PHOENIX Library

API Reference Manual

Active Silicon

v3.6 6-Jan-2010

Revision History

Version	Comments
v1.0 26-Jun-2000	Initial release
v2.0 1-May-2002	Implemented error reporting. More support for exposure and acquisition trigger control.
v2.1 14-Jun-2002	New release.
v2.2 10-Oct-2002	Added *PHX_CAM_DATA_VALID* and *PHX_EVENT_CONTEXT* parameters
v2.3 7-Apr-2003	Implemented PHX_CameraConfigSave. Added *PHX_BUFFER_READY_COUNT*. Updated I/O port API.
v2.4 20-May-2003	Added *PHX_IO_METHOD_BIT_ACQTRIG* PhxIoMethods and *PHX_ACQTRIG_ALIGN* parameter.
v3.0 19-Jan-2005	New PHX_CameraConfigLoad options (including new board types). New PHX_Acquire commands. New *PHX_DST_FORMAT* values. Updates to *PHX_EXPTRIG_SRC* and *PHX_ACQTRIG_SRC*. Added : *PHX_BOARD_INFO*, *PHX_CAM_CLOCK_MAX*, *PHX_CHAN_SYNC_MODE*, *PHX_DATARATE_TEST*.
v3.1 25-Jul-2005	New PHX_Acquire command (*PHX_AUTO_WHITE_BALANCE, updated PHX_CHECK_AND_WAIT*). New Bayer *PHX_DST_FORMAT* values. Implemented *PHX_TIMEOUT_DMA* and *PHX_INTRPT_TIMEOUT*. Added : *PHX_CAM_SRC_COL*, *PHX_ACQ_BUFFER_START*. Updated *PHX_LINETRIG_TIMER_CTRL*.
v3.2 24-Aug-2005	Added *PHX_LUT_BYPASS*.
v3.3 17-Oct-2005	Renamed to API Reference Manual.
v3.4 26-Oct-2005	Updated *PHX_ACQTRIG_SRC*.
v3.5 16-Jul-2008	Updated the Example Code Fragment.
v3.6 6-Jan-2010	New US Office details.

Disclaimer

While every precaution has been taken in the preparation of this manual, Active Silicon assumes no responsibility for errors or omissions. Active Silicon reserves the right to change the specification of the product described within this manual and the manual itself at any time without notice and without obligation of Active Silicon to notify any person of such revisions or changes.

Copyright Notice

Trademarks

All trademarks and registered trademarks are the property of their respective owners.

Part Information

Part Number: PHX-MAN-API Version v3.6 6-Jan-2010

Contact Details

Web www.activesilicon.com
Sales sales@activesilicon.com
Support phoenix.support@activesilicon.com

Europe:

Active Silicon Limited
Pinewood Mews, Bond Close, Iver,
Bucks, SL0 0NA, UK.

Tel: +44 (0)1753 650600
Fax: +44 (0)1753 651661

North America:

Active Silicon.
73 Princeton Street, Suite 304,
North Chelmsford, MA 01863, USA

Tel +1 978 244 0490
Fax +1 978 244 0491

Table of Contents

Introduction

SCOPE

This document describes the functional specification of the PHOENIX software library. This library provides a platform independent interface to the acquisition and control features of the DIG36 Phoenix board (and future boards in the Phoenix family).

OVERALL CONCEPT

The DIG36 Phoenix digital camera image acquisition board is a highly configurable system allowing image capture from a wide range of digital cameras in many formats. Data widths from 8 to 36 bits are supported, and multiple boards can be chained together to permit simultaneous capture from more than one camera.

This software library provides the application programmer with a set of function calls, which permit flexible access to the advanced features of the Phoenix image acquisition system without in-depth knowledge of the underlying libraries/hardware.

DOCUMENT STRUCTURE

The document is broken down into the following major sections

- Function definitions. A detailed description of each of the functions in the PHOENIX API.
- Error return descriptions.
- Example code snippet. A small code segment to illustrate the operation of the PHOENIX API
- Configuration file format.

TYPE DEFINITIONS

The following basic types are used in this API document

ui32	A 32 bit unsigned integer.
tHandle	A 32 bit handle.
etStat	Enumerated error return, defined in section Error Handling

In addition, numerous enumerated types are used to pass parameters to functions, and these are defined within each of the separate function definitions.

RELATED DOCUMENTATION

Phoenix User Manual, PHX-MAN-USER.

Phoenix Display Library API Reference Manual, PDL-MAN-API.

Overview

The functionality of the PHOENIX library is subdivided into four sections; Initialisation, Control, Acquisition and Serial Communications.

Note that any functions illustrated in grey are not yet implemented.

INITIALISATION

PHX_CameraConfigLoad	Configure the hardware. If an optional configuration file is supplied, it can also initialise the hardware with user settings.
PHX_CameraConfigSave	Save the current hardware settings to a configuration file.
PHX_CameraRelease	Release the hardware and system resources.

CONTROL

PHX_ParameterGet	Read parameter values directly from the hardware or software libraries.
PHX_ParameterSet	Control all aspects of the acquisition system.

ACQUISITION

PHX_Acquire	Initiate and control the acquisition of video data. The destination of the data can be system memory, graphics memory, or other user addressable memory, such as slave PCI cards.

SERIAL COMMUNICATIONS

PHX_CommsReceive	Receive serial characters from the embedded serial port.
PHX_CommsTransmit	Send serial characters over the embedded serial ports.

PHX_CameraConfigLoad

USAGE

*etStat PHX_CameraConfigLoad (tHandle *phCamera, char* szConfigFile, etCamConfigLoad eCamConfig, void (*pFnErrorHandler) (const char *, etStat, const char *))*

ARGUMENTS

phCamera	Pointer to camera handle return value.
szConfigFile	Configuration file name.
eCamConfig	The board type, bus position and channel options.
pFnErrorHandler	Pointer to an error handler function.

DESCRIPTION

Finds and allocates a Phoenix board, and performs configuration. The board selection is determined by *eCamConfig*, which is a combination (i.e. bitwise OR) of the following values :

Physical board type: One of these values must be specified, and must correspond to the board type in any configuration file used (Parameter PHX_BOARD_TYPE).

PHX_DIGITAL	Any digital interface board (i.e. AS-PHX-D* or AS-PHX-B-D*).
PHX_D24CL_PCI32	Any AS-PHX-B-D24CL-PCI32 variant board.
PHX_D32_PCI32	Any AS-PHX-B-D32-PCI32 variant board.
PHX_D36_PCI32	Any AS-PHX-B-D36-PCI32 variant board.
PHX_D36_PCI64	Any AS-PHX-B-D36-PCI64 variant board.
PHX_D36_PCI64U	Any AS-PHX-B-D36-PCI64U variant board.
PHX_D48CL_PCI64	Any AS-PHX-B-D48CL-PCI64 variant board.
PHX_D48CL_PCI64U	Any AS-PHX-B-D48CL-PCI64U variant board.

Board number:

PHX_BOARD_AUTO	The first free Phoenix board available is allocated. A board with one channel already allocated is not considered to be free. Default.
PHX_BOARDX	Specify a particular Phoenix board. *X* is between 1 and *PHX_BOARD_MAX*.

Channel number:

PHX_CHANNEL_AUTO	The first free channel available is allocated. Default.
PHX_CHANNEL_Y	Specify a particular channel of the board. Currently this may be *PHX_CHANNEL_A* or *PHX_CHANNEL_B*.

Board mode:

PHX_MODE_NORMAL	Open both the acquisition engine and the Phoenix com port(s). Default
PHX_COMMS_ONLY	Open only the Phoenix com ports(s).
PHX_ACQ_ONLY	Open only the acquisition engine.

Upon successfully finding and allocating a board, *szConfigFile* is opened, and the configuration within used to setup the board. If *szConfigFile* is NULL default settings will be used. The format of configuration files is described in Configuration File Format. The settings can be queried using PHX_ParameterGet.

The handle created for the camera is returned in *phCamera*.

If *pFnErrorHandler* is *NULL*, then no error handler will be used.

A camera configuration is considered to be independent of the actual board architecture so that the same configuration can be used for various boards (i.e. CameraLink versions, or possible future versions of the hardware).

EXAMPLES

To just find and initialise the first board found with the camera configuration file provided, using the default supplied error handler.

```
PHX_CameraConfigLoad( &hCamera, "file.pcf", PHX_BOARD_AUTO | PHX_DIGITAL,
       &PHX_ErrHandlerDefault);
```

To find and initialise channel A on the first board found using the camera configuration file provided, without any error handler installed.

```
PHX_CameraConfigLoad( &hCamera, "file.pcf", PHX_BOARD_AUTO | PHX_CHANNEL_A |
       PHX_DIGITAL, NULL);
```

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CameraConfigSave

PHX_CameraConfigSave

USAGE

etStat PHX_CameraConfigSave(tHandle hCamera, char szConfigFile, etCamConfigSave eCamConfig)*

ARGUMENTS

hCamera	Camera handle.
szConfigFile	File name of the configuration file.
eCamConfig	Indicates which parameters are to be saved to the configuration file.

DESCRIPTION

This function saves the complete camera setup for the camera hCamera to the configuration file szConfigFile. The selection of parameters to be saved is determined by *eCamConfig*, which is a combination (i.e. bitwise OR) of the following values :

PHX_SAVE_CAM	Saves only the camera specific parameters.
PHX_SAVE_SYS	Saves only the system specific parameters.
PHX_SAVE_APP	Saves only the application specific parameters.
PHX_SAVE_ALL	Saves all three of the above types of parameters.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CameraConfigLoad

PHX_CameraRelease

USAGE

*etStat PHX_CameraRelease(tHandle *phCamera)*

ARGUMENTS

phCamera Pointer to camera handle.

DESCRIPTION

Deallocates the hardware and software resources associated with *phCamera*, and resets the camera handle value to “Null” handle.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CameraConfigLoad, PHX_CameraConfigSave

PHX_ParameterGet

USAGE

etStat PHX_ParameterGet(tHandle hCamera, etParam eParam, void *pvData);

ARGUMENTS

hCamera	Camera handle.
eParam	Parameter to return.
pvData	Pointer to data.

