TELEDYNE PRINCETON INSTRUMENTS
Everywhereyoulook™

Part of the Teledyne Imaging Group

Introduction to Automating LightField 5 with MATLAB®

4411-0151
Issue 4
January 7, 2020

www.princetoninstruments.com

Revision History

Issue	Date	List of Changes
Issue 4	January 7, 2020	Issue 4 of this document incorporates the following changes: • Updated the copyright year.
Issue 3	July 24, 2019	Issue 3 of this document incorporates the following changes: • Converted to standard FrameMaker template.
Issue 2	January 2, 2018	Issue 2 of this document incorporates the following changes: • Updated the copyright year.
Issue 1	June 9, 2015	This is the initial release of this document.

Teledyne Princeton Instruments
3660 Quakerbridge Rd
Trenton, NJ 08619
TEL: 800-874-9789 / 609-587-9797

Table of Contents

List of Figures

List of Tables

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.

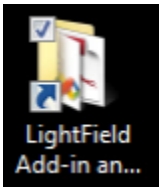
This page is intentionally blank.

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.

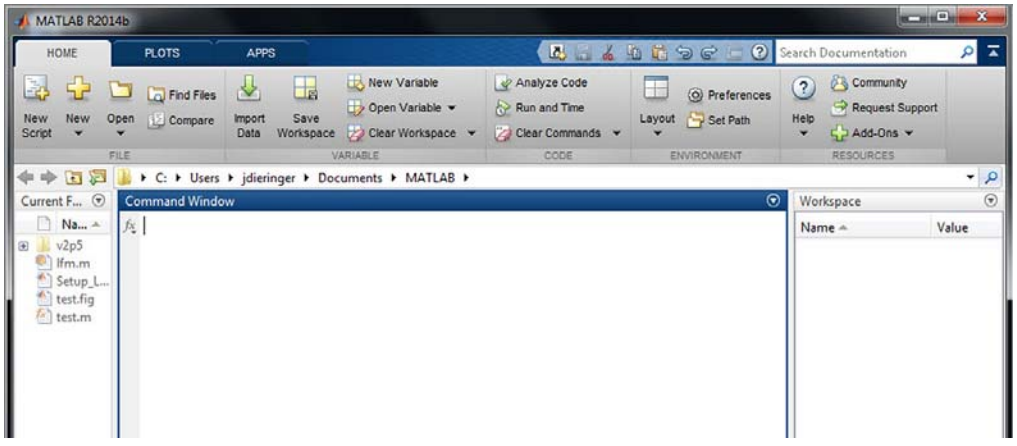
Perform the following procedure to develop a simple automation application that will:

- 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



4411-0151_001

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

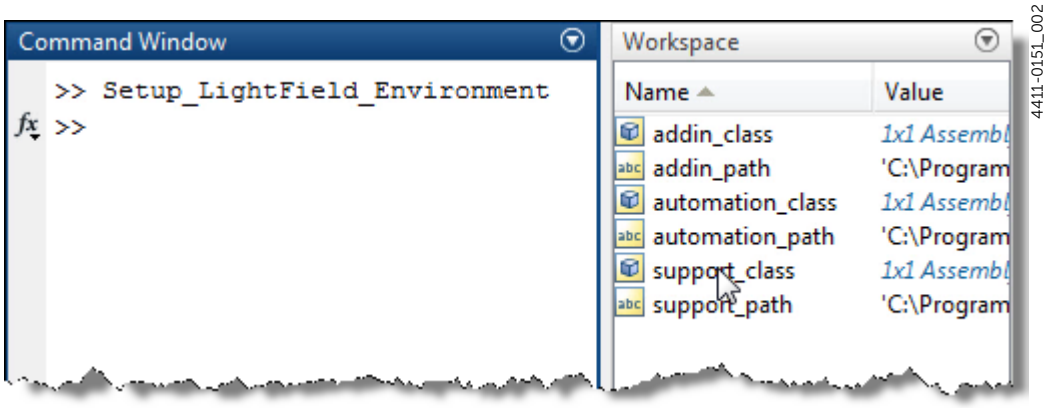
```
Setup_LightField_Environment
```
Enter

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:

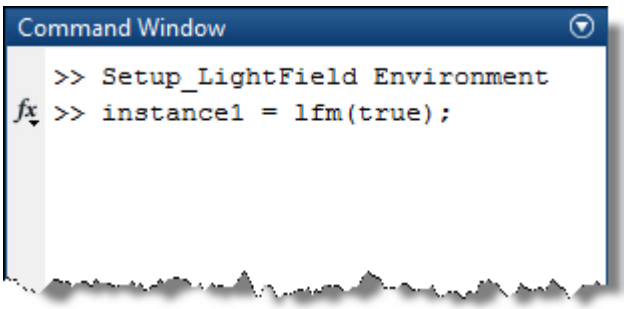
```
instance1 = lfm(visible);
```

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 Enter to launch the LightField instance.

