

**RSE 600/300 MATLAB Toolkit Documentation**

**FLUKE19A MATLAB Toolkit**

**Fluke**

**Everett, Washington**

**Date: Mar, 26, 2019**

Table of Contents

[1 Introduction 3](#_Toc4503850)

[2 File Hierarchy 3](#_Toc4503851)

[3 Toolkit Function Descriptions 3](#_Toc4503852)

[3.1 LoadAssemblies 3](#_Toc4503853)

[3.2 DiscoverDevices 4](#_Toc4503854)

[3.3 SelectDevice 4](#_Toc4503855)

[3.4 StartStream 4](#_Toc4503856)

[3.5 Net2MatImage 4](#_Toc4503857)

[3.6 GetData 4](#_Toc4503858)

[3.7 StopStream 5](#_Toc4503859)

[3.8 Capture Image 5](#_Toc4503860)

[3.9 FireNUC 5](#_Toc4503861)

[3.10 Disable NUC 5](#_Toc4503862)

[3.11 EnableNUC 5](#_Toc4503863)

[3.12 AdjustFocus 5](#_Toc4503864)

[3.13 SetFirstRange 6](#_Toc4503865)

[3.14 SetSecondRange 6](#_Toc4503866)

[3.15 SetEBT 6](#_Toc4503867)

[3.16 TempAlarm 6](#_Toc4503868)

[3.17 GetMinMax 6](#_Toc4503869)

[3.18 RecordMovie 7](#_Toc4503870)

[3.19 BooleanTrigger 7](#_Toc4503871)

[3.20 TempTrigger 7](#_Toc4503872)

[3.21 RelativeTimeTrigger 7](#_Toc4503873)

[3.22 AbsoluteTimeTrigger 8](#_Toc4503874)

[3.23 TriggerCallback 8](#_Toc4503875)

[3.24 GetROI 8](#_Toc4503876)

[4 Example Uses 9](#_Toc4503877)

[4.1 Begin Streaming Data 9](#_Toc4503878)

[4.2 Generate a Region of Interest 10](#_Toc4503879)

[4.3 Get Min and Max Temperatures 10](#_Toc4503880)

[4.4 Configure Emissivity, Background Temperature, and Transmissivity 11](#_Toc4503881)

[4.5 Save Image 11](#_Toc4503882)

[4.6 Save a Recording 12](#_Toc4503883)

[4.7 Enable Boolean Trigger 12](#_Toc4503884)

[4.8 Enable Temperature Threshold Trigger 13](#_Toc4503885)

[4.9 Enable Relative Time Trigger 14](#_Toc4503886)

[4.10 Enable Absolute Time Trigger 15](#_Toc4503887)

[4.11 Opening the GUI Interface 15](#_Toc4503888)

# Introduction

The purpose of this document is to explain how to use the Fluke MATLAB Toolkit when developing MATLAB applications that interface with the Fluke line of thermal imagers.

The MATLAB Toolkit is a series of functions that act as wrappers for the Fluke OpenAccess64 DLL for GigE camera communication. The toolkit is divided into two general types of functions: basic camera connectivity functions, and functions that illustrate how to achieve certain common tasks with the camera, such as alarming, region-of-interest masking, and triggered recording.

A demonstration application is included that rolls up all of these functions into a single application. This application has been designed using the MATLAB App Designer, a free add on for MATLAB that allows the user to create GUI applications using a new, more user friendly development environment. In order to guarantee compatibility, the version of MATLAB used to create the application was MATLAB 9.5 (R2018b).

# File Hierarchy

The files in the project are arranged according to the following hierarchy:

* MATLAB Toolkit
  + Source Code
    - FlukeStreamingGUI
      * FlukeMatlabToolkit.mlapp
      * IsAssemblyAdded.m
      * ToolkitFunctions.m
      * OpenAccess64
        + <Fluke OpenAccess64 DLLs>

# Toolkit Function Descriptions

All Toolkit Functions have been wrapped into a MATLAB script(.m) file. To call a function located within the Toolkit Functions script file, type ToolkitFunctions.<name of function>. For example, to adjust the focus of the camera, the user would execute the command “ToolkitFunctions.AdjustFocus(500)”.