DESCRIPTION

For the camera *hCamera,* return the parameter information indicated by *eParam* in *pvdata*. See PHX_ParameterSet for the complete list of parameters that may be requested. Each of the parameter IDs that may be requested ONLY is described below.

Note that parameters shown in grey are not yet implemented.

BOARD INFORMATION

PHX_BOARD_INFO

Type : etBoardInfo

Returns information about the PCI bus slot, the camera interface, the chaining status, and the board capabilities. The bit settings for the return values are

PHX_BOARD_INFO_PCI_3V	PCI slot is 3.3V.
PHX_BOARD_INFO_PCI_5V	PCI slot is 5V.
PHX_BOARD_INFO_PCI_33M	PCI slot is 33MHz.
PHX_BOARD_INFO_PCI_66M	PCI slot is 66MHz.
PHX_BOARD_INFO_PCI_32B	PCI slot is 32bit.
PHX_BOARD_INFO_PCI_64B	PCI slot is 64bit.
PHX_BOARD_INFO_LVDS	LVDS camera interface.
PHX_BOARD_INFO_CL	Camera Link interface.
PHX_BOARD_INFO_CHAIN_MASTER	Chaining jumper is set to Master.
PHX_BOARD_INFO_CHAIN_SLAVE	Chaining jumper is set to Slave.
PHX_BOARD_INFO_BOARD_3V	Board is 3.3V compatible.
PHX_BOARD_INFO_BOARD_5V	Board is 5V compatible.
PHX_BOARD_INFO_BOARD_33M	Board is 33MHz compatible.
PHX_BOARD_INFO_BOARD_66M	Board is 66MHz compatible.
PHX_BOARD_INFO_BOARD_32B	Board is 32bit compatible.
PHX_BOARD_INFO_BOARD_64B	Board is 64bit compatible.

REVISION INFORMATION

PHX_REV_HW_MAJOR

Type : ui32

Hardware revision information i.e. 4 from 4.0.1 Range 0-65535.

PHX_REV_HW_MINOR

Type : ui32

Hardware revision information i.e. 0 from 4.0.1 Range 0-64.

PHX_REV_HW_SUBMINOR

Type : ui32

Hardware revision information i.e. 1 from 4.0.1 Range 0-64.

PHX_REV_HW

Type : ui32

Hardware revision information. Returns both PHX_REV_HW_MAJOR and PHX_REV_HW_MINOR.

The Format is :

BITS 31-16	BITS 15-8	BITS 7-0
PHX_REV_HW_MAJOR	PHX_REV_HW_MINOR	PHX_REV_HW_SUBMINOR

PHX_REV_SW_MAJOR

Type : ui32

Software revision information i.e. 4 from 4.0.1. Range 0-65535.

PHX_REV_SW_MINOR

Type : ui32

Software revision information i.e. 0 from 4.0.1. Range 0-64.

PHX_REV_SW_SUBMINOR

Type : ui32

Software revision information i.e. 1 from 4.0.1. Range 0-64.

PHX_REV_SW

Type : ui32

Software revision information. Returns PHX_REV_SW_MAJOR, PHX_REV_SW_MINOR and PHX_REV_SW_SUBMINOR.

The Format is :

BITS 31-16	BITS 15-8	BITS 7-0
PHX_REV_SW_MAJOR	PHX_REV_SW_MINOR	PHX_REV_SW_SUBMINOR

IMAGE INFORMATION

PHX_ROI_XLENGTH_SCALED

Type : ui32

The effective ROI X length (i.e. after subsampling) in pixels.

PHX_ROI_YLENGTH_SCALED

Type : ui32

The effective ROI Y length (i.e. after subsampling) in lines.

INPUT/OUTPUT

See the **I/O PORT ACCESS** section of the Phoenix User Manual for an overview of Absolute and Relative I/O port access, and the etPhxIoMethods used in the parameter values.

Note that PHX_IO_METHOD_READ is used with PHX_ParameterGet instead of PHX_IO_METHOD_WRITE.

PHX_IO_TTL

Type : ui32

The current value of the TTL port assigned to the current channel is returned in the low 8 bits. All other bits are 0.

PHX_IO_TTL_A

Type : ui32

The current value of the TTL port A is returned in the low 8 bits. All other bits are 0.

PHX_IO_TTL_B

Type : ui32

The current value of the TTL port B is returned in the low 8 bits. All other bits are 0.

PHX_IO_CCIO

Type : ui32

Variant : LVDS Phoenix board

Returns the status of the Camera Control I/O port assigned to the current channel. The parameter value passed to this function is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Camera Control I/O value in the low 2 bits. The options are:

PHX_IO_METHOD_READ | 0x0 (no low bits set)	The current value of the Camera Control I/O port assigned to the current channel is returned in the low 2 bits.
PHX_IO_METHOD_READ | 0x1 or 0x2 (only one low bit set)	The etPhxIoMethod for the specified bit is returned.
PHX_IO_METHOD_BIT_SET | 0x0 (no low bits set)	A ‘1’ in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_SET.
PHX_IO_METHOD_BIT_CLR | 0x0 (no low bits set)	A ‘1’ in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_CLR.
PHX_IO_METHOD_BIT_TIMER_POS | 0x0 (no low bits set)	A ‘1’ in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_POS.

PHX_IO_METHOD_BIT_TIMER_NEG | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_NEG.
PHX_IO_METHOD_BIT_ACQTRIG_POS | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_ACQTRIG_POS.
PHX_IO_METHOD_BIT_ACQTRIG_NEG | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_ACQTRIG_NEG.

PHX_IO_CCIO_A

Type : ui32

Variant : LVDS Phoenix board

Returns the status of Camera Control I/O port A. See ***PHX_IO_CCIO*** for the parameter values.

PHX_IO_CCIO_B

Type : ui32

Variant : LVDS Phoenix board

Returns the status of Camera Control I/O port b. See ***PHX_IO_CCIO*** for the parameter values.

PHX_IO_CCOUT

Type : ui32

Variant : Camera Link Phoenix board

Returns the status of the Camera Control output port assigned to the current channel. The parameter value passed to this function is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Camera Control output value in the low 4 bits. The options are:

PHX_IO_METHOD_READ | 0x0 (no low bits set)	The current value of the Camera Control I/O port assigned to the current channel is returned in the low 4 bits.
PHX_IO_METHOD_READ | 0x1 or 0x2 or 0x4 or 0x8 (only one low bit set)	The etPhxIoMethod for the specified bit is returned.
PHX_IO_METHOD_BIT_SET | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_SET.
PHX_IO_METHOD_BIT_CLR | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_CLR.
PHX_IO_METHOD_BIT_TIMER_POS | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_POS.
PHX_IO_METHOD_BIT_TIMER_NEG | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_NEG.
PHX_IO_METHOD_BIT_ACQTRIG_POS | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_ACQTRIG_POS.
PHX_IO_METHOD_BIT_ACQTRIG_NEG | 0x0 (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_ACQTRIG_NEG.

PHX_IO_CCOUT_A

Type : ui32

Variant : Camera Link Phoenix board

Returns the status of Camera Control output port A. See ***PHX_IO_CCOUT*** for the parameter values.

PHX_IO_CCOUT_B

Type : ui32

Variant : Camera Link Phoenix board

Returns the status of Camera Control output port B. See ***PHX_IO_CCOUT*** for the parameter values.

PHX_IO_OPTO_OUT

Type : ui32

The direction of each bit of the Opto port assigned to the current channel is returned in the low 2 bits. All other bits are 0. The return value is a combination (i.e. bitwise OR) of the following etParamValues:

PHX_IO_OPTO_OUT1	Indicates that Opto bit 1 is an output
PHX_IO_OPTO_OUT2	Indicates that Opto bit 2 is an output.

PHX_IO_OPTO_A_OUT

Type : ui32

The direction of each bit of Opto port A is returned in the low 2 bits. See ***PHX_IO_OPTO_OUT*** for the parameter values.

PHX_IO_OPTO_B_OUT

Type : ui32

The direction of each bit of Opto port B is returned in the low 2 bits. See ***PHX_IO_OPTO_OUT*** for the parameter values.

PHX_IO_OPTO

Type : ui32

Returns the status of the Opto port assigned to the current channel. The parameter value passed to this function is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Opto value in the low 2 bits. The options are:

PHX_IO_METHOD_READ (no low bits set)	The current value of the Opto port assigned to the current channel is returned in the low 2 bits.
PHX_IO_METHOD_READ | PHX_IO_OPTO1 or PHX_IO_METHOD_READ | PHX_IO_OPTO2 (only one low bit set)	The etPhxIoMethod for the specified bit is returned.
PHX_IO_METHOD_BIT_SET (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_SET.
PHX_IO_METHOD_BIT_CLR (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_CLR.
PHX_IO_METHOD_BIT_TIMER_POS (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_POS.
PHX_IO_METHOD_BIT_TIMER_NEG (no low bits set)	A '1' in any of the returned lower bit positions indicates that the bit was set with PHX_IO_METHOD_BIT_TIMER_NEG.

PHX_IO_OPTO_A

Type : ui32

Returns the status of Opto port A. See ***PHX_IO_OPTO*** for the parameter values.

PHX_IO_OPTO_B

Type : ui32

Returns the status of Opto port B. See ***PHX_IO_OPTO*** for the parameter values.

SERIAL COMMUNICATIONS

PHX_COMMS_STANDARD

Type : etParamValue

Returns the current serial interface communications standard as selected using hardware jumpers. Possible return values are

PHX_COMMS_STANDARD_RS232	RS232 comms.
PHX_COMMS_STANDARD_LVDS	LVDS comms.

PHX_COMMS_INCOMING

Type : ui32

Number of characters waiting in the serial interface receive buffer.

PHX_COMMS_OUTGOING

Type : ui32

Number of characters waiting in the serial interface transmit buffer.

