Communication

ELAD

# ELAD FDM-S1


## API Documentation for Linux Libraries

All needed to run FDM-S1 under Linux operating system was developed in collaboration with the CSP research center, Torino - Italy

CSP innovazione nelle ICT

# Index

## 1 Summary



© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

# 2 Files

## 2.1 Runtime

After installing "**libfdms1_1.0-1_i386.deb**" the following libraries will be created in "**/usr/local/lib**":

| Filename | Description |
|---|---|
| libfdms1_hw_init_192000.so.1.0<br>libfdms1_hw_init_384000.so.1.0<br>libfdms1_hw_init_768000.so.1.0<br>libfdms1_hw_init_1536000.so.1.0<br>libfdms1_hw_init_3072000.so.1.0<br>libfdms1_hw_init_6144000.so.1.0 | The initialization library, provided in different versions depending on the sample rate. Each on this library will upload a different firmware into the FDMS1 hardware. |
| libfdms1_hw_ctrl.so.1.0 | The control library, which provides those functions that are independent from the firmware version. |

## 2.2 SDK files

The following files are provided as part of the development kit.

| Filename | Description |
|---|---|
| API Documentation for Linux Libraries.pdf | The present document. |
| example.c | Sample code showing initialization operation. |

# 3 Initialization

Client application must initialize the FDMS1 before issuing any control operation. The initialization consists of the following steps:

1. Obtain a working connection to the device using the libusb API;
2. Dynamically load the initialization library based on the sample rate, this is done using **dlopen()** Linux API call defined in **dlfcn.h**;
3. Fetch the only function exported by the initialization library "**fdms1_hw_init**" by using the **dlsym()** Linux API call (see **dlfcn.h**) ;
4. Call the *fdms1_hw_init()* function.

The *fdsm1_hw_init()* function has the following signature:

***int*** *fdms1_hw_init* ***(*** *libusb_device_handle* ******* *pUSB* ***)***

The function returns "0" on error.

The sample code in file "example.c" describes how to dynamically load the initialization library, obtain a pointer to the initialization function and call it. The example is not working because it does not contain the code to initialize the connection to the USB device using libusb (please refer to the **libusb** (*www.libusb.org*) documentation for more information).

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

A shorter example follows.

```
/* define a type for this function for convenience  */
typedef int (* FDMS1_HW_INIT)(libusb_device_handle *);

/* define the pointer to the library and to the init function */
void *init_lib;
FDMS1_HW_INIT FDMS1HWInit;

/* Load the library */
init_lib = dlopen("libfdms1_hw_init_192000.so.1.0", RTLD_NOW);
if (init_lib != NULL ) {

   /* fetch the init function */
   init = (FDMS1_HW_INIT) dlsym (init_lib, "fdms1_hw_init");

   if (init != NULL ) {
         /* Initialize the firmware */
         init ( usb_dev_handle );
   }

   dlclose (init_lib);
}
```

# 4 Control

To control the device client application should load the library " **fdms1_hw_init.so.1.0**" using **dlopen()** call afterwards a pointer to the needed control function should be obtained by using **dlsym()** call.

## 4.1 Loading the library

```
...
#include <dlfcn.h>
...
void *control_lib;
...

control_lib = dlopen("libfdms1_hw_control.so.1.0", RTLD_NOW);
if (control_lib != NULL) {
  /* Obtain the pointer to the required functions and call them ...
*/
}
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

## 4.2 30MHz filter

Synopsis

```
void set_en_ext_io_LP30 (
        libusb_device_handle *deviceHandle,
        int *d_en_ext_io_LP30
)
```

Enables 30MHz filter.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef void (* SET_EN_EXT_IO_LP30)(libusb_device_handle *, int *);

/* define the pointer */
SET_EN_EXT_IO_LP30 set_en_ext_io_LP30;

/* fetch the pointer using the library handle and call it*/
set_en_ext_io_LP30 =
   (SET_EN_EXT_IO_LP30) dlsym(control_lib , "set_en_ext_io_LP30");

/* Call */
if (set_en_ext_io_LP30 != NULL) {
   int d_en_ext_io_LP30 = 1;
   set_en_ext_io_LP30(usb_dev_handle, &d_en_ext_io_LP30)
}
```

## 4.3 20dB Attenuator

Synopsis

```
void set_en_ext_io_ATT20 (
        libusb_device_handle *deviceHandle,
        int *d_en_ext_io_ATT20
)
```

Enables 20dB attenuator.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef void (* SET_EN_EXT_IO_ATT20)(libusb_device_handle *, int *);

/* define the pointer */
SET_EN_EXT_IO_ATT20 set_en_ext_io_ATT20;

/* fetch the pointer using the library handle and call it*/
set_en_ext_io_ATT20 =
   (SET_EN_EXT_IO_ATT20) dlsym(control_lib , "set_en_ext_io_ATT20");
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

```
/* Call */
if (set_en_ext_io_ATT20 != NULL) {
   int d_en_ext_io_ATT20 = 1;
   set_en_ext_io_ATT20(usb_dev_handle, &d_en_ext_io_ATT20)
}
```

## 4.4 I/Q Swap

Synopsis

```
void set_en_ext_io_SWAPIQ (
        libusb_device_handle *deviceHandle,
        int *d_en_ext_io_SWAPIQ
)
```

