libcpen_backend

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

API of the proprietary backend library for C-Pen scanner

Author

Andreas Christ software@quantentunnel.de

Date

2017

Version

0.1.3

Copyright

C-Pen is a nice, small hand-held scanner commonly used to scan short lines of text, e.g., to copy quotes from a book into a document or from invoice slips to online banking. Unluckily there is no Linux support by the manufacturer. Reading the USB data is easy, converting the image to text, too. The step in between: decoding a proprietary encoded image. This library attempts to do the latter, based on proprietary code kindly provided by the manufacturer (thus, the library is closed-source).

See also

http://cpen.com

How to use (what *you* need to do)

- Write an application that reads the data from the USB port
  The Linux command 'lsusb -v' provides all required information (endpoint number and type, interface id, vendor and product id, ...). Recommendation: make use of libusb that allows to write the application in user space (i.e., without kernel modules)
- Forward the read data frames to this library (see cpen_scandata)
  This is a non-blocking function. The scanned image is saved to disk and/or available by calling cpen_↩ retrieve_output.
- Convert image to text
  Personal recommendation: use tesseract-ocr.

Error codes

Some functions return a negative value to signal an error or another non-standard condition. The code is 0 minus the returned value, i.e., a code >= 0. 'frame' refers to a single scan obtained from the C-Pen and forwarded to this library. Error codes may be combined, i.e., more than one error might have occured.

Code	Hex	Mnemonic	Meaning
0	0x0000	E_SUCCESS	No error
1	0x0001	E_UNSPECIFIED	Unspecified error
2	0x0002	E_HEADER	Wrong content of header
4	0x0004	E_FRAMEHEADERSHORT	Wrong length of header (too short)
8	0x0008	E_OUTOFMEMORY	Out of memory
16	0x0010	E_NOIMAGE	No image (no call of end-of-scan frame submitted?)
32	0x0020	E_OUTOFIMAGEID	Too many images (more than max integer or than files per directory)
64	0x0040	E_NOOUTPUT	No output available
128	0x0080	E_PATHTOOLONG	File path ot output file too long
256	0x0100	E_NOMATCH	Could not stitch the individual scans together
512	0x0200	E_PIXELFRAMESHORT	Too few pixels in frame
1024	0x0400	E_PIXELHEADERSHORT	Too short image-specific subheader of frame
2048	0x0800	E_IMWRITE	Write-error of OpenCV library
4096	0x1000	E_IMENCODE	Image encoding error of OpenCV library
8192	0x2000	E_FILEERROR	Error when writing file
16384	0x4000	E_NOBITMAP	Could not convert submitted pixels to bitmap

Values $>$ 0xFFFF are not an error but return the button status of the C-Pen (in the two least significant bits)

Attention

these codes may change in minor versions of the library

If the binary (compiled) library going with this header file is not available for your favorite processor architecture (on linux), feel free to drop me a mail. The library requires that OpenCV is installed (i.e., libopencv-core, libopencv-imgproc, libopencv-video, libopencv-stitching)

Chapter 2

Todo List

Member cpen_Scanmode

2D scanning is not yet implemented in the library (and may not be possible due to whitespace frames)

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 cpen_Displacement Struct Reference

Displacement of current single image (cpen_char) to the preceding.

`#include <cpen_backend.h>`

Public Attributes

- unsigned int **microseconds**
- int **x**
- int **y**

5.1.1 Detailed Description

Displacement of current single image (cpen_char) to the preceding.

See also

cpen_Output_flag

The documentation for this struct was generated from the following file:

- cpen_backend.h

Chapter 6

File Documentation

6.1 cpen_backend.h File Reference

Classes

- struct cpen_Displacement

  Displacement of current single image (cpen_char) to the preceding.

Enumerations

- enum cpen_Output_flag {
  cpen_raw = 1, **cpen_char** = 2 ∗ cpen_raw, **cpen_displacement** = 2 ∗ cpen_char, **cpen_word** = 2 ∗ cpen↩
  _displacement,
  cpen_line = 2 ∗ cpen_word, **cpen_resized** = 2 ∗ cpen_line }

  Flags defining the type of output that should be produced.
- enum cpen_Scanmode { **cpen_1D_r2l**, **cpen_1D_l2r**, **cpen_2D** }

  Available scan modes.

Functions

- void cpen_set_signal (sig_atomic_t signal)

  Sets a ∗nix signal (optional)
- void cpen_set_fileoutput (sig_atomic_t outputflags)

  Sets the outputs (images) that should be saved to file.
- void cpen_set_memoryoutput (sig_atomic_t outputflags)

  Sets the outputs (images) to be available for in-memory retrieval.
- void cpen_set_scanmode (enum cpen_Scanmode scanmode)

  Sets the scanmode.
- int cpen_retrieve_output (unsigned char ∗∗data, unsigned int ∗size)

  Pops the oldest library output.
- int cpen_scandata (const unsigned char ∗data, size_t length)

  To feed USB frames (obtained from C-Pen) to the library.
- const char ∗ cpen_getbitmapdirectory (void)

  cpen_getbitmapdirectory returns the directory, where all bitmaps are stored

6.1.1 Enumeration Type Documentation

6.1.1.1 enum cpen_Output_flag

Flags defining the type of output that should be produced.

