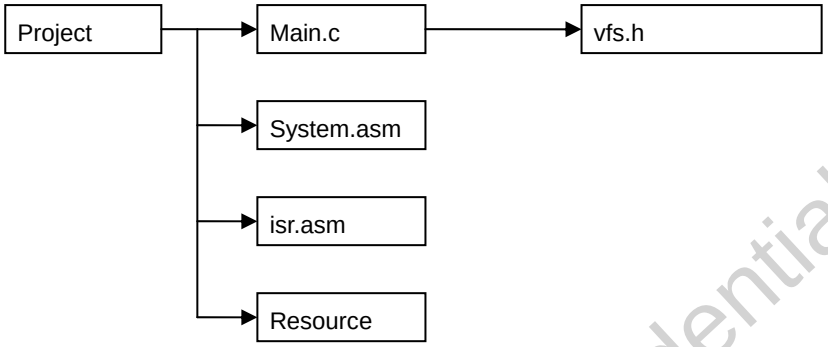
	Timer	TimeBase	IRQ	FIQ	Others
File system					

4.4 Other

Generalplus Confidential Use Only
For 立奕企業股份有限公司

5 Project Architecture

5.1 C Architecture



5.2 ASM Architecture

Generalplus Confidential
For 立奕企業股份有限公司 Use Only

6 Application Interface

6.1 open

API Name		open
Function Description		Open the specified file with the specified mode.
Header File	C	vfs.h
	ASM	
Syntax	C	int open (LPSTR *pathname*, int *flags*);
	ASM	
Parameters	C	pathname pointer to a path name string flags open flag O_OPEN Open a exists file O_TRUNC Open the file and truncate the file to zero length. O_CREAT If set, the file will be created if it doesn't already exist. O_RDONLY Open the file for read access O_WRONLY Open the file for write access. O_RDWR Open the file for both reading and writing. O_EXCL If set, the open operation will fail when trying to create an existent file
Return Values		The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of -1 is returned instead. **NOTE: Return value is a file node index. Maybe zero.** **Error Code List:** EACCES The file exists but is not readable/writable as requested by the flags argument, the file does not exist and the directory is unwritable so it cannot be created. EEXIST Both O_CREAT and O_EXCL are set, and the named file already exists. EISDIR The flags argument specified write access, and the file is a directory. ENFILE The entire system, or perhaps the file system which contains the directory, cannot support any additional open files at the moment. ENOENT The named file does not exist, and O_CREAT is not specified. ENAMETOOLONG Filename specified too long. EIO For many devices, and for disk files, this error code indicates a hardware error. ENAMEINVALID Invalid character detected in the filename string. ENOSPC No free directory entry left in root directory on a FAT16 or

For Generalplus Confidential 立奇企業股份有限公司 Use Only

	FAT12 file system or the file system doesn't have enough room to extend the directory.
Remarks	

Example	#include <stdio.h>
	#include <string.h>
	#include "vfs.h"
	
	void main (void)
	{
	int fp, err;
	char string[] = "Hello , world\n";
	// open with create options
	if ((fp = open ("A:\\test.txt", O_CREAT | O_EXCL)) == -1)
	printf ("The file 'test.txt' was not created\n");
	else
	printf ("The file 'test.txt' was created\n");
	// write string to file
	write (fp, string, strlen(string));
	close (fp);
	}

6.2 close

API Name		**close**
Function Description		The function close the file specified by the file node index and flush the buffers associates to the file.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int close (int *filedes*);
	ASM	
Parameters	C	*filedes* File node index, maybe it's "open" 's return value.
	ASM	None
Return Values		The normal return value from close is 0; a value of -1 is returned in case of failure. **Error Code List** EBADF The *filedes* argument is not a valid file descriptor.
Remarks		

Example		fp = open (".\\Hello.txt", O_CREAT | O_RDWR);
		write (fp, "Hello, world.", 14);
		close (fp); // system will write the data to disk

