Technical documentation

libfmio User Manual

19 December 2011

Norwegian
Meteorological Institute
met.no

Revision history

Version	Date	Comment	Responsible
Version 1.0	02.05.2007	First documentation.	Øystein Godøy
Version 1.1	03.01.2011	Add proj string paragraph for fm_MITIFF_create_head()	Thomas Lavergne
Version 1.2	19.12.2011	Added fm_readHLHDFdata to interface HLHDF files.	Øystein Godøy

Table of Contents

1 Introduction 4
1.1 What 4
1.2 Why 4
1.3 Where 4
1.4 When 4
1.5 Who 4
2 Compilation and implementation 5
2.1 Requirements 5
2.2 Compilation 5
3 Library variables and structures 5
3.1 Definitions 5
3.2 Containers 5
4 Library functions 6
4.1 High level generic functions 6
4.1.1 fm_init_fmio_img 6
4.1.2 fm_readheader 6
4.1.3 fm_readdata 7
4.1.4 fm_img2fmtime 7
4.1.5 fm_img2slopes 7
4.1.6 fm_byte2float 7
4.2 METSAT 7
4.2.1 fm_readMETSATheader 7
4.2.2 fm_readMETSATdata 7
4.3 MIHDF5 8
4.4 HLHDF 8
4.4.1 fm_readHLHDFdata 8
4.5 MITIFF 8
4.5.1 fm_MITIFF_read 8
4.5.2 fm_MITIFF_read_imagepal 8
4.5.3 fm_MITIFF_fillhead 8
4.5.4 fm_MITIFF_fillhead_imagepal 8
4.5.5 fm_MITIFF_create_head 9
4.5.6 fm_MITIFF_write 9
4.5.7 fm_MITIFF_write_rgb 9
4.5.8 fm_MITIFF_write_imagepal 9
4.5.9 fm_MITIFF_write_multi 10

1 Introduction

1.1 What

`libfmio` is a C library containing various functions for file format handling of remote sensing data. It is specifically developed for the needs at the Norwegian Meteorological Institute, but might be useful to other as well.

1.2 Why

`libfmio` have been developed to help handling remote sensing data file formats. Specifically it is focused at storage and presentation formats in use at the Norwegian Meteorological Institute. However, the commercial file formats that have been used earlier are not supported by this library.

1.3 Where

`libfmio` have been developed for the following purposes:

- Research and development at the Norwegian Meteorological Institute
- Operational processing of remote sensing data at the Norwegian Meteorological Institute

1.4 When

`libfmio` builds on an earlier software library `libsatimg`. However `libsatimg` contained both utility functions and file format specifications. The latter complicated the building and use of the library as some of the file formats were defined by commercial companies delivering software to the Norwegian Meteorological Institute. This also complicated use of the library in projects involving external partners. These file formats have been excluded from the new library which solely builds on NCSA HDF5 and TIFF 6.0.

1.5 Who

`libfmio` have been developed by scientists employed by the Norwegian Meteorological Institute and Danish Meteorological Institute. However, the library is free to be used by anyone that would find it useful. It is distributed using GNU General Public License. See the NCSA HDF5 and TIFF homepages for details on the licenses of this software.

The following persons have been involved in the software development:

Name	Function	Affiliation
Øystein Godøy	Coordinator, Software development	Norwegian Meteorological Institute (http://www.met.no/)
Steinar Eastwood	Software development	Norwegian Meteorological Institute (http://www.met.no/)
Søren Andersen	Software development	Danish Meteorological Institute (http://www.dmi.dk/)

Name	Function	Affiliation
Thomas Lavergne	Software development	Norwegian Meteorological Institute (http://www.met.no/)

2 Compilation and implementation

2.1 Requirements

`libfmio` have functionality that relies on `libfmutil`, NCSA HDF5 and `libtiff`. However, the library can be built without this software. The functionality requiring this software will then be disabled.

`libfmio` have been developed on a Linux system running Fedora Core 3 and Fedora Core 5. Although being developed for portability, this has not been thoroughly tested yet.