## LoadAssemblies

* Loads in the .DLL files required to interface with the camera
  + Inputs
    - N/A
  + Outputs
    - N/A

## DiscoverDevices

* Discovers Fluke Thermal Imagers that are on the same network as the current machine. Returns a string containing the serial number of the first thermal imager found.
  + Inputs
    - N/A
  + Outputs
    - deviceList (String)

## SelectDevice

* Sends a command to the camera interface to select the camera with the given seral number. Returns a Boolean to determine success.
  + Inputs
    - camSerial (String)
  + Outputs
    - Success (Boolean)

## StartStream

* Connects to the camera and begins streaming data
  + Inputs
    - N/A
  + Outputs
    - cameraStream (ImageSourceManager)

## Net2MatImage

* Internal Toolkit Function used by the GetData function. MATLAB does not natively support.net bitmaps. This function berforms some data manipulation on the bits and returns an image that can be displayed on an axes.
  + Inputs
    - Bitmap (Bitmap Image)
  + Outputs
    - Img (MATLAB image)

## GetData

* Reads streaming data from the camera and returns a 2D Matrix of Temperature Data and an image that can be displayed on an axes.
  + Inputs
    - temperatureUnits(String)
  + Outputs
    - TempArray (2D Matrix of double)
    - MatImage (MATLAB Image)

## StopStream

* Stops the stream and closes the connection to the camera
  + Inputs
    - N/A
  + Outputs
    - N/A

## Capture Image

* Sends a command to the camera to save a .IS2 image to disk in the location specified.
  + Inputs
    - Path (string)
  + Outputs
    - N/A

## FireNUC

* Sends a command to the camera to manually fire a NUC
  + Inputs
    - N/A
  + Outputs
    - N/A

## Disable NUC

* Sends a command to the camera to disable the NUC functionality
  + Inputs
    - N/A
  + Outputs
    - N/A

## EnableNUC

* Sends a command to the camera to enable the NUC functionality
  + Inputs
    - N/A
  + Outputs
    - N/A

## AdjustFocus

* Sends a command to the camera to adjust the focus distance in mm.
  + Inputs
    - Distance (int)
  + Outputs
    - N/A

## SetFirstRange

* Sends a command to the camera to set the range to -4°F-176°F
  + Inputs
    - N/A
  + Outputs
    - N/A

## SetSecondRange

* Sends a command to the camera to set the range to -4°F-2192°F
  + Inputs
    - N/A
  + Outputs
    - N/A

## SetEBT

* Sends a command to the camera to configure the Emissivity and Transmission Coefficients as well as the Background Temperature.
  + Inputs
    - Emissivity (double [0-1.0])
    - BackgroundTemp (double)
    - TempUnits (String [Celcsius, Fahrenheit, Kelvin])
    - Transmission (double)
  + Outputs
    - N/A

## TempAlarm

* Returns a Boolean Value that is true when a temperature crosses the desired threshold. Also returns the min and max Temperatures.
  + Inputs
    - TempArray (Array of doubles)
    - minTempThreshold (double)
    - maxTempThreshold (double)
  + Outputs
    - Alarm (Boolean)
    - minValue (double)
    - maxValue (double)

## GetMinMax

* Returns the minimum and maximum values of an array
  + Inputs
    - TempArray (Array of doubles)
  + Outputs
    - minValue (double)
    - maxValue (double)

## RecordMovie

* Sends a command to the camera to save a .IS3 file to disk in the location specified.
  + Inputs
    - filePath (String)
    - filePrefix (String)
    - movieLength (Int) [0-60000mS]
  + Outputs
    - N/A

## BooleanTrigger

* Allows the user to capture a series of images at a specified interval starting when the Boolean input is true
  + Inputs
    - filePath (String)
    - interval (Int) [mS]
    - numCaptures (Int)
    - enableIn (Boolean)
  + Outputs
    - N/A