For 立奕企業股份有限公司 Generalplus Confidential Use Only

6.3 read

API Name		read
Function Description		The read function reads up to size bytes from the file with descriptor filedes, storing the results in the buffer.
Header File	C	vfs.h
	ASM	
Syntax	C	int read (int *filedes*, unsigned long *buffer*, unsigned int *size*);
	ASM	
Parameters	C	*Filedes* File node index, maybe it's "open" 's return value. *Buffer* Buffer pointer. It's data pointer given to specify an offset in the SRAM. *Size* Reads up to *size* bytes from the file.
	ASM	None
Return Values		The return value is the number of bytes actually read. This might be less than *size*; In case of an error, read returns -1. EBADF The filedes argument is not a valid file descriptor, or is not open for reading. EIO For many devices, and for disk files, this error code indicates a hardware error.
Remarks		

Example		int fp;
		fp = open (".\\Hello.txt", O_CREAT | O_RDWR);
		write (fp, 0, 14); // write 14 bytes start form SRAM 0 to file
		close (fp); // system will write the data to disk
		fp = open(".\\Hello.txt", O_OPEN | O_RD);
		read (fp, 0, 14); // read 14 byte form the file into the first 14 bytes of the SRAM
		close (fp);

6.4 write

API Name		write
Function Description		The write function writes up to size bytes from buffer to the file with descriptor filedes.
Header File	C	vfs.h
	ASM	
Syntax	C	int write (int *filedes*, unsigned long *buffer*, unsigned int *size*);
	ASM	

For Generalplus Confidential 凌通企業股份有限公司 Use Only

Parameters	C	*filedes* File node index, maybe it's "open" 's return value *buffer* Buffer pointer. It's data pointer given to specify an offset in the SRAM. *size* Writes up to *size* bytes from buffer to the file.
	ASM	
Return Values		The return value is the number of bytes actually written. This may be *size*, but can always be smaller. Your program should always call write in a loop, iterating until all the data is written. In the case of an error, write returns -1. **Error code list** EBADF The *filedes* argument is not a valid file descriptor, or is not open for writing EIO For many devices, and for disk files, this error code indicates a hardware error. EACCES File access mode error. Write to a file which is opened with O_RDONLY access mode. ENOSPC Media is full. File system cannot allocate any free cluster for the new write operation.
Remarks		

Example		// creat a new file
		fp = open (".\\Hello.txt", O_CREAT | O_RDWR);
		// write a string data to file
		write (fp, 0, 14); // Write the first 14 bytes in SRAM into the file
		close (fp);

6.5 lseek

API Name		**lseek**
Function Description		The lseek sets the read-write file pointer for the open file specified by the fd.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	long lseek (int *fd*, long *offset*, int *whence*);
	ASM	

For Generalplus Confidential 立奕公司股份有限公司 Use Only

Parameters	C	*fd* Index of the file node. *offset* The offset will be set. *Whence* SEEK_SET Sets the file pointer to the value of the *offset* parameter. SEEK_CUR Sets the file pointer to its current location plus the value of the *offset* parameter. SEEK_END Sets the file pointer to the size of the file plus the value of the *offset* parameter.
	ASM	None
Return Values		The return value from lseek is normally the resulting file position, measured in bytes from the beginning of the file. You can use this feature together with SEEK_CUR to read the current file position. If the file position cannot be changed, or the operation is in some way invalid, lseek returns a value of -1. **Error code list** EBADF The *fd* parameter is not an open file descriptor EINVAL The *whence* argument value is not valid or seek to an invalid position
Remarks		When a file was "lseeked" to a place where is out of the size of the file and a write operation followed, file size extending operation will be performed. It means that file system will read and adapt the FAT for free cluster allocation. The clusters are dirty because file system never try to clean the rubbish data on the free clusters.

