

Introduction to Automating LightField 5 with MATLAB®



Revision History

Issue	Date	List of Changes
Issue 3	August 29, 2018	Issue 3 of this document incorporates the following changes: <ul style="list-style-type: none">• Corrected typos;
Issue 2	January 2, 2018	Issue 2 of this document incorporates the following changes: <ul style="list-style-type: none">• Updated the copyright year.
Issue 1	June 9, 2015	This is the initial release of this document.

©Copyright 2015-2018

Princeton Instruments, a division of Roper Scientific, Inc.
3660 Quakerbridge Rd
Trenton, NJ 08619
TEL: 800-874-9789 / 609-587-9797
FAX: 609-587-1970

All rights reserved. No part of this publication may be reproduced by any means without the written permission of Princeton Instruments, a division of Roper Scientific, Inc. ("Princeton Instruments").

Printed in the United States of America.

Pentium is a registered trademark of Intel Corporation.

MATLAB is a registered trademark of The MathWorks, Inc.

PVCAM is a registered trademark of Photometrics, Ltd. Corporation

Scientific Imaging ToolKit and SITK are trademarks of R Cubed Software Consultants, LLC.

IntelliCal, PICam, SuperSYNCHRO, and SyncMASTER are trademarks and IsoPlane, LightField, and PI-MAX are registered trademarks of Roper Scientific, Inc.

Windows and Windows Vista are registered trademarks of Microsoft Corporation in the United States and/or other countries.

The information in this publication is believed to be accurate as of the publication release date. However, Princeton Instruments does not assume any responsibility for any consequences including any damages resulting from the use thereof. The information contained herein is subject to change without notice. Revision of this publication may be issued to incorporate such change.

Table of Contents

Chapter 1:	Introduction to Automation	5
1.1	Prerequisites	5
Chapter 2:	Prepare LightField	7
2.1	Verify SDK Installation	7
2.2	Create Default Experiment	8
Chapter 3:	Build a Simple Automation Application	9
Chapter 4:	Using LightField Experiment Settings Help File	15
4.1	Build a Sample Automation Application	15

List of Figures

Figure 3-1:	Typical MATLAB® Workspace	10
Figure 3-2:	Newly Available Variables	10
Figure 3-3:	Create an Instance of LightField	11
Figure 3-4:	Data Variable Type Declaration Information	12
Figure 3-5:	Typical Pop-up Plot Dialog	13
Figure 4-1:	Typical LightField Experiment Settings.chm Help File	16
Figure 4-2:	Typical Classes Table	16
Figure 4-3:	CameraSettings Class Information	17
Figure 4-4:	Partial View: CameraSettings Members Table	18
Figure 4-5:	CameraSettings Members: AdcQuality Entry	18
Figure 4-6:	AdcQuality Camera Setting Information	19
Figure 4-7:	AdcQuality Camera Setting Information	19
Figure 4-8:	Initializing LightField: MATLAB® Command Window	20
Figure 4-9:	Configuring ADC Quality: MATLAB® Command Window	20

List of Tables

Revision History	2
----------------------------	---

This page is intentionally blank.

Chapter 1: Introduction to Automation

LightField is able to interface with several third-party applications in order to automate the configuration and performance of experiments within LightField.

Automation is the ability to use a programming environment (e.g., MATLAB[®],) to control LightField externally and performing such tasks as:

- Modifying experiment configuration parameters during such as:
 - Exposure Time
 - Frames to Save
- Connecting/disconnecting hardware
- Displaying live data as it is being acquired
- Importing previously acquired data that have been saved

Automation differs from LightField Add-in in that LightField is actually being controlled by an external program as opposed to running what is, effectively, a LightField subroutine.

This document provides an introduction to developing automation routines using National Instruments' MATLAB[®] programming environment.



NOTE:

This document is not intended to be a tutorial about using MATLAB[®]. It is written with the assumption that the reader possesses a basic knowledge of the MATLAB[®] environment.

This document provides information about the following:

1.1 Prerequisites

In order to automate LightField using MATLAB[®], the following requirements must be satisfied:

- LightField 5 must be installed with the Add-Ins and Automation SDK option included.



NOTE:

For complete information, refer to the installation instructions included with LightField.

- MATLAB[®] 2014 (or later) must be installed.

This page is intentionally blank.

