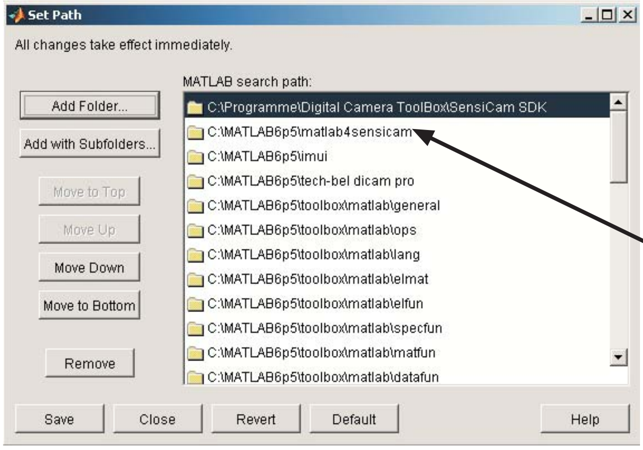
Matlab4sensicam

Before starting...

The following conditions have to be fullfilled to enable the operation from sensicam family, dicam pro family and hsfc pro family of pco camera systems:

[1] the pci board of the camera is installed properly
[2] the corresponding drivers from the CD or the website and the correpsonding SDK from the CD or the website are installed
[3] Matlab operation has been checked with Matlab v6.5.0 and Image Processing Toolbox v4.0 and higher
[4] The GenericDLL.exe has to be installed properly, (see the information given by The Mathworks at the end of this manual)
[5] the directories where the M-files from this zip.file are unpacked as well as the directory where the pco SDK is installed have to be added with the "Set Path" function of Matlab (see image below) to announce them to Matlab:



All changes take effect immediately.

MATLAB search path:

```
C:\Programme\Digital Camera ToolBox\SensiCam SDK
C:\MATLAB6p5\matlab4sensicam
C:\MATLAB6p5\imui
C:\MATLAB6p5\tech-bel dicam pro
C:\MATLAB6p5\toolbox\matlab\general
C:\MATLAB6p5\toolbox\matlab\ops
C:\MATLAB6p5\toolbox\matlab\lang
C:\MATLAB6p5\toolbox\matlab\elmat
C:\MATLAB6p5\toolbox\matlab\elfun
C:\MATLAB6p5\toolbox\matlab\specfun
C:\MATLAB6p5\toolbox\matlab\matfun
C:\MATLAB6p5\toolbox\matlab\datafun
```

directory, where the SDK is installed

directory, where for example the M-files are unpacked

The functions...

The presented M-files to control and access the PCO sensicam, dicam pro & hsfc pro camera family represent a subset of commands, which is available with the PCO SDK (software development kit, can be downloaded for free at www.pco.de). It uses the GenericDLL interface of Matlab and contains therefore only the SDK functions, which do not use Windows handles.
The structure for the naming for the functions always corresponds to the used name in the SDK and adding the letters "sc" in front, therefore the SDK function "GET_IMAGE_SIZE" becomes for Matlab "scGET_IMAGE_SIZE". The help text of the Matlab functions given as M-files contain information about calling convention, differences to the original SDK calling and page numbers of the SDK manual. The parameter and value transfer is a little bit changed compared to the SDK functions, mainly to suit the way Matlab acts.
If you find errors in the scripts or something is not working as described, it would be nice to inform support@pco.de about it. We'll try to check and improve if possible and necessary.

Corresponding to the advice on page 6 in the SDK manual a simple program, which dscribes how to record an image with a camera, and a simple GUI-application is added in the zip-file. For more detailed information please have a look into the SDK manual.
The application files are:
simpleMatlab4sensicam.m - this is a simple take one image application
Matlab4sensicam.m - this is a GUI application to show some of the functionality of the camera, Matlab4sensicam.fig - corresponding GUI figure file

Matlab4sensicam

The functions are...

The following pco SDK functions have been turned into Matlab functions

scSET_Board

If there is more than one PCI interface board installed in the computer, it is possible to set the board, which is addressed by a scSET_BOARD() call. The first board starts with the number 0. After calling scSET_BOARD() all following SDK commands will be directed to the selected board, until the next scSET_BOARD() command is called.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 8}

syntax error_code = scSET_BOARD(board_number)
input board_number [uint32]: number of PCI interface boards to work with, range 0..4 + (flag * 256)
flag = 0 - with COC reprogramming
flag = 1 - without COC reprogramming
output error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scGET_BOARD

This function returns the number of the PCI board, which is currently in use.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 8}

syntax [board_number, error_code] = scGET_BOARD
input none
output board_number [uint32]: number of the currently active PCI interface board
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scSET_INIT

