DATA TRANSLATION

UM-13442-J

Frame Grabber SDK™ User's Manual

Ninth Edition
June, 2002

Data Translation, Inc.
100 Locke Drive
Marlboro, MA 01752-1192
(508) 481-3700
www.datatranslation.com
Fax: (508) 481-8620
E-mail: info@datx.com

Table of Contents

About this Manual

This manual describes how to get started using the Frame Grabber SDK™ (Software Development Kit) to develop application programs for frame grabber boards.

Intended Audience

This document is intended for engineers, scientists, technicians, or others responsible for developing imaging application programs using the Microsoft® C compiler. The Microsoft C compiler is included in the following development environments:

- Microsoft Visual C++®, Versions 1.5 and later, and
- Microsoft Developer's Studio™, Version 4.0 and later.

It is assumed that you are a proficient programmer, that you are experienced in programming in the Windows® operating environment on the IBM® PC or compatible computer platform, that you are familiar with imaging principles, and that you have clearly defined the requirements of your application.

What You Should Learn from this Manual

Using this manual, you should be able to successfully install the Frame Grabber SDK and get started writing an imaging application program.

This manual is intended to be used with the online help for the Frame Grabber SDK. The online help contains all the specific reference information for each of the functions and status codes.

Organization of this Manual

This manual is organized as follows:

- Chapter 1, "Getting Started," describes how to install the Frame Grabber SDK in Windows 95, Windows 98, Windows NT, Windows 2000, Windows Me (Millennium Edition), and Windows XP.
- Chapter 2, "Function Summary," summarizes the functions provided in the Frame Grabber SDK.
- Chapter 3, "Using the Frame Grabber SDK," describes the operations that you can perform using the Frame Grabber SDK.
- Chapter 4, "Programming Flowcharts," illustrates how to use the functions provided in the Frame Grabber SDK in a program.
- Chapter 5, "Product Support," describes how to get help if you have trouble using the Frame Grabber SDK.
- Appendix A, "Example Programs," describes the example programs included with the Frame Grabber SDK.
- An index completes this document.

Conventions Used in this Manual

The following conventions are used in this manual:

- Notes provide useful information or information that requires special emphasis, cautions provide information to help you avoid losing data or damaging your equipment, and warnings provide information to help you avoid catastrophic damage to yourself or your equipment.
- Items that you select or type are shown in **bold**. Function names are also shown in bold.
- Code fragments are shown in `courier font.`

Related Information

Refer to the following documentation for more information on using the Frame Grabber SDK:

- Frame Grabber SDK Online Help. This contains all the specific reference information for each of the functions and status codes provided by the Frame Grabber SDK. Refer to page 9 for information on how to open this help file.
- Board-specific documentation, which consists of a getting started manual and a user's manual. The getting started manual describes how to install the frame grabber board, how to install the device driver for the board, and how to get started using the board. The user's manual describes the features of the board and the capabilities supported by the device driver for the board. These manuals are on the Imaging OMNI CD™.

Refer to the following documentation for programming information:

- *Microsoft C Reference*, Document Number LN06515-1189, Microsoft Corporation, and *The C Programming Language*, Brian W. Kernighan and Dennis Ritchie, Prentice Hall, 1988, 1987 Bell Telephone Laboratories, Inc, ISBN 0-13-109950-7.
- For Windows NT programmers, the *Microsoft Windows NT Training Kit*, ISBN 1-55615-864-5, Microsoft Corporation.
- For Windows 95 and Windows 98 programmers, the *Microsoft Windows User's Guide*, Document Number SY06851-0290, Microsoft Corporation.
- For Windows 95 programmers, *Programming Windows 95,* Charles Petzold and Paul Yao, Microsoft Press, 1996, ISBN 1-55625-676-6.