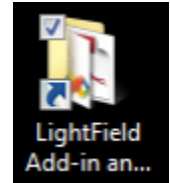
Chapter 2: Prepare LightField

This chapter describes the steps necessary to prepare LightField for automation using MATLAB®.

2.1 Verify SDK Installation

Before developing any automation routines, verify the **Add-Ins and Automation SDK** has been included as part of the LightField installation.

When the SDK has been included, LightField places a shortcut on the host computer desktop. This shortcut points to a directory in which information required by the automation application developer is stored. This information includes:



- LightField Add-ins and Automation Programming Manual.pdf
Provides detailed programming information about the API and dotNET function required to develop an automation application.
- Experiment XML Specification.pdf
This is the specification that defines how a LightField experiment file is structured.
- SPE 3.0 File Format Specification.pdf
This is the specification that defines how LightField saves acquired data.
In order to access data after it has been saved, how it is stored/save must first be understood which is detailed in this document.
- LightField Experiment Settings.chm
This Windows Help File provides information about all settings used when interacting with LightField. The information included in this help file is required when developing automation applications.

If the shortcut is not located on the host computer's desktop, it is a good indication that the SDK has not been included in the current LightField installation. Therefore, uninstall LightField and reinstall it with the custom option to add the **LightField Add-in and Automation SDK** selected.

2.2 Create Default Experiment

It is recommended that a baseline, default LightField experiment be created and saved prior to beginning any automation application development. Doing so will reduce the number of preliminary and initial configuration steps required by the automation application.

Perform the following procedure to create a default experiment that will serve as the basic experiment for automation routines:

1. Launch LightField.
2. Once LightField has finished initializing, verify that all devices are properly installed and have been placed on the Device Grid within the Experiment Workspace.
3. Configure a baseline set of experiment parameters.
These should be appropriate for the specific experiment and application for which an automation application is to be developed.
4. Once the baseline configuration is set, save the experiment with a descriptive name that will be easily recognizable for future use. The name of this file will be required as an input when developing the MATLAB[®] automation application.

Chapter 3: Build a Simple Automation Application

Once a preliminary LightField project template has been created and saved as described in [Chapter 2, Prepare LightField](#), on page 7, development of an automation application can begin.

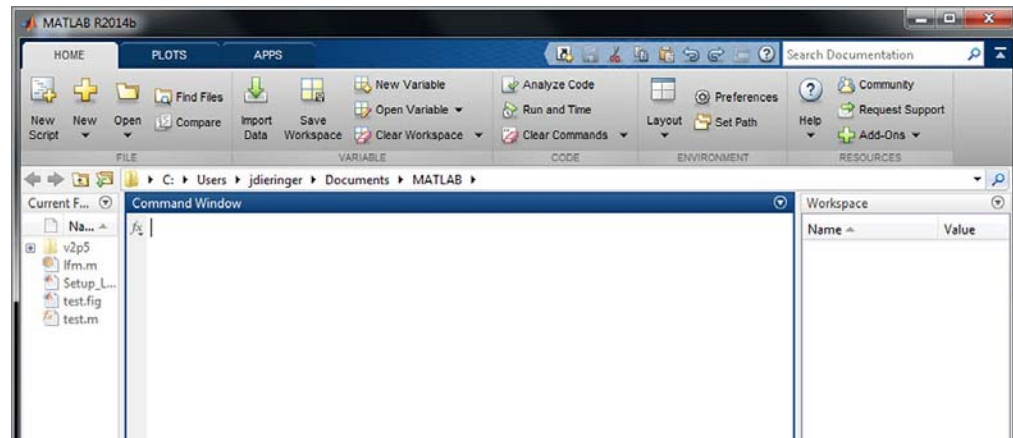


NOTE:


This document is not intended to be a tutorial about using MATLAB[®]. It is written with the assumption that the reader possesses a basic knowledge of the MATLAB[®] environment.