This function resets the PCI interface board hardwrae as well as the camera to default values. It checks whether a camera is connected and a PCI interface board is installed. Note: scSET_INIT(1) or (2) has to be called before any other function calls (except scSET_BOARD). Call scSET_INIT(0) to close a selected board before closing an application.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 9}

syntax error_code = scSET_INIT(init_mode)
input init_mode [int32] : camera initialization mode
0 - terminate driver, shutdown
1 - initialize camera, start with the standard parameters, without dialog DLLs
2 - initialize camera, start with stored parameters, loading dialog DLLs
output error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scSET_COC

This function generates a COC (camera operation code) which is loaded into the program memory of the camera.
{...for sensicam: pco SDK for sensicam, dicam pro, hsfc pro manual page 10}
{...for dicam pro: pco SDK for sensicam, dicam pro, hsfc pro manual page 20}

syntax error_code = scSET_COC(type, gain, submode, trig, roix1,...
roix2, roiy1, roiy2, hbin, vbin, table)

input type [int32]: part of mode, defines camera type
gain [int32]: part of mode, defines gain status of camera
submode [int32]: part of mode, defines operational behavior of
camera type
trig [int32]: sets camera trigger mode
roix1 [int32]: start value for horizontal ROI
roix2 [int32]: end value for horizontal ROI
roiy1 [int32]: start value for vertical ROI
roiy2 [int32]: end value for vertical ROI
hbin [int32]: set horizontal binning
vbin [int32]: set vertical binning
table [char array]: table that contains delay and exposure times in
Milliseconds (sensicam) as strings

output error_code [int32]: zero on success, nonzero indicates failure,
returned value is the error code

scTEST_COC

This function tests all parameters that should be used for a scSET_COC() command. If the parameters have a valid value, they will be accepted, otherwise the value, next close to a valid one, will be used.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 25}

syntax [r_type, r_gain, r_submode, r_trig, r_roix1, r_roix2, r_roiy1,...
r_roiy2, r_hbin, r_vbin, r_table, error_code] = scTEST_COC(type,...
gain, submode, trig, roix1, roix2, roiy1, roiy2, hbin, vbin, table)

input type [int32]: part of mode, defines camera type
gain [int32]: part of mode, defines gain status of camera
submode [int32]: part of mode, defines operational behavior of
camera type
trig [int32]: sets camera trigger mode
roix1 [int32]: start value for horizontal ROI
roix2 [int32]: end value for horizontal ROI
roiy1 [int32]: start value for vertical ROI
roiy2 [int32]: end value for vertical ROI
hbin [int32]: set horizontal binning
vbin [int32]: set vertical binning
table [char array]: table that contains delay and exposure times in
Milliseconds (sensicam) as strings

output r_type [int32]: part of mode, defines camera type
r_gain [int32]: part of mode, defines gain status of camera
r_submode [int32]: part of mode, defines operational behavior of
camera type
r_trig [int32]: sets camera trigger mode
r_roix1 [int32]: start value for horizontal ROI
r_roix2 [int32]: end value for horizontal ROI
r_roiy1 [int32]: start value for vertical ROI
r_roiy2 [int32]: end value for vertical ROI
r_hbin [int32]: set horizontal binning
r_vbin [int32]: set vertical binning
r_table [char array]: table that contains delay and exposure times
in Milliseconds (sensicam) as strings
error_code [int32]: zero on success, nonzero indicates failure,
returned value is the error code

scRUN_COC

Processing of the COC is started with scRUN_COC. The COC program describes the read out procedure for the CCD image sensor as well as the delay and exposure times for capturing an image. In continuous mode the COC program is starting repeatingly until a scSTOP_COC command is executed.
Note: scRUN_COC does not transfer images to the image buffers of the PCI interface board as long as tehre are all buffers occupied by images! In order to release buffers call scREAD_IMAGE... (releases one buffer) or scSTOP_COC (releases all buffers) or scCLEAR_BOARD_BUFFER (releases one buffer).
run_mode = 4 (single) should not be called in simultaneous mode, otherwise this causes the camera and DLL to make some processing, which will decrease performance.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 26}

syntax	error_code = scRUN_COC(run_mode)	
input	run_mode [int32]:	0 - continuous 1 - single
output	error_code [int32]: zero on success, nonzero indicates failure,	returned value is the error code

scSTOP_COC

This function interrupts a running exposure (execution of the COC program, COC = camera operation code). It can be used as a break option, e.g. in case of very long delay and exposure times. Additionally, the image buffers of the PCI interface board are released and stored images are lost!
Note: After scSTOP_COC is called the BUSY signal of the PCI interface board indicates "busy state, not ready for taking images" until scSET_COC is called again. For description of the BUSY signal see description of the PCI interface board.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 27}

syntax	error_code = scSTOP_COC	
input	none	
output	error_code [int32]: zero on success, nonzero indicates failure,	returned value is the error code