2.2 Compilation

The following steps are required to build the software:

1. Unpack the library (distributed as a gzipped tarball) using `tar xvzf libfmio.tgz`
2. Run `autoreconf --install --verbose`
3. execute `configure`. The `–help` option of `configure` will list options. If e.g. PROJ.4 support is not needed, use `–without-proj`. If reconfiguration of the system is required, remember to run `make distclean`.
4. `make`
5. `make install`

3 Library variables and structures

3.1 Definitions

```
#define FMIO_MISSING -32767
#define FMIO_MEMERROR -999
#define FMIO_REARTH 6371.
#define FMIO_SATSIZE 81
#define FMIO_NCHAN 8
#define FMIO_MAXCHANNELS 6
#define FMIO_MAXIMGSIZE 1440000
#define FMIO_TIFFHEAD 1024
#define FMIO_FIELDS 19
```

3.2 Containers

```
typedef struct fmio_imgh {
    unsigned int iw; /* image width (pixels) */
```

```
    unsigned int ih; /* image height (pixels) */
    unsigned int size; /* image size (pixels) */
    struct sat_track *track; /* subsatellite track (lat, lon) */
    int numtrack; /* no. of subsattracks */
    float rga; /* albedo gain value */
    float ria; /* albedo intercept value */
    float rgt; /* temperature gain value */
    float rit; /* temperature intercept value */
    char sa[20];  /* satellite name */
    char area[20];  /* area name */
    /*char source[20];*/  /* process name */
    usi ho; /* satellite hour */
    usi mi; /* satellite minute */
    usi dd; /* satellite day */
    usi mm; /* satellite month */
    usi yy; /* satellite year */
    usi z; /* satellite no. of channels */
    usi ch[NCHAN]; /* satellite channels */
    float Bx; /* UCS Bx */
    float By; /* UCS By */
    float Ax; /* UCS Ax */
    float Ay; /* UCS Ay */
    unsigned short *image[NCHAN]; /* image data for all channels */
} fmio_strimgh;

typedef struct fmio_miheadpal {
    char *name;
    usi noofcl;
    char **clname;
    unsigned short cmap[3][256];
} fmio_strmiheadpal;
```

4 Library functions

4.1 High level generic functions

4.1.1 fm_init_fmio_img

```
int fm_init_fmio_img(fmio_img *h);
```

4.1.2 fm_readheader

```
int fm_readheader(char *filename, fmio_img *h);
```

Read header information from the file types supported into the `fmio_img` structure. Currently MITIFF and METSAT[1] files are supported. The code is adapted for support of the Kongsberg Spacetec fileformats (in HDF and HDF5), but this part has been disabled due to the fact that these libraries are only available as binary files. They may be supported in time,

1 The METSAT file format was defined by the EUMETSAT NWCSAF PPS team.

if feasible.

Remember to use `fm_init_fmio_img` prior to this function.

4.1.3 fm_readdata

```
int fm_readdata(char *satfile, fmio_img *h);
```

Read header and data information from the file types supported into the fmio_img structure. Currently MITIFF and METSAT[2] files are supported. The code is adapted for support of the Kongsberg Spacetec fileformats (in HDF and HDF5), but this part has been disabled due to the fact that these libraries are only available as binary files. They may be supported in time, if feasible.

Remember to use `fm_init_fmio_img` prior to this function.

4.1.4 fm_img2fmtime

```
int fm_img2fmtime(fmio_img imghead, fmtime *newdate)
```

Extract the information relevant in the `fmtime` structure from the `fmio_img structure`.

4.1.5 fm_img2slopes

```
int fm_img2slopes(fmio_img imghead, fmscale *newcal);
```

Extract the information relevant in the `fmscale` structure from the `fmio_img` structure.

4.1.6 fm_byte2float

```
float fm_byte2float(unsigned short value, fmscale byte2float, char
*keyword);
```

Convert character values to floating point values by using the relevant information from the `fmio_img` structure (as extracted by `fm_img2slopes`).

4.2 METSAT

4.2.1 fm_readMETSATheader

```
int fm_readheaderMETSAT(char *filename, fmio_img *h);
```

Read header from METSAT files, store the results into a `fmio_img` structure (the data part of the structure is left unchanged and will not be allocated until data are being read by `fm_readMETSATdata`). Remember to use `fm_init_fmio_img` prior to this function.

4.2.2 fm_readMETSATdata

```
int fm_readdataMETSAT(char *satfile, fmio_img *h);
```

Read header and data from METSAT files, store the results into a `fmio_img` structure. Remember to use `fm_init_fmio_img` prior to this function.

2 The METSAT file format was defined by the EUMETSAT NWCSAF PPS team.

4.3 MIHDF5

TBW MEOS specific formats have been removed, own HDF5 format based upon METSAT header and libosihdf5 will be added in time...

4.4 HLHDF

The HLHDF (NCSA HDF5 files generated using SMHIs HLHDF library) is decoded using the function defined below.

