PMD TECHNOLOGIES

PMD[vision] CamCube

PMDSDK2 Plugins

PROGRAMMING MANUAL

For 3D time-of-flight cameras

Technical information subject to change without notice.
This document may also be changed without notice.
November 2006

Version:	3.0-1.1
Created:	20/Febuary/2009
Changed:	26/April/2010
Author:	Pro

We would like to advise you that all equipment related to the camera demonstrator, including the camera itself, is intended for internal use on an experimental basis only. Please note, that PMDTechnologies has assigned exclusive rights to third party companies for the use of PMDTechnologies systems. Permission must be granted by these companies for any further use or development of these systems, and lack thereof may result in PMDTechnologies being unable to supply further revisions of the equipment in the future.

Table of Contents

1 Introduction

1.1 Description

This manual describes the special commands and properties of the *camcube/camcubeproc* plugins for PMD[vision] CamCube cameras. For further information about software development with the PMDSDK2, consult the *PMDSDK2 Programming Manual*.

There are two plugins. The *camcube* plugin is a PMDSDK2 source plugin, and the camcubeproc plugin is a PMDSDK2 processing plugin.

The plugin files (*camcube.W32.pap* and *camcubeproc.W32.ppp*) are located on the CD-ROM in the *\PMDSDK2\plugins* directory.

1.2 Configuration parameters

The PMD[vision] CamCube supports integration times between 12 and 50000 microseconds. *pmdSetIntegrationTime()* will fail with an error code if out of range integration times are attempted to be set.

Currently, the PMD[vision] CamCube can operate at fixed modulation frequencies of 18 MHz, 19 MHz, 20 MHz and 21 MHz. The modulation frequencies can be set with *pmdSetModulationFrequency()*.

1.3 System requirements

- PMDSDK 2.2.0 for Windows or later

2 Initialization

Both the source plugin and the processing plugin do not need an initialization parameter.

To connect through the PMDSDK2, use a call similar to:

```
res = pmdOpen (&handle,
            “camcube”, “”,
            “camcubeproc”, “”);
```

This will open the first available CamCube connected to the computer.

It is also possible to specify which camera to open by supplying the CamCube’s serial number in the call:

```
res = pmdOpen (&handle,
            “camcube”, “1031.0952”,
            “camcubeproc”, “”);
```

This is only useful if more than one CamCube is connected to the computer.

The pmdOpen call might take while to process. This is due to the calibration of the CamCube.

3 Command Reference

3.1 Source plugin commands

A number of source commands (for use with the *pmdSourceCommand()* function in the PMDSDK2) is available with the *camcube* plugin. These commands modify either the behaviour of the camera or the behaviour of the access plugin itself. Some commands have persistent results, i.e. the behaviour remains intact if the application is restarted (but not when the camera is restarted). Other commands only affect the active connection.

The following source commands are available with this plugin:

GetSerialNumber

Retrieve the serial number of the camera.

This command does not change the camera state.

Example:

```
char serial[16];
pmdSourceCommand (hnd, serial, 16, “GetSerialNumber”);
```

SetSoftOffset

Set an additional distance offset to be used (soft offset). The value will be added to all calculated distance values. This is useful if the light sources are not attached to the camera but placed at a known distance from it.

The offset is given in meters. The format for this command is:

SetSoftOffset <index> <offset>

<index> is the index of the exposure. This is usually 0, except in multi exposure modes.

This command is non-persistent.

Example:

```
pmdSourceCommand (hnd, 0, 0, “SetSoftOffset 0 1.3”);
```

GetSoftOffset

Retrieve the currently set soft offset (see also *SetSoftOffset*). If no soft offset has been set, the result will be “0”. Otherwise, the result will be a string representation of the soft offset in meters. The format for this command is:

GetSoftOffset <index>

<index> is the index of the exposure. This is usually 0, except in multi exposure modes.

This command does not change the camera state.

Example:

```
char offset[16];
pmdSourceCommand (hnd, offset, 16, “GetSoftOffset 0”);
```

SetROI

Set the region of interest.

The camera can be configured to output only a certain area of the image. This has the effect of lowering the readout time of the sensor and therefore increasing the frame rate for smaller image sizes and short integration times. The format for this command is:

SetROI <left> <top> <width> <height>

<left> and <width> have to be divisible by 3 for the CamCube 2.0. This restriction does not apply to the CamCube 3.0.