scGET_STATUS

This function gets status information from the camera and the PCI interface board and read the temperature of the camera circuits and of the CCD image sensor.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 28}

syntax	[cam_type, temp_ele, temp_ccd, error_code] = scGET_STATUS	
input	none	
output	cam_type [int32]: camera status information, for explanation of	bit meaning see SDK manual p.28
	temp_ele [int32]: temperature value of camera electronic	valid range -30..+65°C (-22...+149°F)
	temp_ccd [int32]: temperature value of CCD-chip	valid range -30..+65°C (-22...+149°F)
	error_code [int32]: zero on success, nonzero indicates failure,	returned value is the error code

scGET_CAMERA_CCD

This function reads the CCD image sensor type of the camera.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 29}

syntax	[ccd_type, error_code] = scGET_CAMERA_CCD(board_number)	
input	board_number [int32]: number of PCI interface board	-1 - board selected with the last scSET_BOARD call 0..3
output	ccd_type [int32]:	1 VGA black & white 2 VGA color.... for more information see SDK manual p.29
	error_code [int32]: zero on success, nonzero indicates failure,	returned value is the error code

Matlab4sensicam

scGET_CAMERA_TYP

This functions determines the camera type
{pco SDK for sensicam, dicam pro, hsfc pro manual page 29}

syntax [camera_type, error_code] = scGET_CAMERA_TYP(board_number)
input board_number [int32]: number of PCI interface board
-1 - board selected with the last scSET_BOARD command
0..3
output camera_type [int32]: type of camera
1 - fast shutter
2 - long exposure
3 - dicam pro
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scGET_IMAGE_SIZE

This function reads the actual image size. The image size depends on the binning and ROI settings, which have been set by the last scSET_COC() command and on the CCD image sensor type. In "double shutter" mode or cameras the height of the double image (2..2048) is returned.
Note: this function returns invalid values, if it is called after a LOAD_USER_COC() command.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 30}

syntax [image_width, image_height, error_code] = scGET_IMAGE_SIZE
input none
output image_width [int32]: width of image in pixel
image_height [int32]: height of image in pixel
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scGET_CCD_SIZE

This function gets the total available pixel number of the CCD image sensor.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 30}

syntax [ccd_size, error_code] = scGET_CCD_SIZE
input none
output ccd_size [int32]: number of pixel of the CCD chip
307200 (VGA)
1310720 (SVGA)
1431040 (QE)
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scGET_IMAGE_STATUS

This function gets the current image status.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 31}

syntax [image_status, error_code] = scGET_IMAGE_STATUS
input none
output image_statusd [int32]: status of camera
bit 0 = 0: no scREAD_IMAGE... function is running
bit 0 = 1: busy, scREAD_IMAGE... is running
bit 1 = 0: image data are available in PCI board buffers
bit 1 = 1: no image data are available
bit 2 = 0: COC is idle, not running
bit 2 = 1: COC is running
bit 3 = 0: none or one buffer is full
bit 3 = 1: both buffers are full
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

Matlab4sensicam

scGET_COC_SETTING

This function reads the actual camera settings, which have been set using the scSET_COC() command. The returned values have the same format as described for scSET_COC().

{pco SDK for sensicam, dicam pro, hsfc pro manual page 32}

syntax	[mode, trig, roix1, roix2, roiy1, roiy2, hbin, vbin, table,...
	error_code] = scGET_COC_SETTING(table_in)
input	table_in [char array]: gives table od delay and exposure times in
		Milliseconds (sensicam) as strings,
		see SDK manual p. 10-24
output	mode [int32]: mode, defines camera type, gain status & oeprational
		behavior
	trig [int32]: sets camera trigger mode
	roix1 [int32]: start value for horizontal ROI
	roix2 [int32]: end value for horizontal ROI
	roiy1 [int32]: start value for vertical ROI
	roiy2 [int32]: end value for vertical ROI
	hbin [int32]: set horizontal binning
	vbin [int32]: set vertical binning
	table [char array]: table that contains delay and exposure times
		in Milliseconds (sensicam) as strings
	error_code [int32]: zero on success, nonzero indicates failure,
		returned value is the error code

scGET_COCTIME

This function gives the COC time in [µs].

{pco SDK for sensicam, dicam pro, hsfc pro manual page 32}

syntax	coc_time = scGET_COCTIME
input	none
output	coc_time [single]: COC time in [µs]

scGET_BELTIME

This function gives the total time: delay + exposure time in [µs].

{pco SDK for sensicam, dicam pro, hsfc pro manual page 32}

syntax	total_exposure_time = scGET_BELTIME
input	none
output	total_exposure_time [single]: delay + exposure time in [µs]

scGET_EXPTIME

This function gives the exposure time in [µs].

