Tinyfits

A simplified library for manipulating FITS files

Mark Galassi

Dan Starr

Sherri Davidoff

Tinyfits: A simplified library for manipulating FITS files

by Mark Galassi, Dan Starr, and Sherri Davidoff

Table of Contents

Chapter 1. Quick Start Guide to Using tinyfits

For now there is no tutorial; there is an auto-generated API reference in Appendix A which will give you an idea of the calling convention for each function.

Appendix A. tinyfits API reference

This appendix is automatically generated from the structured comments in the C source code, using the gnome-doc program.

tinyfits

tf_img_new

Name

`tf_img_new` —

Synopsis

```
TinyFitsImg * tf_img_new (void);
```

Arguments

None

Description

Creates a new empty TinyFitsImg object

Return value

the newly created image object

tf_get_n_keys

Name

`tf_get_n_keys` —

Synopsis

```
int tf_get_n_keys (TinyFitsImg * img);
```

Arguments

`img`

Description

Returns the number of keys in the given FITS image as stored in img->n_keys. See `tf_get_n_keys_from_header()` to obtain the number of keys by actually counting them in the header.

Return value

the number of keys in the given FITS image

tf_img_load

Name

`tf_img_load` —

Synopsis

```
TinyFitsImg * tf_img_load (char * filename);
```

Arguments

`filename`

Description

Loads a FITS file into a TinyFitsImg object

Return value

a newly created TinyFitsImg object with the contents of the file

tf_img_load_short

Name

`tf_img_load_short` —

Synopsis

```
TinyFitsImg * tf_img_load_short (char * filename);
```

Arguments

`filename`

Description

Loads a FITS file with short integers into a TinyFitsImg object. This differs from `tf_img_load_short()` in that it will rescale the shorts into floats if it finds the TF-SCALE and TFZERO keywords in the FITS header. These are written by tinyfits's `tf_img_save_short()` routine for just this purpose.

Return value

a newly created TinyFitsImg object with the contents of the file

tf_img_save

Name

`tf_img_save` —

Synopsis

```
void tf_img_save (TinyFitsImg * img, char * filename);
```

Arguments

`img`

`filename`

Description

Saves a TinyFitsImg object into a FITS file

tf_img_save_short

Name

`tf_img_save_short` —

Synopsis

```
void tf_img_save_short (TinyFitsImg * img, char * filename);
```

Arguments

`img`

`filename`

Description

saves the image *`img`* as a FITS file of type TSHORT

tf_img_save_short2

Name

`tf_img_save_short2` —

Synopsis

```
void tf_img_save_short2 (TinyFitsImg * img, char * filename);
```

Arguments

`img`

`filename`

Description

saves the image *`img`* as a FITS file of type TSHORT -- this routine is experimental and tries to use the "proper" CFITSIO approach -- if it works then I will use the technique for `tf_img_save_short()` and get rid of this routine

tf_img_free

Name

`tf_img_free` —

Synopsis

```
void tf_img_free (TinyFitsImg * img);
```

Arguments

`img`

Description

Frees up the memory used by the image data and keys

tf_get_key_position

Name

`tf_get_key_position` —

Synopsis

```
int tf_get_key_position (TinyFitsImg * img, char * key_name);
```

Arguments

img

key_name

Description

Return value

the position of the given key in the list of keys

tf_key_exists

Name

`tf_key_exists` —

Synopsis

```
int tf_key_exists (TinyFitsImg * img, char * key_name);
```

Arguments

img

key_name

Description

check if a key exists in the header

Return value

1 if it exists, 0 otherwise

tf_get_key_double

Name

`tf_get_key_double` —

Synopsis

```
double tf_get_key_double (TinyFitsImg * img, char * key_name);
```

Arguments

img

key_name

Description

Given a key name, returns its value as a double

Return value

the value of the requested key

tf_get_key_long

Name

`tf_get_key_long` —

Synopsis

```
long tf_get_key_long (TinyFitsImg * img, char * key_name);
```

Arguments

`img`

`key_name`

Description

Given a key name, returns its value as a long

Return value

the value of the requested key

tf_get_key_int

Name

`tf_get_key_int` —

Synopsis

```
int tf_get_key_int (TinyFitsImg * img, char * key_name);
```

Arguments

`img`

`key_name`

Description

Given a key name, returns its value as an int

Return value

the value of the requested key

tf_get_key_string

Name

`tf_get_key_string` —

Synopsis

```
void tf_get_key_string (TinyFitsImg * img, char * key_name, char *
string_out);
```