Example		int fid;
		// open a text file for reading
		fid = open ("a:\\tempfile.txt", O_OPEN | D_RDONLY);
		// set the read-write file pointer
		lseek (fid, 100, SEEK_SET);

6.6 mkdir

API Name		**mkdir**
Function Description		Directories are created with the mkdir function.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int mkdir (LPSTR *pathname*);
	ASM	
Parameters	C	*pathname* pointer to a string specify the directory name will be created.
	ASM	None
Return Values		A return value of 0 indicates successful completion, and -1 indicates failure. **Error code list**

For Generalplus Confidential 立普企業股份有限公司 Use Only

EACCES	Write permission is denied for the parent directory in which the new directory is to be added.
EEXIST	A file named filename already exists.
ENOSPC	The file system doesn't have enough room to create the new directory.
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
ENAMETOOLONG	Filename specified too long.
ENAMEINVALID	Invalid character detected in the filename string.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".

Remarks

Example

```
void create_temp_directory () {
        if (mkdir ("a:\\temp" ) == -1)    {
                printf ("can not create temporary directory\n");
        }
        else  {
                printf ("temporary directory created\n");
        }
}
```

6.7 rmdir

API Name		**rmdir**
Function Description		The rmdir function deletes a directory.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int rmdir (LPSTR *pathname*);
	ASM	
Parameters	C	*pathname* pointer to a string specify the directory name will be deleted.
	ASM	None

For Generalplus Confidential 立奕企業股份有限公司 Use Only

Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** ENOTEMPTY The directory to be deleted is not empty. EACCES Write permission is denied for the directory from which the file is to be removed. ENOENT This error is reported when a file referenced as a directory component in the file name doesn't exist. ENFILE The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions". EIO For many devices, and for disk files, this error code indicates a hardware error.
Remarks		
Example		/* delect a temporary directory */ if (rmdir ("a:\\temp") == -1) printf ("rmdir failed\n");

6.8 chdir

API Name		**chdir**
Function Description		This function is used to set the process's working directory to *filename*.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int chdir (LPSTR *pathname*);
	ASM	
Parameters	C	*pathname* pointer to a string specify the directory name will be deleted.
	ASM	None
Return Values		The normal, successful return value from chdir is 0. A value of -1 is returned to indicate an error. **Error code list** ENOENT This error is reported when a file referenced as a directory component in the file name doesn't exist, ENOTDIR A file that is referenced as a directory component in the file name exists, but it isn't a directory. EIO For many devices, and for disk files, this error code indicates a hardware error. ENFILE The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".

For Generalplus Confidential 立奇企業股份有限公司 Use Only

Remarks		

Example		// change current directory
		if (chdir ("a:\\temp") == -1) { // return value -1 means error occurred
		printf ("change current directory failed\n");}
		else {
		printf ("change current directory successful\n");
		}

6.9 getcwd

API Name		**getcwd**
Function Description		Get current directory if success return buffer address else return NULL.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	LPSTR getcwd (LPSTR *buffer*, int *size*);
	ASM	
Parameters	C	*buffer* Pointer to directory string buffer. *size* Maximum length of the directory string can be stored.
	ASM	None
Return Values		The return value is buffer on success and a null pointer on failure. **Error code list** EINVAL The *size* argument is zero and *buffer* is not a null pointer. ERANGE The *size* argument is less than the length of the working directory name. You need to prepare a bigger array and try again.
Remarks		

Example		int size = 100;
		char buffer[100];
		if (getcwd (buffer, size) == buffer) {
		printf ("\nCurrent path: %s", buffer);
		}

6.10 unlink

API Name		**unlink**
Function Description		Delete the specified file.
Header File	**C**	vfs.h
	ASM	

For Generalplus Confidential 立普公司股份有限公司 Use Only

Syntax	**C**	int unlink (LPSTR *pathname*);
	ASM	
Parameters	C	*pathname* The file will be unlinked
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** EACCES Write permission is denied for the directory from which the file is to be removed or the file is busy. ENOENT The *filename* to be deleted doesn't exist. EISDIR Unlink cannot be used to delete the name of a directory. To avoid such problems, use rmdir to delete directories. ENFILE The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".
Remarks		