{pco SDK for sensicam, dicam pro, hsfc pro manual page 32}

syntax	exposure_time = scGET_EXPTIME
input	none
output	exposure_time [single]: exposure time in [µs]

scGET_DELTIME

This function gives the exposure time in [µs].

{pco SDK for sensicam, dicam pro, hsfc pro manual page 32}

syntax	delay_time = scGET_DELTIME
input	none
output	delay_time [single]: delay time in [µs]

scLOAD_OUTPUT_LUT

This function copies values of the look-up-table (LUT) into the internal look-up-table (LUT) memory used by the black & white convert functions of the SDK, which convert the pixel values from 12bit to 8bit, e.g. scREAD_IMAGE_8BIT(). The size of the allocated memory for the look-up-table must be at least 4kByte. Only the first 4kByte of the lokk-up-table are copied.

{pco SDK for sensicam, dicam pro, hsfc pro manual page 33}

syntax	error_code = scLOAD_OUTPUT_LUT(look_up_table)
input	look_up_table [uint8]: 1-dimensional vector containing the 4096
		values for a valid look-up_table
output	delay_time [single]: delay time in [µs]
	error_code [int32]: zero on success, nonzero indicates failure,
		returned value is the error code

scLOAD_COLOR_LUT

This function copies values of Red LUT, Blue LUT and Green LUT (LUT = look-up-table) to the corresponding internal LUT memory used by the color convert functions of the SDK, which convert the pixel values from 12bit to 3x8bit (RGB), e.g. scCONVERT_BUF_12TOCOL() or scREAD_IMAGE_COL. The size of the allocated memory for the LUT must be at least 4kByte. Only the first 4kByte of the LUT are copied.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 33}

syntax error_code = scLOAD_COLOR_LUT(Red_table, ... Green_table, Blue_table)

input Red_table [uint8]: 1-dimensional vector containing 4096 values for a valid Red look-up-table
Green_table [uint8]: 1-dimensional vector containing 4096 values for a valid Green look-up-table
Blue_table [uint8]: 1-dimensional vector containing 4096 values for a valid Blue look-up-table

output error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scLOAD_PSEUDO_COLOR_LUT

This function copies values of Red LUT, Blue LUT and Green LUT (LUT = look-up-table) to the corresponding internal LUT memory used by the pseudo color convert functions of the SDK, which convert the pixel values from 12bit to 3x8bit (RGB), e.g. scCONVERT_BUF_12TOCOL() or scREAD_IMAGE_COL. The size of the allocated memory for the LUT must be at least 256Byte. Only the first 256Byte of the LUT are copied.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 33}

syntax error_code = scLOAD_PSEUDO_COLOR_LUT(Red_table, ... Green_table, Blue_table)

input Red_table [uint8]: 1-dimensional vector containing 4096 values for a valid Red look-up-table
Green_table [uint8]: 1-dimensional vector containing 4096 values for a valid Green look-up-table
Blue_table [uint8]: 1-dimensional vector containing 4096 values for a valid Blue look-up-table

output error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scCONVERT_BUFFER_12TO8

When this function is called, a 16bit memory area (12bit pixel) with a size "width x height" in pixel is converted into 8bit memory area (8bit pixel) with the use of the internal black & white LUT.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 34}

syntax [image_8bit, error_code] = scCONVERT_BUFFER_12TO8(... convert_mode, image_12bit)

input convert_mode [int32]: convert mode, combination of the following flags
0x0000 - normal
0x0001 - flip (change columns)
0x0008 - mirror (change rows)
image_12bit [uint16]: input image, 16bit

output image_8bit [uint8]: output image, 8bit
error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

scCONVERT_BUFFER_12TOCOL

When this function is called, a 16bit memory area (12bit pixel) with a size "width x height" in pixel is converted into a COLOR memory area with 3 colors of 8bit each (BGR) using internal LUTs (look-up-tables) , which have been loaded with scLOAD_COLOR_LUT or scLOAD_PSEUDO_COLOR_LUT. The missing intermediate values of colors red, green and blue are interpolated.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 35}

syntax [image_8bit, error_code] = scCONVERT_BUFFER_12TOCOL(... convert_mode, image_12bit)

input convert_mode [int32]: convert mode, combination of the following flags
0x0000 - normal
0x0001 - flip (change columns)
0x0008 - mirror (change rows)