Perform the following procedure to develop a simple automation application that will:

- Launch LightField
 - Modify the experiment exposure time
 - Close LightField.
1. From the host computer's desktop, double-click on the **Add-in and Automation SDK** folder shortcut and then open the **Samples** folder.
 2. Open the MATLAB[®] folder and locate the following two files:
 - `lfm.m`
Contains the MATLAB[®] class for automating LightField.
 - `Setup_LightField_Environment.m`
This is the code necessary for importing the dotNet assemblies and preparing the MATLAB[®] environment to automate LightField.
- Copy each of these files into the working directory for MATLAB[®].
3. Close the **Add-in and Automation SDK** folder.
 4. Launch MATLAB[®]. See [Figure 3-1](#)

Figure 3-1: Typical MATLAB® Workspace

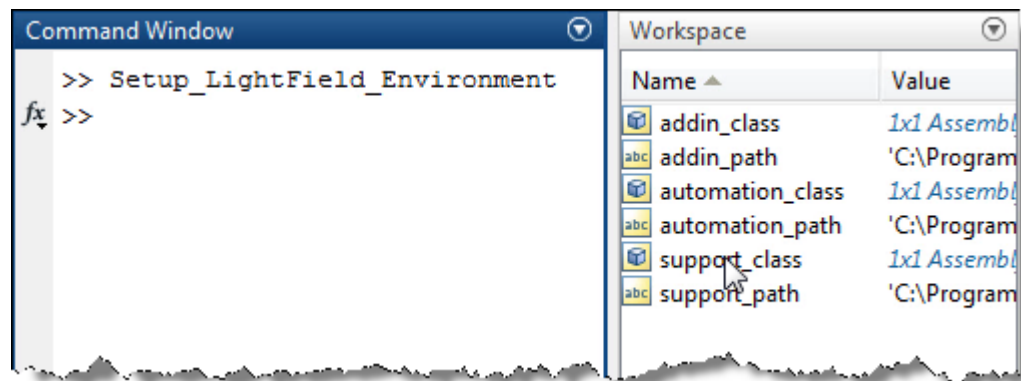
5. Within the **Command Window**, issue the following command:

Setup_LightField_Environment 

Once this setup is complete, a number of variables are created within the Workspace and includes the following information:

- The paths from which the assemblies were loaded;
- Information about the different classes in the dotNet assembly supplied by Princeton Instruments.

See [Figure 3-2](#).

Figure 3-2: Newly Available Variables

6. Issue the following command to create an instance of LightField:

```
instance1 = lfm(visible);
```

where:

- `instance1` is the user-defined variable name assigned to the instance of LightField;
- `visible` indicates whether the instance of LightField will be visible or not.

Valid values are:

— `true`

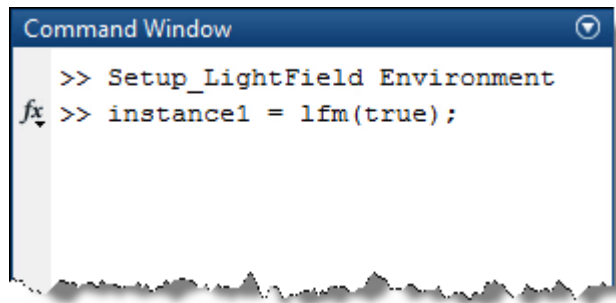
The instance of LightField will be visible.


— false

The instance of LightField will be hidden.

See [Figure 3-3](#).

Figure 3-3: Create an Instance of LightField



7. Press  to launch the LightField instance.
8. Issue the following command to load the sample experiment:

```
instance1.load_experiment('Demo');
```

where `Demo` is the name of the saved sample experiment.

The device will be setup and configured within LightField per the settings saved in the experiment being loaded.

9. Issue the following command to change the Exposure Time to 1500 mS:

```
instance1.set_exposure(1500);
```

10. Issue the following command to change the number of frames to save to 10:

```
instance1.set_frames(10);
```

11. To acquire data, issue the following command:

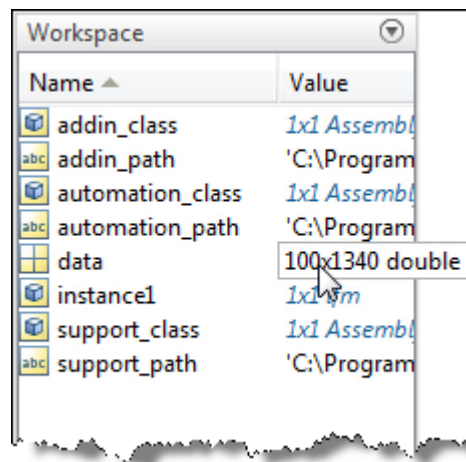
```
data = instance1.acquire;
```

where:

- `data` is a user-defined variable in which the acquired data are to be stored after it has been transferred from LightField into MATLAB®.

When acquiring data for a LightField frame that measures 1340 x 100 pixels, the variable `data` is added to the workspace column where it is declared as an array of size and type **100x1340 double** which corresponds to the size of the acquired frame. See [Figure 3-4](#).

