Camera Driver API

for

"SL-5000D/SL-5500"

(revision 1.00 2002.4.11)

"SL-5000D" and "SL-5500" are part numbers for Sharp Linux™/Java™ based multimedia PDA.

Linux™ is a registered trademark owned by Linus Torvalds.

Java™ and all Java-related marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

---

1.   Overview

This document describes the driver API to utilize Sharp's camera card (P/N: CE-AG06) on the SL-5500, or SL-5000D with romimage 1.12 or higher.
Please note that this document describes functions of the "CameraIO" class libraries as its API specifications, but the actual data transaction with the driver is executed by a read/write access to "/dev/sharp_zdc" device.

2.   API Specifications

2-1. Opening Driver

Format :
      bool CameraIO::Open()

Argument :
      No Need

Return Value :
      true   (success)
      false   (fail)

Function :
      Opens camera device.
      Automatically called by the CameraIO class constructor.
      This API is used to re-open the device in the event that the unit is once turned off etc.

2-2. Closing Driver

Format :
      bool CameraIO::Close()

Argument :
      No Need

Return Value :
      true

Function :
      Closes camera device.
      Automatically called by CameraIO class destructor.
      This API is used to re-open the device in the event that the unit is once turned off, etc.

2-3. Checking Camera Driver I/O Availability

Format :
  bool CameraIO::IoAvalable()

Argument :
  No Need

Return Value :
  true (Opened)
  false (Closed)

Function :
  Checks the camera driver I/O availability (device open status)

2-4. Setting the Capture Image Resolution

Format :
  bool CameraIO::setCaptureFrame(QImage *f, bool r, int w, int h, int z)

Argument :
QImage *f	Buffer to store image
bool r	true is willing to capture the image with 90 degree rotation
int w, h	height and width of the image **(multiplied value of 16)**
int z	digital zoon **(set value multiplied by 256)**

Return Value :
  true (success)
  false (fail)

Function :
  Setting the resolution of the image to be captured.
  Image resolution must be set by the value multiplied by 16.

Notice :
  Note that the camera's CCD or unit LCD is actually rotated by 90 degrees on the SL-5000D/SL-5500.
  If one wishes to pass "through" the camera's image to LCD "AS-IS", one must set **r=false**.
  In case of saving images in the supported image format (such as JPEG), one must set **r=true**.

Information :
  This process is implemented by writing the following String to the device:
    "S=*w*,*h*,*z*,*l*" (r=false: capture camera image as-is)
    "R=*w*,*h*,*z*,*l*" (r=true: capture camera image with 90-degree rotation)
  w,**h**,**z** are the above mentioned Argument in number String, and **l** is to specify number of bytes per one line by number String. (If omitted, driver will calculate with 0 byte gap by **w**)
  Also note that one must write data at one time. (e.g. "R=480,640,256,960")

2-5. Setting Driver Mode

Format :

bool CameraIO::setReadMode(int rm, int cs, int hr, int vr)

Argument :

int rm   Set the read mode (0: image, 1: status, -1: no changes)
int cs   Set the capturing speed (sensitivity) (0: faster (lower sensitivity), 1: lower (higher sensitivity), -1: no changes)
int hr   Set horizontal flip (0: no flip, 1: horizontal flip, -1: no changes)
int vr   Set the vertical flip (0: no flip, 1: vertical flip, -1: no changes）

Return Value :

true

Function :

Set the driver operation mode. Initialized by the CameraIO constructor to rm=1, cs=0, hr=0, vr=0.

Information :

This process is implemented by writing the following String to the device:

"M=x"

x is a number String of the above mentioned Argument in bit field representation.

bit0=rm, bit1=cs, bit2=hr, bit3=vr

Also note that one must write data at one time. (e.g."M=0")

2-6. Start Camera Capturing

Format :

bool CameraIO::CaptureStart()

Argument :

No Need

Return Value :

true

Function :

Start the camera to capture

Information :

This process is implemented by writing following String to the device:

"C"

2-7. Clearing Shutter Latch Flag

Format :
> **bool CameraIO::ShutterLatchClear()**

Argument :
> No Need

Return Value :
> true

Function :
> Clear the latch flag of camera's shutter button

Information :
> This process is implemented by writing following String to the device:
>> "B"

2-8. Obtaining Camera Status

Format :
> **int CameraIO::GetCameraStatus()**

Argument :
> No Need

Return Value :
> Following bit field value will be returned:
> **bit0**   shutter button (latched)
> **bit1**   direction of the camera finder (0: normal, 1: reverse)
> **bit2**   in capturing process
> **bit3**   driver available (1 when camera card is ejected, or unit turned off)

Function :
> Clears the camera's shutter button latch flag.
> When the driver is not available, one must re-open the device by Close/Open, or the camera device will not be available.

Information :
> This process is implemented by reading the data from the driver
> (Driver status set to read mode by StatusCameraIO constructor. See section 2-5.)
> The read data format is as follows (in byte order)
> [0]   'S': shutter pressed 's': shutter not pressed
> [1]   'M': finder directed to reverse side, 'm': finder directed to normal side
> [2]   'C': now capturing, 'c': not capturing
> [3]   'A': driver available, 'a': driver not available

2-9. Obtaining Image

Format :
>   **bool CameraIO::GetPhotoData(QImage *f)**

Argument :
>   **QImage *f**   buffer to store the image

Return Value :
>   **true**   success
>   **false**   fail

Function :
>   Obtains an image.
>   If capturing is not in process, it will also proceed the capturing process.

Information :
>   This process is implemented by reading the data from the driver.
>   (Driver status tentatively is released from the read mode by StatusCameraIO constructor. See section 2-5.)   This read process must be executed at one time.   An error occurs when the specified size is not sufficient for the required number of bytes for reading data.