Matlab4sensicam

scCONVERT_BUFFER_12TOCOL

When this function is called, a 16bit memory area (12bit pixel) with a size "width x height" in pixel is converted into a COLOR memory area with 3 colors of 8bit each (BGR) using internal LUTs (look-up-tables) , which have been loaded with scLOAD_COLOR_LUT or scLOAD_PSEUDO_COLOR_LUT. The missing intermediate values of colors red, green and blue are interpolated.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 35}

syntax   [color_image, error_code] = scCONVERT_BUFFER_12TOCOL(...
                  convert_mode, image_12bit)
input   convert_mode [int32]: convert mode, combination of the following
                  flags
                  0x0000 - normal
                  0x0001 - flip (change columns)
                  0x0008 - mirror (change rows)
            image_12bit [uint16]: input image, 16bit
output   color_image [uint8]: output image, 8bit, converted
            error_code [int32]: zero on success, nonzero indicates failure,
                  returned value is the error code

scREAD_IMAGE_8BIT

This function reads an image with the selected "width" and "height" (in pixel) from the PCI interface board buffer, converts the data from 12bit to 8bit using the internal LUT (look-up-table) loaded with the scLOAD_OUTPUT_LUT. If the function was successful, the PCI interface board buffer containing the image is released and the image can't be read again. The number of bytes which are read equals "width x height". In "double shutter" mode the two half images are read as one data set of double height, when this function is called.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 36}

syntax   [result_image, error_code] = scREAD_IMAGE_8BIT(...
                  read_image_mode, image_width,...
                  image_height)
input   read_image_mode [int32]: convert mode, combination of the
                  following flags
                  0x0000 - normal
                  0x0001 - flip (change columns)
                  0x0008 - mirror (change rows)
            image_width [int32]: horizontal size of image
            image_height [int32]: vertical size of image
output   result_image [uint8]: 8bit image, which has been read
            error_code [int32]: zero on success, nonzero indicates failure,
                  returned value is the error code

scREAD_IMAGE_12BIT

This function reads an image with the selected "width" and "height" (in pixel) from the PCI interface board buffer, converts the data from 12bit to 8bit using the internal LUT (look-up-table) loaded with the scLOAD_OUTPUT_LUT. If the function was successful, the PCI interface board buffer containing the image is released and the image can't be read again. The number of bytes which are read equals "width x height". In "double shutter" mode the two half images are read as one data set of double height, when this function is called.
(for differences in Win95/98 and WinNT/2000 see SDK manual, p.37)
{pco SDK for sensicam, dicam pro, hsfc pro manual page 37}

syntax   [result_image, error_code] = scREAD_IMAGE_12BIT(...
                  read_image_mode, image_width,...
                  image_height)
input   read_image_mode [int32]: convert mode, combination of the
                  following flags
                  0x0000 - normal
                  0x0001 - flip (change columns)
                  0x0008 - mirror (change rows)
            image_width [int32]: horizontal size of image
            image_height [int32]: vertical size of image
output   result_image [uint16]: 12bit image, which has been read
            error_code [int32]: zero on success, nonzero indicates failure,
                  returned value is the error code

scREAD_IMAGE_COL

This function reads an image with the selected "width" and "height" (in pixel). Internally the READ_IMAGE_12BIT() is used to read data from the PCI interface board. The 12bit data are then converted into 3 (4) x 8bit data using the lUT (look-up-table) that has been loaded by the scLOAD_COLOR_LUT() command. The resulting array is changed to a 3 layer array, which directly can be used as a RGB image with functions like imview() or imshow(). If the function was successful, the PCI interface board buffer containing the image is released and the image can't be read again. The number of bytes which are read equals "3 (4) x width x height" depending on the 32bit flag. In "double shutter" mode the two half images are read as one data set of double height, when this function is called.
{pco SDK for sensicam, dicam pro, hsfc pro manual page 38}

syntax	[result_image, error_code] = scREAD_IMAGE_COL(... read_image_mode, image_width,... image_height)
input	read_image_mode [int32]: convert mode, combination of the following flags 0x0000 - normal 0x0001 - flip (change columns) 0x0002 - 32bit (convert BGR0) 0x0008 - mirror (change rows) 0x0010 - pseudo (convert via pseudo) 0x0020 - low_av (low average enable)
	image_width [int32]: horizontal size of image
	image_height [int32]: vertical size of image
output	result_image [uint16]: 12bit image, which has been read
	error_code [int32]: zero on success, nonzero indicates failure, returned value is the error code

Simple Matlab program...

