EFSL

5 EFSL utilities 39
5.1 Notations . . . 39
5.2 cpo . . . 39
5.3 cpi . . . 39
5.4 cpa . . . 40
5.5 list . . . 40
5.6 mkdir . . . 40
5.7 rmfile . . . 41

6 Developer notes 42
6.1 Integer types . . . 42
6.2 Debugging . . . 42
6.2.1 Debugging on Linux . . . 42
6.2.2 Debugging on AVR . . . 43
6.2.3 Debugging on DSP . . . 43
6.3 Adding support for a new endpoint . . . 43
6.3.1 hwInterface . . . 45
6.3.2 if_initInterface . . . 45
6.3.3 if_readBuf . . . 46
6.3.4 if_writeBuf . . . 46
6.4 I/O Manager . . . 47
6.4.1 General operation . . . 47
6.4.2 Cache decisions . . . 47
6.4.3 Functions . . . 48
6.5 C library for EFSL . . . 51

1 Preface

1.1 Project aims

The EFSL project aims to create a library for filesystems, to be used on various

2 Getting started

2.1 On Linux (file)

Debugging efsl on embedded devices is a rather hard job, because you can't just printf debug strings or watch memory maps easily. Because of that, core development has been performed under the Linux operating system. Under Linux, efsl can be compiled as library and used as a userspace filesystem handler. On Unix- style operating system (like Linux), all devices (usb stick, disc, ...) can be seen as a file, and as such been opened by efsl.

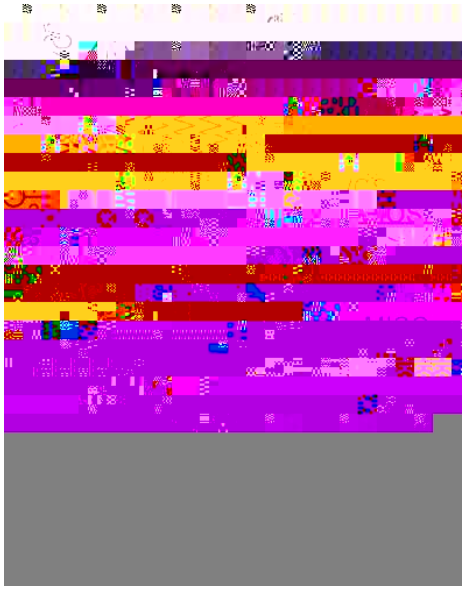
In the following section, we will explain how to get started using efsl as userspace

2.2 On AVR (SD-Card)

This section describes how to implement Efsl on a AVR μC connected to an SD-

Connect the following lines on the SD-card:

- Pin 9 (DAT2) - NC
  (or pull-up to 3.3V)
- Pin 1 (CD) - Any pin on the Atmega128
- Pin 2 (CMD) - MOSI
  (pin 12 on the Atmega128)
- Pin 3 (Vss) - GND
- Pin 4 (Vdd) - +3.3V
- Pin 5 (CLK) - SCK
  (pin 11 on the Atmega128)
- Pin 6 (Vss) - GND
- Pin 7 (DAT0) - MISO
  (pin 12 on the Atmega128)
- Pin 8 (DAT1) - NC
  (or pull-up to 3.3V)



Remark: this schematic includes pull-up's to 3.3V, which can be left o.

Remark 1: Mak0-11.95a-3eeat95a-3y7rk-ur4.54/F977.69-3145.2.435

First, create a new directory in which you put the compiled efsl-library (

- Line 7: The object efs is created, this object will contain information about the hardware layer, the partition table and the disc.
- Line 8: The objects `file_r` and `file_w` are created, these objects will contain information about the files that we will open on the efs-object.
- Line 9: A buffer of 512 bytes is allocated. This buffer will be used for reading and writing blocks of data.
- Line 12: Call of `efs_init()`, which will initialize the efs-object. To this function we pass:

  1. A pointer to the efs-object.
  2. A pointer to the file that contains the partition table / file system (in this example, we select a device as file).

  If this function returns 0, it means that a valid fat partition is found on the SD-card connected. If no valid fat-filesystem is found, or the file does not exist, the function returns a negative value. In this example we then go to an infinite loop to prevent the program to continue.

-

2.2.4 Testing

So now let's test the program:

1. Make sure that your directory contains both the example from above called avrtest.c and the library libefsl.a .
2. Compile the program:
   - On Linux (with avr-gcc): avr-gcc -I/home/user/efsl/inc/ -I/home/user/efsl/conf -reestanding -mmcu=atmega128 -Os -o avrtest.o avrtest.c -L./ -lefsl
   - On Windows (with WinAVR): avr-gcc -Ic:\efsl\inc -Ic:\efsl\conf -reestanding -mmcu=atmega128 -Os -o avrtest.o avrtest.c -L.\ -lefsl
3. Generate(-221(-I/homa-I/h3(hexest70(frohomvr-gcc)-4oefs55(jcop-gccy-I/homejfrohom.text-I/h3(ejfroh

2.3.2 McBSP configuration

McBSP Register Explanations			
SPCR			Serial Port Control Register
Name	Bit	Value	Value (0x00001800 | 0x00410001)
RRST	0	1b	The serial port receiver is enabled
XRST	16	1b	The serial port transmitter is enabled

2.4 On ARM7 (SD-Card)

This section describes how the ARM7 port of EFSL works. This documentation was written by Martin Thomas, as is the port to the ARM7 and the examples included with EFSL. The examples are pretty large, so we will not print them here, they have their own subdirectory in the examples sections and should be quite understandable.