Example		if (unlink ("a:\\tempfile.txt") == -1) {
		switch (_getfserrcode ()) {
		case EACCES:
		…
		}
		}
		else {
		printf ("file has been deleted\n");
		}

6.11 rename

API Name		**rename**
Function Description		The function can be used to move or rename a file or a directory.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int rename (LPSTR *oldname*, LPSTR *newname*);
	ASM	
Parameters	C	*oldname* The old name of the file. *newname* The new name of the file.
	ASM	
Return Values		-1 returned when error occurred. **Error code list** EACCES One of the directories containing *newname* or *oldname* refuses

For Generalplus Confidential 凌通科技股份有限公司 Use Only

write permission; or *newname* and *oldname* are directories and write permission is refused for one of them.

EEXIST	The file or directory *newname* is already existed.
ENOENT	The file *oldname* doesn't exist.
ENOSPC	The directory that would contain *newname* has no room for another entry, and there is no space left in the file system to expand it.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".

Remarks

Example

```
// change the filename of "file1" into "file2"
int res;
res = rename ("a:\\file1", "a:\\file2");
if (res == -1)
        printf ("rename failed\n");
```

6.12 stat

API Name		**stat**
Function		The stat function fills the specified structure with the information about the specified file.
Description		
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int stat (LPSTR *filename*, struct stat **buf*);
	ASM	

Parameters	C	*Filename* Pointer to a pathname string.
		Buf Pointer to a stat structure.
		Data Structure
		struct stat {
		unsigned short st_mode;// access attribute of the file, see **file mode bitwise mask**
		long st_size; // the size of the normal file in byte
		unsigned long st_mtime; // the last modification time
		};
		File Mode Bitwise Mask
		S_READ_ONLY Read only attribute.
		S_HIDDEN Hide attribute. A hidden file has this attribute.
		S_SYSTEM System file attribute.
		S_DIRECTORY Directories have this attribute.
		S_ARCHIVE Normal file attribute.

	ASM	
Return Values		The return value is 0 if the operation is successful, or -1 on failure.

For Generalplus Confidential 立奇企業股份有限公司 Use Only

Error code list

EINVAL	Parameter list error: neither the *filename* nor the *buf* can be NULL.
ENOENT	The file named by filename doesn't exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".

Remarks

Example

```
/* Compare two files' last modification times */
struct stat statbuf;
time_t time1;
int res;
res = stat ("file1.txt", &statbuf);
if (res)
        return -1;
time1 = statbuf.st_mtime;
res = stat ("file2.txt", &statbuf);
if (res)
        return -1;
if (time1 > statbuf.st_mtime)
        printf ("file1.txt is more recent");
else
        printf ("file2.txt is more recent");
```

6.13 fs_init

API Name		fs_init
Function Description		Initialize file system.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	void fs_init(void);
	ASM	
Parameters	C	None
	ASM	None
Return Values		None
Remarks		All the global variables of file system will be forced into the initial values. Typically this function should be used only once at the initialization part of your program.

Example

```
fs_init();
_devicemount (0);
fp = open(".\\Hello.txt" , O_CREAT | O_RDWR);
write (fp, 0, 14);
close(fp);
_deviceunmount (0);
```

For Generalplus Confidential 立奕企業股份有限公司 Use Only

6.14 fs_uninit

API Name		fs_uninit
Function Description		uninitialize file system.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	void fs_uninit(void);
	ASM	
Parameters	C	None
	ASM	None
Return Values		None
Remarks		Uninitialize the file system to release all resources used by file system.

