FLI CCD Library for Linux

— Version 0.3 —

FLI CCD Library for Linux Documentation.

Finger Lakes Instrumentation

Contents

1 Introduction 4
2 Library Functions 5
3 Example 36

1 Introduction

This library provides a core set of functions for programming FLI CCD cameras under Linux. The type definitions, function prototypes, and definitions/enumerations of constant values used by library functions are spcified in `fli.h` All library functions return zero on sucessful completion, and non-zero if the function failed. The exact nature of a failure can be found by treating the negative of the function's return value as a system error code, for example:

```
if ((err = FLIOpen(port, dev, cam)))
{
  fprintf(stderr, "Error FLIOpen: %s\n", strerror(-err));
  exit(1);
}
```

Library functions use three types of handles to manage how FLI CCD camera hardware is accessed; a port handle of type `fliport_t`, a device handle of type `flidev_t` and a camera handle of type `flicam_t`. The port handle represents an abstract interface to a parallel port managed by the FLI CCD device driver module. The device handle represents an abstract interface to a physical FLI CCD camera device. The camera handle represents an abstract interface to a virtual FLI CCD camera device, which allows multiple virtual cameras to share a single physical camera.

A simple example of how the library functions can be used is given in the Example section.

2

Library Functions

Names

2.1 int **FLIGetLibVersion** (char* ver, size_t len)
Get the current library version. 7

2.2 int **FLIInit** (char* file, fliport_t* port)
Initialize the library. 8

2.3 int **FLIOpen** (fliport_t port, flidev_t dev, flicam_t* cam)
Get a handle to a virtual camera. 9

2.4 int **FLIClose** (flicam_t cam)
Close/Release a camera handle. 10

2.5 int **FLIExit** (void)
Exit the library. 11

2.6 int **FLIGetNextDevice** (fliport_t port, flidev_t dev, flidev_t* nextdev)
Get a handle to the next physical device. 12

2.7 int **FLIGetDevice** (flicam_t cam, flidev_t* dev)
Get a handle to the physical device for a virtual camera. 13

2.8 int **FLISetDevice** (flicam_t cam, flidev_t dev)
Set the physical device for a virtual camera. 14

2.9 int **FLIGetModel** (flidev_t dev, char* model, size_t len)
Get the model of a given physical camera device. 15

2.10 int **FLIGetArrayArea** (flicam_t cam, int* ul_x, int* ul_y, int* lr_x, int* lr_y)
Get the array area of the given virtual camera. 16

2.11 int **FLIGetVisibleArea** (flicam_t cam, int* ul_x, int* ul_y, int* lr_x, int* lr_y)
Get the visible area of the given virtual camera. 17

2.12 int **FLIGetSerialNum** (flidev_t dev, int* serialnum)
Get the serial number of a given physical camera device. 18

2.13 int **FLIGetFirmwareRev** (flidev_t dev, int* firmrev)
Get the firmware revision number of a given physical camera device. 19

2.14 int **FLIExposeFrame** (flicam_t cam)
Expose a frame. 20

2.15 int **FLICancelExposure** (flicam_t cam)
Cancel an exposure. 21

2.16 int **FLIGetExposureStatus** (flicam_t cam, long* timeleft)
Find the remaining exposure time. 22

2.17 int **FLIGrabRow** (flicam_t cam, void* buff, size_t width)
Grab a row of an image. 23

2.18 int **FLIGrabFrame** (flicam_t cam, void* buff, size_t buffsize, size_t* bytesgrabbed)
Grab a frame. 24

2.19 int **FLISetTemperature** (flidev_t dev, double temperature)
Set the temperature of a given physical camera device. 26

2.20 int **FLIGetTemperature** (flidev_t dev, double* temperature)
Get the temperature of a given physical camera device. 27

2.21 int **FLIFlushRow** (flicam_t cam, int rows, int repeat)
Flush rows of a given virtual camera. 28

2.22 int **FLISetExposureTime** (flicam_t cam, long exptime)
Set the exposure time for a given virtual camera. 29

2.23 int **FLISetFrameType** (flicam_t cam, fliframe_t frametype)
Set the frame type for a given virtual camera. 30

2.24 int **FLISetImageArea** (flicam_t cam, int ul_x, int ul_y, int lr_x, int lr_y)
Set the image area for a given virtual camera. 31

2.25 int **FLISetHBin** (flicam_t cam, int hbin)
Set the horizontal bin factor for a given virtual camera. 32

2.26 int **FLISetVBin** (flicam_t cam, int vbin)
Set the vertical bin factor for a given virtual camera. 33

2.27 int **FLISetNFlushes** (flicam_t cam, int nflushes)
Set the number of flushes for a given virtual camera. 34

2.28 int **FLISetBitDepth** (flicam_t cam, flibitdepth_t bitdepth)
Set the gray-scale bit depth for a given virtual camera. 35

2.1

`int` **`FLIGetLibVersion`** `(char* ver, size_t len)`

Get the current library version.

