



**Future Technology Devices  
International Ltd.  
Application Note**

**AN\_189**

**Vinculum-II Using the LFAT Driver**

**Document Reference No.: FT\_000533**

**Version 1.0**

**Issue Date: 2011-10-31**

This application note provides an example of how to use the FTDI Vinculum-II (VNC2) LFAT driver. Sample source code is included.

**Future Technology Devices International Limited (FTDI)**

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

**E-Mail (Support): [support1@ftdichip.com](mailto:support1@ftdichip.com) Web: <http://www.ftdichip.com>**

Copyright © 2011 Future Technology Devices International Limited

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use.

---

## **Table of Contents**

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
<b>2</b>	<b>LFAT File System Concepts.....</b>	<b>3</b>
2.1	LFAT Driver hierarchy .....	3
<b>3</b>	<b>LFAT File System API .....</b>	<b>4</b>
3.1	ifat_dirChangeDir.....	5
3.2	ifat_dirCreateDir .....	6
3.3	ifat_dirTableFind.....	7
3.4	ifat_dirTableFindFirst.....	8
3.5	ifat_dirTableFindNext .....	9
3.6	ifat_fileOpen .....	10
3.7	ifat_fileRename.....	12
3.8	ifat_dirEntryName .....	13
3.9	ifat_dirDeleteEntry .....	14
<b>4</b>	<b>Using the LFAT Driver .....</b>	<b>15</b>
4.1	Obtaining a License.....	15
4.2	Requesting the Library.....	15
4.3	Installing the Library .....	15
<b>5</b>	<b>Contact Information.....</b>	<b>16</b>
<b>6</b>	<b>Appendix A – References.....</b>	<b>18</b>
	Document References .....	18
	Acronyms and Abbreviations .....	18
<b>7</b>	<b>Appendix B – Revision History .....</b>	<b>19</b>

## **1 Introduction**

The LFAT driver is an extension of the VNC2 FAT File System driver that supports FAT Long Directory Entries (see [1]). This is commonly referred to as long filename (LFN) support.

This note describes specifically the extensions to the VNC2 FAT driver that enable support for LFNs. For full details of the FAT driver, see the VNC2 Help System.

The sample source code contained in this application note is provided as an example and is neither guaranteed nor supported by FTDI.

---

## 2 LFAT File System Concepts

In an application that supports LFNs, the LFAT driver replaces the FAT driver. To handle LFNs, the LFAT driver extends the FAT API.

### 2.1 LFAT Driver hierarchy

The LFAT driver hierarchy is as follows:

LFAT Driver		
LFAT/FAT API		
BOMS Class Driver	SD Card Driver	Other MSI Driver
USB Host Driver	SPI Master Driver	
VOS Kernel		
USB Host Hardware	SPI Master Hardware	Other Hardware

The LFAT driver is implemented as an extension of the FAT driver. The basic functionality of the FAT driver remains present in the LFAT driver, and this is documented elsewhere in the VNC2 help system. The extensions necessary to implement LFN support are the subject of this application note, and are documented in the following sections.

---

### 3 LFAT File System API

The LFAT File System API provides direct access to the file system commands. LFAT-specific functions are defined in the following sections.

#### File Functions

lfat\_fileOpen()            Open a file and return a handle to the file  
lfat\_fileRename()        Rename a file

#### Directory Table Functions

lfat\_dirChangeDir()      Changes the current directory  
lfat\_dirCreateDir()      Make a new directory  
lfat\_dirTableFind()      Find a file or directory with a specified name in the current directory  
lfat\_dirTableFindFirst() Find the first file or directory in the current directory  
lfat\_dirTableFindNext() Find subsequent files or directories in the current directory  
lfat\_dirDeleteEntry()    Remove long directory entries from FAT table  
lfat\_dirEntryName()      Copy the name of an open file

#### Note

In the following definitions, long filenames are declared as *unsigned char \**. Long filenames are actually stored in UNICODE, and the conversion from *unsigned char \** to UNICODE is performed by the LFAT driver. This limitation may be removed in future versions of the LFAT driver.