8. Issue the following command to load the sample experiment:

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

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);
```
Enter

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

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

11. To acquire data, issue the following command:

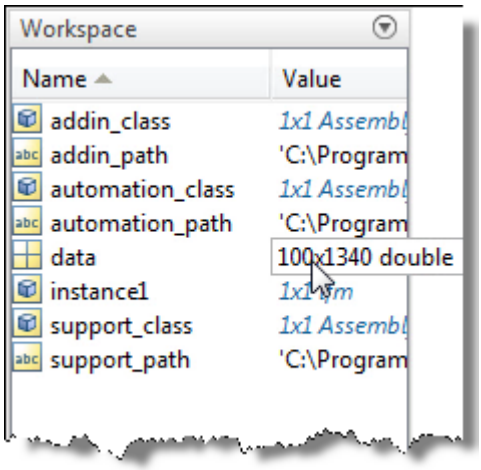
```
data = instance1.acquire;
```
Enter

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



4411-0151_004

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

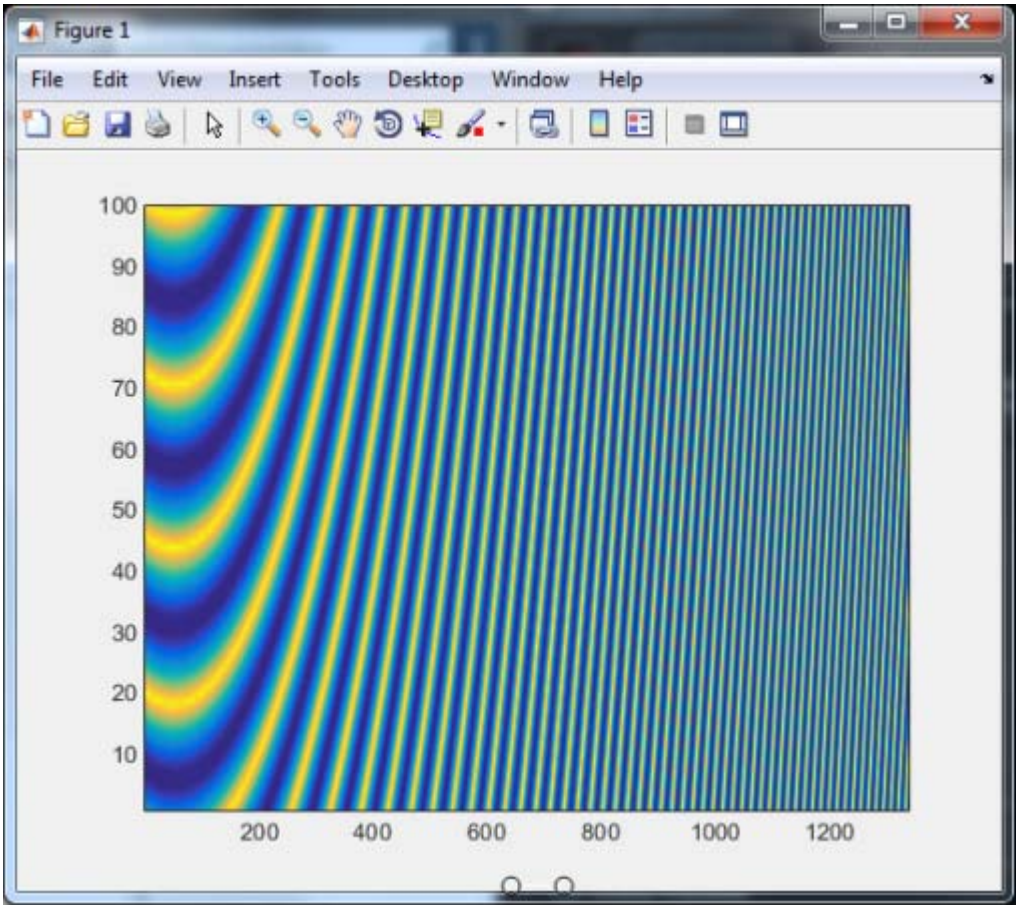
```
pcolor(data); shading interp;
```
Enter

Figure 3-5 illustrates the output of this command.

NOTE:

The displayed output dialog is able to be resized. 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;
```
Enter