Arguments

`img`

`key_name`

`string_out`

Description

returns a FITS string key in *`string_out`*

tf_add_key

Name

`tf_add_key` —

Synopsis

```
void tf_add_key (TinyFitsImg * img, TfSingleKey key);
```

Arguments

`img`

key

Description

tf_add_key_double

Name

`tf_add_key_double` —

Synopsis

```
void tf_add_key_double (TinyFitsImg * img, char * key_name, double
keyval, char * comment);
```

Arguments

img

key_name

keyval

comment

tf_add_key_string

Name

`tf_add_key_string` —

Synopsis

```
void tf_add_key_string (TinyFitsImg * img, char * key_name, char *
keyval, char * comment);
```

Arguments

`img`

`key_name`

`keyval`

`comment`

Description

tf_add_bscale

Name

`tf_add_bscale` —

Synopsis

```
void tf_add_bscale (TinyFitsImg * img, double bscale_val);
```

Arguments

`img`

`bscale_val`

Description

tf_add_bzero

Name

`tf_add_bzero` —

Synopsis

```
void tf_add_bzero (TinyFitsImg * img, double bzero_val);
```

Arguments

img

bzero_val

Description

tf_img_copy_header_without_clobber

Name

`tf_img_copy_header_without_clobber` —

Synopsis

```
void tf_img_copy_header_without_clobber (TinyFitsImg * source,
TinyFitsImg * destination);
```

Arguments

source

destination

Description

not yet implemented

tf_img_copy

Name

`tf_img_copy` —

Synopsis

```
TinyFitsImg * tf_img_copy (TinyFitsImg * old_img);
```

Arguments

`old_img`

Description

Copy constructor for images

Return value

tf_tab_float_new

Name

`tf_tab_float_new` —

Synopsis

```
TinyFitsTabFloat * tf_tab_float_new (int n_columns);
```

Arguments

n_columns

Description

Creates a new floating point table

Return value

the table object

tf_tab_float_free

Name

`tf_tab_float_free` —

Synopsis

```
void tf_tab_float_free (TinyFitsTabFloat * tab);
```

Arguments

tab

Description

Frees up a tinyfits table and the memory it uses

tf_tab_float_load

Name

`tf_tab_float_load` —

Synopsis

```
TinyFitsTabFloat * tf_tab_float_load (char * fname);
```

Arguments

fname

Description

Loads a tinyfits table from file

Return value

the TinyFitsTabFloat object

tf_tab_float_add_row

Name

`tf_tab_float_add_row` —

Synopsis

```
void tf_tab_float_add_row (TinyFitsTabFloat * tab, float * row);
```

Arguments

tab

row

Description

tf_tab_float_get

Name

`tf_tab_float_get` —

Synopsis

```
float tf_tab_float_get (TinyFitsTabFloat * tab, long int row, int
col);
```

Arguments

tab

row

col

Description

tf_imgtype2tabletype

Name

`tf_imgtype2tabletype` —

Synopsis

```
int tf_imgtype2tabletype (int imgtype);
```

Arguments

imgtype

Description

Return value

tf_tabletype2imgtype

Name

`tf_tabletype2imgtype` —

Synopsis

```
int tf_tabletype2imgtype (int tabletype);
```

Arguments

`tabletype`

Description

Return value

tf_tab_float_save_ascii

Name

`tf_tab_float_save_ascii` —

Synopsis

```
void tf_tab_float_save_ascii (TinyFitsTabFloat * tab, char * fname);
```

Arguments

`tab`

`fname`

Description

tf_tab_float_save_image

Name

`tf_tab_float_save_image` —

Synopsis

```
void tf_tab_float_save_image (TinyFitsTabFloat * tab, long n_planes,
long x_dim, long y_dim, char * fname);
```

Arguments

`tab`

`n_planes`

`x_dim`

`y_dim`

`fname`

Description

print_all_columns

Name

`print_all_columns` —

Synopsis

`void print_all_columns (fitsfile * fptr, long row, int ncols);`

Arguments

fptr

row

ncols

Description

tf_delete_key

Name

`tf_delete_key` —

Synopsis

`void tf_delete_key (TinyFitsImg * img, char * tag);`

Arguments

img

tag

Description

tf_delete_mem_key

Name

`tf_delete_mem_key` —

Synopsis

```
int tf_delete_mem_key (TfKeyArray keys, int n_keys, char * tag);
```