---

## 3.1 lfat\_dirChangeDir

### Syntax

```
unsigned char lfat_dirChangeDir(fat_context *fat_ctx, unsigned char *dirname)
```

### Description

Changes the current directory to a subdirectory specified in the *dirname* parameter.

A special case value of NULL is used to change to the top level directory.

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*dirname*

The destination directory name. The value of NULL will change the current directory to the volume's root directory

### Returns

There is no data returned by this call. The return value will be one of the following:

FAT\_OK successfully changed the current directory

FAT\_NOT\_FOUND directory not changed as destination directory not found

### Example

```
char changetoroot(fat_context *fatctx)
{
    if (lfat_dirChangeDir(fatctx, NULL) == FAT_OK)
    {
        return 0;
    }

    return 1;
}

char changetosubdir(fat_context *fatctx)
{
    char file[12] = "MYBACKUPDIR";

    if (lfat_dirChangeDir(fatctx, file) == FAT_OK)
    {
        return 0;
    }

    return 1;
}
```

---

## 3.2 Ifat\_dirCreateDir

### Syntax

```
unsigned char Ifat_dirCreateDir(fat_context *fat_ctx, unsigned char *name)
```

### Description

Make a new subdirectory in the Current directory. The name of the subdirectory is specified in the *name* parameter.

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*name*

The new directory name. This must not exist in the current directory.

### Returns

There is no data returned by this call. The return value will be one of the following:

FAT\_OK successfully created the new directory

FAT\_EXISTS directory not created as a directory or file with that name already exists

FAT\_DISK\_FULL there were no free clusters found for creating a new directory table

### Example

```
char makesub(fat_context *fatctx)
{
    char dirname[22] = "A_long_directory_name";

    if (Ifat_dirCreateDir(fatctx, dirname) == FAT_OK)
    {
        return 0;
    }

    return 1;
}
```

---

### 3.3 lfat\_dirTableFind

#### Syntax

```
unsigned char lfat_dirTableFind(fat_context *fat_ctx, file_context_t *file_ctx, char *name)
```

#### Description

Searches in the current directory for a file or directory matching the name specified in the parameters of the call. The filename is specified in the name parameter.

#### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to a FAT File Handle structure

*name*

Contains a pointer to a file name.

#### Returns

The return value will be one of the following:

FAT\_OK successfully received current file pointer

FAT\_NOT\_FOUND a matching file was not found

FAT\_EOF no matching file was found but directory table is full

A FAT File Handle is returned in the *file\_ctx* parameter if a matching file is found. This can be used for subsequent access to the file or directory. The file handle is opened with a file mode of FILE\_MODE\_HANDLE.

#### Example

```
char checkforfile(fat_context *fatctx)
{
    char file[20] = "The_quick_brown.bat";
    file_context_t fd;

    if (lfat_dirTableFind(fatctx, &fd, file) == FAT_OK)
    {
        // file exists
        return 0;
    }

    return 1;
}
```

---

## 3.4 lfat\_dirTableFindFirst

### Syntax

```
unsigned char lfat_dirTableFindFirst(fat_context *fat_ctx, file_context_t file_ctx)
```

### Description

Searches in the current directory for all files and directories. This function initialises the search, and lfat\_dirTableFindNext() is used to continue searching through the files in the current directory.

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to a FAT File Handle structure

### Returns

The return value will be one of the following:

FAT\_OK successfully received current file pointer

FAT\_NOT\_FOUND a matching file was not found

FAT\_EOF no matching file was found but directory table is full

A FAT File Handle is returned in the file\_ctx parameter if any file is found. This can be used for subsequent access to the file or directory. The file handle is opened with a file mode of FILE\_MODE\_HANDLE. Subsequent calls to lfat\_dirTableFindNext() must reuse the same file handle.