Swaps I and Q samples.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef void (* SET_EN_EXT_IO_SWAPIQ)(libusb_device_handle *, int
*);

/* define the pointer */
SET_EN_EXT_IO_SWAPIQ set_en_ext_io_SWAPIQ;

/* fetch the pointer using the library handle and call it*/
set_en_ext_io_SWAPIQ =
   (SET_EN_EXT_IO_SWAPIQ) dlsym(control_lib ,
"set_en_ext_io_SWAPIQ");

/* Call */
if (set_en_ext_io_SWAPIQ != NULL) {
   int d_en_ext_io_SWAPIQ = 1;
   set_en_ext_io_SWAPIQ(usb_dev_handle, &d_en_ext_io_SWAPIQ)
}
```

## 4.5 Set LO Frequency

Synopsis

```
int set_HWLO (
        libusb_device_handle *deviceHandle,
        long *lo_freq
)
```

Modify the LO frequency. Returns 1 on success.

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef int (* SET_HWLO)(libusb_device_handle *, long *);

/* define the pointer */
SET_HWLO set_HWLO;

/* fetch the pointer using the library handle and call it*/
set_HWLO =
   (SET_HWLO) dlsym(control_lib , "set_HWLO");

/* Call */
if (set_HWLO != NULL) {
   long freq = ...;
   set_HWLO(usb_dev_handle, &freq)
}
```

## 4.6 OpenHW

Synopsis

```
int OpenHW (
        libusb_device_handle *deviceHandle,
        long sample_rate
)
```

Opens the FDMS1 hardware.
Returns **1** on success.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef int (* OPEN_HW)(libusb_device_handle *, long );

/* define the pointer */
OPEN_HW OpenHW;

/* fetch the pointer using the library handle and call it*/
OpenHW =
   (OPEN_HW) dlsym(control_lib , "OpenHW");

/* Call */
if (OpenHW != NULL) {
   OpenHW(usb_dev_handle, sample_rate)
}
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

## 4.7 Cmd_ExtIO

Synopsis

```
int Cmd_ExtIO (
        libusb_device_handle* deviceHandle,
        int command
        char* data
        int unused
)
```

This routine is provided as an extensible way to provide access to some low level functions and registers.
Returns **1** on success.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef int (* CMD_EXTIO)(libusb_device_handle *, int, char *, int);

/* define the pointer */
CMD_EXTIO Cmd_ExtIO;

/* fetch the pointer using the library handle and call it*/
Cmd_ExtIO =
   (CMD_EXTIO) dlsym(control_lib , "Cmd_ExtIO");

/* Call to read FDMS1 SN */
if (Cmd_ExtIO != NULL) {
   char sn[32];
   if Cmd_ExtIO(usb_dev_handle, 0x09, sn, 0) {
         printf("Device SN: %s\n", sn);
   }
}
```

The following table contains supported command codes.

| | | |
|---|---|---|
| 0x03 | Reads USB status | “data” points to a buffer that will contain 4 bytes representing a 32 bit integer:<br>• -1 , generic USB error<br>• -2 , generic FDMS1 error |
| 0x09 | Read SN | “data” is a 32 byte buffer that will contain a 0 termed string representing the serial number |

## 4.8 StartFIFO

Synopsis

```
void StartFIFO ( libusb_device_handle* deviceHandle )
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

This routine starts the I/Q samples transmission from the FDMS1 to the client application through the USB channel.

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef void (* START_FIFO)(libusb_device_handle *);

/* define the pointer */
START_FIFO StartFIFO;

/* fetch the pointer using the library handle and call it*/
StartFIFO =
   (START_FIFO) dlsym(control_lib , "StartFIFO");

/* Call to read FDMS1 SN */
if (StartFIFO != NULL) {
   StartFIFO( usb_dev_handle );
}
```

## 4.9 Finalization functions

Synopsis

void **StopFIFO** ( libusb_device_handle* *deviceHandle* )
void **StopHW** ( )
void **CloseHW** ( )

How to obtain the pointer and call it.

```
/* define a type for this function for convenience  */
typedef void (* STOP_FIFO)(libusb_device_handle *);
typedef void (* STOP_HW)();
typedef void (* CLOSE_HW)();

/* define the pointer */
STOP_FIFO StopFIFO;
STOP_HW StopHW;
CLOSE_HW CloseHW;

/* fetch the pointer using the library handle and call it*/
StopFIFO = (STOP_FIFO) dlsym(control_lib , "StopFIFO");
StopHW = (STOP_HW) dlsym(control_lib , "StopHW");
CloseHW = (CLOSE_HW) dlsym(control_lib , "CloseHW");

/* Call */
if (StopFIFO != NULL) StopFIFO( usb_dev_handle );
if (StopHW != NULL) StopHW();
if (CloseHW != NULL) CloseHW();
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.

## 4.10 Release

To release the library simply call **dlclose()** Linux API.

```
...
dlclose( control_lib );
...
```

© 2013 ELAD S.r.l. All rights reserved. No part of this document may be reproduced, published, used, disclosed or disseminated in any form or by any means, electronic, photocopying or otherwise, without prior written permission of ELAD S.r.l.