DCC and DCU Cameras

MATLAB Interface Guide

Table of Contents

Chapter 1 Warning Symbol Definitions

Below is a list of warning symbols you may encounter in this manual or on your device.

Symbol	**Description**
⎓	Direct Current
∼	Alternating Current
≂	Both Direct and Alternating Current
⏚	Earth Ground Terminal
⏚	Protective Conductor Terminal
⊥	Frame or Chassis Terminal
▽	Equipotentiality
|	On (Supply)
○	Off (Supply)
▭	In Position of a Bi-Stable Push Control
▯	Out Position of a Bi-Stable Push Control
⚠	Caution: Risk of Electric Shock
⚠	Caution: Hot Surface
⚠	Caution: Risk of Danger
⚠	Warning: Laser Radiation
⚠	Caution: Spinning Blades May Cause Harm

Chapter 2 Safety

Precautions of a general nature should be gathered here. Wherever possible, however, safety warnings, cautions, and notes should only appear immediately before the instructions to which they apply (versus being listed in this section).

SHOCK WARNING

Warning is given when there is danger of injury to users.

CAUTION

Caution is given when there is a possibility of damage to the product.

WARNING

Given when there is danger of injury to users.

Use either exclamation points or laser warning if laser.

Chapter 3 Preface

This is a brief overview of how to get started making a custom MATLAB script to communicate with a Thorlabs DCC or DCU USB camera.

The example script is for reference only; the user is encouraged to extend or modify the script to fit their specific needs.

This guide and script was written using the following camera model andsoftware versions.

Camera: DCU224C

ThorCam: Version 2.6.7064

uc480 Driver: Version 4.60.7

uc480DotNet: Version 1.6.1.0

MATLAB: Version R2014a 64-bit

The basic steps will be similar for all DCC or DCU cameras. Functionality and procedures might vary when using other software versions.