The output types to be retrieved by calling cpen_retrieveOutput and/or to be saved to disk. Images are in png format; from 'raw' to 'resized' more and more processed. Displacement is a struct cpen_displacement.

Code	Description of the image types	Filename format
raw	image as coming from an single static scan of C-Pen	d-d+d.png
char	despeckled single image (roughly one printed character)	d-d-d.png
word	'char' images stitched together up to the first white space (as reported by C-Pen)	d-d.png
line	'word'-images stitched together to form an entire line, until the C-Pen is lifted	d.png
resized	'line'-image downsampled to a reasonable resolution	scan_d.png
displacement	not an image but the information how two adjacent 'char' images overlap	d-d-d.txt

See also

cpen_set_fileoutput
cpen_set_memoryoutput

6.1.1.2 enum cpen_Scanmode

Available scan modes.

The library cannot know, how the C-Pen is held. If set to cpen_1D_l2r, the library assumes that the pen is hold with the right hand and the scan is from bottom to top (which equals a scan with the left hand from top to bottom). Thus, right-handed persons scanning a western language need to rotate the images clockwise by 90°, e.g. using OpenCV.

mode	Description
cpen_1D_r2l	scan in one direction, from right to left
cpen_1D_l2r	scan in one direction, from left to right
cpen_2D	scan a two-dimensional image

If you wish to implement autodetection of the scan direction, you will need to retrieve the cpen_char images, analyze them and stitch them together.

Todo 2D scanning is not yet implemented in the library (and may not be possible due to whitespace frames)

See also

cpen_set_scanmode

6.1.2 Function Documentation

6.1.2.1 const char* cpen_getbitmapdirectory (void)

cpen_getbitmapdirectory returns the directory, where all bitmaps are stored

The filenames of the images are the image id plus file extension. The image id is returned by the function cpen_↩ scandata(...). All files are located in the directory whose name is returned by cpen_getbitmapdirectory(...).

If the directory is empty when the library gets unloaded (i.e., all bitmap file have been consumed and removed by the calling process or by other means) then the directory is removed when the library gets unloaded.

Returns

Full path of the directory, where output is stored

6.1.2.2 int cpen_retrieve_output (unsigned char ** *data*, unsigned int * *size*)

Pops the oldest library output.

Retrieves one image from the library (FIFO buffer). The caller takes ownership of data. Notably the caller is responsible for free'ing data. Data format: if data would be written to disk, it could be opened as png image.

Returns

an enum cpen_Output_flag describing the type of retrieved bitmap or a negative number if an error occured (e.g., no output is available).

See also

Error_codes

6.1.2.3 int cpen_scandata (const unsigned char * *data*, size_t *length*)

To feed USB frames (obtained from C-Pen) to the library.

Asynchroneus image processing: a complete image becomes soon (!) available after the end-of-scan has been submitted to the library. To detect end-of-scan: wait until the return value is 0 (see table on return values), then poll repeatedly; or request to receive a *nix signal once done. In principle, USB frames of button status need not be submitted to the ibrary with this function, but it also not hurts. The return value will then be a negative number with the button status coded in the least significant bits (2 bits for the 2 buttons of the C-Pen 3.0)

Parameters

frame	Frame as received from the device; library becomes owner
length	Length of the frame received from the device

Returns

see table

Return values

Code	Name
>0	Image id of current scan
0	Scan completed successfully
<0	Error code or button status; see separate table

See also

Error_codes

6.1.2.4 void cpen_set_fileoutput (sig_atomic_t *outputflags*)

Sets the outputs (images) that should be saved to file.

Sets one or several (or'ed) flags from cpen_Outputflag. The corresponding images will be written to the output directory and are numbered. The filenames of cpen_raw and cpen_char contain three numbers (line-word-char), cpen_word two numbers and cpen_line one number.

6.1.2.5 void cpen_set_memoryoutput (sig_atomic_t *outputflags*)

Sets the outputs (images) to be available for in-memory retrieval.

Sets one or several (or'ed) flags from cpen_Outputflag. The corresponding images will become available for retrieval; if set, a *nix signal will be emitted to let the caller now that an image is available for retrieval.

Reporting of errors cannot be masked.

See also

cpen_set_signal
cpen_retrieve_output

6.1.2.6 void cpen_set_scanmode (enum cpen_Scanmode *scanmode*)

Sets the scanmode.

Default of the library is cpen_1D_l2r

Parameters

scanmode	One of the available scan modes

See also

cpen_Scanmode

6.1.2.7 void cpen_set_signal (sig_atomic_t *signal*)

Sets a ∗nix signal (optional)

Sets a signal to be informed when a new image becomes available; set to 0 to stop receiving signals. The application may want to call cpen_retrieveOutput when the signal is received. Values outside of the signal range (0 - 31) are ignored; it might be reasonable to restrict to user signals (SIGUSR1, SIGUSR2).

Index