## TempTrigger

* Allows the user to capture a series of images at a specified interval starting when the temperature crosses a given threshold
  + Inputs
    - filePath (String)
    - interval (Int) [mS]
    - numCaptures (Int)
    - minTempThreshold (Double)
    - maxTempThreshold (Double)
    - tempArray (Array of Double)
  + Outputs
    - triggerActivated (Boolean)

## RelativeTimeTrigger

* Allows the user to capture a series of images at a specified interval starting a specified amount of time after the function is called
  + Inputs
    - filePath (String)
    - interval (Int) [mS]
    - numCaptures (Int)
    - numMilliSecondsToWait (Int)
  + Outputs
    - N/A

## AbsoluteTimeTrigger

* Allows the user to capture a series of images at a specified interval starting at the given time
  + Inputs
    - filePath (String)
    - interval (Int) [mS]
    - numCaptures (Int)
    - startTime (1D Array of Int [Year, Month, Day, Hour, Minute, Second])
  + Outputs
    - N/A

## TriggerCallback

* Internal function used in the trigger functions
  + Inputs
    - Src (Timer)
    - Event (Timer Function)
    - filePath (String)
  + Outputs
    - N/A

## GetROI

* Creates a mask by drawing a filled in polygon with vertices at the input coordinates. This mask is then used to determine whether or not a given point in the temperature data array is within the desired ROI. Points within the ROI remain unchanged while points outside the ROI have their value changed to -500000. This 2D Matrix is then returned. This is useful if the actual indices of the ROI are required.
* This function also returns a 1D array that contains only the temperature elements that are within the ROI. This enables easy manipulation on the ROI
  + Inputs
    - TempArray (2D Array of Double)
    - xPoints (1D Array of Int)
    - yPoints (1D Array of Int)
  + Outputs
    - maskedData (1D Array of Double)
    - TempArray (2D Array of Double)

# Example Uses

## Begin Streaming Data

1. Create a variable and set it equal to ToolkitFunctions.DiscoverDevices
   1. This gives you the camera serial number as a string
2. Pass the camera serial number into the ToolkitFunctions.SelectDevice Function
   1. This sends a command to the camera telling it to prepare for streaming
3. Call the function ToolkitFunctions.StartStream
   1. This actually begins the stream from the camera
4. Create 2 variables (TempArray and IRImage) and set them equal to ToolkitFunctions.GetData
   1. This function returns a 2D array of Temperatures and an IRImage that can be displayed on a MATLAB axes.
   2. This function only returns one array and image. In order to update the data as the camera is streaming, it is recommended to create a timer that calls this function periodically.

|  |
| --- |
| CODE EXAMPLE |
| CameraSerial = ToolkitFunctions.DiscoverDevices  ToolkitFunctions.SelectDevice(CameraSerial)  ToolkitFunctions.StartStream()  [TempArray, IRImage] = ToolkitFunctions.GetData() |

## Generate a Region of Interest

1. Create an array of X-Coordinates.
2. Create an array of Y-Coordinates.
3. Call the function ToolkitFunctions.GetROI(TempArray, X-Coordinates, Y-Coordinates)
   1. TempArray is the original, unmasked array of temperatures
4. Set the output of this function to 2 variables (ROIElementsOnly and MaskedArray)
   1. The GetROI function returns 2 arrays of temperatures
      1. ROIElementsOnly is a 1D array containing only the temperatures that lie within the ROI
         1. This is useful when you only care about the temperature data within the ROI and not its actual position within the ROI
      2. MaskedArray is a 2D array with all elements that lie outside of the ROI set to -500000 and elements that lie within the ROI unchanged
         1. This is useful when you need to know the exact location of elements within the ROI, however data manipulation may be more difficult because elements outside of the ROI (value of -500000) will need to be manually excluded.

|  |
| --- |
| CODE EXAMPLE |
| xCoordinates = [0,5,5,0]  yCoordinates = [0,0,5,5]  [ROIElementsOnly, MaskedArray] =…  ToolkitFunctions.GetROI(TempArray, xCoordinates, yCoordinates) |