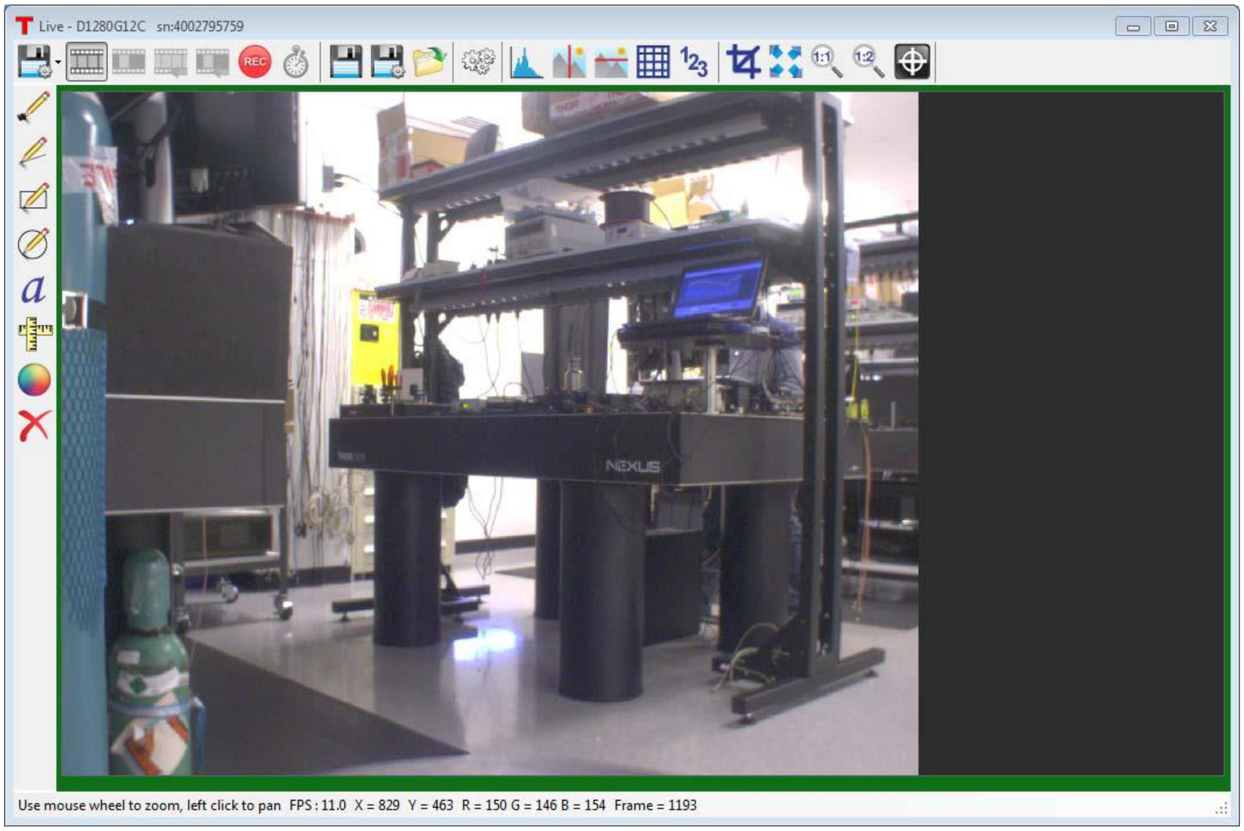
Chapter 4 Step-by-Step instructions

1. Download and install the software for the ThorCam Software for Scientific and USB Cameras located on the Software tab here:

   http://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

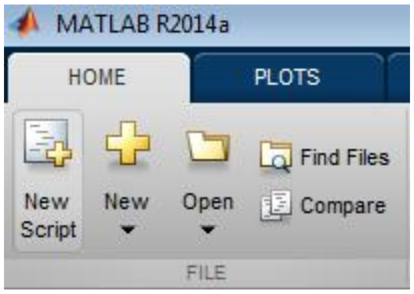
   Be sure to select the USB camera option, since this installs all support for the DCx line of cameras.

2. Run the Thorlabs ThorCam software to verify that your instrument is working with the computer correctly and become familiar with the operation of the device.

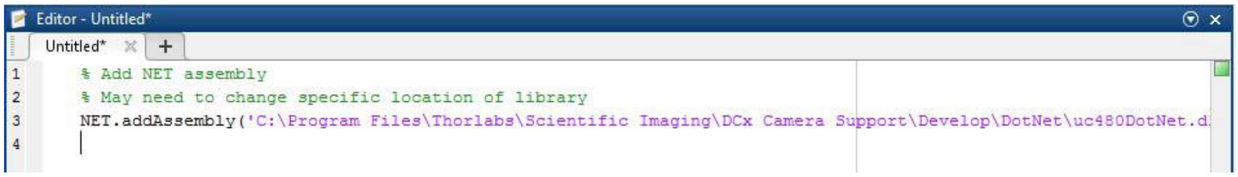


Use mouse wheel to zoom, left click to pan FPS : 11.0 X = 829 Y = 463 R = 150 G = 146 B = 154 Frame = 1193

3. Open MATLAB and start a new script.



4. Make the uc480DotNet assembly visible to MATLAB. If you are using 64 bit MATLAB you will need to use the 64 bit uc480DotNet.dll located in C:\Program Files\Thorlabs\Scientific Imaging\DCx Camera Support\Develop\DotNet (on a 64 bit machine). If you are using 32 bit MATLAB on a 64-bit machine, you will need to use the 32 bit uc480DotNet.dll located in C:\Program Files (x86)\Thorlabs\Scientific Imaging\DCx Camera Support\Develop\DotNet



```
% Add NET assembly
% May need to change specific location of library
NET.addAssembly('C:\Program Files\Thorlabs\Scientific Imaging\DCx Camera Support\Develop\DotNet\uc480DotNet.d
```

5. Create a camera object handle.

```
% Create camera object handle
cam = uc480.Camera;
```

6. Open a camera. The Init method parameter is an integer which indicates which camera to open. If it is 0 the first available camera will be opened. The integer can also be the Cam.ID from the DCx Camera Manager if a particular camera would like to be used.

```
% Open the 1st available camera
cam.Init(0);
```

7. Set the display mode to bitmap (DiB). The Display.Mode.Set method parameter is a uc480.Defines.DisplayMode enumeration which defines the display mode.

```
% Set display mode to bitmap (DiB)
cam.Display.Mode.Set(uc480.Defines.DisplayMode.DiB);
```

8. Set the color mode to 8-bit RGB. The PixelFormat.Set method parameter is a uc480.Defines.ColorMode enumeration which defines the color mode.

```
% Set color mode to 8-bit RGB
cam.PixelFormat.Set(uc480.Defines.ColorMode.RGBA8Packed);
```

9. Set the trigger mode to software so we can take a single acquisition. The Trigger.Set method parameter is a uc480.Defines.TriggerMode enumeration which defines the trigger mode.

```
% Set trigger mode to software (single image acquisition)
cam.Trigger.Set(uc480.Defines.TriggerMode.Software);
```

10. Allocate memory for the camera image. The Memory.Allocate method parameter is a Boolean which indicates if the created memory is active. The method returns a status indicator (not used in the example) and the memory id as an integer.

```
% Allocate image memory
[~, MemId] = cam.Memory.Allocate(true);
```