Get the current library version. This function copies up to `len - 1` characters of the current library version string, and a terminating `NULL` character, into the buffer pointed to by `ver`. If `len` is less than or equal to zero this function has no effect.

Parameters:

`ver`	Pointer to a character buffer where the library versionstring is to be placed.
`len`	Size in bytes of buffer pointed to by `ver`.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

2.2

`int` **`FLIInit`** `(char* file, fliport_t* port)`

Initialize the library.

Initialize the library. This function initializes the library and places a port handle in the location pointed to by `port`. The port handle is used in subsequent library calls and acts as an interface to the parallel port. This function must be called before any other library functions are used.

Parameters:

`file`	Filename of character special file that corresponds tothe major/minor number controlled by FLI device driver module(e.g. `/dev/ccd0`).
`port`	Pointer to where a port handle will be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIExit

2.3

int **FLIOpen** (fliport_t port, flidev_t dev, flicam_t* cam)

Get a handle to a virtual camera.

Get a handle to a virtual camera. The caller of this function can optionally request a handle associated with the specific physical camera `dev`. If `dev` has a `NULL` value, the default (first) camera is used. An application may use any number of handles associated with the same physical camera.

Parameters:

`port`	Port handle from a previous call to `FLIInit`.
`dev`	Device handle to request a specific device, or`NULL` for the default (first) device.
`cam`	Pointer to where a camera handle will be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIClose

2.4

`int` **`FLIClose`** `(flicam_t cam)`

Close/Release a camera handle.

Close/Release a camera handle. This function invalidates the given camera handle and releases all resources associated with it. It should be called when the camera handle `cam` will no longer be used.

Parameters: `cam` Camera handle to be closed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIOpen

2.5

`int` **`FLIExit`** `(void)`

Exit the library.

Exit the library. This function releases all resources held by the library. After calling this function, library functions will not operate until `FLIInit` is called.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIInit

2.6

```
int FLIGetNextDevice (fliport_t port, flidev_t
                      dev, flidev_t* nextdev)
```

Get a handle to the next physical device.

Get a handle to the next physical device. This function places a handle to the physical device after `dev`, in the location pointed to by `nextdev`. If `dev` is the last physical device, the next device becomes the first physical device.

Parameters:

`port`	Port handle from a previous call to `FLIInit`.
`dev`	Device handle of the current device.
`nextdev`	Pointer to where a handle to the next will beplaced.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetDevice
FLISetDevice

2.7

`int` **`FLIGetDevice`** `(flicam_t cam, flidev_t* dev)`

Get a handle to the physical device for a virtual camera.

Get a handle to the physical device for a virtual camera. A handle to the physical device associated with the virtual camera `cam` is place in the location pointed to by `dev`.

Parameters:

`cam`	Camera handle to find physical device for.
`dev`	Pointer to where a device handle will be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLISetDevice
FLIGetNextDevice

2.8

`int` **`FLISetDevice`** `(flicam_t cam, flidev_t dev)`

Set the physical device for a virtual camera.

Set the physical device for a virtual camera. This function sets the physical device associated with the virtual camera `cam` to `dev`. Note that the previous image area setting for `cam` may not be valid for new device `dev` since `dev` may have a different array area and visible area.

Parameters:

`cam`	Camera handle to set physical device for.
`dev`	Device handle to associate `cam` with.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetDevice
FLIGetNextDevice

2.9

```
int FLIGetModel (flidev_t dev, char* model, size_t
                 len)
```

Get the model of a given physical camera device.

Get the model of a given physical camera device. This function copies up to `len - 1` characters of the model string for physical device `dev`, and a terminating `NULL` character, into the buffer pointed to by `model`. If `len` is less than or equal to zero this function has no effect.