Where to Get Help

Should you run into problems installing or using the Frame Grabber SDK, the Technical Support Department is available to provide technical assistance. Refer to Chapter 5, "Product Support," for information on how to contact the Technical Support Department. If you are outside the U.S. or Canada, call your local distributor, whose number is listed in your Data Translation® product handbook or contact the Data Translation Web site (www.datatranslation.com).

Getting Started

What is the Frame Grabber SDK?

The Frame Grabber SDK is a DLL (Dynamically Linked Library) that supports the programming of Data Translation PCI frame grabber boards under Microsoft Windows 95, Windows 98, Windows NT 4.0, Windows 2000, Windows Me, and Windows XP.

The Frame Grabber SDK contains the following types of functions:

- **Frame Grabber SDK standard functions** – These are a core group of common functions that allow you to perform standard operations. You can use these functions with all or most frame grabber boards. The Frame Grabber SDK standard functions are fully compatible with DT-Open Layers™, which is a set of standards for developing integrated, modular application programs under Windows.
- **Frame Grabber SDK extension functions** – These functions allow you to perform unique operations that are specific to a particular frame grabber board or to a small set of frame grabber boards.

  Currently, the following sets of extension functions are available:

  – *Color SDK extensions* – These functions are used by color frame grabber boards, such as the DT3130 Series, the DT3153, and the DT3154.

  – *DT3152 SDK extensions* – These functions are used by the DT3152 and DT3152-LS frame grabber boards.

  – *DT3157 SDK extensions* – These functions are used by the DT3157 frame grabber boards.

  – *Line-scan SDK extensions* – These functions are used by frame grabber boards that support line-scan operations, such as the DT3152-LS frame grabber board.

Because DT-Open Layers is modular and uses Windows DLLs, you can add support for a new frame grabber board at any time. Just add the new DT-Open Layers device driver, modify your code to incorporate the features of the new board (using any new Frame Grabber SDK standard functions or any Frame Grabber SDK extension functions), and then recompile the code. All calls to Frame Grabber SDK standard functions currently in your application program can remain untouched.

The Frame Grabber SDK currently supports the following Data Translation PCI frame grabber boards:

- **DT3155** – an industrial-accuracy monochrome frame grabber board.
- **DT3152** – a high-accuracy variable-scan monochrome frame grabber board.
- **DT3152-LS** – a high-accuracy line-scan/variable-scan monochrome frame grabber board.
- **DT3153** – a composite color frame grabber board.
- **DT3154** – a high-performance RGB color frame grabber board.
- **DT3157** – an RS-422 input, 8-bit to 16-bit, single-channel/dual-channel, monochrome frame grabber board for digital cameras.
- **DT3130 Series** – a family of basic-feature monochrome or color frame grabber boards. Some models provide three simultaneous camera inputs and/or isolated trigger input and strobe output signals.

The list of supported frame grabber boards and their associated extension functions is constantly expanding. Refer to the Data Translation web site (www.datatranslation.com) for the latest information.

Quick Start

The following is an overview of the tasks required to install and use the Frame Grabber SDK:

1. Make sure that your system meets the requirements for installing the Frame Grabber SDK. For more information, refer to the next section.
2. Install the Frame Grabber SDK. For more information, refer to page 5.
3. Use the example programs provided with the Frame Grabber SDK. For more information, refer to page 171.
4. Create your application program. For more information, refer to your C compiler documentation.

If you have problems installing or using the Frame Grabber SDK, refer to page 9 for information on opening the online help, or refer to page 165 for information on contacting the Data Translation Technical Support Department.

What You Need

To use the Frame Grabber SDK, you need the following items:

- Pentium-based computer with a minimum of 20 MB of RAM (more recommended),
- CD-ROM drive,
- Minimum of 5 MB of hard drive space,
- One or more supported frame grabber boards with appropriate cables, installed in a PCI slot,
- Video input source (camera),
- Microsoft Visual C++, version 2.0 or higher, and

- Microsoft SDK (or equivalent) and resource kit (recommended but not required).