2.4.1 License

1. Example lpc2138

3 Configuring EFSL

On architectures that do have the alignment problem, you should turn this flag o. Failure to do so will result in undefined behavior.

3.3 Cache configuration

This section is dedicated to configuring the cache memory for the library. Caching is performed by the IOMan object, see section 6.4.

IOMAN_NUMBUFFER

This number determines how much memory will be used for caching. Since this is sector based one `IOMAN_NUMBUFFER`

3.5 Endianness

The Microsoft FAT filesystem was originally created to be run on Intel compatible hardware. Therefore the Microsoft programmers decided to record all data on the disc in little endian format. Our library supports running on big endian devices. Here you can select whether your target CPU is little or big endian.

Running on big endian will cause some performance lose because (rather simple)

- On AVR debug will be sent over a selected UART
  Make sure youT

4.2 efs

4.4 file

```
25          fs_umount(&efs.myFs);
26      }
```

4.5 file_read

Purpose

Reads a file and puts it's content in a buffer.

Prototype

```
euint32 file_read (File *file, euint32 size, euint8 *buf);
```

Arguments

```
24          DBG((TXT("File opened for reading.\n")));
25
26          /* Write buffer to file */
27          if( file\n")));
/
```

4.7 mkdir

Purpose

Creates a new directory.

Prototype

```
esint8 mkdir(FileSystem *fs,eint8* dirname);
```

Arguments

Objects passed to `mkdir` :

- `fs` : pointer to the FileSystem object
- `dir` : pointer to the path + name of the new directory

Return value

Returns 0 if no errors are detected.

Returns non-zero if an error is detected:

- Returns -1 if the directory already exists.
-

```
18              mkdir(&efsl.myFs,"dir1/subdir2");
19              mkdir(&efsl.myFs,"dir1/subdir3");
20          }
21
22          /* Close filesystem */
23          fs_umount(&efsl.ms                    8
```

4.8 ls_openDir

Purpose

This function opens a directory for viewing, allowing you to iterate through it's contents.

Prototype

```
esint8 ls
```

21 list

4.10 rmfile

Purpose

Deletes a file.

Prototype

```
esint16 rmfile(FileSystem *fs,euint8* filename);
```

Arguments

Objects passed to `rmfile` :

- `fs` : pointer to the FileSystem object
- `filename` : pointer to the path + name of the file to be removed

Return value

Returns 0 if no errors are detected.

Returns non-zero if an error is detected, most likely that the file does not exist.

Note

If you have opened a file with `fopen()` , and you wish to delete it, first close

```
18
19          /* Close filesystem */
20          fs_umount(&efsl.myFs);
21      }
```

4.11 Getting the free space

To get the free space left on EFSL 0.2 is a bit tricky. This feature was implemented after it had gone into stable, so it couln't interfere with other library functions.

5 EFSL utilities

5.1 Notations

The utilities can be compiled and run on any POSIX compliant system. Al-

sourcefile on your local filesystem. The third argument (

6 Developer notes

6.1 Integer types

- Initialize the hardware
- Read sectors from disc
- Write sectors to disc

All requests are *sector*based, a sector is a 512 byte piece from the disc, that is

6. Add your object file to the Makefile Take the Makefile that works best on your platform (they should all work with GNU/Make), or create a new one, using the existing one's as a template. Make sure to include your new pigeon object to the library. If you have an 'ar' like utility you can create a static library, else you may have to create a new project containing all required source files.

The basic framework is now complete, now all that's left to do is to write the code that will perform the actual flying work.

6.3.1 hwInterface

This structure represents the underlying hardware. There are some field that

```
11
12      /  Initialize hardware  /
13      feed(hw->pigeon);
14      pet (hw->pigeon);
15
16      /  Get sectors count  /
17
```

6.4 I/O Manager

The IOManager that is the second lowest layer of the embedded filesystems library is responsible for coordinating disk input and output, as well as managing a caching system. This documentation describes the second implementation of IOMan, which includes features such as :

- Delayed write
- Buffer reference statistics
- Buffer exportable to users
- Support for cached direct I/O as well as indirect I/O
- Can allocate memory itself (on the stack), or you can do it yourself (heap)

3.

7 Legal notes

distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source

libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies (y)-5’sn are n5t

application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is