Parameters:

`dev`	Physical camera device to find model of.
`model`	Pointer to a character buffer where the model stringis to be placed.
`len`	Size in bytes of buffer pointed to by `model`.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetFirmwareRev
FLIGetSerialNum

2.10

int **FLIGetArrayArea** (flicam_t cam, int* ul_x, int* ul_y, int* lr_x, int* lr_y)

Get the array area of the given virtual camera.

Get the array area of the given virtual camera. This function finds the *total* area of the CCD array for virtual camera `cam`. This area is specified in terms of a upper-left point and a lower-right point. The upper-left x-coordinate is placed in `ul_x`, the upper-left y-coordinate is placed in `ul_y`, the lower-right x-coordinate is placed in `lr_x`, and the lower-right y-coordinate is placed in `lr_y`.

Parameters:

`cam`	Virtual camera to find array area of.
`ul_x`	Pointer to where the upper-left x-coordinate is to beplaced.
`ul_y`	Pointer to where the upper-left y-coordinate is to beplaced.
`lr_x`	Pointer to where the lower-right x-coordinate is to beplaced.
`lr_y`	Pointer to where the lower-right y-coordinate is to beplaced.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetVisibleArea
FLISetImageArea

2.11

```
int FLIGetVisibleArea (flicam_t cam, int* ul_x,
                       int* ul_y, int* lr_x, int*
                       lr_y)
```

Get the visible area of the given virtual camera.

Get the visible area of the given virtual camera. This function finds the *visible* area of the CCD array for virtual camera `cam`. This area is specified in terms of a upper-left point and a lower-right point. The upper-left x-coordinate is placed in `ul_x`, the upper-left y-coordinate is placed in `ul_y`, the lower-right x-coordinate is placed in `lr_x`, the lower-right y-coordinate is placed in `lr_y`.

Parameters:

`cam`	Virtual camera to find visible area of.
`ul_x`	Pointer to where the upper-left x-coordinate is to beplaced.
`ul_y`	Pointer to where the upper-left y-coordinate is to beplaced.
`lr_x`	Pointer to where the lower-right x-coordinate is to beplaced.
`lr_y`	Pointer to where the lower-right y-coordinate is to beplaced.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLISetImageArea
FLIGetArrayArea

2.12

`int` **`FLIGetSerialNum`** `(flidev_t dev, int* serialnum)`

Get the serial number of a given physical camera device.

Get the serial number of a given physical camera device. This function copies the serial number for physical device `dev` into the location pointed to by `serialnum`.

Parameters:

`dev`	Physical camera device to find serial number of.
`serialnum`	Pointer to a integer location where the serialnumber is to be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetModel
FLIGetFirmwareRev

2.13

```
int FLIGetFirmwareRev (flidev_t dev, int*
                       firmrev)
```

Get the firmware revision number of a given physical camera device.

Get the firmware revision number of a given physical camera device. This function copies the firmware revision number for physical device `dev` into the location pointed to by `firmrev`.

Parameters:

`dev`	Physical camera device to find firmware revision numberof.
`firmrev`	Pointer to a integer location where the firmwarerevision number is to be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetModel
FLIGetSerialNum

2.14

`int` **`FLIExposeFrame`** `(flicam_t cam)`

Expose a frame.

Expose a frame. This function exposes a frame according to the settings (image area, exposure time, bit depth, etc.) of virtual camera `cam`. The settings of `cam` must be valid for the physical camera device `cam` represents. They are set by calling the appropriate set library function. This function returns after the exposure has started.

Parameters: `cam` Virtual camera to expose frame of.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also:
FLIGrabFrame
FLICancelExposure
FLIGetExposureStatus
FLISetExposureTime
FLISetFrameType
FLISetImageArea
FLISetHBin
FLISetVBin
FLISetNFlushes
FLISetBitDepth

2.15

`int` **`FLICancelExposure`** `(flicam_t cam)`

Cancel an exposure.

Cancel an exposure. This function cancels an exposure in progress by closing the shutter.

Parameters:

`cam`	Virtual camera to cancel exposure of.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also:

FLIExposeFrame
FLIGetExposureStatus
FLISetExposureTime

2.16

```
int FLIGetExposureStatus (flicam_t cam, long*
                          timeleft)
```

Find the remaining exposure time.

Find the remaining exposure time. This functions places the remaining exposure time (in milliseconds) in the location pointed to by `timeleft`.