Figure 3-4: Data Variable Type Declaration Information



12. Issue the following command to plot this acquired data:

```
pcolor(data); shading interp;
```



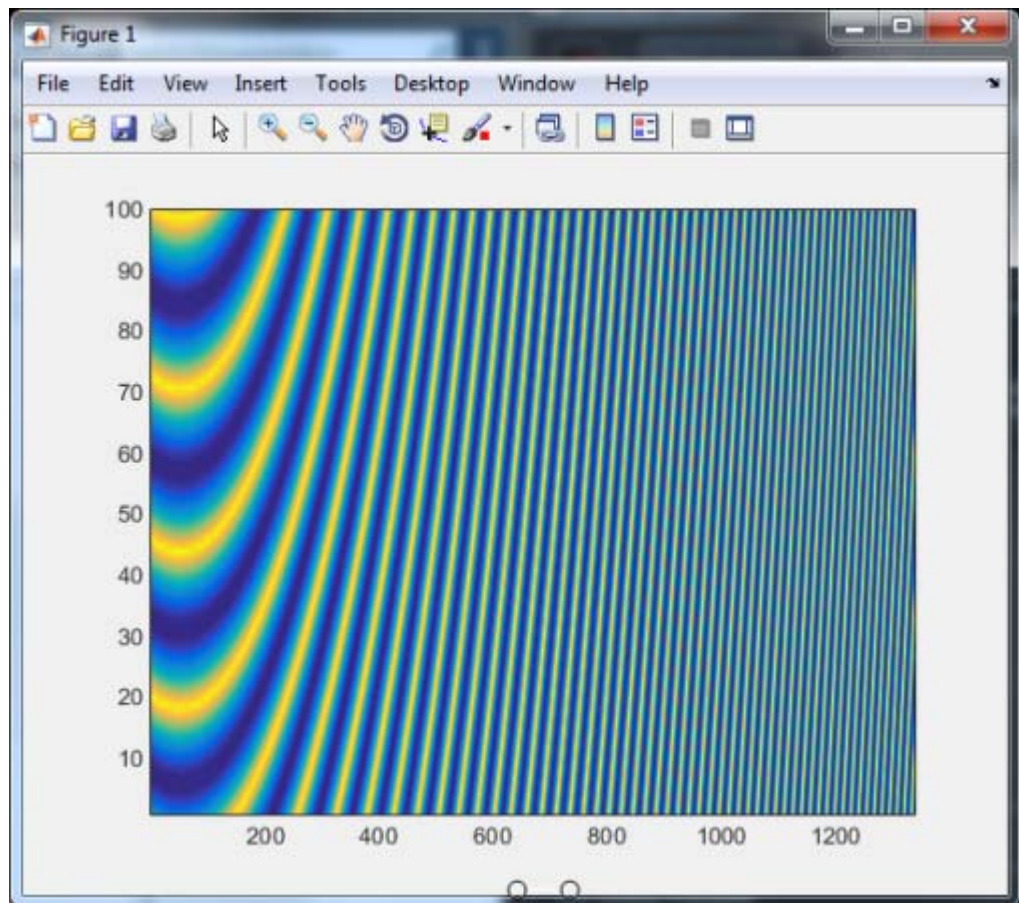
Figure 3-5 illustrates the output of this command.



NOTE:

The displayed output dialog is resizable. When the window is resized, the data displayed within it will be appropriately scaled to reflect the new window size.

Figure 3-5: Typical Pop-up Plot Dialog



When displaying data, MATLAB® determines if there is a single Region of Interest (ROI) or multiple ROIs within the data being displayed.

- If the data comprise a single ROI, the MATLAB® library creates a standard array (e.g., the **100x1340 double** data object in step 11,) that is appropriately sized to store the data being received from LightField.
- When there are multiple ROIs, the data object (refer to step 11) is then declared as a cell array.

For example, when two ROIs are defined and data are acquired, the data object is then declared as a **2x1 cell** array.

When displaying data comprised of multiple ROIs, the command to plot the data must be modified. Issue the following command to display data comprised of multiple ROIs:

```
pcolor(data{#}); shading interp;
```



where:

- # represents the ROI number to be plotted (e.g., 1 for **ROI 1**, 2 for **ROI 2**, etc.)