### Example

```
char processXfiles(fat_context *fatctx)
{
    char file[11];
    file_context_t fd;
    char *file = (char*)&fd;

    if(lfat_dirTableFindFirst(fatctx, &fd) == FAT_OK)
    {
        // file exists
        do
        {
            if (file[0] == 'X')
            {
                // process files beginning with X
            }

        } while (lfat_dirTableFindNext(fatctx, &fd) == FAT_OK);
    }
}
```

## 3.5 lfat\_dirTableFindNext

### Syntax

```
unsigned char lfat_dirTableFindNext(fat_context *fat_ctx, file_context_t file_ctx)
```

### Description

Searches in the current directory for subsequent files and directories continuing a lfat\_dirTableFindFirst() search.

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to a FAT File Handle structure

### Returns

The return value will be one of the following:

FAT\_OK successfully received current file pointer

FAT\_NOT\_FOUND a matching file was not found

FAT\_EOF no matching file was found but directory table is full

The FAT File Handle in the file\_ctx parameter is updated. This can be used for subsequent access to the file or directory. The file handle is opened with a file mode of FILE\_MODE\_HANDLE. Subsequent calls to lfat\_dirTableFindNext() must reuse the same file handle.

### Example

See example in lfat\_dirTableFindFirst().

---

## 3.6 lfat\_fileOpen

### Syntax

```
unsigned char lfat_fileOpen(fat_context *fat_ctx, file_context_t file_ctx, unsigned char *name, unsigned char mode)
```

### Description

Opens a file or directory by name in the current directory. It can be used for opening files for read, write access or simply obtaining a handle to a file without allowing access to the contents.

Directories may only be opened by mode FILE\_MODE\_HANDLE. This can be used to rename directories (with lfat\_fileRename()) or change the attributes of a directory (with fat\_fileMod()).

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to memory allocated to store a file handle

*name*

Contains a pointer to a file name.

*mode*

The mode member is one of the File Mode Values defined in the FAT File Handle structure.

### Returns

The return value will be one of the following:

FAT\_OK successful file open

FAT\_NOT\_FOUND file system type invalid or file system not attached, open a file for reading which does not exist

FAT\_INVALID\_FILE\_TYPE attempt to open a volume ID directory entry or directory as a file

FAT\_READ\_ONLY opening a read only file with a write or append mode

FAT\_DISK\_FULL no free clusters found in which to store file data

FAT\_DIRECTORY\_TABLE\_FULL root directory on FAT12 and FAT16 disks has no free entries

FAT\_INVALID\_PARAMETER the set value of the fat\_ioctl\_cb\_t IOCTL structure is NULL

A FAT File Handle is returned in the *file\_ctx* parameter. This is used for subsequent access to the file. The memory is allocated in the calling procedure.

### Example

```
file_context_t *openfile(fat_context *fatctx)
{
    char file[20] = "The_quick_brown.fox";
    file_context_t *fd = malloc(sizeof(file_context_t));

    if (lfat_fileOpen(fatctx, fd, file, FILE_MODE_APPEND_PLUS) == FAT_OK) {
        return fd;
    }
    else {
        return NULL;
    }
}
```

```
void closefile(file_context_t *fd)
{
    fat_fileClose(fd);
    free(fd);
}
```

---

## 3.7 lfat\_fileRename

### Syntax

```
unsigned char lfat_fileRename(file_context_t file_ctx, char *name)
```

### Description

Renames a file or directory using a FAT File Handle obtained from `fat_fileOpen()`. The rename function does not rename a file or directory based on a name but rather a handle. The file or directory must be opened first and then renamed. The file or directory must be opened with a file mode of `FILE_MODE_HANDLE` to ensure no changes to the file are made before deletion.

The file handle must be closed afterwards with `fat_fileClose()`. This will also synchronise the directory table and remove the file or directory from there.

### Parameters

*file\_ctx*

Pointer to a valid FAT file handle.

*name*

New name of file or directory.

