Location: `ECam.h`

Class `CCamera`

```
class CCamera : public CBase;
```

Description

Base class for camera devices.

Provides the interface that an application uses to control, and acquire images from, the camera.

An application must supply an implementation of MCameraObserver2 (or `MCameraObserver`).

Derivation

- `CBase` - Base class for all classes to be instantiated on the heap
  - `CCamera` - Base class for camera devices

Members

Defined in `CCamera`:
`Brightness()`, `BuffersInUse()`, `CameraInfo()`, `CamerasAvailable()`, `CancelCaptureImage()`, `CaptureImage()`, `Contrast()`, `CustomInterface()`, `DigitalZoomFactor()`, `EBrightnessAuto`, `EContrastAuto`, `EExposureAuto`, `EExposureBacklight`, `EExposureCenter`, `EExposureNight`, `EFlashAuto`, `EFlashFillIn`, `EFlashForced`, `EFlashNone`, `EFlashRedEyeReduce`, `EFormat16BitRGB565`, `EFormat16bitRGB444`, `EFormat32BitRGB888`, `EFormatExif`, `EFormatFbsBitmapColor16M`, `EFormatFbsBitmapColor4K`, `EFormatFbsBitmapColor64K`, `EFormatJpeg`, `EFormatMonochrome`, `EFormatUserDefined`, `EFormatYUV420Interleaved`, `EFormatYUV420Planar`, `EFormatYUV420SemiPlanar`, `EFormatYUV422`, `EFormatYUV422Reversed`, `EFormatYUV444`, `EWBAuto`, `EWBCloudy`, `EWBDaylight`, `EWBFlash`, `EWBFluorescent`, `EWBTungsten`, `EnumerateCaptureSizes()`, `EnumerateVideoFrameRates()`, `EnumerateVideoFrameSizes()`, `Exposure()`, `Flash()`, `FrameRate()`, `FramesPerBuffer()`, `GetFrameSize()`, `Handle()`, `JpegQuality()`, `NewDuplicateL()`, `NewL()`, `PowerOff()`, `PowerOn()`, `PrepareImageCaptureL()`, `PrepareImageCaptureL()`, `PrepareVideoCaptureL()`, `PrepareVideoCaptureL()`, `Release()`, `Reserve()`, `SetBrightnessL()`, `SetContrastL()`, `SetDigitalZoomFactorL()`, `SetExposureL()`, `SetFlashL()`, `SetJpegQuality()`, `SetViewFinderMirrorL()`, `SetWhiteBalanceL()`, `SetZoomFactorL()`, `StartVideoCapture()`, `StartViewFinderBitmapsL()`, `StartViewFinderBitmapsL()`, `StartViewFinderDirectL()`, `StartViewFinderDirectL()`, `StopVideoCapture()`, `StopViewFinder()`, `TBrightness`, `TContrast`, `TExposure`, `TFlash`, `TFormat`, `TWhiteBalance`, `VideoCaptureActive()`, `ViewFinderActive()`, `ViewFinderMirror()`, `WhiteBalance()`, `ZoomFactor()`

Inherited from `CBase`:
`operator new()`

Construction and destruction

`NewL()`

```
static IMPORT_C CCamera *NewL(MCameraObserver &aObserver, TInt aCameraIndex);
```

Description

Creates an object representing a camera.

Parameters

`MCameraObserver &aObserver`

Reference to class derived from `MCameraObserver` designed to receive notification of asynchronous event completion.

`TInt aCameraIndex`

Index from 0 to `CamerasAvailable()`-1 inclusive specifying the camera device to use.

Return value

`CCamera *`

Pointer to a fully constructed CCamera object. Ownership is passed to the caller.

Leave codes

`May`

leave with KErrNoMemory or KErrNotSupported if aCameraIndex is out of range.

Member functions

`CamerasAvailable()`

```
static IMPORT_C TInt CamerasAvailable();
```

Description

Determines the number of cameras on the device.

Return value

`TInt`

Count of cameras present on the device.

`NewDuplicateL()`

```
static IMPORT_C CCamera *NewDuplicateL(MCameraObserver &aObserver, TInt
aCameraHandle);
```

Description

Duplicates the original camera object for use by, for example, multimedia systems.

May leave with KErrNoMemory or KErrNotFound if aCameraHandle is not valid.

Parameters

`MCameraObserver &aObserver`
Reference to an observer.

`TInt aCameraHandle`
Handle of an existing camera object.

Return value

`CCamera *`
Duplicate of the original camera object.

CameraInfo()

```
virtual void CameraInfo(TCameraInfo &aInfo) const=0;
```

Description

Gets information about the camera device.

Parameters

`TCameraInfo &aInfo`
On return, information about the camera device. See `TCameraInfo`.

Return value

```
void
```

Reserve()

```
virtual void Reserve()=0;
```

Description

Asynchronous function that performs any required initialisation and reserves the camera for exclusive use.

Calls `MCameraObserver`:: ReserveComplete() when complete.

Return value

```
void
```

Release()

```
virtual void Release()=0;
```

Description

De-initialises the camera, allowing it to be used by other clients.

Return value

```
void
```

PowerOn()

```
virtual void PowerOn()=0;
```

Description

Asynchronous method to switch on camera power.

User must have successfully called Reserve() prior to calling this function.

Calls MCameraObserver::PowerOnComplete() when power on is complete.

Return value

```
void
```

PowerOff()

```
virtual void PowerOff()=0;
```

Description

Synchronous function for switching off camera power.

Return value

```
void
```

Handle()

```
virtual TInt Handle()=0;
```

Description

Gets the device-unique handle of this camera object.

Return value

TInt

The device-unique handle of this camera object.

SetZoomFactorL()

```
virtual void SetZoomFactorL(TInt aZoomFactor=0)=0;
```

Description

Sets the zoom factor.

This must be in the range of `TCameraInfo::iMinZoom` to `TCameraInfo::iMaxZoom` inclusive. May leave with KErrNotSupported if the specified zoom factor is out of range.

Parameters

`TInt aZoomFactor`
Required zoom factor.

Return value

`void`

ZoomFactor()

```
virtual TInt ZoomFactor() const=0;
```

Description

Gets the currently set zoom factor.

Return value

`TInt`
The currently set zoom factor.

SetDigitalZoomFactorL()

```
virtual void SetDigitalZoomFactorL(TInt aDigitalZoomFactor=0)=0;
```

Description

Sets the digital zoom factor.

This must be in the range of 0 to `TCameraInfo::iMaxDigitalZoom` inclusive.

May leave with KErrNotSupported if the zoom factor is out of range.

Parameters

`TInt aDigitalZoomFactor`
The required digital zoom factor.

Return value

`void`

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.