Parameters:	`cam`	Virtual camera to find remaining exposure time of.
	`timeleft`	Pointer to where the remaining exposure time inmsec will be placed.
Return Value:	`Zero`	on success.
	`Non-zero`	on failure.
See Also:	FLIExposeFrame FLICancelExposure FLISetExposureTime	

2.17

```
int FLIGrabRow (flicam_t cam, void* buff, size_t
                width)
```

Grab a row of an image.

Grab a row of an image. This function grabs the next available row of the image from virtual camera device `cam`. The row of width `width` is placed in the buffer pointed to by `buff`. The size of the buffer pointed to by `buff` must take into account the bit depth of the image, meaning the buffer size must be at least `width` bytes for an 8-bit image, and at least 2*`width` for a 16-bit image.

Parameters:

`cam`	Virtual camera whose image to grab the next availablerow from.
`buff`	Pointer to where the next available row will be placed.
`width`	Row width in pixels.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGrabFrame

2.18

```
int FLIGrabFrame (flicam_t cam, void* buff, size_t
                  buffsize, size_t* bytesgrabbed)
```

Grab a frame.

Grab a frame. This function grabs a complete frame (image) according to the settings (image area, exposure time, bit depth, etc.) of virtual camera `cam`. Up to `buffsize` bytes of the grabbed image are placed into the buffer pointed to by `buff`, and the actual number of bytes grabbed is placed in the location pointed to by `bytesgrabbed`. The settings of `cam` must be valid for the physical camera device `cam` represents. They are set by calling the appropriate set library function. This function returns after the frame has been exposed and the image is acquired. This includes starting the exposure, waiting until the exposure is complete, flushing rows above the image area, grabbing all rows of the image, and flushing any remaining rows below the image.

Parameters:

`cam`	Virtual camera to grab frame of.
`buff`	Pointer to where the grabbed frame will be placed.
`buffsize`	The size in bytes of the buffer pointed to by`buff`.
`bytesgrabbed`	Pointer to where the actual number of bytesgrabbed will be placed.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also:

FLISetExposureTime
FLISetFrameType
FLISetImageArea
FLISetHBin
FLISetVBin
FLISetNFlushes
FLISetBitDepth
FLIExposeFrame
FLIGetExposureStatus
FLIFlushRow
FLIGrabRow

2.19

```
int FLISetTemperature (flidev_t dev, double
                       temperature)
```

Set the temperature of a given physical camera device.

Set the temperature of a given physical camera device. This function sets the temperature of the CCD camera cold finger for physical device `dev` to `temperature` degrees Celsius. The valid range of the `temperature` parameter is from -55 C to 45 C.

Parameters:

`dev`	Physical camera device to set temperature of.
`temperature`	Temperature in Celsius to set CCD camera coldfinger to.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetTemperature

2.20

```
int FLIGetTemperature (flidev_t dev, double*
                       temperature)
```

Get the temperature of a given physical camera device.

Get the temperature of a given physical camera device. This function places the temperature of the CCD camera cold finger of physical device `dev` in the location pointed to by `temperature`.

Parameters:	`dev`	Physical camera device to get temperature of.
	`temperature`	Pointer to where the temperature will be placed.
Return Value:	`Zero`	on success.
	`Non-zero`	on failure.
See Also:	FLISetTemperature	

2.21

```
int FLIFlushRow(flicam_t cam, int rows, int
                repeat)
```

Flush rows of a given virtual camera.

Flush rows of a given virtual camera. This function flushes `rows` rows of virtual camera `cam`, `repeat` times.

Parameters:

`cam`	Virtual camera to flush rows of.
`rows`	Number of rows to flush.
`repeat`	Number of times to flush each row.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

2.22

```
int FLISetExposureTime (flicam_t cam, long
                        exptime)
```

Set the exposure time for a given virtual camera.

Set the exposure time for a given virtual camera. This function sets the exposure time for virtual camera `cam` to `exptime` msec. The valid range of the `exptime` parameter is from 8 to 68669153.

Parameters:

`cam`	Virtual camera to set exposure time of.
`exptime`	Exposure time in msec.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIExposeFrame
FLICancelExposure
FLIGetExposureStatus

2.23

int **FLISetFrameType** (flicam_t cam, fliframe_t frametype)

Set the frame type for a given virtual camera.

Set the frame type for a given virtual camera. This function sets the frame type for virtual camera `cam` to `frametype`. The `frametype` parameter is either `FLI_FRAME_TYPE_NORMAL` for a normal frame where the shutter opens or `FLI_FRAME_TYPE_DARK` for a dark frame where the shutter remains closed.