ERROR INFORMATION

PHX_ERROR_FIRST_ERRNUM

Type : etStat

Returns the first error, which occurred since the last time this parameter, was queried.

PHX_ERROR_LAST_ERRNUM

Type : etStat

Returns the last error to have occurred.

PHX_ERROR_FIRST_ERRSTRING

Type : char**

Returns a string describing the first error, which occurred since the last time this parameter, was queried.

PHX_ERROR_LAST_ERRSTRING

Type : char**

Returns a string describing the last error to have occurred.

MISCELLANEOUS

PHX_NUM_BOARDS

Type : ui32

The number of boards being used for this camera.

PHX_STATUS

Type : etParamValue

Checks the current status of the hardware, live display, single snapshot etc. etc. Possible return values are

PHX_STATUS_ACQ_IN_PROGRESS	An acquisition is in progress
PHX_STATUS_WAITING_FOR_TRIGGER	Waiting for a trigger event
PHX_STATUS_IDLE	Idle

PHX_EVENTCOUNT

Type : ui32

The current value of the event counter. This counter can be programmed to count image lines or frames. See PHX_EVENTCOUNT_SRC, which determines the operating mode of the counter.

PHX_EVENTCOUNT_AT_GATE

Type : ui32

The gated value of the event counter. See PHX_EVENTGATE_SRC, which determines the gating signal used to enable counting of events.

PHX_BUFFER_READY_COUNTER

Type : ui32

The current value of the *BUFFER_READY* interrupt event counter. This counter decrements every time a *PHX_INTRPT_BUFFER_READY* event occurs. It is loaded with *PHX_BUFFER_READY_COUNT* when *PHX_Acquire* is called with *PHX_START*.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_ParameterSet

PHX_ParameterSet

USAGE

*etStat PHX_ParameterSet(tHandle hCamera, etParam eParam, void *pvData)*

ARGUMENTS

hCamera	Camera handle.
eParam	Parameter to set.
pvData	Pointer to the data to be set.

DESCRIPTION

For the camera *hCamera,* set the parameter information indicated by *eParam* to the contents of *pvdata*. Note that the PHX library uses a cache to optimise access to the hardware. Parameter updates are stored and any necessary hardware writes only occur when a call to PHX_ParameterSet is made with PHX_CACHE_FLUSH OR'd in with the parameter being set.

Note that parameters shown in grey are not yet implemented.

BASIC CAMERA SETTINGS

These parameters describe the physical characteristics of the camera. These are usually determined by consulting the technical specifications provided by the camera manufacturer.

PHX_CAM_TYPE

Type : etParamValue

Selects the type of camera connected. The options are

PHX_CAM_LINESCAN_ROI	Linescan mode with a region of interest.
PHX_CAM_LINESCAN_NO_ROI	Linescan mode no region of interest (i.e. datastream)
PHX_CAM_AREASCAN_ROI	Areascan mode with a region of interest. Default.
PHX_CAM_AREASCAN_NO_ROI	Areascan mode with no region of interest (i.e. datastream)

PHX_CAM_FORMAT

Type : etParamValue

Selects the camera data format. The options are

PHX_CAM_INTERLACED	Interlaced frame.
PHX_CAM_NON_INTERLACED	Non-interlaced frame. Default.

PHX_CAM_CLOCK_POLARITY

Type : etParamValue

Defines the polarity of the camera clock. The options are

PHX_CAM_CLOCK_POS	Image data read on the rising edge of pixel clock. Default.
PHX_CAM_CLOCK_NEG	Image data read on the falling edge of pixel clock.

PHX_CAM_CLOCK_MAX

Type : etParamValue

Defines which firmware design is used during board configuration. This parameter allows Phoenix to be compatible with Camera Link cameras with clock rates up to 85MHz for example. The options are

PHX_CAM_CLOCK_MAX_DEFAULT	Use the standard camera link firmware design. Default.
PHX_CAM_CLOCK_MAX_85MHZ	Use the 85MHz camera link firmware design.

PHX_CAM_SRC_COL

Type : etParamValue

Selects the camera colour format. The options are

PHX_CAM_SRC_MONO	Monochrome. Default.
PHX_CAM_SRC_RGB	Colour (three channel RGB).
PHX_CAM_SRC_BAY_RGGB	Bayer (Red first pixel).
PHX_CAM_SRC_BAY_GRBG	Bayer ($Green_r$ first pixel).
PHX_CAM_SRC_BAY_GBRG	Bayer ($Green_b$ first pixel).
PHX_CAM_SRC_BAY_BGGR	Bayer (Blue first pixel).

PHX_CAM_SRC_DEPTH

Type : ui32

Defines the depth of the image data output from the camera. Default 8.

PHX_CAM_ACTIVE_XOFFSET

Type : ui32

Defines the camera active area X offset. Default 0.

PHX_CAM_ACTIVE_YOFFSET

Type : ui32

Defines the camera active area Y offset. Default 0.

PHX_CAM_ACTIVE_XLENGTH

Type : ui32

Defines the camera active area X length. Default 640.

PHX_CAM_ACTIVE_YLENGTH

Type : ui32

Defines the camera active area Y length. Default 480.

PHX_CAM_DATA_VALID

Type : etParamValue

Defines how valid camera data is captured. Certain cameras (e.g. Camera Link) output a 'Data Enable' control signal to indicate when the camera data is valid. This parameter allows Phoenix to make use of such a control signal. The options are

PHX_DISABLE	All camera data is captured. Default.
PHX_ENABLE	Only valid camera data is captured.

TAP CONFIGURATION

See the **CAMERA TAP CONFIGURATION** section of the Phoenix User Manual for an overview of tap configuration.

Note that not all combinations of the following settings describe legal tap configurations and these will be flagged as an error e.g. 1 single horizontal tap with direction both.

PHX_CAM_HTAP_NUM

Type : ui32

Determines the number of horizontal taps. Default 1.

PHX_CAM_HTAP_DIR

Type : etParamValue

Determines the direction of the horizontal taps along the imaging array line. The options are

PHX_CAM_HTAP_LEFT	Taps moving left to right. Default.
PHX_CAM_HTAP_RIGHT	Taps moving right to left.
PHX_CAM_HTAP_CONVERGE	Taps moving towards each other.
PHX_CAM_HTAP_DIVERGE	Taps moving away from each other.

PHX_CAM_HTAP_TYPE

Type : etParamValue

Determines the pixel extraction method for the horizontal taps. The options are

PHX_CAM_HTAP_LINEAR	Pixels extracted in a linear fashion along the line, e.g. [0], [1], [2], [3], etc. Default.
PHX_CAM_HTAP_OFFSET	Pixels extracted from offsets along the line, e.g. [0,N], [1,N+1], [2,N+2], etc.
PHX_CAM_HTAP_ALTERNATE	Pixels extracted from alternate pixel positions along the line, e.g. [0,1], [2,3], [4,5], etc

PHX_CAM_HTAP_ORDER

Type : etParamValue

Determines the physical ordering of the horizontal taps

PHX_CAM_HTAP_ASCENDING	Lower numbered taps generate image data that is earlier in time
PHX_CAM_HTAP_DESCENDING	Lower numbered taps generate image data that is later in time

PHX_CAM_VTAP_NUM

Type : ui32

Determines the number of vertical taps. Default 1.

PHX_CAM_VTAP_DIR

Type : etParamValue

Determines the direction of the vertical taps over the imaging array. The options are

PHX_CAM_VTAP_TOP	Taps moving top to bottom. Default.
PHX_CAM_VTAP_BOTTOM	Taps moving bottom to top.
PHX_CAM_VTAP_BOTH	Taps moving towards each other.
PHX_CAM_VTAP_DIVERGE	Taps moving away from each other.

PHX_CAM_VTAP_TYPE

Type : etParamValue

Determines the line extraction method for the vertical taps. The options are

PHX_CAM_VTAP_LINEAR	Lines extracted in a linear fashion from the imaging array, e.g. [1], [2], [3], etc. Default.
PHX_CAM_VTAP_OFFSET	Lines extracted from offsets in the imaging array, e.g. [0,N], [1,N+1], [2,N+2], etc.
PHX_CAM_VTAP_ALTERNATE	Lines extracted alternately from the imaging array, e.g. [0,1], [2,3], [4,5], etc.

PHX_CAM_VTAP_ORDER

Type : etParamValue

Determines the physical ordering of the vertical taps

PHX_CAM_VTAP_ASCENDING	Lower numbered taps generate image data that is earlier in time
PHX_CAM_VTAP_DESCENDING	Lower numbered taps generate image data that is later in time

IMAGE DATA CONTROL

The following parameters are used to describe the format of the physical image data generated by the camera, and its relationship to that captured in memory.

See the **IMAGE DATA CONTROL** section of the Phoenix User Manual for more information.

PHX_ACQ_XSUB

Type : etParamValue

Defines the horizontal subsampling ratio. The options are

PHX_ACQ_X1	No pixel subsampling.
PHX_ACQ_X2	Subsample pixels by 2.
PHX_ACQ_X4	Subsample pixels by 4.
PHX_ACQ_X8	Subsample pixels by 8.

PHX_ACQ_YSUB

Type : etParamValue

Defines the vertical subsampling ratio. The options are

PHX_ACQ_X1	No line subsampling.

PHX_ACQ_X2	Subsample lines by 2.
PHX_ACQ_X4	Subsample lines by 4.
PHX_ACQ_X8	Subsample lines by 8.

PHX_ROI_SRC_XOFFSET

Type : ui32

Defines the source ROI X offset in image pixels. Default 0.

PHX_ROI_SRC_YOFFSET

Type : ui32

Defines the source ROI Y offset in image lines. Default 0.

PHX_ROI_DST_XOFFSET

Type : ui32

Defines the destination ROI X offset in image pixels. Default 0.

PHX_ROI_DST_YOFFSET

Type : ui32

Defines the destination ROI Y offset in image lines. Default 0.