Installing the Software

Before you install the Frame Grabber SDK, make sure that you have installed your frame grabber board and device driver. Refer to your board-specific documentation for installation instructions. In addition, if a previous version of the Frame Grabber SDK is installed on your system, it is recommended that you remove the old version before you continue with this procedure. For more information, refer to page 6.

To install the Frame Grabber SDK, perform the following steps:

1. Insert the Imaging OMNI CD into your CD-ROM drive.
2. Select **Start** from the Task Bar, then select **Run**.
   The Run dialog box appears.
3. In the **Command Line** edit box, enter **D:\LAUNCH.EXE.**
   If your CD-ROM is not in drive D:, enter the letter of the drive where your CD-ROM is located.
4. Click **OK**.
   The Imaging OMNI CD splash screen appears.
5. Click **Install Products**.
6. Click **Frame Grabber SDK**, then click **Next**.
   You are prompted for the installation folder.
7. Either accept the default installation directory or browse to a new directory, then click **Next**.
8. Either accept the default program group or enter a new program group, then click **Next**.
   The files are copied to the installation directory.
9. Click **Finish**.
10. Click **Main Menu**, then click **Exit**.

About the Distribution Files

The Frame Grabber SDK installation directory contains the following subdirectories:

- **Bin** – Contains the executable form of the Acquire to Host, and Passthru example programs. Acquire To Host (ACQ2HST.EXE) acquires a single image and stores it in memory. Passthru (PASSTHRU.EXE) performs a passthru operation, allowing you to see a continuous image; the image is not saved. Refer to Appendix A for more information on the example programs.
- **CamFiles** – Contains the DT3157 setup files for the 640 x 480 standard frame size, Kodak model 1.0 camera, Kodak model 1.6l camera, Pulnix model 9701 camera, and Pulnix model 1001 camera.
- **Examples** – Contains the source code for the Acquire to Host, DT Capture, and Passthru example programs.
- **Include** – Contains the include files required by projects that reference Frame Grabber SDK functions or structures.

  OLFGAPI.H, OLIMGAPI.H, and OLWINTYP.H are standard include files. Make sure that you add these include files to any projects that reference Frame Grabber SDK functions or structures.

  The DTCOLORSDK.H include file is currently used by the DT3153, DT3154, and DT3130 Series boards. Make sure that you add this include file to any projects that reference the Color SDK extension functions or structures.

  DT3152API.H and DT3152TYP.H are include files that are currently used by the DT3152 and DT3152-LS boards. Make sure that you add these include files to any projects that reference DT3152 SDK extension functions or structures.

DT3157API.H and DT3157TYP.H are include files that are currently used by the DT3157 board. Make sure that you add these include files to any projects that reference the DT3157 SDK extension functions or structures.

The DTLINESCAN.H include file is currently used by the DT3152-LS board. Make sure that you add this include file to any projects that reference the Line-Scan SDK extension functions or structures.

- **Lib** – Contains the import libraries that must be linked to programs that reference Frame Grabber SDK functions or structures.

  OLFG32.LIB and OLIMG32.LIB are standard import libraries. Make sure that you link all programs that reference Frame Grabber SDK functions or structures with these import libraries.

  The DTCOLORSDK.LIB import library is currently used by the DT3153, DT3154, and DT3130 Series boards. Make sure that you link this import library to any programs that reference the Color SDK extension functions or structures.

  The DT315232.LIB import library is currently used by the DT3152 and DT3152-LS boards. Make sure that you link this import library to any programs that reference DT3152 SDK extension functions or structures.

  The DT315732.LIB import library is currently used by the DT3157 board. Make sure that you link this import library to any programs that reference the DT3157 extension functions or structures.

  The DTLINESCAN.LIB import library is currently used by the DT3152-LS board. Make sure that you link this import library to any programs that reference Line-Scan SDK extension functions or structures.