13. Once all updates and data collection/display is complete, issue the following command to close LightField:

```
instance1.close;
```



Chapter 4: Using LightField Experiment Settings Help File

Information necessary to configure LightField settings using MATLAB® Command Line instructions is included in the **LightField Experiment Settings.chm** help file found in the **Add-in and Automation SDK** folder.

The help file includes the following Classes available for configuration updates via MATLAB®:

- CameraSettings
- ExperimentSettings
- SpectrometerSettings

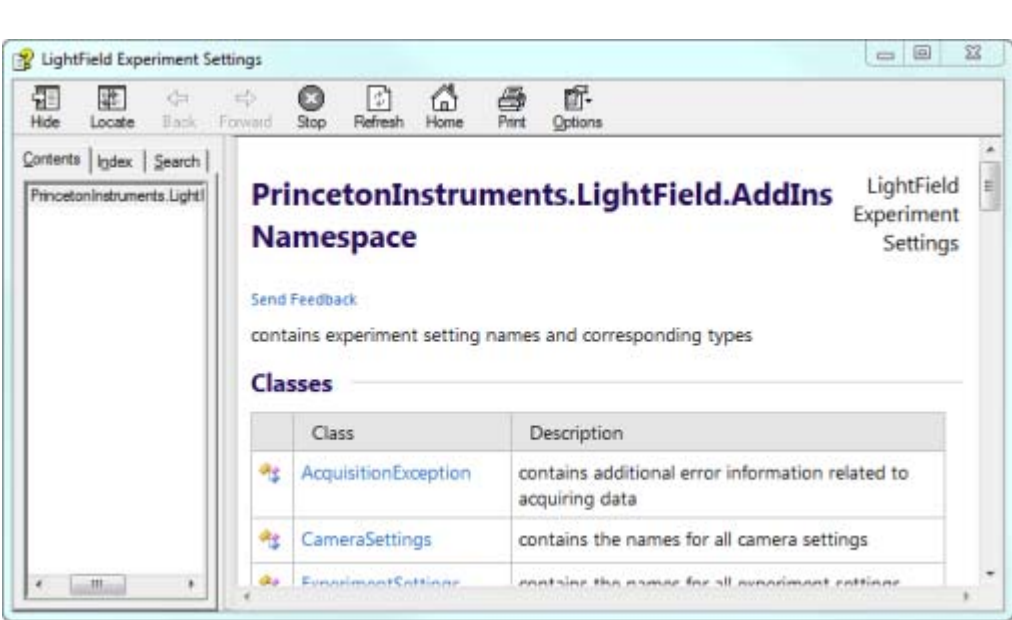
This chapter describes how to use this information when creating a MATLAB® automation application.

4.1 Build a Sample Automation Application

Perform the following procedure to use MATLAB® to change the **Analog to Digital Conversion Quality** from Low Noise to High Capacity.

1. Verify that MATLAB® and LightField have both been launched and are running. If necessary, launch each of them.
2. If necessary, open the **Add-in and Automation SDK** folder using the shortcut on the host computer's desktop.
3. Locate the file named **LightField Experiment Settings.chm** and double-click on it to open the Help file. See [Figure 4-1](#).







Figure 4-1: Typical LightField Experiment Settings.chm Help File



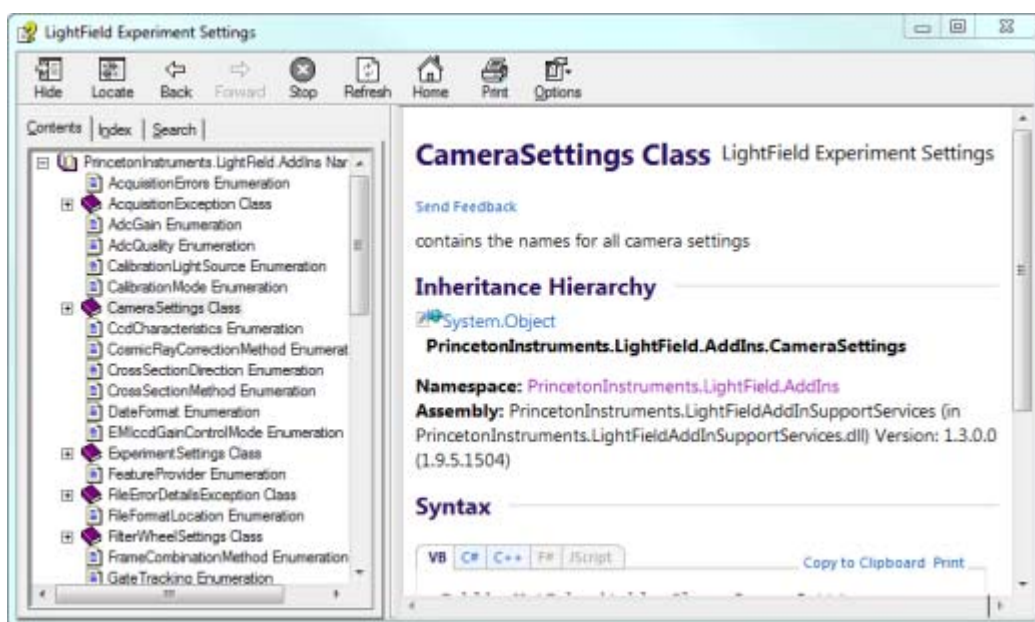
4. The ADC Quality setting is a Camera setting. Within the Classes table, click on CameraSettings. See Figure 4-2.