PHX_ROI_XLENGTH

Type : ui32

Defines the ROI X length in image pixels. Default 640. Note that the image buffer size must be divisible by 4.

PHX_ROI_YLENGTH

Type : ui32

Defines the ROI Y length in image lines. Default 480.

PHX_BUF_DST_XLENGTH

Type : ui32

Defines the destination image buffer X length in bytes.

PHX_BUF_DST_YLENGTH

Type : ui32

Defines the destination image buffer Y length in lines.

PHX_DST_FORMAT

Type : etParamValue

Defines the destination image format. The options are

PHX_DST_FORMAT_Y8	8 bit mono.
PHX_DST_FORMAT_Y10	10 bit mono.
PHX_DST_FORMAT_Y12	12 bit mono.

PHX_DST_FORMAT_Y14	14 bit mono.
PHX_DST_FORMAT_Y16	16 bit mono.
PHX_DST_FORMAT_Y32	32 bit mono.
PHX_DST_FORMAT_Y36	36 bit mono.
PHX_DST_FORMAT_BAY8	8 bit Bayer (i.e. 8 bit pixels, each pixel is R, G_r, G_b or B).
PHX_DST_FORMAT_BAY10	10 bit Bayer.
PHX_DST_FORMAT_BAY12	12 bit Bayer.
PHX_DST_FORMAT_BAY14	14 bit Bayer.
PHX_DST_FORMAT_BAY16	16 bit Bayer.
PHX_DST_FORMAT_RGB15	15 bit RGB.
PHX_DST_FORMAT_RGB16	16 bit RGB.
PHX_DST_FORMAT_RGB24	24 bit RGB.
PHX_DST_FORMAT_RGB32	32 bit RGB.
PHX_DST_FORMAT_RGBX32	32 bit RGBX.
PHX_DST_FORMAT_XRGB32	32 bit XRGB.
PHX_DST_FORMAT_RGB48	48 bit RGB.
PHX_DST_FORMAT_BGR15	15 bit BGR.
PHX_DST_FORMAT_BGR16	16 bit BGR.
PHX_DST_FORMAT_BGR24	24 bit BGR.
PHX_DST_FORMAT_BGR32	32 bit BGR.
PHX_DST_FORMAT_BGRX32	32 bit BGRX.
PHX_DST_FORMAT_XBGR32	32 bit XBGR.
PHX_DST_FORMAT_BGR48	48 bit BGR.

PHX_DST_ENDIAN

Type : etParamValue

Defines the byte order of the data in the destination image buffer. The options are

PHX_DST_LITTLE_ENDIAN	Intel format (Default).
PHX_DST_BIG_ENDIAN	Motorola format.

PHX_DST_PTR_TYPE

Type : etParamValue

Determines the types of the image buffers. They can either be internally allocated by the library, or provided by the user (virtual or physical addresses). The options are

PHX_DST_PTR_INTERNAL	Internally (library) allocated buffers.
PHX_DST_PTR_USER_VIRT	User allocated buffers – virtual addresses.
PHX_DST_PTR_USER_PHYS	User allocated buffers – physical addresses.

If user buffers are declared, any internally allocated buffers are automatically destroyed.

PHX_DST_PTRS_VIRT

Type : stImageBuff*[]

Defines an array of pointers to stImageBuff structures. Each element in the array describes a users buffer. stImageBuff is defined as follows.

```
typedef struct {
   void *pvAddress;
   void *pvContext;
} stImageBuff;
```

pvAddress is the virtual address of the buffer.

pvContext can be set to return user defined data. For example, when using PDL, pvContext is used to return the handle of the display buffer that has been filled with acquisition data.

A NULL address pointer is used to designate the last structure in the array.

PHX_DST_PTRS_PHYS

Type : stImageBuff*[]

Defines an array of pointers to stImageBuff structures. Each element in the array describes a users buffer. stImageBuff is defined as follows.

```
typedef struct {
   void *pvAddress;
   void *pvContext;
} stImageBuff;
```

pvAddress is a pointer to a scatter gather list. The scatter gather list consists of a number of ui32 physical Address Length pairs, followed by two ui32 NULL values as terminators. For example :

```
{dwAddress1, dwLength1, dwAddress2, dwLength2, NULL, NULL}
```

defines a scatter gather list with two physical entries.

pvContext can be set to return user defined data.

A NULL address pointer is used to designate the last structure in the stImageBuff array.

PHX_CAM_XBINNING

Type : ui32

Enables binning of image pixels. When set to 1, full image resolution is produced. When set to 2 or greater, the camera combines neighbouring pixels to give better signal to noise ratio at the expense of resolution. Note that this feature is camera dependent. Default 1.

PHX_CAM_YBINNING

Type : ui32

Similar to PHX_CAM_YBINNING but applies to image lines. Default 1.

LOOK-UP TABLE CONTROL

See the **LOOK_UP TABLE CONTROL** section of the Phoenix User Manual for more information.

PHX_LUT_INFO

Type : struct tLutInfo *

Modify an internal LUT or declare a custom LUT.

PHX_LUT_COUNT

Type : ui32

Defines the number of logical LUTs to be used across a line of data. This setting is independent of the number of camera taps and colour bands, but is limited by the camera bit depth. The example values below are the number of LUTs available per tap for mono cameras, or per component (i.e. R, G, or B) for RGB cameras.

8 bit camera data	256 LUTs available
12 bit camera data	16 LUTs available
16 bit camera data	1 LUT available

Changing this value will force the libraries to reallocate memory (if internal LUTs are being used). Default 1 (i.e. the same LUT is used for every pixel in an image line)

PHX_LUT_CORRECT

Type : etParamValue

Turn on automatic switching of the LUT across the line. The number of available LUTs can be determined by calling PHX_ParameterGet with PHX_LUT_MAX_NUM.

PHX_DISABLE	Disable multiple LUT correction. Default.
PHX_ENABLE	Enable multiple LUT correction.

PHX_LUT_SHIFT

Type : i32

Defines a positive or negative bit shift which may be applied to the camera image data through the LUTs. The value must lie in the range –15 to +15. Default is 0.

PHX_LUT_BYPASS

Type : etParamValue

Allows the Look-up Tables to be bypassed.

PHX_DISABLE	Disable the LUT bypass mode. Default.
PHX_ENABLE	Enable the LUT bypass mode.

INPUT/OUTPUT CONTROL

See the **I/O PORT ACCESS** section of the Phoenix User Manual for an overview of Absolute and Relative I/O port access, and the etPhxIoMethods used in the parameter values.

PHX_IO_TTL_OUT

Type : etParamValue

Defines the direction of the TTL port assigned to the current channel. The options are

PHX_DISABLE	Input. Default.
PHX_ENABLE	Output.

PHX_IO_TTL_A_OUT

Type : etParamValue

Defines the direction of the TTL port A. See ***PHX_IO_TTL_OUT*** for the parameter values.

PHX_IO_TTL_B_OUT

Type : etParamValue

Defines the direction of the TTL port B. See ***PHX_IO_TTL_OUT*** for the parameter values.

PHX_IO_TTL

Type : ui32

Overwrites, sets or clears bits of the TTL port assigned to the current channel. The parameter value is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant TTL value in the bottom 8 bits. The port must be defined as an output. The etPhxIoMethods allowed are:

- PHX_IO_METHOD_WRITE
- PHX_IO_METHOD_BIT_SET / PHX_IO_METHOD_BIT_CLR

PHX_IO_TTL_A

Type : ui32

Overwrites, sets or clears bits of TTL port A. See ***PHX_IO_TTL*** for the parameter values.

PHX_IO_TTL_B

Type : ui32

Overwrites, sets or clears bits of TTL port B. See ***PHX_IO_TTL*** for the parameter values.

PHX_IO_OPTO

Type : ui32

Defines the behaviour of the Opto port assigned to the current channel. The parameter value is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Opto value in the bottom 2 bits. The relevant bits must be defined as outputs. The etPhxIoMethods allowed are:

- PHX_IO_METHOD_WRITE
- PHX_IO_METHOD_BIT_SET / PHX_IO_METHOD_BIT_CLR
- PHX_IO_METHOD_BIT_TIMER_POS / PHX_IO_METHOD_BIT_TIMER_NEG

PHX_IO_OPTO_A

Type : ui32

Defines the behaviour of Opto port A. See ***PHX_IO_OPTO*** for the parameter values.

PHX_IO_OPTO_B

Type : ui32

Defines the behaviour of Opto port B. See ***PHX_IO_OPTO*** for the parameter values.

PHX_IO_TIMER_1_PERIOD

Type : ui32

Defines the pulse width of the CcIo_1 signal sent to the camera to begin exposure. This signal is generated upon receiving either a line or acquisition trigger. The units are microseconds.

PHX_IO_TIMER_A1_PERIOD

Type : ui32

Defines the pulse width of the CcIo_A1 signal sent to the camera to begin exposure. This signal is generated upon receiving either a line or acquisition trigger. The units are microseconds.

PHX_IO_TIMER_B1_PERIOD

Type : ui32

Defines the pulse width of the CcIo_B1 signal sent to the camera to begin exposure. This signal is generated upon receiving either a line or acquisition trigger. The units are microseconds.

PHX_IO_TIMER_2_PERIOD

Type : ui32

Defines the delay from the deassertion of the CcIo_1 signal (start of exposure), to the deassertion of CcIo_2 (end of exposure). The units are microseconds.

PHX_IO_TIMER_A2_PERIOD

Type : ui32

Defines the delay from the deassertion of the CcIo_A1 signal (start of exposure), to the deassertion of CcIo_A2 (end of exposure). The units are microseconds.

PHX_IO_TIMER_B2_PERIOD

Type : ui32

Defines the delay from the deassertion of the CcIo_B1 signal (start of exposure), to the deassertion of CcIo_B2 (end of exposure). The units are microseconds.