The following simple Matlab program is a simple demonstration how to read out images from the camera and corresponds to the description on page 6 of the SDK manual:

a typical sequence is:

loadlibrary	- to have access to the DLL functions
scSET_BOARD	- set the appropiate PCI interface board
scSET_INIT	- maybe twice, one for stopping and one for starting
scGET_STATUS	- read status and type of camera
scTEST_COC	- define and test values for COC
scSET_COC	- set the camera adjustments
scRUN_COC	- start an exposure
scGET_IMAGE_STATUS	- check image status until exposure and image transfer are ready
scGET_IMAGE_SIZE	- read the resulting image size
scREAD_IMAGE_12BIT	- read image data
scSTOP_COC	- stop an exposure and reset image memory
scSET_INIT	- end the program

Matlab4sensicam

Simple Matlab program...

```
% load library "Senntcam.dll" & corresponding Header-File "Sencam.h"
loadlibrary('Senntcam','Sencam');
% set board number, in this case one board is installed => 0
% only necessary if multiple boards are used
board_number=0;
error_code = scSET_BOARD(board_number);
% in case a process is running, stop everything by scSET_INIT(0)
init_mode_terminate = int32(0);
error_code = scSET_INIT(init_mode_terminate);
% initialization of the camera and of the hardware
init_mode_initcam   = int32(1);
error_code = scSET_INIT(init_mode_initcam);
% call scGET_STATUS for reading the camera type as well as the CCD &
% electronics temperatures
[cam_type, temp_ele, temp_ccd, error_code] = scGET_STATUS;
% it makes sense to call a scTEST_COC before the scSET_COC is called,
% because in case of false value input, the corresponding return values
% will contain allowed values....
roix1 = 1;  % horizontal left limit of ROI
roix2 = 40; % horizontal right limit of ROI
roiy1 = 1;  % vertical lower limit of ROI
roiy2 = 32; % vertical upper limit of ROI
hbin = int32(1); % horizontal binning
vbin = int32(1); % vertical binning
exposure_str = num2str(1); % exposure time in ms as a string
delay_str = num2str(0);    % delay in ms as a string
exposure_table = [delay_str, ',', exposure_str, ', -1, -1'];
set_type = int32(4);
set_gain = int32(1);
set_submode = int32(0);
trig = int32(0);
[r_type, r_gain, r_submode, r_trig, r_roix1, r_roix2, r_roiy1, r_roiy2,...
          r_hbin, r_vbin, r_table, error_code] = scTEST_COC(set_type,...
          set_gain, set_submode, trig, roix1, roix2,...
          roiy1, roiy2, hbin, vbin, exposure_table);
% write back the results
set_type = int32(r_type);
set_gain = int32(r_gain);
set_submode = int32(r_submode);
trig = r_trig;
roix1 = r_roix1;
roix2 = r_roix2;
roiy1 = r_roiy1;
roiy2 = r_roiy2;
hbin = r_hbin;
vbin = r_vbin;
exposure_table = r_table;
% set the camera adjustments
error_code = scSET_COC(set_type, set_gain, set_submode, trig, roix1,
roix2,...
                        roiy1, roiy2, hbin, vbin, exposure_table);
% start an exposure
run_mode = 4; % single exposure
error_code = scRUN_COC(run_mode);
% Check for image status, to be sure that image is ready
image_busy = uint32(1);           % if b0001, then camera is busy
no_image_available = uint32(2);   % if b0010, then buffer empty
COC_runs = uint32(4);             % if b0100, then COC running
both_buffers_full = uint32(8);    % if b1000, then both buffers are full
bit_test = 1;
lower8bit_mask=uint32(255);
image_status = int32(99);
[ret_image_status, error_code] = scGET_IMAGE_STATUS;
while bit_test == 1
    [ret_image_status, error_code] = scGET_IMAGE_STATUS;
    ret_image_status = bitand(ret_image_status, lower8bit_mask);
    image_busy_test = bitand(ret_image_status, image_busy);
    no_image_available_test = bitand(ret_image_status, no_image_available);
    COC_runs_test = bitand(ret_image_status, COC_runs);
    both_buffers_full_test = bitand(image_status, both_buffers_full);
    if  (no_image_available_test == 0) & (image_busy_test ==0)
        bit_test=0;
    end
end
```