Parameters:

`cam`	Virtual camera to set exposure time of.
`frametype`	Frame type: `FLI_FRAME_TYPE_NORMAL` or`FLI_FRAME_TYPE_DARK`.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

2.24

`int` **`FLISetImageArea`** `(flicam_t cam, int ul_x, int ul_y, int lr_x, int lr_y)`

Set the image area for a given virtual camera.

Set the image area for a given virtual camera. This function sets the image area for virtual camera `cam` to an area specified in terms of a upper-left point and a lower-right point. The upper-left x-coordinate is `ul_x`, the upper-left y-coordinate is `ul_y`, the lower-right x-coordinate is `lr_x`, and the lower-right y-coordinate is `lr_y`. Note that the given lower-right coordinate must take into account the horizontal and vertical bin factor settings, but the upper-left coordinate is absolute. In other words, the lower-right coordinate used to set the image area is a virtual point (lr'_x, lr'_y) determined by:

$$lr'_x = ul_x + (lr_x - ul_x)/hbin$$

$$lr'_y = ul_y + (lr_y - ul_y)/vbin$$

Where (lr'_x, lr'_y) is the coordinate to pass to the `FLISetImageArea` function, (ul_x, ul_y) and (lr_x, lr_y) are the absolute coordinates of the desired image area, *hbin* is the horizontal bin factor, and *vbin* is the vertical bin factor.

Parameters:

`cam`	Virtual camera to set image area of.
`ul_x`	Upper-left x-coordinate of image area.
`ul_y`	Upper-left y-coordinate of image area.
`lr_x`	Lower-right x-coordinate of image area (lr'_x from above).
`lr_y`	Lower-right y-coordinate of image area (lr'_y from above).

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLIGetVisibleArea
FLIGetArrayArea

2.25

```
int FLISetHBin (flicam_t cam, int hbin)
```

Set the horizontal bin factor for a given virtual camera.

Set the horizontal bin factor for a given virtual camera. This function sets the horizontal bin factor for virtual camera `cam` to `hbin`. The valid range of the `hbin` parameter is from 1 to 4095. Note that the horizontal bin factor effects the image area.

Parameters:

`cam`	Virtual camera to set horizontal bin factor of.
`hbin`	Horizontal bin factor.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLISetVBin
FLISetImageArea

2.26

`int` **`FLISetVBin`** `(flicam_t cam, int vbin)`

Set the vertical bin factor for a given virtual camera.

Set the vertical bin factor for a given virtual camera. This function sets the vertical bin factor for virtual camera `cam` to `vbin`. The valid range of the `vbin` parameter is from 1 to 4095. Note that the vertical bin factor effects the image area.

Parameters:

`cam`	Virtual camera to set vertical bin factor of.
`vbin`	Vertical bin factor.

Return Value:

`Zero`	on success.
`Non-zero`	on failure.

See Also: FLISetHBin
FLISetImageArea

2.27

```
int FLISetNFlushes (flicam_t cam, int nflushes)
```

Set the number of flushes for a given virtual camera.

Set the number of flushes for a given virtual camera. This function sets the number of times the CCD array of virtual camera `cam` is flushed *before* exposing a frame to `nflushes`. The valid range of the `nflushes` parameter is from 1 to 4095.

Parameters:	`cam`	Virtual camera to set the number of flushes of.
	`nflushes`	Number of times to flush CCD array before anexposure.
Return Value:	`Zero`	on success.
	`Non-zero`	on failure.

2.28

`int` **`FLISetBitDepth`** `(flicam_t cam, flibitdepth_t bitdepth)`

Set the gray-scale bit depth for a given virtual camera.

Set the gray-scale bit depth for a given virtual camera. This function sets the gray-scale bit depth of virtual camera `cam` to `bitdepth`. The `bitdepth` parameter is either `FLI_MODE_8BIT` for 8-bit mode or `FLI_MODE_16BIT` for 16-bit mode.

Parameters:	`cam`	Virtual camera to set the number of flushes of.
	`bitdepth`	Gray-scale bit depth: `FLI_MODE_8BIT` or`FLI_MODE_16BIT`.
Return Value:	`Zero`	on success.
	`Non-zero`	on failure.