PHX_IO_CCIO_OUT

Type : etParamValue

Variant : LVDS Phoenix board

Defines the direction of the Camera Control I/O port assigned to the current channel. The two signals on this port can be used either for exposure control outputs for the current camera (hardware control), or as general I/O (software control). The options are

PHX_DISABLE	Input. Default.
PHX_ENABLE	Output.

PHX_IO_CCIO_A_OUT

Type : etParamValue

Variant : LVDS Phoenix board

Defines the direction of the Camera Control I/O port A. The two signals on this port can be used either for exposure control outputs for camera 1 (hardware control), or as general I/O (software control). See ***PHX_IO_CCIO_OUT*** for the parameter values.

PHX_IO_CCIO_B_OUT

Type : etParamValue

Variant : LVDS Phoenix board

Defines the direction of the Camera Control I/O port B. The two signals on this port can be used either for exposure control outputs for camera 2 (hardware control), or as general I/O (software control). See ***PHX_IO_CCIO_OUT*** for the parameter values.

PHX_IO_CCIO

Type : ui32

Variant : LVDS Phoenix board

Defines the behaviour of the Camera Control I/O port assigned to the current channel. The parameter value is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Camera Control I/O value in the bottom 2 bits. The port must be defined as an output. The etPhxIoMethods allowed are :

- PHX_IO_METHOD_WRITE
- PHX_IO_METHOD_BIT_SET / PHX_IO_METHOD_BIT_CLR
- PHX_IO_METHOD_BIT_TIMER_POS / PHX_IO_METHOD_BIT_TIMER_NEG
- PHX_IO_METHOD_BIT_ACQTRIG_POS / PHX_IO_METHOD_BIT_ACQTRIG_NEG

PHX_IO_CCIO_A

Type : ui32

Variant : LVDS Phoenix board

Defines the behaviour of Camera Control I/O port A. See ***PHX_IO_CCIO*** for the parameter values.

PHX_IO_CCIO_B

Type : ui32

Variant : LVDS Phoenix board

Defines the behaviour of Camera Control I/O port B. See ***PHX_IO_CCIO*** for the parameter values.

PHX_IO_CCOUT

Type : ui32

Variant : Camera Link Phoenix board

Defines the behaviour of the Camera Control output port assigned to the current channel. The parameter value is a combination (i.e. bitwise OR) of an etPhxIoMethod and the relevant Camera Control output value in the bottom 4 bits. The etPhxIoMethods allowed are:

- PHX_IO_METHOD_WRITE
- PHX_IO_METHOD_BIT_SET / PHX_IO_METHOD_BIT_CLR
- PHX_IO_METHOD_BIT_TIMER_POS / PHX_IO_METHOD_BIT_TIMER_NEG
- PHX_IO_METHOD_BIT_ACQTRIG_POS / PHX_IO_METHOD_BIT_ACQTRIG_NEG

PHX_IO_CCOUT_A

Type : ui32

Variant : Camera Link Phoenix board

Defines the behaviour of Camera Control output port A. See ***PHX_IO_CCOUT*** for the parameter values.

PHX_IO_CCOUT_B

Type : ui32

Variant : Camera Link Phoenix board

Defines the behaviour of Camera Control output port B. See ***PHX_IO_CCOUT*** for the parameter values.

SERIAL COMMUNICATIONS

Phoenix has a standard UART which can be used to communicate with the camera or control external equipment. The following parameters are used to define the configuration of the UART.

PHX_COMMS_DATA

Type : etParamValue

Defines the number of bits in the serial data word. The options are

PHX_COMMS_DATA_5	5 data bits.
PHX_COMMS_DATA_6	6 data bits.
PHX_COMMS_DATA_7	7 data bits.
PHX_COMMS_DATA_8	8 data bits. Default.

PHX_COMMS_STOP

Type : etParamValue

Defines the number of stop bits to use. The options are

PHX_COMMS_STOP_1	1 stop bit. Default.
PHX_COMMS_STOP_1_5	1.5 stop bits.
PHX_COMMS_STOP_2	2 stop bits.

PHX_COMMS_PARITY

Type : etParamValue

Defines whether parity is used. The options are

PHX_COMMS_PARITY_EVEN	Even parity.
PHX_COMMS_PARITY_ODD	Odd parity.
PHX_COMMS_PARITY_NONE	No parity. Default.

PHX_COMMS_SPEED

Type : ui32

Defines the baud rate of the serial port. Default 9600.

PHX_COMMS_FLOW

Type : etParamValue

Defines the flow control method used. The options are

PHX_COMMS_FLOW_HW	Hardware flow control (RTS/CTS).
PHX_COMMS_FLOW_SW	Software flow control (XON/XOFF).
PHX_COMMS_FLOW_NONE	No flow control. Default.

The following parameters cause pre-set serial command streams to be transmitted to the camera at the specified time. The messages are defined in the INI file TBD.

PHX_COMMS_PREACQ

Type : char*

Defines a character string that is transmitted to the camera via the serial interface immediately prior to an acquisition.

PHX_COMMS_POSTACQ

Type : char*

Defines a character string that is transmitted to the camera via the serial interface immediately following a non-sequential acquisition i.e. after the users callback has completed. This parameter has no relevance in sequential mode.

ACQUISITION CONTROL

These parameters define the method of image acquisition.

PHX_ACQ_CONTINUOUS

Type : etParamValue

Defines the capture mode.

When disabled, Phoenix captures PHX_ACQ_NUM_IMAGES and then halts acquisition.

When enabled, acquisition continuously cycles through PHX_ACQ_NUM_IMAGES image buffers. If PHX_ACQ_BLOCKING is disabled, then successive buffers are overwritten, irrespective of whether they have been processed. If PHX_ACQ_BLOCKING is enabled, then acquisition only continues if the next buffer is available i.e. after a call to `PHX_Acquire( hCamera, PHX_BUFFER_RELEASE, NULL );`

PHX_DISABLE	Capture a single sequence.
PHX_ENABLE	Repeatedly capture the image sequence. Default.

PHX_ACQ_BLOCKING

Type : etParamValue

Controls the behaviour of the acquisition engine if PHX_ACQ_CONTINUOUS is enabled.

When PHX_ACQ_BLOCKING is enabled, Phoenix captures into the image buffers and only continues if the next buffer is available.

When disabled, Phoenix always captures into the next buffer in the image sequence, irrespective of whether that buffer has been processed.

PHX_DISABLE	Repeatedly capture the image sequence, overwriting data if necessary.
PHX_ENABLE	Capture the image sequence, and continue capturing if each buffer is available. Default.

PHX_ACQ_TYPE

Type : etParamValue

Defines which of the image fields is captured in area scan mode.

This parameter is only valid if PHX_CAM_FORMAT is PHX_CAM_INTERLACED.

The options are

PHX_ACQ_FRAME_12	Capture the next frame (interleave 1st field and 2nd field). Default.
PHX_ACQ_FRAME_21	Capture the next frame (interleave 2nd field and 1st field).
PHX_ACQ_FIELD_12	Capture fields sequentially in order 1st field, 2nd field.
PHX_ACQ_FIELD_21	Capture fields sequentially in order 2nd field, 1st field.

PHX_ACQ_FIELD_1	Capture 1st fields only.
PHX_ACQ_FIELD_2	Capture 2nd fields only.

PHX_ACQ_NUM_IMAGES

Type : ui32

Defines the total number of images Phoenix captures during acquisition. Each image is stored in its own capture buffer.

PHX_ACQ_BUFFER_START

Type : ui32

Defines the particular buffer of an image sequence that Phoenix will capture into first. Default 1.

PHX_ACQ_SKIP

Type : ui32

Defines the number of images to skip in at the start of a capture in continuous mode. Default 0.

PHX_DATASTREAM_VALID

Type : etParamValue

Determines the conditions for capturing image data. The options are

PHX_DATASTREAM_ALWAYS	Always capture image data. Default.
PHX_DATASTREAM_LINE_ONLY	Only capture image data when LINE is asserted.
PHX_DATASTREAM_FRAME_ONLY	Only capture image data when FRAME is asserted.
PHX_DATASTREAM_FRAME_AND_LINE	Only capture image data when FRAME AND LINE are both asserted.

PHX_TIMEOUT_TRIGGER

Type : ui32

Defines the maximum time between software enabling a capture and a trigger occurring. If this time is exceeded, an interrupt is generated.

PHX_TIMEOUT_CAPTURE

Type : ui32

Defines the maximum time between a trigger occurring and the start of data capture. If this time is exceeded, an interrupt is generated.

PHX_TIMEOUT_DMA

Type : ui32

Defines the maximum time between a trigger occurring and the end of data capture (i.e. the image buffer having been filled). If this time is exceeded, an interrupt is generated.

PHX_BUFFER_READY_COUNT

Type : ui32

This parameter allows an interrupt to be generated every *PHX_BUFFER_READY_COUNT* times that an image buffer is filled with data. For example, with a sequence capture of 10 images and a *PHX_BUFFER_READY_COUNT* of 2, 5 interrupts would be generated. Default 1.

PHX_INTRPT_SET, PHX_INTRPT_CLR

Type : etParamValue

Enables or disables each of the possible interrupt sources. Multiple interrupt sources can be enabled (either by ORing together the parameter values below, or by calling PHX_ParameterSet multiple times with the individual parameter values). By default, *PHX_INTRPT_BUFFER_READY* is enabled. *PHX_INTRPT_GLOBAL_ENABLE* is also enabled by default if not running under the DOS32 operating system. The options are

PHX_ INTRPT_TEST	This generates an interrupt immediately allowing callback functions to be exercised. The interrupt is automatically cleared in hardware.
PHX_INTRPT_BUFFER_READY	Generates an interrupt when an image buffer has been filled with data. e.g. with a sequence capture of 10 images, 10 interrupts would be generated.
PHX_ INTRPT_FIFO_OVERFLOW	Generates an interrupt when the current channel FIFO overflows.
PHX_ INTRPT_CAPTURE_COMPLETE	Generate an interrupt upon reaching the end of a sequence capture. Note this interrupt is not generated if continuous acquisition is enabled (see below).
PHX_ INTRPT_FRAME_START	Generate an interrupt when frame enable becomes active i.e. at the start of the cameras active area.
PHX_ INTRPT_FRAME_END	Generate an interrupt when frame enable becomes deasserted i.e. at the end of the cameras active area.
PHX_ INTRPT_LINE_START	Generate an interrupt at the start of each line.
PHX_ INTRPT_LINE_END	Generate an interrupt at the end of each line.
PHX_ INTRPT_ACQ_TRIG_START	Generate an interrupt when the selected acquisition trigger becomes asserted.
PHX_ INTRPT_ACQ_TRIG_END	Generate an interrupt when the selected acquisition trigger becomes deasserted.
PHX_INTRPT_INITIAL_DATA	Generate an interrupt when image data is first acquired into the capture buffer i.e. at the start of the ROI.
PHX_ INTRPT_TIMEOUT	Generate an interrupt when a timeout occurs.
PHX_ INTRPT_GLOBAL_ENABLE	A global setting which must be enabled for any of the above interrupts to be generated.