11. Obtain the image width, height and bit width. The Memory.Inquire method parameter is the memory id as an integer. The method returns a status indicator (not used in the example), the image width as an integer, the image height as an integer, the image bit width as an integer and the line increment of the image memory (not used in example).

```
% Obtain image information
[~, Width, Height, Bits, ~] = cam.Memory.Inquire(MemId);
```

12. Acquire an image from the camera. The Acquisition.Freeze method parameter is a uc480.Defines.DeviceParameter enumeration which defines if the program should wait or not wait for the image to be acquired.

```
% Acquire image
cam.Acquisition.Freeze(uc480.Defines.DeviceParameter.Wait);
```

13. Copy the image data from the memory buffer to an array. The Memory.CopyToArray method parameter is the memory id as an integer. It returns a status indicator (not used in the example) and the image data as an array of System.Byte.

```
% Copy image from memory
[~, tmp] = cam.Memory.CopyToArray(MemId);
```

14. Reshape the image array to a three dimensional array of RGB images.

```
% Reshape image
Data = reshape(uint8(tmp), [Bits/8, Width, Height]);
Data = Data(1:3, 1:Width, 1:Height);
Data = permute(Data, [3,2,1]);
```

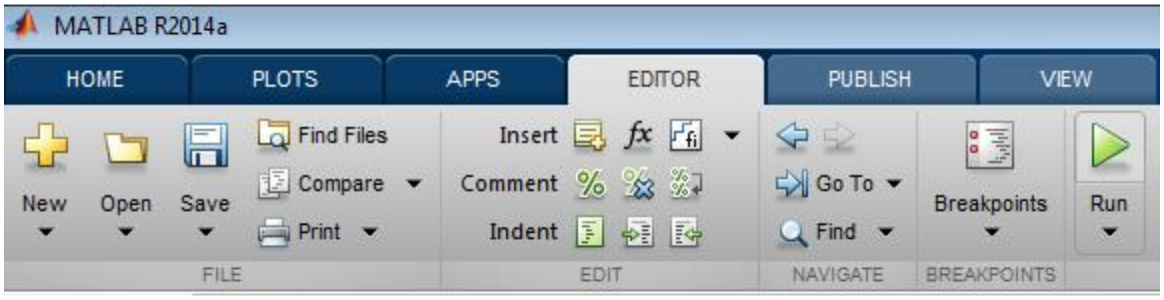
15. Display the image.

```
% Display Image
himg = imshow(Data);
```