This command is persistent.

Example: Retrieve a 48x64 array from the middle of the image

```
pmdSourceCommand (hnd, 0, 0, “SetROI 24 42 48 64”);
```

GetROI

Get the region of interest.

The camera can be configured to output only a certain area of the image. This command retrieves the current setting.

This command does not change the camera state.

Example:

```
char result[64];
pmdSourceCommand (hnd, result, 64, “SetROI”);
```

If the ROI had been set using the example from SetROI, the result will be:

```
 “24 42 48 64”
```

SetExposureMode

Set the exposure mode of the CamCube.

The camera can be configured to operate in different exposure modes:

- *Normal*: One exposure with integration time 0 and modulation frequency 0.
- *SMB*: Only available with the CamCube 2.0, not with earlier or later versions. *SMB* stands for *Suppression of Motion Blur* and differs from *Normal* mode in that it minimizes the time needed to capture a distance image from the camera. This results in a significantly reduced amount of motion artefacts. The downside is, that there might be increased levels of noise and that the minimum integration time is higher than in *Normal* mode (depending on the ROI, up to 4 ms). The CamCube 3.0 has replaced this functionality with a sensitivity based approach to limit motion artefacts.
- *DoubleExposure*: Only available with the CamCube 3.0. Two measurements will be made directly following each other, one with integration time 0 and one with integration time 1. The source data size will increase to almost twice the size. Use the processing command *SetFrameSkip* to select which image will be used for the calculation of distances etc.

More complex configurations are possible in conjunction with *SetFrequencyMode*. See *SetFrequencyMode* for more information.

This command is persistent.

Example:

```
pmdSourceCommand (hnd, 0, 0, “SetExposureMode SMB”);

pmdSourceCommand (hnd, 0, 0, “SetExposureMode Normal”);
```

GetExposureMode

Retrieve the current exposure mode (see *SetExposureMode*).

This command does not change the camera state.

Example:

```
char mode[16];
pmdSourceCommand (hnd, serial, 16, “GetExposureMode”);
```

SetFrequencyMode

Set the frequency mode of the CamCube 3.0. Not available on earlier models.

The camera can be configured to operate in different frequency modes:

- *Normal*: Do exposures modulation frequency 0 only. Depending on wether *DoubleExposure* (see *SetExposureMode*) is active, there will be one or two exposures, each with modulation frequency 0.
- *Double:* Use modulation frequencies 0 and 1. In *Normal* exposure mode, there will be two exposures with integration time 0 and 2. In *DoubleExposure* mode, there will be four exposures.

The following states are possible:

Exposure mode / frequency mode	**Integration time indexes**	**Modulation frequency indexes**	**Total number of exposures**
Normal / Normal	I0	F0	1
DoubleExposure / Normal	I0 → I1	F0 → F0	2
Normal / Double	I0 → I2	F0 → F1	2
DoubleExposure / Double	I0 → I1 → I2 → I3	F0 → F0 → F1 → F1	4

This command is persistent.

Example:

```
pmdSourceCommand (hnd, 0, 0, “SetFrequencyMode Double”);

pmdSourceCommand (hnd, 0, 0, “SetFrequencyMode Normal”);
```

GetFrequencyMode

Retrieve the current frequency mode (see *SetFrequencyMode*).

This command does not change the camera state.

Example:

```
char mode[16];
pmdSourceCommand (hnd, serial, 16, "GetFrequencyMode");
```

SetTriggerMode

Set the trigger mode of the CamCube 3.0.

The following modes are available:

- *Freerun* (default): The camera keeps on running. *pmdUpdate()* will retrieve the newest frame. If *pmdUpdate()* is not called quickly enough, frames will be dropped.
- *Soft*: The camera only captures a frame when *pmdUpdate()* is called. There will be no frame drops, but the maximum frame rate is lower than in *Freerun* mode.
- *Hard*: The camera only captures a frame when a hardware trigger is sent. Call *pmdUpdate()* to wait for an image and transfer it to the computer. Use *SetFrameTimeout* to set an appropriate timeout value to ensure that there is enough time for the hardware trigger to arrive.

Note: *Hard* trigger mode does not support multiple exposures/frequencies. When *Hard* is selected, the exposure and frequency modes will be reset to *Normal*/*Normal*.

