

# Developer Guide

## Phantom SDK for Matlab

---

Version 13.4.787.0 (June 2019)

Copyright (C) 1992-2019 Vision Research Inc. All Rights Reserved.

*Notice*

The information contained in this document file includes data that is proprietary of Vision Research Inc and shall not be duplicated, used, or disclosed – in whole or in part – without the prior written permission of Vision Research Inc. The data subject to this restriction is contained in all pages of this file.

THIS PUBLICATION AND THE INFORMATION HEREIN ARE FURNISHED AS IS, ARE FURNISHED FOR INFORMATIONAL USE ONLY, ARE SUBJECT TO CHANGE WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY VISION RESEARCH INC. VISION RESEARCH INC ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR INACCURACIES THAT MAY APPEAR IN THE INFORMATIONAL CONTENT CONTAINED IN THIS GUIDE, MAKES NO WARRANTY OF ANY KIND (EXPRESS, IMPLIED, OR STATUTORY) WITH RESPECT TO THIS PUBLICATION, AND EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSES, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

*Licenses and Trademarks*

Windows is a trademark of Microsoft Corporation.

*Software release*

Some new fields of the SETUP structure may be added in new software releases. This document is based on software release 787 (PhCon.dll, PhFile.dll, PhInt.dll version 13.4.787.0, PCC version 3.4.787.0).



**AMETEK**<sup>®</sup>  
MATERIALS ANALYSIS DIVISION

100 Dey Road  
Wayne, NJ 07470 USA  
+1.973.696.4500  
[phantom@visionresearch.com](mailto:phantom@visionresearch.com)

[www.phantomhighspeed.com](http://www.phantomhighspeed.com)

## Contents

<b>1.</b>	<b>Introduction .....</b>	<b>4</b>
<b>2.</b>	<b>Package content .....</b>	<b>5</b>
<b>3.</b>	<b>Installation.....</b>	<b>6</b>
<b>4.</b>	<b>SDK description .....</b>	<b>10</b>
4.1.	C header files for Matlab interface to shared libraries .....	10
4.2.	Matlab function wrappers for PhantomSDK .....	10
4.3.	Object-oriented layer for PhantomSDK .....	10
4.4.	Matlab demo scripts.....	10
4.4.1.	HelloPhantom.....	10
4.4.2.	PhDemoMatlab .....	11
4.4.3.	SimpleMatlabDemo .....	11
4.4.4.	Function to read images from a cine file.....	11
<b>5.</b>	<b>Tips for the Matlab developers .....</b>	<b>13</b>
5.1.	Calling functions which have a pointer to an array of data .....	13
5.2.	Working with C structures and arrays not supported in Matlab .....	13
5.2.1.	Array of structures inside structure .....	13
5.2.2.	Multidimensional array inside structure .....	14
5.2.3.	Structure with array or pointer fields inside another structure .....	14
5.3.	Compatibility with Matlab versions older than 2009a.....	14

## 1. Introduction

The Matlab SDK is designed to be used by Matlab developers who want to write their own code to control Phantom high speed cameras or to play, analyze and do measurements on cine files.

It contains header files needed to call Phantom SDK functions from Matlab, function wrappers, a simple object-oriented layer and demo scripts. This SDK uses the Matlab interface to shared libraries to call functions from Phantom libraries.

This Matlab SDK was written and tested in Matlab 2009a. For older versions of Matlab, the compatibility of the calls with calllib to Phantom dlls functions will be low and crashes may occur. The code should work with all Matlab versions after version 2009a.

This document provides a description of what you will find in the package, setup instructions, information about how to write functions wrappers for calls from Matlab, a description of the object oriented layer used by the demo application written in Matlab.

## 2. Package content

The package for MatlabSDK contains the following items:

- C header files adjusted for Matlab interface to shared libraries
- Matlab wrappers for a large number of functions from Phantom SDK
- An object-oriented Matlab layer for PhantomSDK
- Matlab demo scripts
  - ✓ A very simple demo application, without user interface, for recording, triggering, obtaining and image and saving the recording to file: `HelloPhantom`
  - ✓ A demo application for camera control and cine player: `PhDemoMatlab`
  - ✓ A simpler demo application for camera control and cine player: `SimpleMatlabDemo`
  - ✓ A simple function to read images from a cine file: `ReadCineFileImage`
- Helper functions and classes
- This documentation file and a copy of all documentation from the Phantom SDK.
- The Phantom dlls and the redist files for the release 787. Newer releases of Phantom dlls will be compatible with 787 so you can replace them with the latest version you have, to get access to new cameras or features.