16. Close the camera.

```
% Close camera
cam.Exit;
```

17. Select Run from the Editor menu bar. You will be asked to save your script (if you have not already).



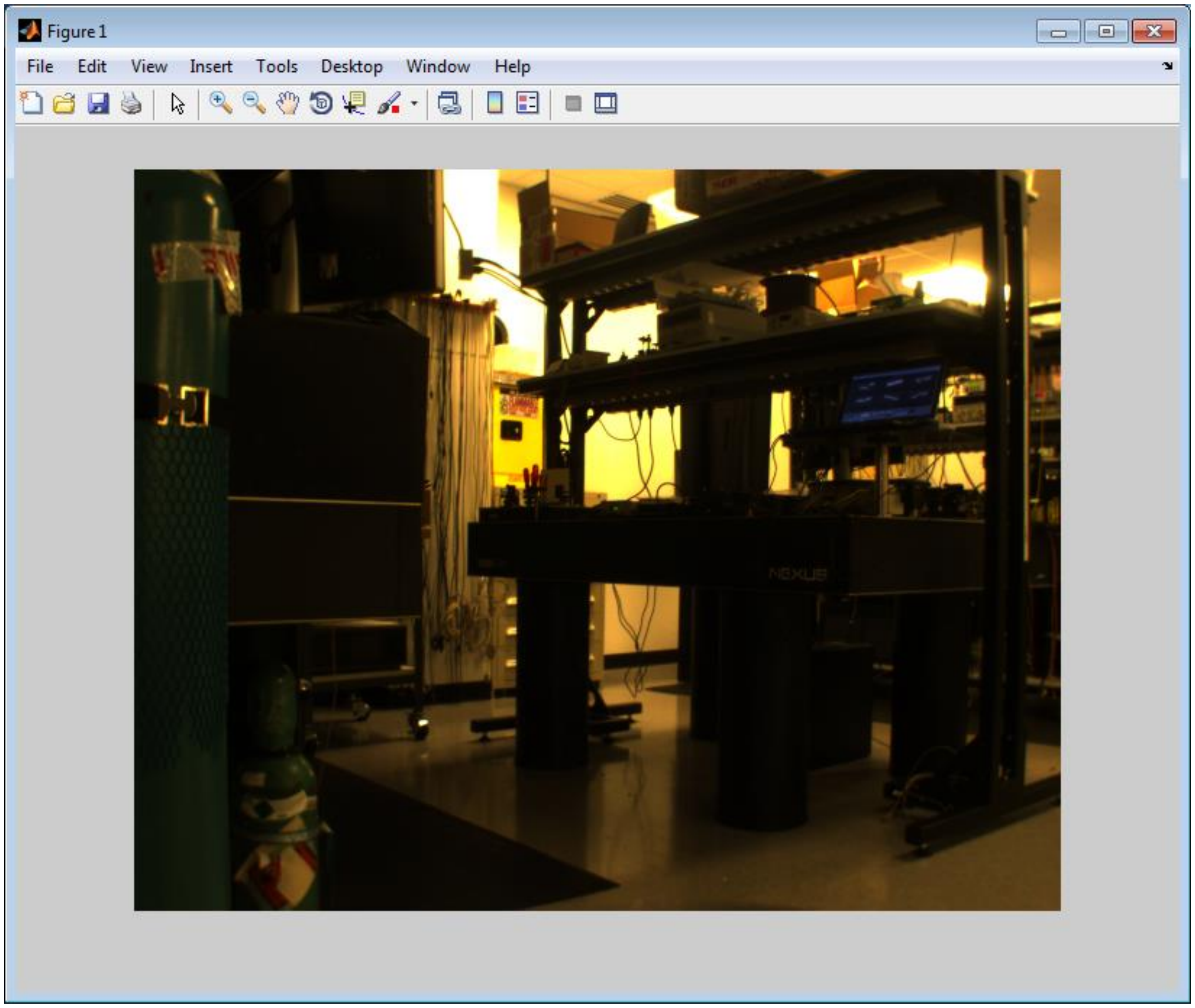


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help

Chapter 5 Methods Used

5.1. uc480.Defines.Status uc480.Camera.Init(int s32Cam)

Summary:

This function starts the driver and establishes the connection to the camera.

Parameters:

s32Cam: This parameter specifies the camera id.

0: The first available camera will be initialized or selected.

1-254: The camera with the specified camera ID will be initialized or selected. The camera ID can be found in the DCx Camera Manager software.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.2. uc480.Defines.Status uc480.DisplayMode.Set(uc480.Defines.DisplayMode mode)

Summary:

This function sets the camera display mode.

Parameters:

mode: This parameter specifies the display mode, see page 103 of the DCx Camera .NET Manual for a full list of display modes.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.3. uc480.Defines.Status uc480.PixelFormat.Set(uc480.Defines.ColorMode mode)

Summary:

This function setts the color mode to be used when image data are saved. For this purpose, the allocated image memory must be large enough to accommodate the data with the selected color mode.

Parameters:

mode: This parameter specifies the color mode, see page 205 of the DCx Camera .NET Manual for a full list of color modes.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.4. uc480.Defines.Status uc480.Trigger.Set(uc480.Defines.TriggerMode mode)

Summary:

This function sets the camera trigger mode. In software trigger mode, an image is captured immediately when Freeze() is called.

Parameters:

mode: This parameter specifies the trigger mode, see page 258 of the DCx Camera .NET Manual for a full list of trigger modes.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.5. uc480.Defines.Status uc480.Memory.Allocate(out int s32MemId, bool bSetAsActiveMem)

Summary:

This function allocates an memory for an image.

Parameters:

s32MemId: This parameter returns the memory id of the allocated memory as an integer. In MATLAB out parameters return on the left side of the function call.

bSetAsActiveMem: This parameter specifies if the allocated memory is active or not as a Boolean.

true: Memory set as active

false: Memory is not set as active

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.6. uc480.Defines.Status uc480.Memory.Inquire(int s32MemId, out int s32X, out int s32Y, out int s32BitsPerPixel, out int s32Pitch)

Summary:

This function returns the properties of an allocated image memory.