About the Library Function Calling Conventions

The Frame Grabber SDK functions adhere to the Microsoft Pascal calling conventions. You can find prototypes for these functions in the include files (in the Data Translation, Inc\DT Frame Grabber 32 SDK\INCLUDE directory). It is recommended that you follow these calling conventions for proper operation.

Frame Grabber SDK functions return an value that indicates the status of the requested function. It is recommended that you check the return status value after each function call for an error condition.

Note: For detailed information on the error codes, refer to the Frame Grabber SDK online help.

Using the Online Help

This manual is intended to be used with the online help for the Frame Grabber SDK. The online help contains all the specific reference information for each of the Frame Grabber SDK functions and status codes.

To open the online help file, double-click **Frame Grabber SDK Help** from the Data Translation, Inc\Frame Grabber 32 SDK program group.

Removing the Software

If you are having problems with the Frame Grabber SDK or if you are upgrading from a previous version of the Frame Grabber SDK, you may want to remove the software and then reinstall it.To remove the Frame Grabber SDK, exit your programming environment, and then select the Uninstall program from the Data Translation, Inc\Frame Grabber 32 SDK program group. All applicable files and file settings are removed.

Function Summary

This chapter provides a summary of the Frame Grabber SDK functions. For more detailed information, refer to the Frame Grabber SDK online help. Refer to page 9 for information on opening the online help.

General Imaging Functions

Function	Description
OlImgOpenDevice	Opens the frame grabber board using its DT-Open Layers alias and returns an ID used to uniquely identify the board for future operations.
OlImgCloseDevice	Closes a frame grabber board previously opened using **OlImgOpenDevice**.
OlImgGetDeviceCount	Returns the number of DT-Open Layers frame grabber boards installed in the system.
OlImgGetDeviceInfo	Returns information about each frame grabber board installed in the system.
OlImgQueryDeviceCaps	Returns information about the general capabilities of the frame grabber board.
OlImgQueryTimeoutPeriod	Returns the timeout period, in seconds, used by the device driver.
OlImgSetTimeoutPeriod	Sets the timeout period, in seconds, used by the device driver.
OlImgGetStatusMessage	Translates a status code into a corresponding message.
OlImgReset	Stops all operations on the frame grabber board and reinitializes the board to its nominal state.
DtColorQueryInterface	For frame grabber boards that use the color SDK extensions, determines whether the specified board supports the specified interface.
Dt3157QueryDigitalCameraType	For frame grabber boards that use the DT3157 SDK extensions, returns the currently defined digital camera type.
Dt3157SetDigitalCameraType	For frame grabber boards that use the DT3157 SDK extensions, specifies the type of digital camera you are using.

Input Control Functions

Function	Description
OlFgQueryInputCaps	Returns information about the input capabilities and characteristics of the frame grabber board and device driver.
OlFgQueryInputControlValue	Returns the current value of a specified control for the specified input channel.
OlFgSetInputControlValue	Specifies the input settings for the frame grabber board.
OlFgQueryBasedSourceMode	Returns the current state of based source mode.
OlFgEnableBasedSourceMode	Specifies whether you want to use the setup of a particular base input channel for all input channels or whether you want a separate setup for each input channel.
OlFgQueryInputVideoSource	Returns the currently selected input channel.
OlFgSetInputVideoSource	Specifies the input channel providing the video signal.
OlFgReadInputLUT	Returns a range of values from the specified ILUT.
OlFgWriteInputLUT	Loads a set of values into the specified ILUT.
OlFgQueryTriggerInfo	Returns the current settings of the triggering capabilities when used for single-frame acquisitions.
OlFgSetTriggerInfo	Specifies the trigger transition for single-frame acquisitions.
OlFgQueryMultipleTriggerInfo	Returns the current settings of the triggering capabilities when used for multiple-frame acquisitions.

Input Control Functions (cont.)