## Get Min and Max Temperatures

1. Create 2 variables called MinValue and MaxValue and set them equal to the ToolkitFunctions.GetMinMax function
   1. NOTE: This function will return -500000 as the max if a masked array is passed in

|  |
| --- |
| CODE EXAMPLE |
| [MinValue, MaxValues] = ToolkitFunctions.GetMinMax(TempArray) |

## Configure Emissivity, Background Temperature, and Transmissivity

1. Create a variable called Emissivity and set it to a value between 0 and 1
2. Create a variable called BackgroundTemp and set it to a numeric value
3. Create a variable called TempUnits and set it to either:
   1. Celsius
   2. Fahrenheit
   3. Kelvin
4. Create a variable called Transmissivity and set it to a value between 0 and 100
5. Pass these values as arguments to the ToolkitFunctions.SetEBT function

|  |
| --- |
| CODE EXAMPLE |
| Emissivity = 0.85  BackgroundTemp = 72  TempUnits = ‘Fahrenheit’  Transmissivity = 95  ToolkitFunctions.SetEBT(Emissivity, BackgroundTemp, TempUnits, Transmissivity) |

## Save Image

1. Create a variable called FilePath and set it equal to a location on disk
   1. Make sure that you include the file prefix that you would like the image to be saved as
      1. For example, if you would like to save an image called “Screenshot” to the Desktop, you would use the path:

‘C:\Users\[UserName]\Desktop\Screenshot’

* + 1. When the function is called, the IR Image can be located in a file called “Screenshot.is2” will be created on the Desktop

1. Pass this variable as an argument into the ToolkitFunctions.CaptureImage function

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop\Screenshot’  ToolkitFunctions.CaptureImage(FilePath) |

## Save a Recording

1. Create a variable called FilePath and set it equal to a location on disk
2. Create a variable called FilePrefix and set it equal to the desired file name
3. Create a variable called MovieLength and set it equal to the desired recording length (in mS, up to 1 minute)
4. Pass these variables as arguments into the ToolkitFunctions.RecordMovie function

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop’  FilePrefix = ‘Recording’  MovieLength = 25000  ToolkitFunctions.RecordMovie(FilePath, FilePrefix, MovieLength) |

## Enable Boolean Trigger

1. The Boolean Trigger function begins capturing images at a specified interval when the EnableIn argument is true
   1. Create a variable called FilePath and set it equal to the desired location on disk. Make sure that you include the file prefix that you would like the image to be saved as
      1. For example, if you would like to save an image called “Screenshot” to the Desktop, you would use the path:

‘C:\Users\[UserName]\Desktop\Screenshot’

* 1. Create a variable called Interval and set it equal to the desired time between captures in mS
  2. Create a variable called NumCaptures and set it equal to the desired number of images that you would like the trigger to capture
  3. Create a variable called EnableIn and set it to True if you would like the trigger to activate, or False if you do not
  4. Pass these variables as arguments into the ToolkitFunctions.BooleanTrigger function

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop\BooleanTrigger’  Interval = 2500  NumCaptures = 5  EnableIn = true  ToolkitFunctions.BooleanTrigger(FilePath, Interval, NumCaptures, EnableIn) |

## Enable Temperature Threshold Trigger

1. The Temperature Threshold Trigger function begins capturing images at a specified interval when the value of an element within the TempArray argument falls outside of a desired range.
   1. Create a variable called FilePath and set it equal to the desired location on disk. Make sure that you include the file prefix that you would like the image to be saved as
      1. For example, if you would like to save an image called “Screenshot” to the Desktop, you would use the path:

‘C:\Users\[UserName]\Desktop\Screenshot’

* 1. Create a variable called Interval and set it equal to the desired time between captures in milliseconds
  2. Create a variable called NumCaptures and set it equal to the desired number of images to be captured
  3. Create a variable called MinTempThreshold and set it equal to the desired minimum temperature
  4. Create a variable called MaxTempThreshold and set it equal to the desired minimum temperature
  5. Create a variable called TempArray and set it equal to the TempArray output of the ToolkitFunctions.GetData function
  6. Create a variable called TriggerActivated and set it equal to the output of the ToolkitFunctions.TempTrigger function, passing in the Interval, NumCaptures, MinTempThreshold, MaxTempThreshold, and TempArray as arguments
     1. This provides feedback to the user about whether or not the trigger was activated