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;
```
Enter

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`

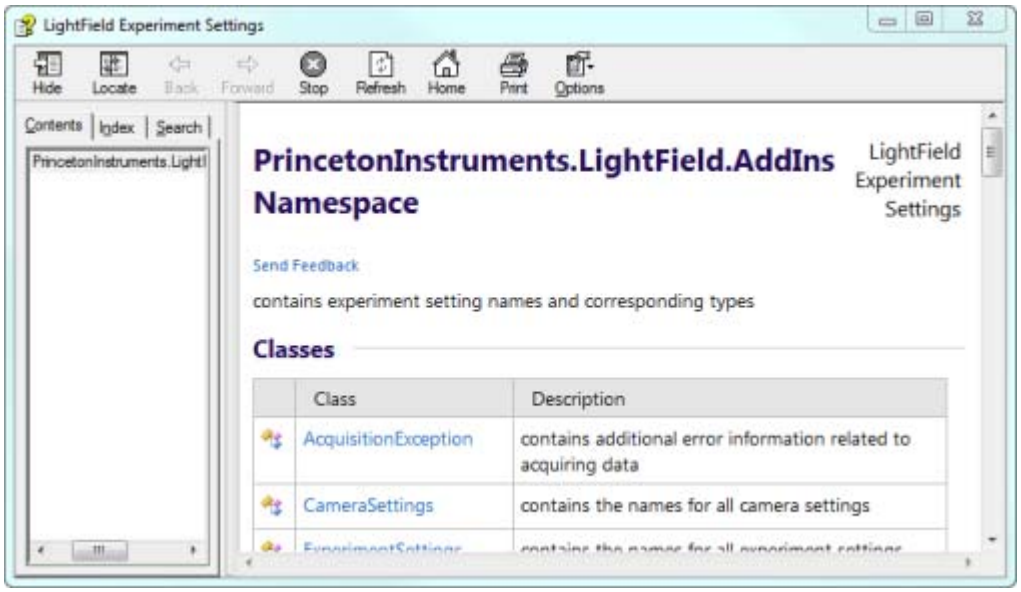
This chapter describes how to use this information when creating a MATLAB automation application.

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.

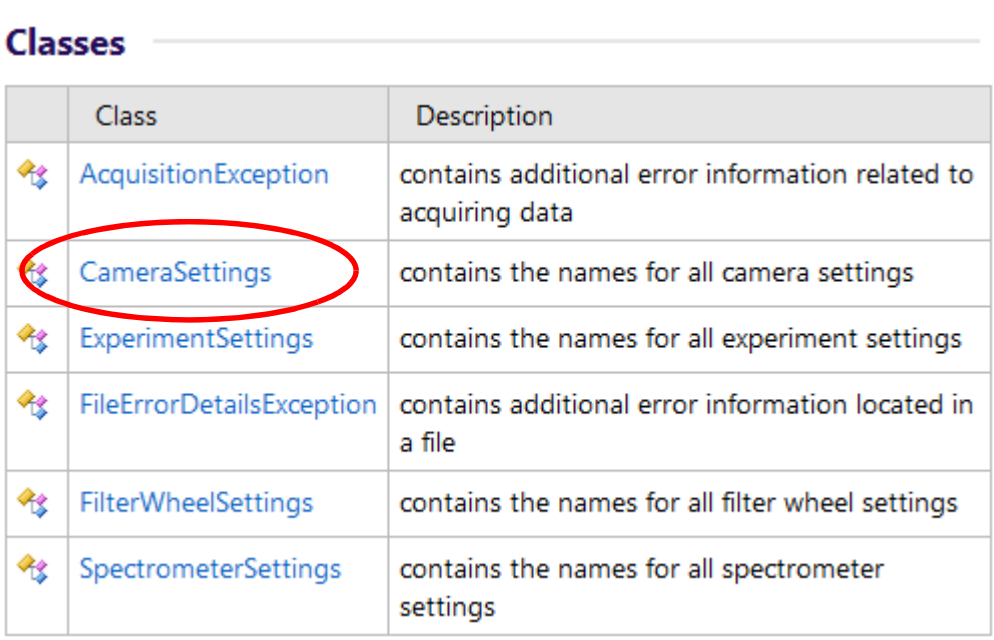
Figure 4-1: Typical LightField Experiment Settings.chm Help File



4411-0151_006

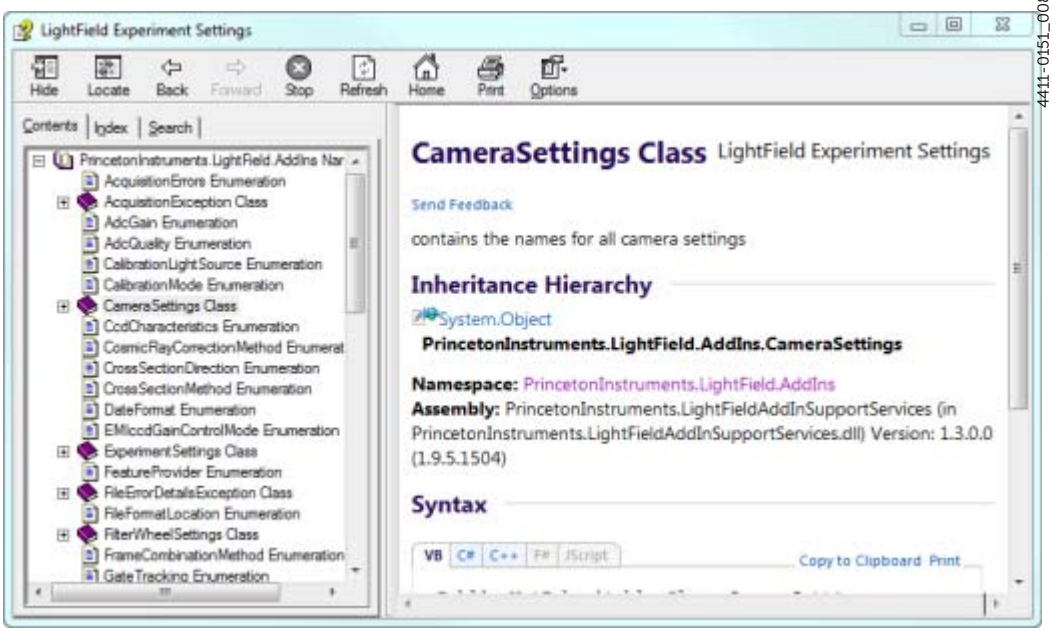
4. The `ADC Quality` setting is a Camera setting. Within the Classes table, click on `CameraSettings`. See Figure 4-2.