Function	Description
OlFgSetMultipleTriggerInfo	Specifies the trigger transition and mode for multiple-frame acquisitions.
DtColorHardwareScaling	For frame grabber boards that use the color SDK extensions, returns the hardware scaling capabilities, returns the current values of the scale factors, or sets the scale factors.
DtColorImageParameters	For frame grabber boards that use the color SDK extensions, returns the capabilities of the color parameters (brightness, contrast, saturation, and hue), returns the current values of the color parameters, or sets the color parameters.
DtColorStorageMode	For frame grabber boards that use the color SDK extensions, returns the current storage mode, sets the storage mode, or determines whether the specified storage mode is supported.
DtColorSignalType	For frame grabber boards that use the color SDK extensions, returns the current video signal type, sets the video signal type, or determines whether the specified video signal type is supported.
DtColorSyncMasterMode	For frame grabber boards that use the color SDK extensions, enables/disables Sync Master mode, returns the current status of Sync Master mode, and determines whether Sync Master mode is supported.
Dt3152EnableSyncMasterMode	For frame grabber boards that use the DT3152 SDK extensions, enables/disables Sync Master mode.

Input Control Functions (cont.)

Function	Description
Dt3152QuerySyncMasterControlValue	For frame grabber boards that use the DT3152 SDK extensions, returns the current value of a Sync Master mode control.
Dt3152SetSyncMasterControlValue	For frame grabber boards that use the DT3152 SDK extensions, sets the current value of a Sync Master mode control.
Dt3157EnableSyncMasterMode	For frame grabber boards that use the DT3157 SDK extensions, enables/disables Sync Master mode.
Dt3157QuerySyncMasterControlValue	For frame grabber boards that use the DT3157 SDK extensions, returns the current value of a Sync Master mode control.
Dt3157SetSyncMasterControlValue	For frame grabber boards that use the DT3157 SDK extensions, sets the current value of a Sync Master mode control.
Dt3152QueryInputControlValue	For frame grabber boards that use the DT3152 SDK extensions, returns the current offset voltage, reference voltage, and gain value.
Dt3152SetInputControlValue	For frame grabber boards that use the DT3152 SDK extensions, specifies the offset voltage, reference voltage, and gain value.

Output Control Functions

Function	Description
OlFgQueryStrobeInfo	Returns the current settings of the type, duration, and polarity of the strobe output signal.
OlFgSetStrobeInfo	Specifies the type, duration, and polarity of the strobe output signal.
Dt3157EnableExposureMode	For frame grabber boards that use the DT3157 SDK extensions, enables the board to generate an exposure pulse to the camera.
Dt3157QueryExposure	For frame grabber boards that use the DT3157 SDK extensions, returns the duration and the polarity of the exposure output pulse.
Dt3157SetExposure	For frame grabber boards that use the DT3157 SDK extensions, sets the duration and the polarity of the exposure output pulse.
Dt3157ResetCamera	For frame grabber boards that use the DT3157 SDK extensions, sends a reset pulse to the attached camera.

Memory Allocation Function s

Function	Description
OlFgAllocateBuiltInFrame	Allocates a frame buffer to be used for image acquisition and returns the frame ID associated with the frame buffer.
OlFgDestroyFrame	Destroys a frame buffer previously created by **OlFgAllocateBuiltInFrame** and releases any resources associated with the frame ID.
OlFgQueryFrameInfo	Returns information about the specified frame buffer.
OlFgQueryMemoryCaps	Returns information about the memory management capabilities and characteristics of the frame grabber board and device driver.