### Returns

There is no data returned by this call. The return value will be one of the following:

FAT\_OK successfully renamed the file

FAT\_INVALID\_FILE\_TYPE file not opened with mode `FILE_MODE_HANDLE`

### Example

```
char renamefile(file_context_t *fatctx)
{
    char filesrc[20] = "The_quick_brown.fox";
    char filedst[20] = "The_quick_brown.dog";
    file_context_t fd;
    char status = -1;

    if (lfat_fileOpen(fatctx, &fd, filesrc, FILE_MODE_HANDLE) == FAT_OK)
    {
        lfat_fileRename(&fd, filedst) == FAT_OK)
        {
            // rename successful
            status = 0;
        }
        fat_fileClose(&fd);
    }

    return status;
}
```

---

## 3.8 lfat\_dirEntryName

### Syntax

```
unsigned char lfat_dirEntryName(fat_context *fat_ctx, file_context_t *file_ctx, char *name)
```

### Description

Copies the name of an opened file into the buffer specified by *name* parameter.

The buffer must be big enough to store a long filename.

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to a valid FAT file handle.

*name*

Pointer to buffer to receive the filename.

### Returns

There is no data returned by this call. The return value will be one of the following:

FAT\_OK successfully copied name to buffer

FAT\_INVALID\_PARAMETER pointer to buffer was invalid

### Example

Find first matching file in the current directory and return the filename.

```
unsigned char getFirst(fat_context *fatSrc, char *name)
{
    file_context_t fileFind;
    unsigned char status;

    if ((status = lfat_dirTableFindFirst(fatSrc, &fileFind)) == FAT_OK)
    {
        status = lfat_dirEntryName(fatSrc, &fileFind, filename);
    }

    return status;
}
```

---

## 3.9 Ifat\_dirDeleteEntry

### Syntax

```
unsigned char Ifat_dirDeleteEntry(fat_context *fat_ctx, file_context_t *file_ctx, char *name)
```

### Description

Deletes the long directory entries for the file specified by *name* parameter. This function is used to clear the long directory entries after the short entry has been deleted (see example).

### Parameters

*fat\_ctx*

Pointer to the instance of the FAT API

*file\_ctx*

Pointer to a valid FAT file handle.

*name*

Pointer to buffer to receive the filename.

### Returns

There is no data returned by this call. The return value will be one of the following:

FAT\_OK successfully copied name to buffer

FAT\_NOT\_FOUND a matching file was not found

### Example

Delete a file.

```
unsigned char delete(fat_context *fatSrc, file_context_t *file_ctx, char *name)
{
    Ifat_fileOpen(fatSrc, file_ctx, name, FILE_MODE_HANDLE)
    fat_fileDelete(file_ctx);
    fat_fileClose(file_ctx);
    Ifat_dirDeleteEntry(fatSrc, file_ctx, name);
}
```

---

## 4 Using the LFAT Driver

To use the LFAT driver, the customer must first obtain a license from Microsoft, then request the LFAT driver library from FTDI, and finally incorporate it into his Vinculum-II development toolchain.

### 4.1 Obtaining a License

Long filename technology is the intellectual property of Microsoft Corporation. Microsoft controls rights to its technologies through its IP licensing group [2]; the technology licensing program includes rights with regard to the implementation of the FAT File System [3].

To use long filenames, the customer must enter into a licensing agreement with Microsoft before FTDI can distribute the LFAT driver library. Consequently, the LFAT driver is not included in the Vinculum-II Toolchain distribution but can be obtained on request from FTDI Support. Prior to distribution, FTDI will obtain an agreement from the customer, stating customer's responsibility to obtain the LFN license from Microsoft. With that agreement signed off, FTDI is relieved of any liability for releasing the LFN driver library to the customer. It must be stressed that it is the customer's responsibility to obtain the license from Microsoft, failing which they could face further consequences from Microsoft.

For information on how to obtain the LFN license from Microsoft, see [3].

### 4.2 Requesting the Library

After finalizing a license agreement with Microsoft, the customer must contact FTDI Support to request the LFAT driver library archive.