EXPOSURE AND ACQUISITION TRIGGER CONTROL

Exposure Control

See the **EXPOSURE CONTROL** section of the Phoenix User Manual for more information.

PHX_EXPTRIG_SRC

Type : etParamValue

Defines the source of the exposure control signal.

PHX_EXPTRIG_NONE	No Trigger. Default.
PHX_EXPTRIG_ACQTRIG	Exposure is started by the acquisition trigger.

PHX_EXPTRIG_SWTRIG	Exposure is started by the software trigger.
PHX_EXPTRIG_TIMER	Exposure is started by the internal timer.
PHX_EXPTRIG_AUXIN_1_RISING	Exposure is started by rising edge of AuxIn1. (relative)
PHX_EXPTRIG_AUXIN_1_FALLING	Exposure is started by falling edge of AuxIn1. (relative)
PHX_EXPTRIG_AUXIN_2_RISING	Exposure is started by rising edge of AuxIn2. (relative)
PHX_EXPTRIG_AUXIN_2_FALLING	Exposure is started by falling edge of AuxIn2. (relative)
PHX_EXPTRIG_AUXIN_A1_RISING	Exposure is started by rising edge of AuxInA1.
PHX_EXPTRIG_AUXIN_A1_FALLING	Exposure is started by falling edge of AuxInA1.
PHX_EXPTRIG_AUXIN_A2_RISING	Exposure is started by rising edge of AuxInA2.
PHX_EXPTRIG_AUXIN_A2_FALLING	Exposure is started by falling edge of AuxInA2.
PHX_EXPTRIG_AUXIN_B1_RISING	Exposure is started by rising edge of AuxInB1.
PHX_EXPTRIG_AUXIN_B1_FALLING	Exposure is started by falling edge of AuxInB1.
PHX_EXPTRIG_AUXIN_B2_RISING	Exposure is started by rising edge of AuxInB2.
PHX_EXPTRIG_AUXIN_B2_FALLING	Exposure is started by falling edge of AuxInB2.

PHX_EXP_LINESTART

Type : etParamValue

Defines the source of the signal used to indicate that the camera is outputting valid image data after an exposure. This is normally the camera's line enable signal, but since some cameras do not generate this, the end of exposure signal (CcIo_2) can be selected.

PHX_EXP_LINESTART_LINE	Valid image data is indicated by the camera's line enable signal. Default.
PHX_EXP_LINESTART_CCIO_2	Valid image data is indicated by the CcIo_2 signal. (relative)
PHX_EXP_LINESTART_CCIO_A2	Valid image data is indicated by the CcIo_A2 signal.
PHX_EXP_LINESTART_CCIO_B2	Valid image data is indicated by the CcIo_B2 signal.

PHX_LINETRIG_SRC

Type : etParamValue

Defines the source of the line trigger. This can be derived either from an external signal (frame or data enable) or from an internally generated pulse.

PHX_LINETRIG_NONE	No line trigger.
PHX_LINETRIG_AUXIN_1_RISING	Use the rising edge of AuxIn1 to begin exposure. (relative)
PHX_LINETRIG_AUXIN_1_FALLING	Use the falling edge of AuxIn1 to begin exposure. (relative)
PHX_LINETRIG_AUXIN_2_RISING	Use the rising edge of AuxIn2 to begin exposure. (relative)
PHX_LINETRIG_AUXIN_2_FALLING	Use the falling edge of AuxIn2 to begin exposure. (relative)
PHX_LINETRIG_AUXIN_A1_RISING	Use the rising edge of AuxInA1 to begin exposure.
PHX_LINETRIG_AUXIN_A1_FALLING	Use the falling edge of AuxInA1 to begin exposure.
PHX_LINETRIG_AUXIN_A2_RISING	Use the rising edge of AuxInA2 to begin exposure.
PHX_LINETRIG_AUXIN_A2_FALLING	Use the falling edge of AuxInA2 to begin exposure.

PHX_LINETRIG_AUXIN_B1_RISING	Use the rising edge of AuxInB1 to begin exposure.
PHX_LINETRIG_AUXIN_B1_FALLING	Use the falling edge of AuxInB1 to begin exposure.
PHX_LINETRIG_AUXIN_B2_RISING	Use the rising edge of AuxInB2 to begin exposure.
PHX_LINETRIG_AUXIN_B2_FALLING	Use the falling edge of AuxInB2 to begin exposure.
PHX_LINETRIG_TIMER	Use the internal timer to begin multiple exposures. Default.

PHX_LINETRIG_TIMER_CTRL

Type : etParamValue

Controls the mode of the line trigger timer. The options are

PHX_LINETRIG_TIMER_DISABLE	The line trigger timer is stopped. Default.
PHX_LINETRIG_TIMER_TIME	The line trigger timer counts microseconds. PHX_LINETRIG_SRC must be set to PHX_LINETRIG_TIMER.
PHX_LINETRIG_TIMER_LINES	The line trigger timer counts lines. PHX_LINETRIG_SRC must be set to one of the external signals (PHX_LINETRIG_AUXIN_*).

PHX_LINETRIG_TIMER_PERIOD

Type : ui32

Defines the period of the line trigger timer. The units are either microseconds or lines depending on the mode of PHX_LINETRIG_TIMER_CTRL.

Acquisition Trigger Control

The acquisition trigger (which defines when to actually begin acquiring camera image data) can be generated by several I/O sources. An internal programmable timer can be used to delay this acquisition by a number of lines or a time period (useful where the camera is physically offset from the trigger source).

The event counter keeps track of the total number of lines or frames. A selectable gating condition determines when to count the specified event. The event counter is reset as soon as this gating condition is true.

Note that there is a tolerance of ±1µs in all timings given below.

PHX_ACQTRIG_SRC

Type : etParamValue

Defines the trigger source. The options are

PHX_ACQTRIG_OPTO_1	Use Opto_1. (relative)
PHX_ACQTRIG_OPTO_2	Use Opto_2. (relative)
PHX_ACQTRIG_OPTO_A1	Use Opto_A1. Default.
PHX_ACQTRIG_OPTO_A2	Use Opto_A2.
PHX_ACQTRIG_OPTO_B1	Use Opto_B1.
PHX_ACQTRIG_OPTO_B2	Use Opto_B2.
PHX_ACQTRIG_AUXIN_1	Use AuxIn1. (relative)
PHX_ACQTRIG_AUXIN_2	Use AuxIn2. (relative)
PHX_ACQTRIG_AUXIN_A1	Use AuxInA1.

PHX_ACQTRIG_AUXIN_A2	Use AuxInA2.
PHX_ACQTRIG_AUXIN_B1	Use AuxInB1.
PHX_ACQTRIG_AUXIN_B2	Use AuxInB2.
PHX_ACQTRIG_CTRLIN_1	Use CtrlIn_1 (frame enable). (relative)
PHX_ACQTRIG_CTRLIN_2	Use CtrlIn_2 (line enable). (relative)
PHX_ACQTRIG_CTRLIN_3	Use CtrlIn_3 (data enable). (relative)
PHX_ACQTRIG_CTRLIN_A1	Use CtrlIn_A1 (frame enable A).
PHX_ACQTRIG_CTRLIN_A2	Use CtrlIn_A2 (line enable A).
PHX_ACQTRIG_CTRLIN_A3	Use CtrlIn_A3 (data enable A).
PHX_ACQTRIG_CTRLIN_B1	Use CtrlIn_B1 (frame enable B).
PHX_ACQTRIG_CTRLIN_B2	Use CtrlIn_B2 (line enable B).
PHX_ACQTRIG_CTRLIN_B3	Use CtrlIn_B3 (data enable B).
PHX_ACQTRIG_CCIO_1	Use CcIo_1. (relative)
PHX_ACQTRIG_CCIO_2	Use CcIo_2. (relative)
PHX_ACQTRIG_CCIO_A1	Use CcIo_A1.
PHX_ACQTRIG_CCIO_A2	Use CcIo_A2.
PHX_ACQTRIG_CCIO_B1	Use CcIo_B1.
PHX_ACQTRIG_CCIO_B2	Use CcIo_B2.
PHX_ACQTRIG_TIMER	Use the internal timer.

PHX_ACQTRIG_TYPE

Type : etParamValue

Defines the trigger type. The options are

PHX_ACQTRIG_NONE	No acquisition trigger. Default.
PHX_ACQTRIG_FIRST_POS_EDGE	Wait for a single positive edge of the trigger, before acquiring the first image in a sequence, then continue ignoring the trigger input.
PHX_ACQTRIG_FIRST_NEG_EDGE	Wait for a single negative edge of the trigger, before acquiring the first image in a sequence, then continue ignoring the trigger input.
PHX_ACQTRIG_EACH_POS_EDGE	Wait for a positive edge of the trigger, before acquiring each image in a sequence.
PHX_ACQTRIG_EACH_NEG_EDGE	Wait for a negative edge of the trigger, before acquiring each image in a sequence.
PHX_ACQTRIG_FIRST_POS_LEVEL	Wait for a single positive trigger level, before acquiring the first image in a sequence, then continue ignoring the trigger input.
PHX_ACQTRIG_FIRST_NEG_LEVEL	Wait for a single negative trigger level, before acquiring the first image in a sequence, then continue ignoring the trigger input.
PHX_ACQTRIG_EACH_POS_LEVEL	Wait for a positive trigger level, before acquiring each

	image in a sequence.
PHX_ACQTRIG_EACH_NEG_LEVEL	Wait for a negative trigger level, before acquiring each image in a sequence.
PHX_ACQTRIG_GATED_POS_LEVEL	Wait for a positive trigger level, then start acquisition and continue until the trigger goes negative.
PHX_ACQTRIG_GATED_NEG_LEVEL	Wait for a negative trigger level, then start acquisition and continue until the trigger goes positive.