Figure 4-2: Typical Classes Table

Classes

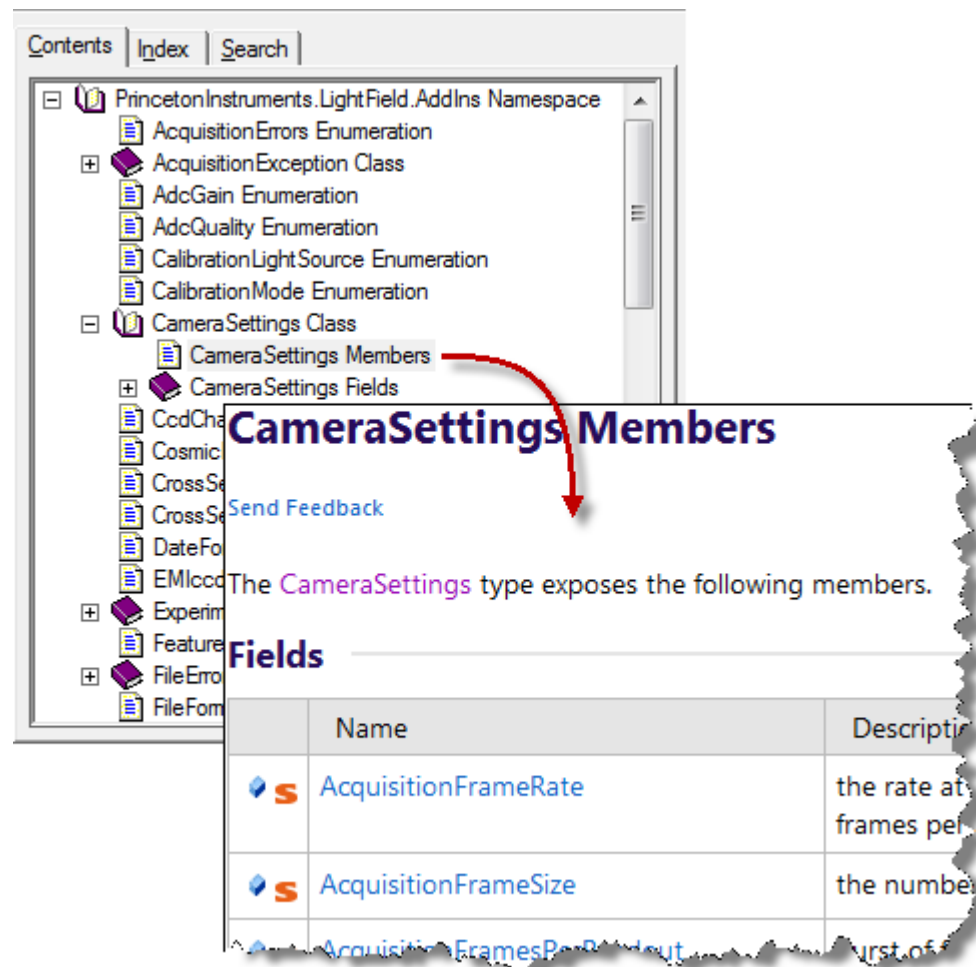
	Class	Description
	AcquisitionException	contains additional error information related to acquiring data
	CameraSettings	contains the names for all camera settings
	ExperimentSettings	contains the names for all experiment settings
	FileErrorDetailsException	contains additional error information located in a file
	FilterWheelSettings	contains the names for all filter wheel settings
	SpectrometerSettings	contains the names for all spectrometer settings

Information about the CameraSettings class is displayed in the Help window primary frame and the Contents tab expands. See Figure 4-3.