### 3. Installation

Do the following steps prior to the use of this SDK.

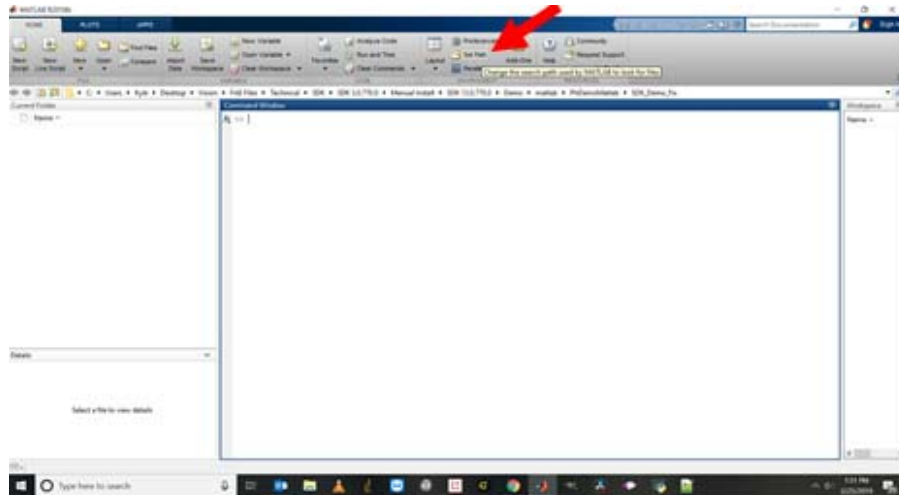
- If you have a software installer for Phantom camera please install it. You can try to run the applications from this SDK and connect to standard Gigabit Ethernet cameras or to simulated cameras without installing the Phantom cameras software
- Install or copy the Phantom SDK to your computer in a folder at your choice. Update Phantom dlls if you have a newer version.
- Make sure you have your appropriate Phantom dlls (32 or 64 bit) in the Matlab search path. Do this by File -> Set Path... -> Add Folder... -> browse for your desired version of phantom dlls. Pay attention to the case when there are other directories in Matlab search path that include phantom dlls - remove those search paths.
- Include the PhMatlabSDK directory into Matlab search path. Do this by using File -> Set Path... -> Add with Subfolders... -> then select the PhMatlabSDK directory from your computer.
- **Important note:** On 64-bit systems and on the latest version of Matlab (i.e., Matlab 2018b), the Win32 folder should be deleted from the SDK file path, located:

SDK 13.4.787.0 >> bin >> Win32

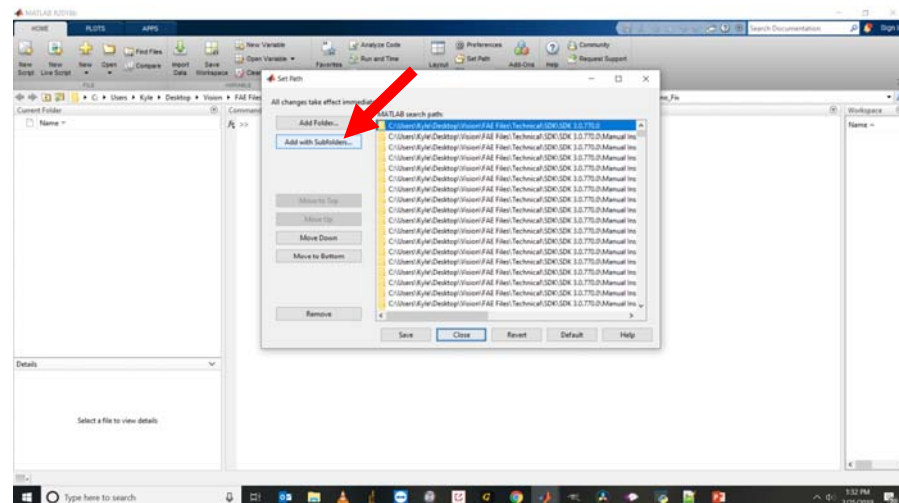
**Note:** You can find more information for Matlab search path on MathWorks site:  
[http://www.mathworks.com/help/techdoc/matlab\\_env/br5tea6-6.html#br5tea6-7](http://www.mathworks.com/help/techdoc/matlab_env/br5tea6-6.html#br5tea6-7)