Example		fs_init();
		_devicemount (0);
		fp = open(".\\Hello.txt" , O_CREAT | O_RDWR);
		write (fp, 0, 14);
		close(fp);
		fs_uninit();

6.15 _getfserrcode

API Name		_getfserrcode
Function Description		Get the last error code of the file system.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _ getfserrcode (void);
	ASM	
Parameters	C	None
	ASM	None
Return Values		Return the last error code value. It can be one of the following values. ENOENT No such file or directory EINVACC Invalid access mode EBADF Bad file number EINVFNC Invalid function number ENOMEM Not enough core ERANGE Not enough core EACCES Permission denied EEXIST File exists EISDIR Target specified not a file but a directory EINVAL Invalid argument

For Generalplus Confidential 立奇企業股份有限公司 Use Only

EMFILE	Too many open files
ENOSPC	No space left on device
ENOTEMPTY	Directory is not empty
EIO	I/O operation error
ENOTDIR	Not directory
ENFILE	File not found
EROFS	Incorrect access mode
EPERM	Target is a directory
EBUSY	Target device is busy
ENAMETOOLONG	Specified path name or file name too long.
ENAMEINVALID	Invalid character detected in the filename string.

Remarks

Example

6.16 _clsfserrcode

API Name		_clsfserrcode
Function Description		Set the global error code value to zero.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	void _clsfserrcode (void);
	ASM	
Parameters	C	None
	ASM	None
Return Values		None
Remarks		
Example		

6.17 _findfirst

API Name		_findfirst
Function Description		Find the first appointed name and attribute's file.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _findfirst (LPSTR *pathname*, struct f_info *f_info, unsigned int *attrib*);
	ASM	

For Generalplus Confidential 凌通企業股份有限公司 Use Only

Parameters

C

pathname

Pointer to a pathname string.

ffblk

Pointer to a f_info structure.

```
struct f_info
{
    char   f_name[256];       /* file name */
    unsigned char f_attrib;   /* file attribute */
    unsigned int f_time;      /* file time */
    unsigned int f_date;      /* file date */
    unsigned long f_fsize;    /* file size */
};
```

attrib

D_RDONLY	Read-only file attribute
D_HIDDEN	Hidden file attribute
D_SYSTEM	System file attribute
D_DIR	Directory attribute
D_ARCHIVE	Archive file attribute

ASM

None

Return Values

This function returns 0 on successful completion, and -1 on error.

Error code list

ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist, or when a component is a symbolic link whose target file does not exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Remarks

Example

```
void list_file (char * pattern)    {
    struct f_info finfo;
    int idx = 0;

    printf ("\nList \"%s\":", pattern);
    if (_findfirst (pattern, &finfo, D_ALL)){
        printf ("\nNo such file");
        return;
    }
    do    {
        idx ++;
        printf ("\n%d\t%s", idx, finfo.f_name);
```

For Generalplus Confidential 立奕企業股份有限公司 Use Only

```
        }
        while (_findnext (&finfo) == 0);
    }
```

6.18 _findnext

API Name		_findnext
Function Description		Find next appointed name and attribute's file.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _findnext (struct f_info *f_info);
	ASM	
Parameters	C	*ffblk* File information struct.
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** ENOENT No more file fit the specified file name pattern and the file attribute condition. EIO For many devices, and for disk files, this error code indicates a hardware error.
Remarks		
Example		See _findfirst () example.

6.19 _copy

API Name		_copy
Function Description		The function can be used to make a copy for a file.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _copy (LPSTR *srcfile*, LPSTR *destfile*);
	ASM	
Parameters	C	*srcfile* It is a source path of file and file name. *destfile* It is a destination path of file and file name.
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error.

For Generalplus Confidential 立唐企業股份有限公司 Use Only

Error code list

ENOENT This error is reported when a file referenced as a directory component in the file name doesn't exist. Or the file named by filename doesn't exist.

EEXIST A file named filename already exists.

Remarks

EMLINK The parent directory has too many links (entries).

ENOSPC The file system doesn't have enough room to create the new directory.

ENFILE The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".