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop\TempTrigger’  Interval = 2500  NumCaptures = 5  MinTempThreshold = 50  MaxTempThreshold = 100  TempArray = ToolkitFunctions.GetData(‘Fahrenheit’)  TriggerActivated = ToolkitFunctions.TempTrigger(FilePath, Interval,… NumCaptures, MinTempThreshold, MaxTempThreshold, TempArray) |

## Enable Relative Time Trigger

1. The Relative Time Trigger function begins capturing images at a specified interval when the a specified amount of time elapses after the function is initially called
   1. Create a variable called FilePath and set it equal to the desired location on disk. Make sure that you include the file prefix that you would like the image to be saved as
      1. For example, if you would like to save an image called “Screenshot” to the Desktop, you would use the path:

‘C:\Users\[UserName]\Desktop\Screenshot’

* 1. Create a variable called Interval and set it equal to the desired time between captures in milliseconds
  2. Create a variable called NumCaptures and set it equal to the desired number of images to be captured
  3. Create a variable called NumMilliSecondsToWait and set it equal to the desired time between the initial call of the function and when the trigger begins capturing images
  4. Pass these variables as arguments into the ToolkitFunctions.RelativeTimeTrigger function

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop\RelativeTrigger’  Interval = 2500  NumCaptures = 5  NumMilliSecondsToWait = 15000  ToolkitFunctions.RelativeTimeTrigger(FilePath, Interval, NumCaptures,…  NumMilliSecondsToWait) |

## Enable Absolute Time Trigger

1. The Absolute Time Trigger function begins capturing images at a specified interval when the system clock reaches a specified time.
   1. Create a variable called FilePath and set it equal to the desired location on disk. Make sure that you include the file prefix that you would like the image to be saved as
      1. For example, if you would like to save an image called “Screenshot” to the Desktop, you would use the path:

‘C:\Users\[UserName]\Desktop\Screenshot’

* 1. Create a variable called Interval and set it equal to the desired time between captures in milliseconds
  2. Create a variable called NumCaptures and set it equal to the desired number of images to be captured
  3. Create a variable called StartTime and set it equal to a 1D array of [Year, Month, Day, Hour, Minute, Second]
  4. Pass these variables as arguments into the ToolkitFunctions.AbsoluteTimeTrigger function

|  |
| --- |
| CODE EXAMPLE |
| FilePath = ‘C:\Users\CurrentUser\Desktop\AbsoluteTrigger’  Interval = 2500  NumCaptures = 5  StartTime = [2019, 03, 15, 16, 30, 0]  ToolkitFunctions.AbsoluteTimeTrigger(FilePath, Interval, NumCaptures,… StartTime) |

## Opening the GUI Interface

1. As mentioned above, a MATLAB application that rolls up all these functions into one easy to use application.
2. If you want to run the application without making any code changes:
   1. Locate the file ‘FlukeMatlabToolkit.mlapp’ on your computer and double click it in Windows Explorer. This will open MATLAB and then open the application.
   2. Set the current MATLAB directory to the location of the ‘FlukeMatlabToolkit.mlapp’ file and type ‘FlukeMatlabToolkit’ in the command window
      1. After setting the current MATLAB directory to this location, the file should appear in the file explorer to the left of the command window
3. If you want to edit the code for the application:
   1. Type ‘appdesigner’ in the command window. This will open the new, user-friendly GUI development environment. Then, click on the ‘Open’ button in the top right corner and browse to the ‘FlukeMatlabToolkit.mlapp’ file and open it.
   2. If the current MATLAB directory is set to the location of the ‘FlukeMatlabToolkit.mlapp’, you can type ‘appdesigner FlukeMatlabToolkit’ and it will automatically open the application in the new development environment.