### Example Installation:

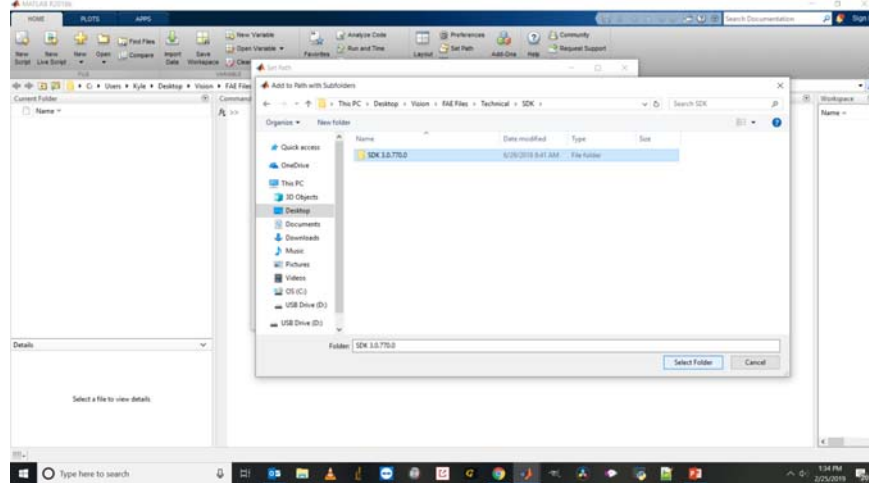
- 1) Download the SDK (make sure you know the download location!)
- 2) Open Matlab (Here Matlab R2018b was used)
- 3) Click, "Set Path" under the home tab



- 4) Click "Add with Subfolders"

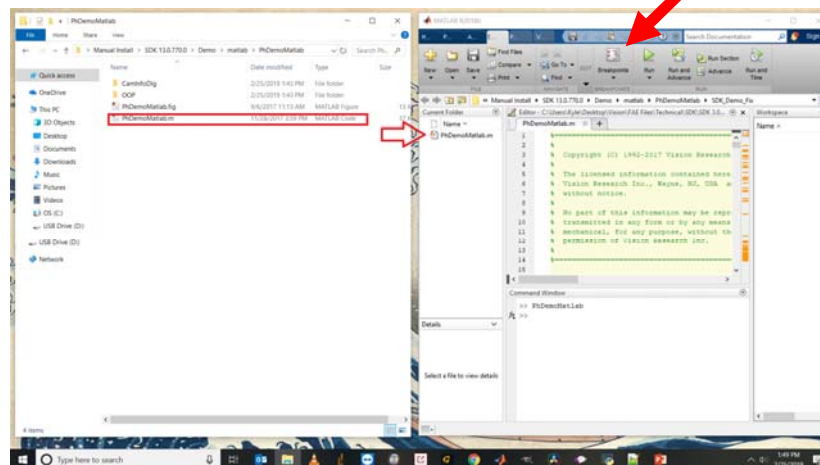


- 5) Select the main SDK folder



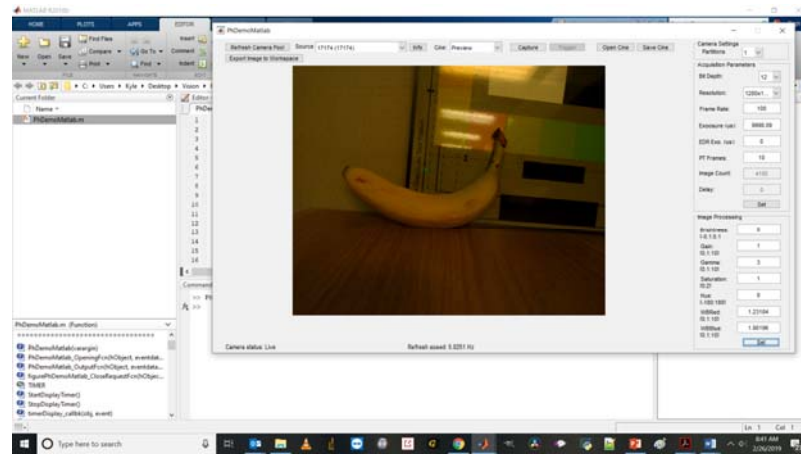
**Note: If using a 64-bit system, make sure you delete the Win32 file from the bin folder within the SDK main folder**