Example

```
int backup_a_c (void)    {
        int res;
        res = _copy ("a:\\a.c", "a:\\backup\\a.c");
        if (!res)       {
                return 0;
        }
        else  {
                printf ("file backup error");
                return –1;
        }
}
```

6.20 _format

API Name		_format
Function Description		Create a file system with the specified driver.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _format (unsigned char *drv*, unsigned char *fstype*);
	ASM	
Parameters	C	*drv* Zero-based driver set index. For example, index for SD card is 0. *fstype* Specify the file system. Should be one of these values. FAT16_Type Format the disk with FAT16 storage format. FAT32_Type Format the disk with FAT32 storage format.
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** EIO For many devices, and for disk files, this error code indicates a

For Generalplus Confidential 立奕企業股份有限公司 Use Only

		hardware error.
	EINVAL	A argument list error detected. May be the *drv* value is larger than NBLKDEV or the *fstype* is greater than FAT32_Type.
	EBUSY	The target device is a mounted device. Can not format a mounted device.
Remarks		

Example		fs_init ();
		if (_format (0, FAT16_Type)) /* make a FAT16 file system on the SD card */
		  while(1);
		/* mount for other operations after _format() */
		_devicemount (0);
		…

6.21 _deleteall

API Name		_deleteall
Function Description		Delete all files and folders in the specified directory.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _deleteall (LPSTR *pathname*);
	ASM	
Parameters	C	*pathname* To appoint path.
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** ENOENT This error is reported when a file referenced as a directory component in the file name doesn't exist. Or the file named by filename doesn't exist. EACCES Write permission is denied for the directory from which the file is to be removed. The deleteall process will terminated after such an accident. EIO Low level I/O error. ENFILE The entire file system cannot allocate any file node structure variable for search at the moment. See "Limits and Suggestions".
Remarks		

Example		/* _deleteall can be used to empty a directory */
		/* Example, empty a temporary directory */
		int res;
		res = _deleteall ("a:\\temp");
		if (res) {
		  …
		}

For Generalplus Confidential 立奇企業股份有限公司 Use Only

6.22 _devicemount

API Name		_devicemount
Function Description		Mount a disk, load the information about the device and the file system information on the device.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _devicemount (unsigned char *diskid*);
	ASM	
Parameters	C	*diskid* To appoint mounted disked. **Access mode** DEVICE_WRITE_ALLOW write permission bit will be masked DEVICE_READ_ALLOW read permission bit will be masked *Note:accessmod* can be a bitwise OR result of DEVICE_WRITE_ALLOW and DEVICE_READ_ALLOW.
	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** EINVAL Parameter list value error. EBUSY The device specified by *diskid* is busy. EIO Low level I/O error.
Remarks		

Example		fs_init ();
		if (_devicemount (0))
		while(1);…

6.23 _deviceunmount

API Name		deviceunmount
Function Description		Umount the specified device, flush all cached data associates to the device.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _deviceunmount (unsigned char *diskID*);
	ASM	
Parameters	C	*diskID* To appoint mounted disked.

For 立奕企業股份有限公司 Use Only Generalplus Confidential

	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** EINVAL Parameter list value error. EBUSY The device specified by *diskid* is busy.
Remarks		
Example		fs_init (); _devicemount (0, 0, DEVICE_READ_ALLOW | DEVICE_WRITE_ALLOW); _deviceunmount(0); …

6.24 _getdiskfree

API Name		_ getdiskfree
Function Description		Get information about the space of the specified device.
Header File	**C**	vfs.h
	ASM	
Syntax	**C**	int _getdiskfree (short *driver*, struct _diskfree_t * *dfreep*);
	ASM	

Parameters	C	*driver*
		Specify the device zero based index.
		dfreep
		The struct of device information
		struct _diskfree_t
		{
		unsigned long total_clusters;
		unsigned long avail_clusters;
		unsigned long sectors_per_cluster;
		unsigned long bytes_per_sector;
		};

	ASM	None
Return Values		This function returns 0 on successful completion, and -1 on error. **Error code list** EINVAL Parameter list value error.
Remarks		

Example		struct _diskfree_t space_info
		_ getdiskfree (0, &space_info); // get space informations about device a
		printf ("Free clusters: %d", space_info.avail_clusters);
		…