4.4.1 fm_readHLHDFdata

```
int fm_readHLHDFdata(char *filename, fmdataset *d, fmbool headeronly);
```

This function reads and decodes HLHDF files containing AVHRR, observation geometry or NWCSAF PPS products. It recognises the products found when opening the file.

Beware that this is a prototype and that it utilises the new dataset container (`fmdataset`) defined within `libfmutil` contrary to the other functions in this library. All functions will in time be updated to use this container.

4.5 MITIFF

4.5.1 fm_MITIFF_read

```
int fm_MITIFF_read(char *infile, unsigned char *image[], fmio_mihead *ginfo);
```

Read and decode MITIFF image files with satellite data on the multichannel format.

4.5.2 fm_MITIFF_read_imagepal

```
int fm_MITIFF_fillhead_imagepal(char *asciifield, char *tag, fmio_mihead_pal *palinfo);
```

To read MITIFF palette colour files containing a categorized satellite imagery or radar imagery.

4.5.3 fm_MITIFF_fillhead

```
int fm_MITIFF_fillhead(char *asciifield, char *tag, fmio_mihead *ginfo);
```

To extract the necessary information from the text string of the MITIFF files header. This header contains information about date, size etc.

4.5.4 fm_MITIFF_fillhead_imagepal

```
int fm_MITIFF_fillhead_imagepal(char *asciifield, char *tag, fmio_mihead_pal *palinfo);
```

To be used after `fm_MITIFF_fillhead`. `fm_MITIFF_fillhead` extracts the standard information in the header, while `fm_MITIFF fillhead_imagepal` extracts the information connected to the classes defined in the image, which color they are represented by and the

classnames.

4.5.5 fm_MITIFF_create_head

```
void fm_MITIFF_create_head(char imginfo[FMIO_TIFFHEAD], char satellite[16], char time[17], short satdir, short zsize, char chdesc[20], int xsize, int ysize, char projection[30], char truelat[10], float rot, int xunit, int yunit, float npx, float npy, float ax, float ay, float bx, float by, char *calib);
```

To define the ASCII textstring that is used to carry header information in the DNMI/TIFF image files. This text string is a specially formatted string that is appended to the image through the TIFF tag IMAGEDESCRIPTION. The first line of the string contains the keyword <Satellite:>.

The textstring which is defined by this function is decoded into data structures by functions `fm_MITIFF_read` and `fm_MITIFF_read_imagepal`. The calibration information may be used to create classnames for Palette-color images. A function that creates this part automatically is however not defined yet.

A proj4 string is added to the textstring for later visualization of the MITIFF file in Diana. Only polar stereographic (NH or SH) projections based on a spherical Earth are supported. The `truelat (+lat_ts=)` and `rot (+lon_0=)` are taken from the parameters. `truelat` is a string with format “XX N” or “XX S”. In case the “S” character appears, then the projection will have origin `(+lat_0=)` at South Pole. Otherwise, at North Pole. Values of `+x_0=` and `+y_0=` are computed for correct positioning in Diana.

4.5.6 fm_MITIFF_write

```
int fm_MITIFF_write(char *outfile, unsigned char *image, char newhead[], fmio_mihead ginfo);
```

Writes image data on TIFF formatted file, ready for visualization on any standard image viewer. TIFF tag number 262 is photometric interpretation. This tag is 1 for grayscale images and 3 for palette images. In situations when grayscale images are wanted TIFF tag number 320 can be added. In present version only single page grayscale images are supported.

4.5.7 fm_MITIFF_write_rgb

```
int fm_MITIFF_write_rgb(char *outfile, unsigned char *image[], char newhead[], fmio_mihead ginfo);
```

To write TIFF 6.0 files in RGB mode with MITIFF header if required. Otherwise this function is similar to the version that writes ordinary grayscale images `fm_MITIFF_write`.

4.5.8 fm_MITIFF_write_imagepal

```
int fm_MITIFF_write_imagepal(char *outfile, unsigned char *class, char newhead[], fmio_mihead ginfo, uint16 cmap[3][256]);
```

Writes image data on TIFF formatted file, ready for visualization on any standard image viewer. TIFF tag number 262 is photometric interpretation. This tag is 1 for grayscale images and 3 for palette images. In situations when grayscale images are wanted TIFF tag number 320 can be commented out.

4.5.9 fm_MITIFF_write_multi

```
int fm_MITIFF_write_multi(char *outfile, unsigned char *image[],
char newhead[], fmio_mihead ginfo);
```

Writes multilevel grayscale TIFF 6.0 images.