Note: In *Hard* trigger mode, the camera expects a trigger signal to arrive. If *pmdUpdate()* is called and no trigger signal arrives, it is impossible to switch back to *Freerun* or *Soft* trigger modes. In this case, the camera needs to be power cycled. However, it is possible to switch back as long as *pmdUpdate()* has *not* been called.

Note: After switching to *Hard* trigger mode, the first frame might contain invalid data.

This command is non-persistent.

Example:

```
pmdSourceCommand (hnd, 0, 0, "SetTriggerMode Freerun");
```

GetTriggerMode

Retrieve the current trigger mode (see *SetTriggerMode*).

This command does not change the camera state.

Example:

```
char mode[16];
pmdSourceCommand (hnd, serial, 16, "GetTriggerMode");
```

SetFrameTimeout

Set the timeout for the frame transfer during *pmdUpdate()*.

This is how long the camera waits for data transfers from the camera. This is needed for hardware trigger modes, where it might take longer for the hardware trigger to arrive than the default frame timeout.

This command is non-persistent.

Example: Set a timeout of 10 seconds

```
pmdSourceCommand (hnd, 0, 0, "SetFrameTimeout 10000");
```

SetLensCalibration

Turn the internal lens calibration on or off.

If the lens calibration is on, the calculation of cartesian coordinates will be more accurate. However, this only works with the supplied lens. If a different lens is used, it is advisable to keep this off.

The lens calibration is off by default.

This command is non-persistent.

Example: Turn lens calibration on

```
pmdProcessingCommand (hnd, 0, 0, "SetLensCalibration On");
```

SetFOV

Set the field of view of the camera lens.

The correct field of view setting is important for the calculation of cartesian coordinates. The default value is 40 degrees.

This command if only needed if the lens of the camera changes and only effective if the lens calibration is off (see *SetLensCalibration*).

This command is non-persistent.

Example: Use a lens with 45 degrees FOV

```
pmdProcesingCommand (hnd, 0, 0, "SetFOV 45.0");
```

3.2 Processing plugin commands

The processing plugin *camcubeproc* also supports commands.

The following processing commands are available:

SetFrameSkip

If there are multiple images in the current frame (see *SetExposureMode* and *SetFrequencyMode*), this command selects the image to be used for processing.

For example, if there are four images in the frame (like with exposure mode *DoubleExposure* and frequency mode *Double*), setting a frame skip value of 2 will select the third image (*I2*/*F1*).

This command is non-persistent.

Examples:

```
pmdProcesingCommand (hnd, 0, 0, "SetFrameSkip 0");

pmdProcesingCommand (hnd, 0, 0, "SetFrameSkip 2");
```

SetSaturationCheck

This activates the saturation check during the calculation of the flag image (see *pmdGetFlags*). Saturated pixels will be marked with *PMD_FLAG_SATURATED* and *PMD_FLAG_INVALID*.

This command is non-persistent.

Examples:

```
pmdProcesingCommand (hnd, 0, 0, "SetSaturationCheck On");

pmdProcesingCommand (hnd, 0, 0, "SetSaturationCheck Off");
```

SetConsistencyCheck

This activates the consistency check during the calculation of the flag image (see *pmdGetFlags*). The raw data of the camera will be analysed for inconsistencies, such as motion artefacts. Inconsistent pixels will be marked with *PMD_FLAG_INCONSISTENT* and *PMD_FLAG_INVALID*.

This command is non-persistent.

Examples:

```
pmdProcesingCommand (hnd, 0, 0, “SetConsistencyCheck On”);

pmdProcesingCommand (hnd, 0, 0, “SetConsistencyCheck Off”);
```

SetConsistencyThreshold

Configures the consistency check during the calculation of the flag image (see *SetConsistencyCheck*). Lower values mean higher sensitivity and more flagged pixels. Inconsistent pixels will be marked with *PMD_FLAG_INCONSISTENT* and *PMD_FLAG_INVALID*.

This command is non-persistent.

Example:

```
pmdProcesingCommand (hnd,0,0, “SetConsistencyThreshold 1.0”);
```

PMDTechnologies GmbH
Am Eichenhang 50

57076 Siegen
Germany
Phone +49(0)271 / 238 538- 800
Fax +49(0)271 / 238 538- 809
info@PMDTec.com
www.PMDTec.com