Figure 4-2: Typical Classes Table



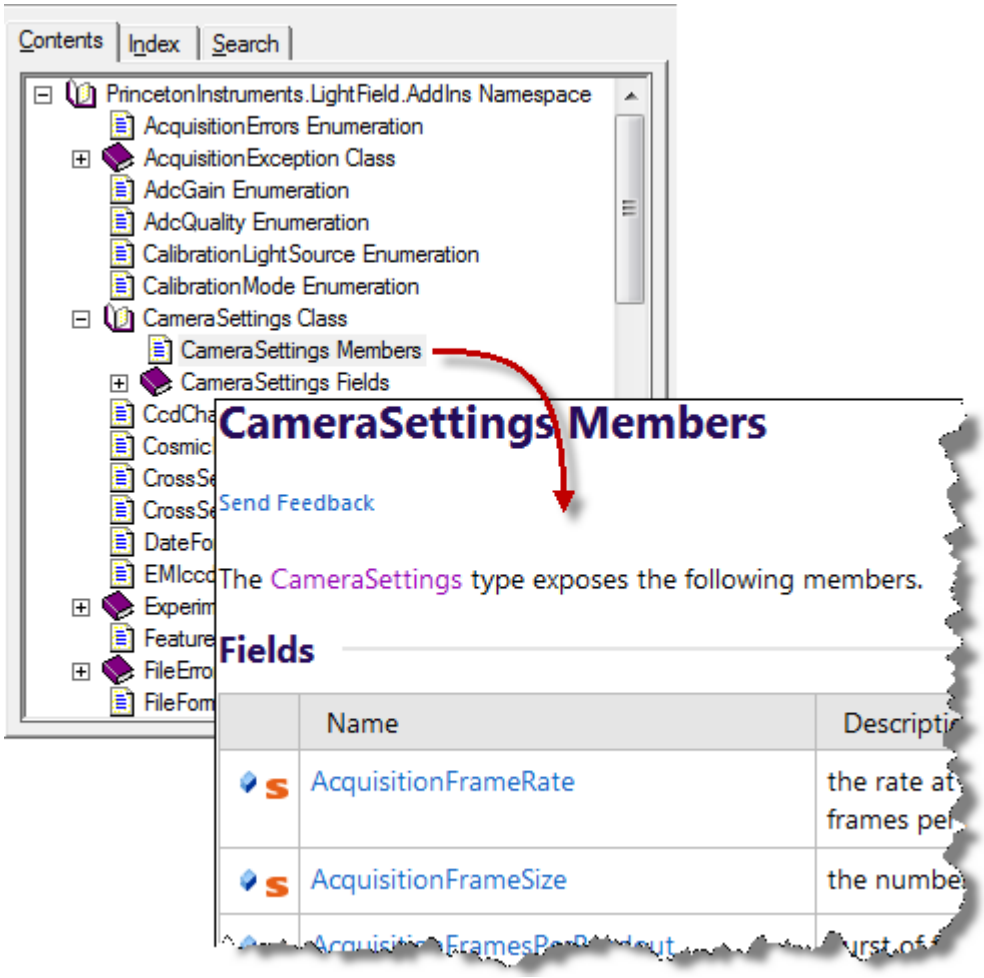
Information about the `CameraSettings` classs is displayed in the Help window primary frame and the Contents tab expands. See Figure 4-3.

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.

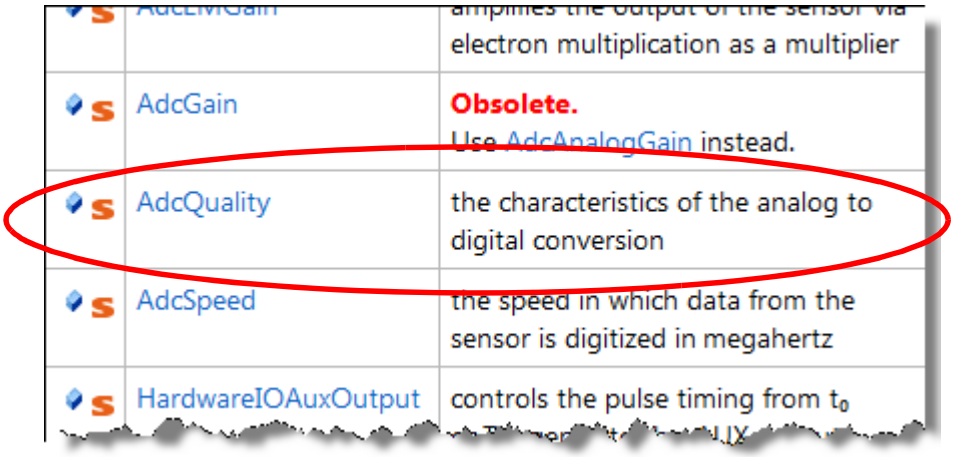
Figure 4-4: Partial View: CameraSettings Members Table



4411-0151_009

6. Scroll the `CameraSettings Members` table until `AdcQuality` is visible. See Figure 4-5.

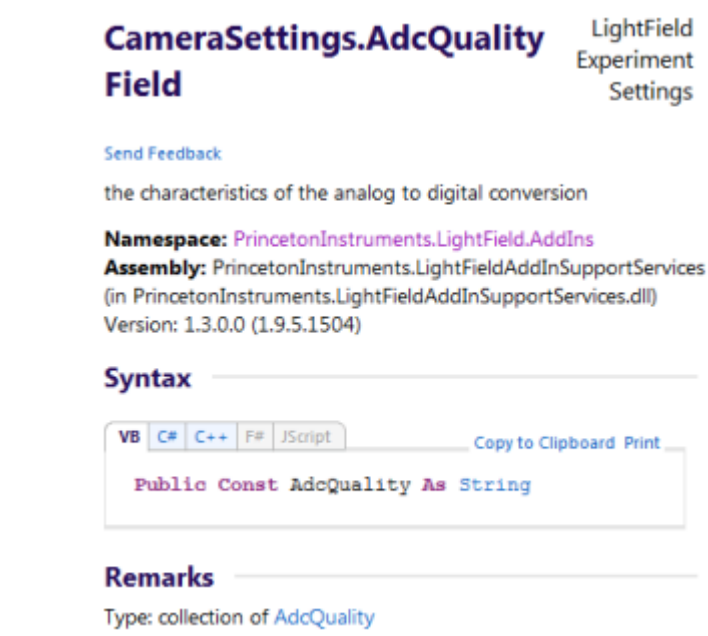
Figure 4-5: CameraSettings Members: AdcQuality Entry



4411-0151_010