Passthru Functio n s

Function	Description
OlFgQueryPassthruCaps	Returns information about the passthru capabilities and characteristics of the frame grabber board and device driver.
OlFgQueryPassthruScaling	Returns the current passthru scale factor.
OlFgSetPassthruScaling	Specifies the passthru scaling coordinates.
OlFgQueryPassthruSourceOrigin	Returns the current passthru source origin.
OlFgSetPassthruSourceOrigin	Sets the passthru source origin.
OlFgLoadDefaultPassthruLUT	Loads the passthru LUT with default grayscale values.
OlFgLoadPassthruLUT	Loads entries into the passthru LUT.
OlFgExtendPassthruPalette	Adds colors to system palette for display during a passthru operation.
OlFgStartSyncPassthruBitmap	Starts a synchronous bitmap passthru from the image source to the specified window.
OlFgStartAsyncPassthruBitmap	Starts an asynchronous bitmap passthru from the image source to the specified window.
OlFgStartAsyncPassthruEx	Starts a continuous-acquire passthru from the image source to the specified window and/or a set a frame buffers.
OlFgPassthruSnapShot	Copies the display memory contents to a frame buffer without stopping the passthru operation.
OlFgStopAsyncPassthru	Stops an asynchronous passthru operation that is currently in progress.

Overlay Functions

Function	Description
OlFgQueryDDICaps	Returns information about the DDI section of the frame grabber board.
OlFgEnableOverlays	Enables/disables the overlay capability.
OlFgCreateSurface	Creates an overlay surface.
OlFgEraseSurface	Erases an overlay surface so that you can change the current contents.
OlFgDestroySurface	Destroys an overlay surface created by **OlFgCreateSurface**.
OlFgGetSurfaceDC	Gets a surface device context (DC) so that you can draw images on the overlay surface.
OlFgReleaseSurfaceDC	Releases a surface DC retrieved by **OlFgGetSurfaceDC**.
OlFgSetOverlayColorKey	Sets the source keying color (the color that is replaced by the passthru image).
OlFgSetTranslucentOverlay	Specifies the state (translucent or opaque) of the overlay surface.
OlFgSetVisibleSurface	Specifies which surface to display.
OlFgAddOverlayToFrame	Copies the contents of an overlay surface into a previously acquired frame buffer.
OlFgGetPassthruSyncEvent	Returns a handle to a thread-synchronization event object.

Acquisition Functions

Function	Description
OlFgAcquireFrameToDevice	Acquires a single frame into device memory synchronously.
OlFgAcquireFrameToHost	Acquires a single frame into host memory synchronously.
OlFgAcquireMultipleToDevice	Acquires multiple frames into device memory synchronously.
OlFgAsyncAcquireFrameToDevice	Acquires a single frame into device memory asynchronously.
OlFgAsyncAcquireFrameToHost	Acquires a single frame into host memory asynchronously.
OlFgAsyncAcquireMultipleToDevice	Acquires multiple frames into device memory asynchronously.
OlFgIsAsyncAcquireJobDone	Determines whether an asynchronous acquisition has completed.
OlFgCancelAsyncAcquireJob	Cancels a previous asynchronous acquisition.

Display Functions

Function	Description
OlFgDrawAcquiredFrame	Draws an acquired frame on a window.
OlFgDrawAcquiredFrameEx	Draws an acquired frame (or part of an acquired frame) on a window.
DtColorDrawAcquiredFrame	For frame grabber boards that use the color SDK extensions, displays a single RGB32, RGB24, RGB16, RGB15, or monochrome frame (from a previous acquisition) in a display window.
DtColorExtractFrametoBuffer	For frame grabber boards that use the color SDK extensions, separates an RGB, triple-monochrome, or dual-monochrome image into up to three independent color planes.
DtColorDrawBuffer	For frame grabber boards that use the color SDK extensions, draws the contents of a user buffer, which contains an RGB, triple-monochrome, or dual-monochrome plane extracted using **DtColorExtractFrametoBuffer**, to a window.