Arguments

`keys`

`n_keys`

`tag`

Description

removes the first occurrence of a key from an array of keys

Return value

0 on success, 1 if the key did not exist

tf_img_find_max_min

Name

`tf_img_find_max_min` —

Synopsis

```
void tf_img_find_max_min (TinyFitsImg * img, double * max, double *
min);
```

Arguments

img

max

min

Description

tf_headers_make_key_list

Name

`tf_headers_make_key_list` —

Synopsis

```
TfKeyList tf_headers_make_key_list (TfKeyArray keys_array, int len);
```

Arguments

keys_array

len

Description

Return value

tf_headers_copy

Name

`tf_headers_copy` —

Synopsis

```
void tf_headers_copy (TfKeyList from, TfKeyList to);
```

Arguments

`from`

`to`

Description

tf_headers_filter_remove

Name

`tf_headers_filter_remove` —

Synopsis

```
void tf_headers_filter_remove (TinyFitsImg * img, char * key_name);
```

Arguments

`img`

key_name

Description

tf_get_n_keys_from_header

Name

`tf_get_n_keys_from_header` —

Synopsis

```
int tf_get_n_keys_from_header (TfKeyArray keys);
```

Arguments

keys

Description

Return value

the number of keys in the header (including the END key)

tf_get_position_from_header

Name

`tf_get_position_from_header` —

Synopsis

```
int tf_get_position_from_header (TfKeyArray keys, int n_keys, char *
key_name);
```

Arguments

`keys`

`n_keys`

`key_name`

Description

Given a header name, it finds and returns its line number.

Return value

tf_modify_string_in_header

Name

`tf_modify_string_in_header` —

Synopsis

```
void tf_modify_string_in_header (TfKeyArray keys, int n_keys, char *
key_name, char * new_val);
```

Arguments

`keys`

`n_keys`

`key_name`

`new_val`

Description

This routine replaces the value for a key line (specified by key_name)

tf_copy_header_key

Name

`tf_copy_header_key` —

Synopsis

```
void tf_copy_header_key (TfKeyArray * keys_to_p, TfKeyArray keys_from,
char * key_name);
```

Arguments

`keys_to_p`

`keys_from`

`key_name`

Description

Given 2 headers, this copies the specified line from one to the other. Remember to increment a 'n_keys' counter in calling program.

tf_img_check_consistency

Name

`tf_img_check_consistency` —

Synopsis

```
void tf_img_check_consistency (TinyFitsImg * img);
```

Arguments

`img`

Description

Asserts that various important consistency conditions are met. For now the only one we check is that img->n_keys matches the number of keys in the header.

tf_img_read_shm

Name

`tf_img_read_shm` —

Synopsis

```
TinyFitsImg * tf_img_read_shm (int mem_id);
```

Arguments

`mem_id`

Description

Reads the data from the shared memory segment "mem_id" and returns an image. The caller function must then use `tf_img_free_shm()` to detach and delete the memory segment after the image has been returned.

Return value

tf_img_write_shm

Name

`tf_img_write_shm` —

Synopsis

```
void * tf_img_write_shm (int mem_id, TinyFitsImg * img, char *
proc_name);
```

Arguments

mem_id

img

proc_name

Description

Takes an image and writes it to the shared memory segment "mem_id". tf_img_read_shm must first be running. The caller of the tf_img_read_shm function must then use `tf_img_free_shm()` to detach and delete the memory segment after the image has been returned.

tf_img_free_shm

Name

`tf_img_free_shm` —

Synopsis

```
int * tf_img_free_shm (int mem_id);
```

Arguments

mem_id

Description

This function will detach and delete the memory segment with mem_id. It should be called after the completion of the tf_img_read_shm function, to ensure that the memory is not freed until after the image is returned.

tf_img_open_net_recv

Name

`tf_img_open_net_recv` —

Synopsis

```
int tf_img_open_net_recv (int portnum);
```

Arguments

`portnum`

Description

This function takes a port number, creates a stream socket and listens for a connection on the specified port. When a connection is requested, this function accepts it and opens a second stream socket, and returns this second socket handle. This function should be used when receiving (not sending) images.