Parameters:

s32MemId: This parameter specifies the memory id of the allocated memory as an integer.

s32X: This parameter returns the image width (in pixels) used to define the image memory as an integer. In MATLAB out parameters return on the left side of the function call.

s32Y: This parameter returns the image height (in pixels) used to define the image memory as an integer. In MATLAB out parameters return on the left side of the function call.

s32BitsPerPixel: This parameter returns the bit width used to define the image memory as an integer. In MATLAB out parameters return on the left side of the function call.

s32Pitch: This parameter returns the s the line increment of the image memory as an integer. The line increment is defined as the number of bytes from the beginning of a line to the beginning of the next line. In MATLAB out parameters return on the left side of the function call.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.7. uc480.Defines.Status uc480.Acquisition.Freeze(uc480.Defines.DeviceParameter param)

Summary:

This function acquires a single image from the camera. In DIB mode, the image is stored in the active image memory.

Parameters:

param: This parameter defines if the program should wait or not wait for the image to be acquired.

uc480.Defines.DeviceParameter.DontWait: Do not wait for image acquisition

uc480.Defines.DeviceParameter.Wait: Wait for image acquisition

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.8. uc480.Defines.Status uc480.Memory.CopyToArray(int s32MemId, out byte[] u8Image)

Summary:

This function copies the image in the specified image memory id to an array.

Parameters:

s32MemId: This parameter specifies memory id of the allocated memory as an integer.

u8Image: This parameter returns a one dimensional byte array containing the image data. In MATLAB out parameters return on the left side of the function call.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

5.9. uc480.Defines.Status uc480.Camera.Exit()

Summary:

This function disables the camera handle and releases the data structures and memory areas taken up by the uc480 camera.

Returns:

The function returns a uc480.Defines.Status enumeration which indicates if the operation has been successful or unsuccessful.

Chapter 6 Full Program Listing

```
% Add NET assembly
% May need to change specific location of library
NET.addAssembly('C:\Program Files\Thorlabs\Scientific Imaging\DCx Camera
Support\Develop\DotNet\uc480DotNet.dll');
% Create camera object handle
cam = uc480.Camera;
% Open the 1st available camera
cam.Init(0);
% Set display mode to bitmap (DiB)
cam.Display.Mode.Set(uc480.Defines.DisplayMode.DiB);
% Set color mode to 8-bit RGB
cam.PixelFormat.Set(uc480.Defines.ColorMode.RGBA8Packed);
% Set trigger mode to software (single image acquisition)
cam.Trigger.Set(uc480.Defines.TriggerMode.Software);
% Allocate image memory
[~, MemId] = cam.Memory.Allocate(true);
% Obtain image information
[~, Width, Height, Bits, ~] = cam.Memory.Inquire(MemId);
% Acquire image
cam.Acquisition.Freeze(uc480.Defines.DeviceParameter.Wait);
% Copy image from memory
[~, tmp] = cam.Memory.CopyToArray(MemId);
% Reshape image
Data = reshape(uint8(tmp), [Bits/8, Width, Height]);
Data = Data(1:3, 1:Width, 1:Height);
Data = permute(Data, [3,2,1]);
% Display Image
himg = imshow(Data);
% Close camera
cam.Exit;
```

Chapter 7 Regulatory

As required by the WEEE (Waste Electrical and Electronic Equipment Directive) of the European Community and the corresponding national laws, Thorlabs offers all end users in the EC the possibility to return “end of life” units without incurring disposal charges.

- This offer is valid for Thorlabs electrical and electronic equipment:
- Sold after August 13, 2005
- Marked correspondingly with the crossed out “wheelie bin” logo (see right)
- Sold to a company or institute within the EC
- Currently owned by a company or institute within the EC
- Still complete, not disassembled and not contaminated

Wheelie Bin Logo

As the WEEE directive applies to self-contained operational electrical and electronic products, this end of life take back service does not refer to other Thorlabs products, such as:

- Pure OEM products, that means assemblies to be built into a unit by the user (e. g. OEM laser driver cards)
- Components
- Mechanics and optics
- Left over parts of units disassembled by the user (PCB’s, housings etc.).

If you wish to return a Thorlabs unit for waste recovery, please contact Thorlabs or your nearest dealer for further information.

Waste Treatment is Your Own Responsibility

If you do not return an “end of life” unit to Thorlabs, you must hand it to a company specialized in waste recovery. Do not dispose of the unit in a litter bin or at a public waste disposal site.

Ecological Background

It is well known that WEEE pollutes the environment by releasing toxic products during decomposition. The aim of the European RoHS directive is to reduce the content of toxic substances in electronic products in the future.

The intent of the WEEE directive is to enforce the recycling of WEEE. A controlled recycling of end of life products will thereby avoid negative impacts on the environment.

Chapter 8 Thorlabs Worldwide Contacts

For technical support or sales inquiries, please visit us at **www.thorlabs.com/contact** for our most up-to-date contact information.

THORLABS
www.thorlabs.com