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

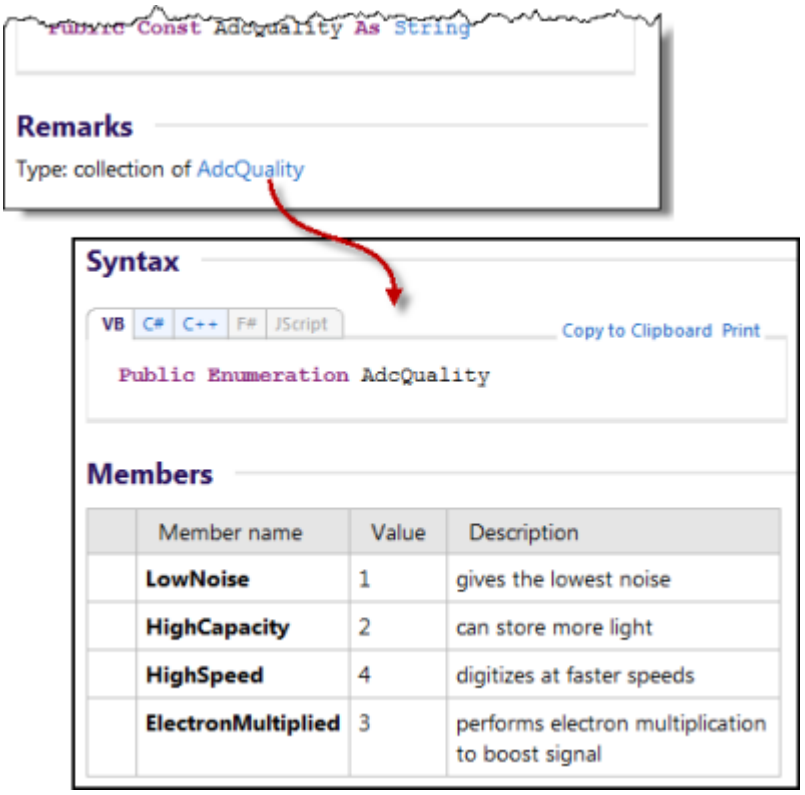
Figure 4-6: AdcQuality Camera Setting Information



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



4411-0151_012

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

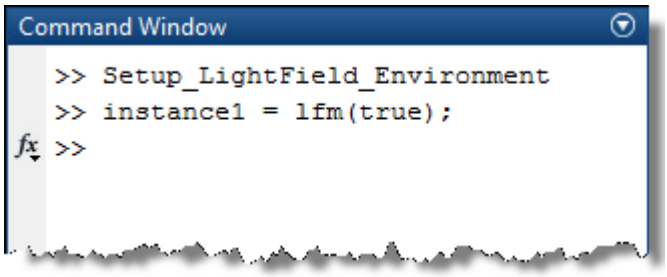
9. Within the MATLAB **Command Window**, issue the following commands to initialize and launch LightField:

```
Setup_LightField_Environment
```
Enter

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

See Figure 4-8.

Figure 4-8: Initializing LightField: MATLAB Command Window



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)
```
Enter

See Figure 4-9.

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

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`

This page is intentionally blank.

This page is intentionally blank.

www.princetoninstruments.com

TELEDYNE PRINCETON INSTRUMENTS
Everywhereyoulook™
Part of the Teledyne Imaging Group

info@princetoninstruments.com
USA +1 877-474-2286 | **France** +33 (1) 60 86 03 65 | **Germany** +49 (0) 89 660 7793 | **UK & Ireland** +44 (0) 1628 472 346
Singapore +65 6408 6240 | **China** +86 10 659 16460 | **Japan** +81 (3) 5639 2741