Figure 4-3: CameraSettings Class Information

5. Within the **Contents** tab, expand the `CameraSettings` Class book and click on the `CameraSettings` Members entry to view the members within the primary help panel. [Figure 4-4](#) illustrates a partial view of the `CameraSettings` Members table.

Figure 4-4: Partial View: CameraSettings Members Table



6. Scroll the `CameraSettings` Members table until `AdcQuality` is visible. See [Figure 4-5](#).

Figure 4-5: CameraSettings Members: AdcQuality Entry

	<code>AdcAnalogGain</code>	amplifies the output of the sensor via electron multiplication as a multiplier
	<code>AdcGain</code>	Obsolete. Use <code>AdcAnalogGain</code> instead.
	<code>AdcQuality</code>	the characteristics of the analog to digital conversion
	<code>AdcSpeed</code>	the speed in which data from the sensor is digitized in megahertz
	<code>HardwareIOAuxOutput</code>	controls the pulse timing from t_0

7. Click on `AdcQuality` to view information about this camera setting in the primary help panel. See [Figure 4-6](#).

Figure 4-6: AdcQuality Camera Setting Information

LightField
Experiment
Settings

CameraSettings.AdcQuality Field

[Send Feedback](#)

the characteristics of the analog to digital conversion

Namespace: `PrincetonInstruments.LightField.AddIns`
Assembly: `PrincetonInstruments.LightFieldAddInSupportServices`
 (in `PrincetonInstruments.LightFieldAddInSupportServices.dll`)
 Version: 1.3.0.0 (1.9.5.1504)

Syntax

[VB](#)
[C#](#)
[C++](#)
[F#](#)
[JScript](#)

[Copy to Clipboard](#)
[Print](#)

```
Public Const AdcQuality As String
```

Remarks

Type: collection of [AdcQuality](#)

4411-0151_011

8. Within the **Remarks** section, click on the [AdcQuality](#) link to view the set of valid Enumeration values/members. See [Figure 4-7](#).

Figure 4-7: AdcQuality Camera Setting Information

```
Public Const AdcQuality As String
```

Remarks

Type: collection of [AdcQuality](#)

Syntax

[VB](#)
[C#](#)
[C++](#)
[F#](#)
[JScript](#)

[Copy to Clipboard](#)
[Print](#)

```
Public Enumeration AdcQuality
```



Members

	Member name	Value	Description
	LowNoise	1	gives the lowest noise
	HighCapacity	2	can store more light
	HighSpeed	4	digitizes at faster speeds
	ElectronMultiplied	3	performs electron multiplication to boost signal

4411-0151_012

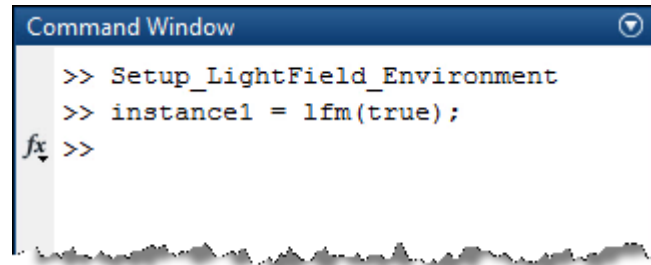
All information required to configure `AdcQuality` using MATLAB® is now known.

9. Within the MATLAB® **Command Window**, issue the following commands to initialize and launch LightField:

```
Setup_LightField_Environment   
  
instance1 = lfm(visible); 
```


See [Figure 4-8](#).

Figure 4-8: Initializing LightField: MATLAB® Command Window



4411-0151_013

10. Next, issue the following command to change LightField's **Analog to Digital Conversion Quality** setting from **Low Noise** to **High Capacity**:

```
instance1.set (CameraSettings.AdcQuality, AdcQuality.  
HighCapacity) 
```

See [Figure 4-9](#).

Figure 4-9: Configuring ADC Quality: MATLAB® Command Window



4411-0151_014

11. To change ADC Quality to any other configuration value is achieved by replacing `HighCapacity` in this command statement with one of the other valid/supported values:

- `LowNoise`
- `HighSpeed`
- `ElectronMultiplied`