### 4.3 Installing the Library

The LFAT driver library distribution consists of the following files:

- LFAT.a LFAT driver library archive
- LFAT.h LFAT header file

LFN support for the Vinculum-II development toolchain can be obtained by copying these files to an appropriate directory. For example:

- Copy the LFAT.a to the FTDI/Vinculum II Toolchain/Firmware/Drivers/lib folder
- Copy the LFAT.h to the FTDI/Vinculum II Toolchain/Firmware/Drivers/include folder

This could also be the current directory of a project. Then, in the Vinculum-II IDE, ProjectManager can be used to add these files to a project.

---

## 5 Contact Information

### Head Office – Glasgow, UK

Future Technology Devices International Limited  
Unit 1, 2 Seaward Place, Centurion Business Park  
Glasgow G41 1HH  
United Kingdom  
Tel: +44 (0) 141 429 2777  
Fax: +44 (0) 141 429 2758

E-mail (Sales) [sales1@ftdichip.com](mailto:sales1@ftdichip.com)  
E-mail (Support) [support1@ftdichip.com](mailto:support1@ftdichip.com)  
E-mail (General Enquiries) [admin1@ftdichip.com](mailto:admin1@ftdichip.com)  
Web Site URL <http://www.ftdichip.com>  
Web Shop URL <http://www.ftdichip.com>

### Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)  
2F, No. 516, Sec. 1, NeiHu Road  
Taipei 114  
Taiwan, R.O.C.  
Tel: +886 (0) 2 8791 3570  
Fax: +886 (0) 2 8791 3576

E-mail (Sales) [tw.sales1@ftdichip.com](mailto:tw.sales1@ftdichip.com)  
E-mail (Support) [tw.support1@ftdichip.com](mailto:tw.support1@ftdichip.com)  
E-mail (General Enquiries) [tw.admin1@ftdichip.com](mailto:tw.admin1@ftdichip.com)  
Web Site URL <http://www.ftdichip.com>

### Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)  
7235 NW Evergreen Parkway, Suite 600  
Hillsboro, OR 97123-5803  
USA  
Tel: +1 (503) 547 0988  
Fax: +1 (503) 547 0987

E-Mail (Sales) [us.sales@ftdichip.com](mailto:us.sales@ftdichip.com)  
E-Mail (Support) [us.support@ftdichip.com](mailto:us.support@ftdichip.com)  
E-Mail (General Enquiries) [us.admin@ftdichip.com](mailto:us.admin@ftdichip.com)  
Web Site URL <http://www.ftdichip.com>

### Branch Office – Shanghai, China

Future Technology Devices International Limited (China)  
Room 408, 317 Xianxia Road,  
Shanghai, 200051  
China  
Tel: +86 21 62351596  
Fax: +86 21 62351595

E-mail (Sales) [cn.sales@ftdichip.com](mailto:cn.sales@ftdichip.com)  
E-mail (Support) [cn.support@ftdichip.com](mailto:cn.support@ftdichip.com)  
E-mail (General Enquiries) [cn.admin@ftdichip.com](mailto:cn.admin@ftdichip.com)  
Web Site URL <http://www.ftdichip.com>

### **Distributor and Sales Representatives**

Please visit the Sales Network page of the [FTDI Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640

---

## 6 Appendix A – References

### Document References

[1] Microsoft Extensible Firmware Initiative FAT32 File System Specification, Microsoft Corporation, 2000.

[2] Microsoft IP Licensing Home is at:

<http://www.microsoft.com/about/legal/en/us/IntellectualProperty/IPLicensing/Default.aspx>

[3] Microsoft Technology Licensing Program, FAT File System is at:

<http://www.microsoft.com/about/legal/en/us/IntellectualProperty/IPLicensing/Programs/FATFileSystem.aspx>

### Acronyms and Abbreviations

Terms	Description
VNC2	Vinculum II
VOS	Vinculum Operating System

---

## 7 Appendix B – Revision History

Revision	Changes	Date
Draft	Initial Draft Release for comment	2011-09-29
1.0	Initial Release	2011-10-31