Image Processing Function s

Function	Description
OlFgCopyFrameRect	Copies pixel data from within a rectangular region of a frame buffer to another rectangular region of a frame buffer.
OlFgReadContiguousPixels	Reads contiguous pixel data from a frame buffer and places the data linearly into a user buffer.
OlFgReadFrameRect	Reads the pixel data from a rectangular region within a frame buffer and places the data linearly into a user buffer.
OlFgReadPixelList	Reads arbitrarily-specified pixel data from a frame buffer and places the data linearly into a user buffer.
OlFgWriteContiguousPixels	Writes pixel data from a user buffer to contiguous pixel locations in a frame buffer.
OlFgWriteFrameRect	Writes pixel data from a user buffer to a rectangular region within a frame buffer.
OlFgWritePixelList	Writes pixel data from a user buffer to arbitrarily-specified pixel locations in a frame buffer.
OlFgMapFrame	Maps the specified frame buffer into an application's virtual address space.
OlFgUnmapFrame	Unmaps a frame buffer that was mapped using **OlFgMapFrame**.

Digital I/O Functio n s

Function	Description
OlFgQueryCameraControlCaps	Returns information about the camera control capabilities and characteristics of the frame grabber board and device driver.
OlFgSetDigitalOutputMask	Sets the digital output lines to the specified pattern.
DtColorDigitalIOControl	For frame grabber boards that use the color SDK extensions, sets the digital I/O configuration, returns the digital I/O configuration, returns the value of the digital input lines, or sets the values of the digital output lines.
Dt3157QueryDigitalIOConfiguration	For frame grabber boards that use the DT3157 SDK extensions, returns the configuration of the digital input and digital output lines.
Dt3157SetDigitalIOConfiguration	For frame grabber boards that use the DT3157 SDK extensions, configures the digital input/output lines.
Dt3157QueryDigitalIO	For frame grabber boards that use the DT3157 SDK extensions, returns the value of the digital input lines.
Dt3157SetDigitalIO	For frame grabber boards that use the DT3157 SDK extensions, writes to the digital output lines.

Line-Scan Functions

Function	Description
OlFgEnableLsMode	For frame grabber boards that use the line-scan SDK extensions, enables/disables line-scan mode.
OlFgGetLsDriveClkDiv	For frame grabber boards that use the line-scan SDK extensions, returns the value of the clock divider used to change the period of the line-sync output pulse and the integration output pulse.
OlFgSetLsDriveClkDiv	For frame grabber boards that use the line-scan SDK extensions, sets the value of the clock divider used to change the period of the line-sync output pulse and the integration output pulse.
OlFgGetLsIntegration	For frame grabber boards that use the line-scan SDK extensions, returns the number of pixel clock pulses that you want to occur before the specified edge of the integration output pulse becomes active.
OlFgSetLsIntegration	For frame grabber boards that use the line-scan SDK extensions, specifies the number of pixel clock pulses that you want to occur before the specified edge of the integration output pulse becomes active.
OlFgGetLsLineDrive	For frame grabber boards that use the line-scan SDK extensions, returns the number of pixel clock pulses that you want to occur before the specified edge of the line-sync output pulse becomes active.
OlFgSetLsLineDrive	For frame grabber boards that use the line-scan SDK extensions, specifies the number of pixel clock pulses that you want to occur before the specified edge of the line-sync output pulse becomes active.

Line-Scan Functions (cont.)

Function	Description
OlFgStartAsyncLsPassthru	For frame grabber boards that use the line-scan SDK extensions, starts a continuous-acquire passthru operation.
OlFgStopAsyncLsPassthru	For frame grabber boards that use the line-scan SDK extensions, stops a continuous-acquire passthru operation.
OlFgAcquireLines	For frame grabber boards that use the line-scan SDK extensions, acquires lines of data to host memory.
OlFgIsAcquireLinesDone	For frame grabber boards that use the line-scan SDK extensions, determines whether the line-scan acquisition has completed.
OlFgDrawAcquiredLines	For frame grabber boards that use the line-scan SDK extensions, draws the contents of a user-allocated buffer to a window.
OlFgGetLsDigIo	For frame grabber boards that use the line-scan SDK extensions, returns the value of the digital I/O lines.
OlFgSetLsDigIo	For frame grabber boards that use the line-scan SDK extensions, sets the value of the digital output lines.