3

Example

```
# if 0

  Copyright (c) 2000, 2002 Finger Lakes Instrumentation (FLI), LLC.
  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

        Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

        Redistributions in binary form must reproduce the above
        copyright notice, this list of conditions and the following
        disclaimer in the documentation and/or other materials
        provided with the distribution.

        Neither the name of Finger Lakes Instrumentation (FLI), LLC
        nor the names of its contributors may be used to endorse or
        promote products derived from this software without specific
        prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
  REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
  ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
  POSSIBILITY OF SUCH DAMAGE.

  ======================================================================

  Finger Lakes Instrumentation (FLI)
  web: http://www.fli-cam.com
```

```
  email: fli@rpa.net

# endif

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "fli.h"

#define DEFAULT_CCD_CHAR_FILE "/dev/ccd0"
#define DEFAULT_EXPOSURE_TIME 50

#define TRY_FUNCTION(function, args...) \
  do { \
    int err; \
    if ((err = function(args))) \
    { \
      fprintf(stderr,\
              "Error " #function ": %s\n", strerror(-err)); \
      exit(1); \
    } \
  } while(0)

void usage(char *name)
{
  printf("\n");
  printf("  Usage: %s [-e <exposure time (msec)>] "
         "[<ccd device filename>] \\\n"
         "         <output file name>\n", name);
  printf("\n");
  printf("    Examples: %s -e 50 /dev/ccd0 testfile.1.out\n", name);
  printf("              %s -e 100 /dev/ccd0 - > testfile.2.out\n", name);
  printf("\n");
  printf("    (An output file name of '-' writes data to standard output.)\n");
  printf("\n");

  exit(0);
}

int main(int argc, char *argv[])
{
  fliport_t port;
  flicam_t cam;
  int ul_x, ul_y, lr_x, lr_y;
```

```
u_int16_t *buff;
int buffsize, bytesgrabbed;
FILE *outfile;
int err;
char *ccd_char_file, *outfilename;
int exptime = DEFAULT_EXPOSURE_TIME;

while (1)
{
  int opt;
  char *endptr;

  if ((opt = getopt(argc, argv, "e:")) == -1)
    break;

  switch (opt)
  {
  case 'e':
    exptime = strtol(optarg, &endptr, 0);
    if (*endptr != '\0')
    {
      printf("Invalid exposure time [%s].\n", optarg);
      usage(argv[0]);
    }
    break;

  case '?':
    usage(argv[0]);
    break;

  default:
    printf("Unknown return value from getopt [%i], exiting.\n", opt);
    exit(1);
  }
}

switch (argc - optind)
{
case 1:
  ccd_char_file = DEFAULT_CCD_CHAR_FILE;
  outfilename = argv[optind];
  break;

case 2:
  ccd_char_file = argv[optind++];
  outfilename = argv[optind];
```

```
    break;

  default:
    printf ("Invalid number of arguments.\n");
    usage(argv[0]);
  }

  TRY_FUNCTION(FLIInit, ccd_char_file, &port);
  TRY_FUNCTION(FLIOpen, port, NULL, &cam);
  TRY_FUNCTION(FLIGetVisibleArea, cam, &ul_x, &ul_y, &lr_x, &lr_y);
  TRY_FUNCTION(FLISetImageArea, cam, ul_x, ul_y, lr_x, lr_y);
  TRY_FUNCTION(FLISetBitDepth, cam, FLI_MODE_16BIT);
  TRY_FUNCTION(FLISetExposureTime, cam, exptime);

  buffsize = (lr_x - ul_x) * (lr_y - ul_y) * sizeof(u_int16_t);

  if ((buff = malloc(buffsize)) == NULL)
  {
    perror("Error malloc");
    exit(1);
  }

  if ((err = FLIGrabFrame(cam, buff, buffsize, &bytesgrabbed)))
    fprintf(stderr, "Error FLIGrabFrame: %s\n", strerror(-err));

  if (bytesgrabbed > 0)
  {
    if (strcmp(outfilename, "-") == 0)
      outfile = stdout;
    else
      if ((outfile = fopen(outfilename, "w")) == NULL)
      {
        perror("Error fopen");
        exit(1);
      }

    if (fwrite(buff, 1, bytesgrabbed, outfile) != bytesgrabbed)
      perror("Error fwrite");
  }

  TRY_FUNCTION(FLIClose, cam);
  TRY_FUNCTION(FLIExit);

  exit(0);
}
```