- 6) *Copy & Paste* PhDemoMatlab.m to the Current Folder in Matlab, click RUN





7) This is a typical screenshot of the PhDemoMatlab live video:



## 4. SDK description

### 4.1. C header files for Matlab interface to shared libraries

The C header files are variants of the ones included in Phantom SDK, adjusted for compatibility with Matlab. They are used in Matlab external interface for shared libraries to define the data structures and functions prototypes from Phantom SDK libraries.

The following files are located in the "PhMatlabSDK\PhMatlab\Inc" directory:

- PhConML.h
- PhIntML.h
- PhFileML.h

The constants declared with #define in the C headers are redefined as Constant properties of some classes in:

- PhFileConst.m
- PhConConst.m
- PhIntConst.m

The way the class is declared simulates an enum. The files with the constant definitions are stored in "PhMatlabSDK\PhMatlab\Constants".

### 4.2. Matlab function wrappers for PhantomSDK

The function wrappers make use of Matlab interface for shared libraries, mainly the calllib function, to call PhantomSDK functions. You can find information on these functions in the "Phantom SDK Reference Manual.pdf".

The function wrappers are located under the "PhMatlabSDK\PhMatlab\" directory in "PhCon", "PhInt" and "PhFile" directories.

### 4.3. Object-oriented layer for PhantomSDK

The object-oriented layer written for Matlab is build on top of function wrappers for Phantom SDK. It provides basic functionality to setup and control a high speed Phantom camera and access the recordings from camera or files. The classes are similar to those used in Phantom SDK demos as described in the document "PhDemo Developer Guide.pdf".

The scripts for the object oriented layer are located in the "PhMatlabSDK\PhMatlab\OOP" directory.

### 4.4. Matlab demo scripts

#### 4.4.1. HelloPhantom

HelloPhantom is a very simple application to control Phantom camera, without any user interface. It prints some information in the output window, in console mode style.

It illustrates the loading/unloading phantom dll's, register/unregister client and a typical operation of a high speed camera:

- set to single cine mode (1 partition) using PhSetPartitions

- get acquisition parameters (resolution, frame rate, exposure) using `PhGetCineParams`, optionally change some of them.
- trigger and waiting the end of recording: `PhSendSoftwareTrigger`, `PhGetCineStatus`
- creating a cine handle in the computer: `PhNewCineFromCamera`
- obtaining an image from the recorded cine and convert it to Matlab format to be displayed with figure function: `PhGetCineImage`
- saving a clip from recording to a RAW cine file: `PhWriteCineFile`
- saving a clip from recording to a number of tiff images: `PhWriteCineFile`
- Some information and results are displayed during the run.

#### **4.4.2. PhDemoMatlab**

PhDemoMatlab contains a graphical user interface with the basic controls you need to get images from a camera, to set and get acquisition parameters and image parameters, to record and save cines to files and to playback the recordings either from camera or from file.

PhDemoMatlab script is based on the included object-oriented layer. Its interface structure and functionality is similar to the one used for the two demo versions in the main SDK – one written in C++ (`PhDemoCPP`) and the other in C Sharp (`PhDemoCS`). These demos are described in the document “PhDemo Developer Guide.doc”.

The PhDemoMatlab files are located under the “PhMatlabSDK\PhDemoMatlab” directory.

##### **4.4.2.1. Running the PhDemoMatlab**

After installation of the SDK on your computer, open the `PhDemoMatlab.m` from the PhDemoMatlab folder and use F5 to run the demo. You can connect to Phantom cameras or you can work on simulated cameras in the absence of the real devices.

#### **4.4.3. SimpleMatlabDemo**

It is a simpler version of PhDemoMatlab without object oriented programming and with fewer features presented – no multicine, no image processing, no play from files.

#### **4.4.4. Function to read images from a cine file**

Most applications of the Matlab are related to the play and analyze of the cine recordings stored in files. If your interest is in this area you can skip the complex, object oriented PhDemoMatlab and start from a simpler example: `ReadCineFileImage` function.