Matlab4sensicam

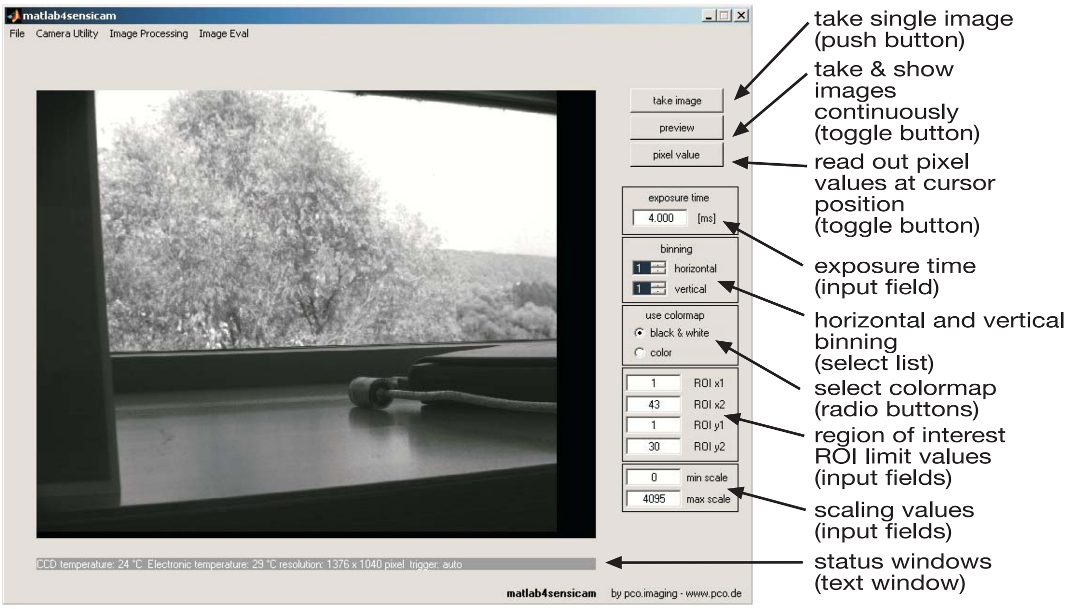
Simple Matlab program continued...

```
% if an image is available
% read the resulting image size
[image_width, image_height, error_code] = scGET_IMAGE_SIZE;
% read the image data
read_image_mode = int32(0);
[result_image, error_code] = scREAD_IMAGE_12BIT(read_image_mode, image_width,
image_height);
show_image = double(result_image);
show_image = mat2gray(show_image, [0 4095]);
%imview(show_image);
imshow(show_image);
% stop an exposure and reset image memory
error = scSTOP_COC;
% in case of presentation of a b/w image on the display
% ...scLOAD_OUTPUT_LUT
% ...scCONVERT_BUFFER_12TO8
% in case of an RGB image
% ...scLOAD_COLOR_LUT
% ...scCONVERT_BUF_12TOCOL
% or use scREAD_IMAGE_8BIT or scREAD_IAMGE_COL instead
% stop everything by scSET_INIT(0)
init_mode_terminate = int32(0);
error_code = scSET_INIT(init_mode_terminate);
```

Simple Matlab GUI program...

The simple Matlab GUI program is just a simple example for the integration of the camera functions with a small amount of image processing functions using the Matlab image processing toolbox. The follwing screen views describe the integrated functionality in the 2 files: matlab4sensicam.fig (the GUI-file) and matlab4sensicam.m (the necessary M-files with all functions):

matlab4sensicam - main window



exposure time 4.000 [ms]

binning: 1 horizontal, 1 vertical

use colormap: ● black & white ○ color

1 ROI x1
43 ROI x2
1 ROI y1
30 ROI y2

0 min scale
4095 max scale

CCD temperature: 24 °C Electronic temperature: 29 °C resolution: 1376 x 1040 pixel trigger: auto

take single image (push button)

take & show images continuously (toggle button)

read out pixel values at cursor position (toggle button)

exposure time (input field)

horizontal and vertical binning (select list)

select colormap (radio buttons)

region of interest ROI limit values (input fields)

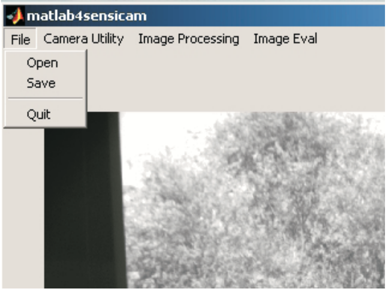
scaling values (input fields)

status windows (text window)

Matlab4sensicam

matlab4sensicam - menus

file menu

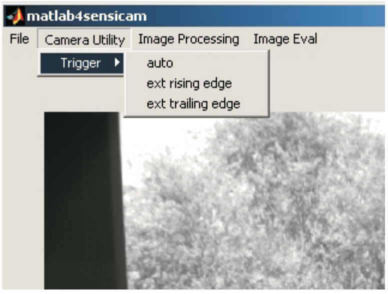


Open
opens and reads TIFF-files for display and processing

Save
saves actual image in TIFF, JPEG or PNG format

Quit
exits program matlab4sensicam

camera utility menu

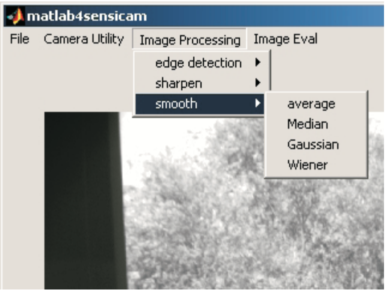


Trigger - auto
sets trigger mode to automatic (see SDK manual)

Trigger - ext rising edge
sets trigger mode to external rising edge trigger (see SDK manual)

Trigger - ext trailing edge
sets trigger mode to external trailing edge trigger (see SDK manual)

Image Processing menu

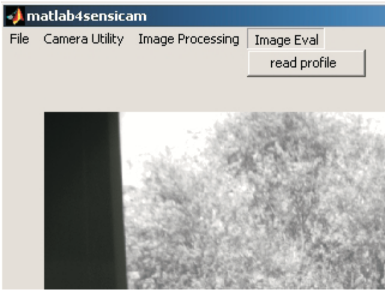


edge detection
Sobel or Prewitt operator applied to the current image

sharpen
unsharp masking applied to the current image

smooth
the current image is smoothed either by an average, Median, Gaussian or Wiener filter

Image Evaluation menu



read profile
calls improfile to read out pixel values along the profile, which is defined by cursor position and mouse click, a double click ends the profile selection

Troubleshooting...