PHX_ACQTRIG_ALIGN

Type : etParamValue

Defines the acquisition trigger alignment. The options are

PHX_ACQTRIG_ALIGN_NONE	No acquisition trigger alignment.
PHX_ACQTRIG_ALIGN_TO_CLK	The acquisition trigger is aligned to the pixel clock. Default.
PHX_ACQTRIG_ALIGN_TO_LINE	The acquisition trigger is aligned to the start of the line enable (i.e. rising edge).
PHX_ACQTRIG_ALIGN_TO_FRAME	The acquisition trigger is aligned to the start of the frame enable (i.e. rising edge).

PHX_ACQTRIG_DELAY_TYPE

Type : etParamValue

Defines the source of the clock used to increment the acquisition trigger delay timer. The options are

PHX_ACQTRIG_DELAY_NONE	No acquisition trigger delay. Default.
PHX_ACQTRIG_DELAY_LINE	The line enable signal from the camera is used to increment the counter.
PHX_ACQTRIG_DELAY_TIMER	An internal 1us clock is used to increment the counter

PHX_ACQTRIG_DELAY

Type : ui32

Defines the delay from receiving an external acquisition trigger to the start of image data capture. If PHX_ACQTRIG_DELAY_TYPE is PHX_ACQTRIG_DELAY_LINE, this is the number of lines to delay capture by. If it is PHX_ACQ_TRIG_DELAY_TIMER, it is in 1us increments. Default 0.

PHX_EVENTCOUNT_SRC

Type : etParamValue

Defines whether to count lines or frames. The options are

PHX_EVENTCOUNT_LINE	Count the number of elapsed lines. Default.
PHX_EVENTCOUNT_FRAME	Count the number of elapsed frames.
PHX_EVENTCOUNT_TIME	Count the number of elapsed microseconds.

PHX_EVENTGATE_SRC

Type : etParamValue

Defines the gating signal used to enable counting of events. The counter is reset when the gating signal is asserted. The options are

PHX_EVENTGATE_ACQTRIG	Count events when acquisition trigger is active.
PHX_EVENTGATE_FRAME	Count events when frame enable is active. Default.
PHX_EVENTGATE_LINE	Count events when line enable is active.
PHX_EVENTGATE_ACQ	Count events when image acquisition is in progress.

PHX_ACQ_CHAIN

Type : etParamValue

The chaining connector allows Phoenix boards to be used together to simultaneously acquire from wider sources. A Master board provides a system wide acquisition trigger for any Slave boards that are connected in the chain. When PHX_ACQ_CHAIN is enabled and an acquisition trigger is selected, the Slave boards acquire when the system acquisition trigger event has occurred.

When disabled, a board is assumed to be a master and operates in standalone mode. The chaining connector must not be fitted, or damage may occur if two boards try to drive the same signal.

PHX_DISABLE	Ignore setting of Master / Slave chain jumper. Board is standalone Master.
PHX_ENABLE	Allows Master / Slave operation. Default.

PHX_CHAN_SYNC_MODE

Type : etParamValue

Both channels on the same board can have their Acquisition & Exposure Triggers synchronised together. This implements the same functionality as the PHX_ACQ_CHAIN features does between two different boards. The options are

PHX_CHAN_SYNC_NONE	Channel triggers are not synchronised. Default.
PHX_CHAN_SYNC_ACQEXPTRIG	Channel triggers are synchronised.

MISCELLANEOUS SETTINGS

PHX_DATASRC

Type : etParamValue

Determines the source of image data. Data can either be captured from the camera, or from an internal image simulator (either a static or rolling grayscale ramp). The options are.

PHX_DATASRC_CAMERA	Capture image data from the camera. Default.
PHX_DATASRC_SIMULATOR_STATIC	Capture image data from the internal simulator. The image is a static 256x256 horizontal greyscale ramp.
PHX_DATASRC_SIMULATOR_ROLL	Capture image data from the internal simulator. The image is a rolling 256x256 horizontal greyscale ramp.

PHX_DUMMY_PARAM

Type : etParamValue

A dummy parameter which can be used to force a cache flush if the PHX_CACHE_FLUSH flag is set.

PHX_EVENT_CONTEXT

Type : void *

The third parameter of the interrupt event handler (i.e. callback).

PHX_DATARATE_TEST

Type : etParamValue

It can be useful to perform a data rate test in order to ascertain the sustained PCI bus bandwidth of a particular system, without having a camera attached to the Phoenix board. If this is enabled, then the DMA engine will generate data continuously, during acquisition. The options are

PHX_DISABLE	Use the DMA engine normally. Default.
PHX_ENABLE	Allow the DMA engine to run flat out.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_ParameterGet

PHX_Acquire

USAGE

*etStat PHX_Acquire(tHandle hCamera, etAcq eAcq, void *pvVar)*

ARGUMENTS

hCamera	Camera handle.
eAcq	Command to be performed.
pvVar	Command specific parameter

DESCRIPTION

The acquire function performs a number of operations

PHX_START	Start the acquisition process, then immediately return control to the user application. If pvVar is non-null, it is interpreted as a pointer to a callback function which is called when an event occurs (e.g. image capture complete). Otherwise, the user must subsequently call PHX_Acquire with either PHX_CHECK_AND_WAIT or PHX_CHECK_AND_RETURN to monitor the progress of the acquisition.
PHX_START_IMMEDIATELY	This is the same as PHX_START, except that a cache flush is not performed. This should only be called if no changes have been made to any Phoenix parameters since the previous cache flush.
PHX_CHECK_AND_WAIT	Wait indefinitely for an event to occur. Only return control to the users application when an event occurs. If pvVar is non-null, it is used as a pointer to return the event that occurred (if interrupts are enabled).
PHX_CHECK_AND_RETURN	Check if an event has occurred then immediately return control to the users application. On exit, pvVar is a pointer to a boolean flag which is set to TRUE if an event has occurred or FALSE otherwise.
PHX_STOP	Stop the acquisition after the current image is acquired.
PHX_ABORT	Abort the acquisition immediately.
PHX_BUFFER_GET	Obtain information describing the next unprocessed buffer containing captured image data. pvVar is a pointer to a (user allocated) stImageBuff structure (see PHX_DST_PTRS) which is filled in with the details of the buffer.
PHX_BUFFER_RELEASE	Indicate that the current image memory buffer has been processed, and can be re-used by the hardware. If not called when *PHX_ACQ_CONTINUOUS* and *PHX_ACQ_BLOCKING* are set to *PHX_ENABLE,* and all buffers have been filled, the hardware will stall acquisition to prevent buffers being overwritten with image data.
PHX_BUFFER_ABORT	This command aborts acquisition to the current buffer, but does NOT terminate the acquisition.
PHX_EXPOSE	This command initiates a software trigger of the exposure control signal.
PHX_UNLOCK	This command unlocks one or all of the DMA buffers allocated to a Phoenix capture buffer. pvVar is a pointer to the DMA buffer to be unlocked. If pvVar is null, all of the buffers are unlocked.

PHX_EVENT_HANDLER	This command allows the user to register a callback function, without starting an acquisition. pvVar is interpreted as a pointer to a callback function which is called when an event occurs (e.g. image capture complete).
PHX_AUTO_WHITE_BALANCE	This command allows the user to perform automatic white balancing for a Bayer camera source. The user must point the camera at a white background. This command will capture an image of the white background, analyse the colour content of the image and adjust the Phoenix Look Up Tables so that the image appears white.

The status of the current acquisition can be monitored by calling PHX_ParameterGet with eParam set to PHX_STATUS.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CommsTransmit

USAGE

etStat PHX_CommsTransmit(tHandle hCamera, char pcMessage, ui32* pdmNum, ui32 dmMsTimeout);*

ARGUMENTS

hCamera	Camera handle.
pcMessage	Message to be sent
pdmNum	Number of characters to send
dmMsTimeout	Timeout in milliseconds

DESCRIPTION

This function sends *p*dmNum* bytes from *pcMessage* to the camera *hCamera* via the serial communications port on the appropriate channel. If the hardware is running in dual camera mode the appropriate port will be selected.

If a data error occurs during transmission, or the timeout period *dmMsTimeout* is exceeded, no further characters will be transmitted, the transmit buffer will be flushed, and an interrupt generated.

The actual number of characters sent within the specified timeout period is returned in *p*dmNum*.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CommsReceive

PHX_CommsReceive

USAGE

etStat PHX_CommsReceive(tHandle hCamera, char pcMessage, ui32* pdmNum, ui32 dmMsTimeout);*

ARGUMENTS

hCamera	Camera handle.
pcMessage	Message buffer to read
pdmNum	Number of characters to read
dmMsTimeout	Timeout in milliseconds

DESCRIPTION

This function receives *dmNum* bytes from the camera *hCamera* via the serial communications port on the appropriate channel, and stores them in *pcMessage*. If the hardware is running in dual camera mode the appropriate port will be selected.

If a data error occurs during reception, or the timeout period *dmMsTimeout* is exceeded, no further characters will be received, the receive buffer will be flushed, and an interrupt generated.

The actual number of characters received within the specified timeout period is returned in *p*dmNum*.

RETURNS

See section Error Handling for a list of error returns.