`ReadCineFileImage` is a simple function to read cine images from a file. The function implementation is based only on the PhantomSDK function wrappers.

##### **4.4.4.1. Function parameters**

- `fileName` – the file path where the cine file is located
- `imageNo` – the image number to be read from the cine file
- `showImage` – if true the function will create a figure where the image is shown

#### 4.4.4.2. Function outputs

- `matlabIm` – a 1D Gray/3D RGB matrix with pixel values left aligned to 16b ready to be displayed with Matlab `image` function
- `unshiftedIm` – a 1D Gray/3D RGB matrix with pixel values unshifted. For example, if the camera provides a pixel bitdepth of 12 you will get pixel values as 16 bit integers from 0 to 4095.

#### 4.4.4.3. Function usage

- Load the phantom libraries  
`LoadPhantomLibraries();`
- Register the Phantom dlls ignoring connected cameras. Use this function once at the beginning of your work  
`RegisterPhantom(true);`
- Read the cine image  
`[matlabIm, origIm] = ReadCineFileImage('D:\Cine\test.cine', -3000, true);`
- Do other work with cine files
- When you finished your work unregister Phantom dlls  
`UnregisterPhantom();`
- Finally unload the Phantom libraries  
`UnloadPhantomLibraries();`

## 5. Tips for the Matlab developers

This section describes a few tips that will help programmers in their development work and in understanding of the actual code.

### 5.1. Calling functions which have a pointer to an array of data

Some functions like: `PhGetCameraID`, `PhGetCameraResolutions`, `PhGetOpenCineName`, `PhGetCineImage` have as parameters a pointer to an array of data.

The steps to call such a function are:

- Allocate sufficient memory by creating a large enough array of data with the type required by the parameter.

Examples:

- ✓ Allocating uint8 array used to read an image buffer with dll function `PhGetCineImage`:  

```
pixels = zeros(BufferSize, 1, 'uint8');
```
- ✓ Allocating a 256 char array which will be used to read a file path with function `PhGetOpenCineName`:  

```
charVector = char(ones(256,1,'int8'));  
or  
charVector = blanks(256);
```
- Create a libpointer to this array.  

```
pPixel = libpointer('uint8Ptr', pixels);
```
- Pass the pointer using `calllib` function.  

```
[HRES, allOutPointers] = calllib('phfile', 'PhGetCineImage',  
CH, pImRng, pPixel, uint32(BufferSize), pIH);
```
- Access returned data using Value member of the pointer. Use Plus operator and/or `setdatatype` function when needed.  

```
readPixels = pPixel.Value;
```

Note: Only an array of simple structures (containing simple data types as `INT` or `float`) are supported for these calls. Array of structures containing different kind of fields (or pointers) cannot be used.

### 5.2. Working with C structures and arrays not supported in Matlab

Some structures declared in the original Phantom C headers are not compatible with `calllib` function use from the 2009a Matlab version. As a result separate compatible headers were created.

#### 5.2.1. Array of structures inside structure

These structures are not supported by `calllib`. Array fields must be converted to an array of basic C type. Examples of such structures: `CAMERAOPTIONS`, `BITMAPINFO`, `IMPARAMS`.

### 5.2.2. Multidimensional array inside structure

These arrays are not supported for calllib calls. These fields must be converted to a single dimensional array.

Example: `char AnaUnit[8][6]`, a member of the `SETUP` structure, will become `char AnaUnit[8*6]`

### 5.2.3. Structure with array or pointer fields inside another structure

These fields can be supported in MatLAB2009a if the below patch is applied:

<http://www.mathworks.com/support/bugreports/537829>

## 5.3. Compatibility with Matlab versions older than 2009a

Arrays of structures are not supported for function calls in versions older than Matlab 2009a. If the structure contains simple data (only int for example) you may change the function declaration from the header file and use simple data pointers (like int pointers) and then parse the content returned in the pointer. Example:

```
PhGetResolutions (UINT CN, PPOINT pRes, PUINT pCnt, PBOOL pCAR, PUINT
pADCBits) function declaration.
```

You may change the function declaration using PINT. One int pair value represent a POINT structure.

**Note:** Other compatibility issues may rise in older versions.