In case you get the following error message by Matlab, when you try to load the libraries:

```
>> lcc preprocessor warning: c:\programme\digital camera toolbox\sensicam
sdk\Sencam.h:650 EOF inside comment
lcc preprocessor warning: c:\programme\digital camera toolbox\sensicam
sdk\Sencam.h:650 No newline at end of file
```

please open the Sencam.h in the corresponding directory:

```
"C:\Programme\Digital Camera ToolBox\SensiCam SDK"
```

with a standard text editor, scroll down to the end of the last line and press the carriage return / enter key of your keyboard twice and store the file onto the old one. Next time, the error message is disappeared.

please note - disclaimer:

The presented software is a free offer to pco ag customers and should faciliate the integration of pco camera systems into applications, which should use Matlab as controlling and image processing software.

pco ag assumes no responsibility for errors or omissions in these materials.

These materials are provided "as is" without warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

pco ag further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. pco shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. The information in this manual is subject to change without notice and does not represent a commitment on the part of pco ag in the future.

Matlab4sensicam

Information given by The Mathworks about how to install the GenericDLL.exe

As of MATLAB 6.5 (R13), it is possible to access functions defined in Windows standard dynamic linked libraries (.dll) through the MATLAB command line. This feature is only available for Windows 98/NT/2000/XP. If you are using MATLAB 6.5 (R13), you should download files needed to add this new functionality from the following site:
ftp://ftp.mathworks.com/pub/tech-support/solutions/s33513/genericDll.exe

To install:
1)Quit any current MATLAB sessions.

2)Download the genericDll.exe file (112 K). The download consists of a self-extracting executable that installs the necessary files into your MATLAB directory.

3)Save the downloadable to your local hard disk.

4)Double-click the executable to start the installation on your local PC. The WinZip Self-Extractor dialog opens, indicating the version of the update.

5)Read the comments in the WinZip Self-Extractor dialog, and then click OK. A dialog opens with the default location for MATLAB root as C:\MATLAB6p5. If MATLAB is installed in a different directory on your machine, specify that location instead.

6)Click the Unzip button to install the interface files in the appropriate directories. After the files are extracted and installed, a new dialog box opens and displays a message indicating that files have been successfully unzipped. The installation is complete. Click OK to close.

7)Exit the installation program by clicking Close in the WinZip Self-Extractor dialog box. Your interface to shared libraries is now ready for use.

8)If you have toolbox caching enabled, or if you obtain the following warning messages when you launch MATLAB

```
%%%BEGIN ERROR%%%
\\System\MATLAB\toolbox\toolboxname not found in Toolbox Path Cache
Warning: MATLAB Toolbox Path Cache is out of date and is not being used.
%%%END ERROR%%%
```

you will need to update the toolbox cache. For information on how to update the toolbox cache, refer to the following website:
http://www.mathworks.com/support/solutions/data/32237.shtml

The new feature allows you to load a Windows standard DLL into the MATLAB memory space and then access any of its functions. Because the DLL may contain functions programmed in languages other than C, the DLL must provide a C interface so that you can load and use the functions from within MATLAB.
Although data types differ between MATLAB and the C language environment used to program the DLL, in most cases, you can seamlessly pass MATLAB types to the functions in the library. MATLAB handles most of the data conversions necessary to marshal data to and from the DLL.

There are some limitations and restrictions that must be noted:
1)Currently, the MATLAB Interface to Shared Libraries is supported on Windows systems only. Windows shared library files have the file extension .dll.

2)Passing a void ** argument (that is, a pointer to a VOID pointer) to a function in a shared library is not supported in this release.

3)Passing a complex structure argument (that is, a structure constructed from other structures) to a shared library function is not supported in this release.

4)MATLAB does not currently support manipulation (e.g., addition, subtraction) of pointers returned by functions in a shared library.

For more information on how to use MATLAB to load and call shared library functions, and for specifics on data conversion, consult the PDF documentation shipped with the files. It is automatically installed in the $MATLAB/toolbox/matlab/general directory (where $MATLAB is your root MATLAB directory). The file is called shared_library_doc.pdf.

pco.
imaging