SEE ALSO

PHX_CommsTransmit

Error Handling

All PHX functions return an etStat value. The following values are defined :

PHX_OK	Successful completion, without errors.
PHX_ERROR_BAD_PARAM	The eParam passed to the function is either unrecognised, or not supported for that function.
PHX_ERROR_BAD_PARAM_VALUE	The eParamValue passed to PHX_ParameterGet or PHX_ParameterSet functions is either unrecognised, or not supported for the eParam value.
PHX_ERROR_READ_ONLY_PARAM	The eParam passed to the PHX_ParameterSet function is read only.
PHX_ERROR_BAD_HANDLE	The handle value is invalid.
PHX_ERROR_OPEN_FAILED	Failed to find the hardware.
PHX_ERROR_INCOMPATIBLE	An illegal combination of parameters was encountered.
PHX_ERROR_HANDSHAKE	Serial communications handshake failure.
PHX_ERROR_INTERNAL_ERROR	An internal error has occurred in the library. Please contact Active Silicon for further support.
PHX_ERROR_OVERFLOW	A data overflow has occurred.
PHX_ERROR_NOT_IMPLEMENTED	The requested feature is not available in the current release of software.
PHX_ERROR_HW_PROBLEM	The hardware is not responding.
PHX_ERROR_NOT_SUPPORTED	The requested option is not supported. This can be caused by invalid options for the hardware, ie analogue sync settings for a PHX_DIGITAL product, or for features which are not implemented on the hardware variant, ie access to a second serial port.
PHX_ERROR_OUT_OF_RANGE	A parameter is not within the permitted range.
PHX_ERROR_MALLOC_FAILED	Failed to allocate system memory.
PHX_ERROR_SYSTEM_CALL_FAILED	A call to the underlying OS failed.
PHX_ERROR_FILE_OPEN_FAILED	Could not find or read file.
PHX_ERROR_FILE_CLOSE_FAILED	Could not close file after reading or writing.
PHX_ERROR_FILE_INVALID	The file contained errors.
PHX_ERROR_BAD_MEMBER	The parameter stored within the libraries is invalid. This is a specific case of PHX_ERROR_INTERNAL_ERROR. Please contact Active Silicon for further support.
PHX_ERROR_HW_NOT_CONFIGURED	The RISC code has not been generated.
PHX_ERROR_INVALID_FLASH_PROPERTIES	The onboard flash memory contains an invalid property string.
PHX_ERROR_ACQUISITION_STARTED	An attempt was made to start an acquisition when

	there was already one in progress.
PHX_ERROR_INVALID_POINTER	A pointer was passed to the library which does not reference valid memory.
PHX_ERROR_LIB_INCOMPATIBLE	An attempt was made to use an out of date software library with the PHX library.
PHX_ERROR_DISPLAY_CREATE_FAILED	PDL failed to create a display instance.
PHX_ERROR_DISPLAY_DESTROY_FAILED	PDL failed to destroy a display instance.
PHX_ERROR_DDRAW_INIT_FAILED	PDL failed to initialise a display instance.
PHX_ERROR_DISPLAY_BUFF_CREATE_FAILED	PDL failed to create a display buffer instance.
PHX_ERROR_DISPLAY_BUFF_DESTROY_FAILED	PDL failed to destroy a display buffer instance.
PHX_ERROR_DDRAW_OPERATION_FAILED	A PDL DirectDraw operation failed (e.g. blit)
PHX_WARNING_TIMEOUT	The requested operation has timed out.
PHX_WARNING_FLASH_RECONFIG	The onboard flash memory has been reprogrammed successfully with new firmware.
PHX_WARNING_ZBT_RECONFIG	The board firmware has been temporarily updated with new firmware.
PHX_WARNING_NOT_PHX_COM	No Phoenix OS specific Communications Ports have been found, e.g. win32 PHX serial ports.
PHX_WARNING_NO_PHX_BOARD_REGISTERED	No Phoenix boards have been registered with an OS specific structure, e.g. a win32 Registry entry was not found.

PHX_CameraConfigLoad takes a pointer to an error handler function. This gets called when an error is generated within the PHX library. The function has the following prototype

*void PHX_EXPORT_FN PHX_ErrHandlerDefault(const char *pszFnName, etStat eErrCode, const char *pszDescString);*

pszFnName – Name of the function in which the error occurred.

eErrCode – The error generated.

pszDescString – A descriptive string describing the error in more detail.

The user may provide his/her own custom error handler, or specify PHX_ErrHandlerDefault which is the built in PHX error handler. This function generates a popup message box (under Win32) describing the error.

The function PHX_ErrCodeDecode is provided to decode an etStat error code into a human readable string. The function is defined as follows

*void PHX_EXPORT_FN PHX_ErrCodeDecode(char *pszDescString, etStat eErrCode);*

pszDescString – Target descriptive string (user allocated)

eErrCode – Error code to decode.

Example Code Fragment

The following example demonstrates how to use the PHOENIX software API. Note that no error checking is performed on any of the function calls in this example! See section Configuration File Format for an example of a camera configuration file.

SINGLE BOARD SETUP FROM SCRATCH

Demonstrates how to create a configuration file for a specific camera.

```
tHandle hCamera = NULL;
etParamValue eParamValue;
ui32 dwParamValue;
ui32 dwDepth = 12;
ui32 dwXOffset = 2;
ui32 dwXLength = 1024;
ui32 dwYOffset = 2;
ui32 dwYLength = 1024;


PHX_CameraConfigLoad( &hCamera, NULL, PHX_BOARD_AUTO | PHX_DIGITAL, NULL );

eParamValue = PHX_CAM_AREASCAN_NO_ROI;
PHX_ParameterSet( hCamera, PHX_CAM_TYPE, &eParamValue );

PHX_ParameterSet( hCamera, PHX_CAM_SRC_DEPTH, &dwDepth );
eParamValue = PHX_ACQTRIG_OPTO1;
PHX_ParameterSet( hCamera, PHX_ACQTRIG_SRC, &eParamValue );
eParamValue = PHX_ACQTRIG_FIRST_POS_EDGE;
PHX_ParameterSet( hCamera, PHX_ACQTRIG_TYPE, &eParamValue );

// Active area
PHX_ParameterSet( hCamera, PHX_CAM_ACTIVE_XOFFSET, &dwXOffset );
PHX_ParameterSet( hCamera, PHX_CAM_ACTIVE_XLENGTH, &dwXLength );
PHX_ParameterSet( hCamera, PHX_CAM_ACTIVE_YOFFSET, &dwYOffset );
PHX_ParameterSet( hCamera, PHX_CAM_ACTIVE_YLENGTH, &dwYLength );

// Set up the I/O
eParamValue = PHX_ENABLE;
PHX_ParameterSet( hCamera, PHX_IO_TTL_A_OUT, &eParamValue );
dwParamValue = 19200;
PHX_ParameterSet( hCamera, PHX_COMMS_SPEED, &dwParamValue );

PHX_CameraConfigSave( hCamera, "camera.pcf", PHX_SAVE_ALL );

PHX_CameraRelease( &hCamera );
```

Configuration File Format

A configuration file is a text file consisting of a list of parameter and parameter values as specified in the PHX_ParameterSet and PHX_ParameterGet function definitions. A special parameter (only used within configuration files) is PHX_BOARD_TYPE. This can take either of the following parameter values

either PHX_DIGITAL or PHX_ANALOGUE

and is used to inform the library of the type of board being configured. This value must correspond to the board type specified in the call to PHX_CameraConfigLoad

Any blank line or line beginning with a # character is deemed to be a comment and is ignored.

An example file is shown below (for a Pulnix 9700 camera)

```
# ==========================
# Phoenix Configuration File
# Pulnix 9700
# ==========================

# Camera Specific Settings
# ------------------------
PHX_BOARD_TYPE,             PHX_DIGITAL
PHX_CAM_TYPE,               PHX_CAM_AREASCAN_ROI
PHX_CAM_SRC_DEPTH,          8
PHX_CAM_SRC_COL,            PHX_CAM_SRC_MONO
PHX_CAM_ACTIVE_XOFFSET,     58
PHX_CAM_ACTIVE_YOFFSET,     39
PHX_CAM_ACTIVE_XLENGTH,     768
PHX_CAM_ACTIVE_YLENGTH,     484
PHX_CAM_HTAP_NUM,           1
PHX_CAM_HTAP_DIR,           PHX_CAM_HTAP_LEFT
PHX_CAM_HTAP_TYPE,          PHX_CAM_HTAP_LINEAR
PHX_CAM_VTAP_NUM,           1
PHX_CAM_VTAP_DIR,           PHX_CAM_VTAP_TOP
PHX_CAM_VTAP_TYPE,          PHX_CAM_VTAP_LINEAR
PHX_COMMS_DATA,             PHX_COMMS_DATA_8
PHX_COMMS_STOP,             PHX_COMMS_STOP_1
PHX_COMMS_PARITY,           PHX_COMMS_PARITY_NONE
PHX_COMMS_SPEED,            9600
PHX_COMMS_FLOW,             PHX_COMMS_FLOW_NONE

# System Specific Settings
# ------------------------
PHX_IO_CCIO_OUT,            PHX_ENABLE
PHX_IO_CCIO,                PHX_IO_METHOD_BIT_CLR,          3

# Application Specific Settings
# -----------------------------
PHX_ACQ_NUM_IMAGES,         1
PHX_ROI_SRC_XOFFSET,        0
PHX_ROI_SRC_YOFFSET,        0
PHX_ROI_DST_XOFFSET,        0
PHX_ROI_DST_YOFFSET,        0
PHX_ROI_XLENGTH,            768
PHX_ROI_YLENGTH,            484
PHX_LUT_COUNT,              1
PHX_INTRPT_SET,             PHX_INTRPT_BUFFER_READY
PHX_INTRPT_SET,             PHX_INTRPT_GLOBAL_ENABLE
PHX_DST_PTR_TYPE,           PHX_DST_PTR_INTERNAL
```