Application Note

AN_189

Vinculum-II Using the LFAT Driver

Version 1.1

Issue Date: 2016-04-05

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
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

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

This application 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 Toolchain Help System (access is shown in Figure 1.1 or refer to AN_151 Vinculum II User Guide.

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

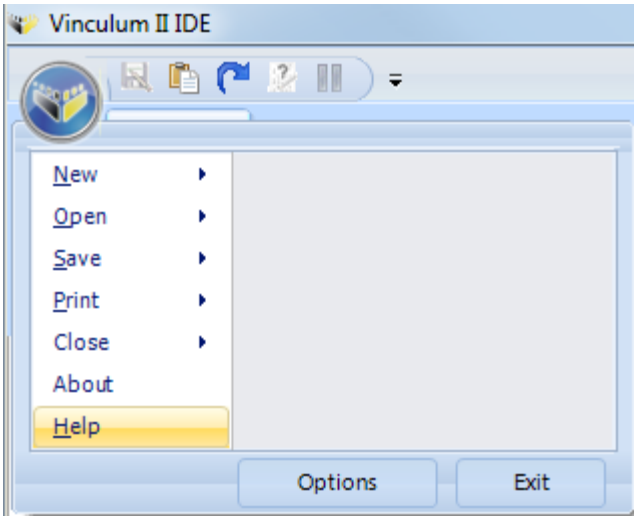

Figure 1.1 VNC2 Toolchain Help System Access

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:

<table>
<tr><td colspan="3">LFAT Driver</td></tr>
<tr><td colspan="3">LFAT/FAT API</td></tr>
<tr><td>BOMS Class Driver</td><td>SD Card Driver</td><td rowspan="2">Other MSI Driver</td></tr>
<tr><td>USB Host Driver</td><td>SPI Master Driver</td></tr>
<tr><td colspan="3">VOS Kernel</td></tr>
<tr><td>USB Host Hardware</td><td>SPI Master Hardware</td><td>Other Hardware</td></tr>
</table>

Table 2.1 LFAT Driver Hierarchy

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 Toolchain help system and in AN_151 Vinculum II User Guide. 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.

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 lfat_dirCreateDir

Syntax

unsigned char lfat_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 exist
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 (lfat_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 initializes 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(fat_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 an 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 synchronize 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(fat_context *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

Copy's 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 lfat_dirDeleteEntry

Syntax

unsigned char lfat_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)
{
   lfat_fileOpen(fatSrc, file_ctx, name, FILE_MODE_HANDLE)

   fat_fileDelete(file_ctx);
   fat_fileClose(file_ctx);
   lfat_dirDeleteEntry(fatSrc, file_ctx, name);
}
```

4 Using the LFAT Driver

To use the LFAT driver, the customer must request the LFAT driver library from FTDI, and finally incorporate it into the Vinculum-II development toolchain.

4.1 Requesting the Library

The customer must contact FTDI Support to request the LFAT driver library archive.

4.2 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 following folder:
  C:\Program Files (x86)\FTDI\Vinculum II Toolchain\Firmware\Drivers\lib
- Copy the LFAT.h to the following folder:
  C:\Program Files (x86)\FTDI\Vinculum II Toolchain\Firmware\Drivers\inc

This could also be the current directory of a project. Then, in the Vinculum-II IDE, Project Manager 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
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited
(USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@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 8797 1330
Fax: +886 (0) 2 8751 9737

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited
(China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

http://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.

Appendix A – References

Document References

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

AN_151 Vinculum II User Guide

Acronyms and Abbreviations

Terms	Description
VNC2	Vinculum II
VOS	Vinculum Operating System
LFN	Long File Name
LFAT	Long File Allocation Table
API	Application Programming Interface

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_189 Vinculum-II Using the LFAT Driver
Document Reference No.: FT_000533
Clearance No.: FTDI# 231
Product Page: http://www.ftdichip.com/FTProducts.htm
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2011-10-31
1.1	Minor Release which removed the need for Microsoft agreement license.	2016-04-05