tf_img_open_net_send

Name

`tf_img_open_net_send` —

Synopsis

```
int tf_img_open_net_send (char * hostname, int portnum);
```

Arguments

`hostname`

`portnum`

Description

This function takes a hostname and a port number and opens a stream socket connection with the host on that port. To transmit images, this function should be called by the sender (not the receiver).

tf_img_close_net

Name

`tf_img_close_net` —

Synopsis

```
int tf_img_close_net (int sock);
```

Arguments

`sock`

Description

This function closes an internet socket. It takes one argument, "sock" which is the socket handle. (Note that the socket handle is different from the port number, which is used by the tf_img_open_net functions.)

tf_img_read_net

Name

`tf_img_read_net` —

Synopsis

```
TinyFitsImg * tf_img_read_net (int msgsock);
```

Arguments

`msgsock`

Description

This function reads from a socket and writes the data into a contiguous segment of memory, which is then converted into TInyFitsImg format and returned. It takes one argument, "msgsock", which is the socket handle.

tf_img_write_net

Name

`tf_img_write_net` —

Synopsis

```
int tf_img_write_net (int sock, TinyFitsImg * img);
```

Arguments

`sock`

`img`

Description

This function sends a series of images over an internet stream socket to a host and port specified in the arguments. The tf_img_read_net function must be running on the host machine, on the appropriate port, in order to accept the connection.

tf_img_pack

Name

`tf_img_pack` —

Synopsis

```
int * tf_img_pack (void * mem_segment, TinyFitsImg * img);
```

Arguments

`mem_segment`

`img`

Description

This function writes an image into a contiguous segment of memory.

tf_img_unpack

Name

`tf_img_unpack` —

Synopsis

```
TinyFitsImg * tf_img_unpack (char * mem);
```

Arguments

`mem`

Description

This function takes data from a contiguous segment of memory and converts it to TinyFitsImg format. It takes one argument, a pointer to the start of the memory segment.

shared memory interface

tf_shm_img_get

Name

`tf_shm_img_get` —

Synopsis

```
TinyFitsImg * tf_shm_img_get (int mem_id);
```

Arguments

mem_id

Description

Return value

tf_shm_img_put

Name

`tf_shm_img_put` —

Synopsis

```
void tf_shm_img_put (int mem_id, TinyFitsImg * img);
```

Arguments

mem_id

img

Description

image ops

tf_ops_img_subtract

Name

`tf_ops_img_subtract` —

Synopsis

```
void tf_ops_img_subtract (TinyFitsImg * original, TinyFitsImg *
to_be_subtracted);
```

Arguments

original

to_be_subtracted

Description

Subtracts an image from the original and puts the result into the original.

tf_ops_img_remove_hotpix

Name

`tf_ops_img_remove_hotpix` —

Synopsis

```
void tf_ops_img_remove_hotpix (TinyFitsImg * original);
```

Arguments

`original`

Description

This routine can be used to remove hotpixels. It consists of simple factor comparison of surrounding pixels.

tf_ops_img_get_median_overall

Name

`tf_ops_img_get_median_overall` —

Synopsis

```
void tf_ops_img_get_median_overall (TinyFitsImg * img, int x_dim, int
y_dim, int median_img_index, float * median_overall);
```

Arguments

`img`

`x_dim`

`y_dim`

`median_img_index`

`median_overall`

Description

tf_ops_part_img_median_add

Name

`tf_ops_part_img_median_add` —

Synopsis

```
void tf_ops_part_img_median_add (float *** pixels, TinyFitsImg * img,
int x_low, int x_high, int x_dim, int y_dim, int median_img_index);
```

Arguments

pixels

img

x_low

x_high

x_dim

y_dim

median_img_index

Description

tf_ops_img_median_calc

Name

`tf_ops_img_median_calc` —

Synopsis

```
void tf_ops_img_median_calc (float *** pixels, int x_dim, int y_dim,
int num_imgs, TinyFitsImg * img, float * median_overall, float *
variance);
```

Arguments

pixels

x_dim

y_dim

num_imgs

img

median_overall

variance

Description

tf_ops_part_img_median_calc

Name

`tf_ops_part_img_median_calc` —

Synopsis

```
void tf_ops_part_img_median_calc (float *** pixels, int x_low, int
x_high, int x_dim, int y_dim, int num_imgs, TinyFitsImg * img, float
* median_overall);
```

Arguments

`pixels`

`x_low`

`x_high`

`x_dim`

`y_dim`

`num_imgs`

`img`

`median_overall`

Description