For 立奕企業股份有限公司 Use Only Generalplus Confidential

7 Program Example

```
#include     "vfs.h"

int main()
{
        int i;
        int ret,fd;
        long len;
        unsigned int buffer[1024];

        System_Initial();

        fs_init();

        for(i = 0; i < 3; i++)
        {
                ret = _devicemount(i);
                //can not mount this disk
        }
        ChangeCodePage(UNI_GBK);

        fd = open((LPSTR)"a:\\test.bin", O_RDWR|O_CREAT);
        if(fd < 0)
        {
                ret = _getfserrcode();
                //error code process
        }

        for(i = 0; i < 10; i++)
        {
                len = write(fd, (UINT32)buffer << 1, 1024*2);
                if(len == -1)
                {
                        ret = _getfserrcode();
```

For Generalplus Confidential 凌通公司 Use Only

```
            //error code process
        }
    }

    close(fd);

    fd = open((LPSTR)"a:\\test.bin", O_RDONLY);
    if(fd < 0)
    {
        ret = _getfserrcode();
        //error code process
    }

    lseek(fd, 512, SEEK_SET);
    for(i = 0; i < 5; i++)
    {
        len = read(fd, (UINT32)buffer << 1, 1024*2);
        if(len == -1)
        {
            ret = _getfserrcode();
            //error code process
        }
    }

    close(fd);
```

For Generalplus Confidential 立奕企業股份有限公司 Use Only

8 Special Note

We need some functions for getting system time. We list the spec here:

8.1 UserGetDate

Function Interface

```
void UserGetdate (struct dosdate * dd);
```

Structure Description

```
struct dosdate    {
     unsigned short year;
     unsigned char monthday, month;
};
```

Function Definition Demo

```
void UserGetDate (struct dosdate *dd)   {
     // function body should be adapted by user
     dd->year = 2004;
     dd->month = 8;
     dd->monthday = 23;
}
```

8.2 UserGetTime

Function Interface

```
void UserGetTime (struct dostime * dt);
```

Structure Description

```
struct dostime    {
  unsigned char minute, hour, hundredth, second;
};
```

Function Definition Demo

```
void UserGetTime (struct dostime *dt)   {
     // function body should be adapted by user
     dt->hour = 16;
     dt->minute = 54;
     dt->second = 37;
     dt->hundredth = 0;
}
```

For Generalplus Confidential 立奕企業股份有限公司 Use Only

Some limits exist in this embedded file system. We list them as following.

8.3 Maximum Open File Number

There are only three file node structure variables in this file system. So at the same time only three files can be opened at most.

8.4 Implied Open Operations

Some functions impliedly require file node structure variables to work properly. With out enough file node structure they will report error. These requirements for file node structure variables are listed in the following table("-" means file node is visibly required):

Function name	Require file node structure variables number
open	1
close	-
read	-
write	-
lseek	-
rmdir	1
mkdir	1
chdir	1
getcwd	0
unlink	-
rename	2
utime	1
stat	1
fs_init	0
_getfserrcode	0
_clserrcode	0
_findfirst	1
_findnext	1
_copy	2
_move	2
_format	0
_setfattr	1
_deleteall	1
_devicemount	0
_deviceunmount	0
_deviceinfoget	0
_getdiskfree	0

For Generalplus Confidential 立奇企業股份有限公司 Use Only

8.5 Copy Operation Performance

We can not make sure which part of the SRAM is available to work as a data buffer when user is calling *_copy()* to duplicate a file. So we declared a 64 byte length unsigned char array as buffer in *_copy()* function (The array size is limited by the size of the memory size on board). It means than we can not make a high performance when user is trying to duplicate a file by *_copy()*. We strongly suggest the user to write a new copy function instead of the *_copy()* in your application.

8.6 Maximum Pathname String Length

The maximum pathname string length is 255 bytes, includes the drive letter and the ':' character and the backslashes but not includes the terminal zero character.

For 立奕企業股份有限公司 Use Only Generalplus Confidential