LibXil FATFile System(FATfs)

EDK 9.1i December 12, 2006

Summary

This document describes the XilFatfs FATFile System access library. This library provides read/write access to files stored on a Xilinx® System ACE™ compact flash or microdrive device.

The document contains the following sections.

- "Overview"
- "XilFatfs Functions"
- "Libgen Customization"

Overview

The XilFatfs filesystem access library provides read/write access to files stored on a Xilinx System ACE CompactFlash or IBM microdrive device. This library requires the underlying hardware platform to contain all of the following:

- OPB SYSACE Interface Controller - Logicore module.
- System ACE controller and CompactFlash connector.
- CompactFlash card or IBM Microdrive formatted with a FAT16, FAT12 or FAT32 file system.

  Caution! FAT16 is required for System ACE to directly configure the FPGA but the XilFatfs library can work with the System ACE hardware to support FAT12 and FAT32 as well.

See the documentation on On-Chip Peripheral Bus (OPB) SYSACE Interface Controller in the *Processor IP Reference Guide* for more details on the hardware.

Users can easily copy files to the flash device from their PC by plugging the flash or microdrive into a suitable USB adapter or similar device.

If the compact flash or microdrive has multiple partitions, each formatted as a FAT12, FAT16 or FAT32 filesystem, XilFatfs allows the partitions to be accessed with partition names. The first partition is always called A:, the second partition is always called B:, and so on. As noted earlier, the first partition must be FAT16 for System ACE to directly configure the FPGA.

XilFatfs Functions

This section presents a list of functions provided by the XilFatfs. The table below provides the function names with signatures. A detailed description of each function follows.

Table 1: **XilFatfs Functions**

Functions
`void *sysace_fopen (const char *file, const char *mode)`
`int sysace_fread (void *buffer, int size, int count, void *file)`
`int sysace_fwrite (void *buffer, int size, int count, void *file)`
`int sysace_fclose (void *file)`
`int sysace_mkdir (const char *path)`
`int sysace_chdir (const char *path)`

void ***sysace_fopen** (const char **file*, const char **mode*)

Parameters

file is the name of the file on the flash device.

mode is “r” or “w”

Returns

A non zero file handle on success.

0 for failure.

Description

The file name should follow the Microsoft 8.3 naming standard with a file name of up to 8 characters, followed by a ‘.’ and a 3 character extension. In this version of the library, the 3 character extension is mandatory so a sample file might be called test.txt.

This function returns a file handle that has to be used for subsequent calls to read/write or close the file.

If mode is “r” and the named file does not exist on the device 0 is returned.

Includes

sysace_stdio.h

int **sysace_fread** (void *buffer, int size, int count, void *file)

Parameters

buffer is a pre allocated buffer that is passed in to this procedure, and is used to return the characters read from the device.

size is restricted to 1.

count is the number of characters to be read.

file is the file handle returned by sysace_fopen.

Returns

Non zero number of characters actually read for success.

0 for failure.

Description

The preallocated buffer is filled with the characters that are read from the device. The return value indicates the actual number of characters read, while *count* specifies the maximum number of characters to read. The buffer size must be at least *count*. *stream* should be a valid file handle returned by a call to sysace_fopen.

Includes

sysace_stdio.h

int **sysace_fwrite** (void *buffer, int size, int count, void *file)

Parameters

buffer is a pre allocated buffer that is passed in to this procedure, and contains the characters to be written to the device.

size is restricted to 1.

count is the number of characters to be written.

file is the file handle returned by **sysace_fopen**.

Returns

Non zero number of characters actually written for success.

0 or -1 for failure.

Description

The pre-allocated buffer is filled (by the caller) with the characters that are to be written to the device. The return value indicates the actual number of characters written, while *count* specifies the maximum number of characters to write. The buffer size must be at least *count*. *stream* should be a valid file handle returned by a call to sysace_fopen. This function might simply return after updating the buffer cache (see CONFIG_BUFCACHE_SIZE). To ensure that the data has been written to the device, a sysace_fclose call has to be done.

Includes

sysace_stdio.h

int **sysace_fclose** (void *file)

Parameters *file:* File handle returned by *sysace_fopen*

Returns 0 On success

-1 On failure

Description Closes an open file. This function also synchronizes the buffer cache to memory. If any files were written to using sysace_fwrite, then it is necessary to synchronize the data to the disk by performing sysace_fclose. If this is not performed, then the disk could possibly become corrupted.

Includes sysace_stdio.h

int **sysace_mkdir** (const char *path)

Parameters *path:* path name of new directory

Returns 0 On success

-1 On failure

Description Create a new directory specified by path. The directory name can be either absolute or relative, and must follow the 8.3 file naming convention.

Examples: "a:\\dirname", "a:\\dirname.dir", "a:\\dir1\\dirnew", "dirname", "dirname.dir"

If a relative path is specified, and the current working directory is not already set, the current working directory defaults to the root directory.

Includes sysace_stdio.h

int **sysace_chdir** (const char *path)

Parameters *path:* path name of new directory

Returns 0 On success

-1 On failure

Description Create a new directory specified by *path*. The directory name can be either absolute or relative, and must follow the 8.3 file naming convention.

Examples: "a:\\dirname", "a:\\dirname.dir", "a:\\dir1\\dirnew", "dirname", "dirname.dir"

If a relative path is specified, and the current working directory is not already set, the current working directory defaults to the root directory.

Includes sysace_stdio.h

Libgen Customization

XilFatfs file system can be integrated with a system using the following snippet in the Microprocessor Software Specification (MSS) file:

```
BEGIN LIBRARY
  parameter LIBRARY_NAME = xilfatfs
  parameter LIBRARY_VER = 1.00.a
  parameter CONFIG_WRITE = true
  parameter CONFIG_DIR_SUPPORT = false
  parameter CONFIG_FAT12 = false
  parameter CONFIG_MAXFILES = 5
  parameter CONFIG_BUFCACHE_SIZE = 10240
  parameter PROC_INSTANCE = powerpc_0
END LIBRARY
```

Parameter description:

- When CONFIG_WRITE is set to true, write capabilities are added to the library.
- When CONFIG_DIR_SUPPORT is set to true, the mkdir and chdir functions are added to the library. For mkdir function to work, CONFIG_WRITE needs to be enabled.
- When CONFIG_FAT12 is set to true, the library is configured to work with FAT12 file systems. Otherwise, the library works with both FAT16 and FAT32 file systems.
- CONFIG_MAXFILES limits the maximum number of files that can be open. This influences the amount of memory allocated statically by XilFatfs.
- CONFIG_BUFCACHE_SIZE defines the amount of memory (in bytes) used by the library for buffering reads and write calls to the System ACE. This improves the performance of both sysace_fread and sysace_fwrite by buffering the data in memory and avoiding unnecessary calls to read the CF device. The buffers are synced up to the device only on a sysace_fclose call. Hence it is essential to perform a sysace_fclose if any file was modified.
- The parameter PROC_INST is not necessary for uniprocessor systems. In a multiprocessor system, set PROC_INST to the processor name for which the library must be compiled. The System ACE peripheral must be reachable from this processor.