Using the Frame Grabber SDK

This chapter provides conceptual information that describes the operations provided by the Frame Grabber SDK.

Use this information with the reference information provided in the Frame Grabber SDK online help when programming your frame grabber board; refer to page 9 for more information on opening the online help file.

Note: Not all operations are supported by all frame grabber boards. Refer to your board-specific documentation for information about the operations supported by your board.

Many of the features described in this chapter are not applicable if you are using line-scan mode. If you are using a frame grabber board that supports line-scan mode, refer to page 116 for specific information about performing line-scan operations.

System Operations

The Frame Grabber SDK provides functions that allow you to perform the following general system operations:

- Initializing a frame grabber board (see the next section),
- Determining a frame grabber board's capabilities (see page 30),
- Defining timeouts (see page 32),
- Setting the digital camera type (see page 32),
- Working with status codes (see page 33), and
- Releasing a frame grabber board (see page 34).

Initializing a Frame Grabber Board

To perform most operations, you must initialize the device driver for the specified frame grabber board using **OlImgOpenDevice**. This function requires an alias, which is the name you gave the board when you configured its device driver. You need one alias for each board. An alias allows you to access more than one board of the same type in your system. **OlImgOpenDevice** returns a unique device ID for the frame grabber board. You use the device ID to identify the board for all subsequent operations.

Before you initialize a frame grabber board, you can retrieve information about the boards in your system using **OlImgGetDeviceCount** and **OlImgGetDeviceInfo**. **OlImgGetDeviceCount** returns the number of DT-Open Layers frame grabber boards installed in your system. **OlImgGetDeviceInfo** returns the board type (monochrome or color), internal driver ID, alias, Data Translation product name, size of allocated device memory, and size of allocated linear memory for each DT-Open Layers frame grabber board in your system.

Note: Avoid using **OlImgGetDeviceInfo** with an open (initialized) frame grabber board. Once you have opened the board with **OlImgOpenDevice**, use **OlImgQueryDeviceCaps** to get information about the open board.

If, during operation, you want to stop all processes and reinitialize the frame grabber board to its nominal state, use **OlImgReset**. Note that when you call **OlImgReset**, any changes you made to your input settings are lost.

Determining a Board's Capabilities

To help you determine which capabilities your frame grabber board supports and the valid range for a particular capability, you can use one of the query functions provided by the Frame Grabber SDK. These query functions include the following:

- **OlImgQueryDeviceCaps** – Returns information about the general capabilities of the frame grabber board.
- **OlFgQueryMemoryCaps** – Returns information about the memory management capabilities and characteristics of the frame grabber board.
- **OlFgQueryInputCaps** – Returns information about the input capabilities and characteristics of the frame grabber board.
- **OlFgQueryCameraControlCaps** – Returns information about the camera control capabilities and characteristics of the frame grabber board.
- **OlFgQueryPassthruCaps** – Returns information about the passthru capabilities and characteristics of the frame grabber board.
- **OlFgQueryDDICaps** – Returns information about the DDI section of the frame grabber board.

- **DtColorQueryInterface**, **DtColorHardwareScaling**, **DtColorImageParameters**, **DtColorSignalType**, **DtColorSyncMasterMode**, and **DtColorStorageMode** – Return information about the color capabilities of the frame grabber board.

Most query functions use a set of query keys. By calling a query function with a particular query key, you can retrieve specific information about your board. Some query functions do not use query keys; they return information about one specific capability only.

The value returned by the query functions varies, depending on the query function and the query key. Some queries return a Boolean indicating yes or no, on or off, and so forth. Other queries return one or more numeric values. Still other queries return a set of bit flags indicating support for several features. When a set of bit flags is returned, if a particular flag is set, the associated function is supported by your frame grabber board.

Note: For some capabilities, the Frame Grabber SDK does not provide a query function. In these cases, refer to your board-specific documentation to determine whether the capability is supported by your frame grabber board.

As an example, if you want to review and modify the frame height, you can perform the following steps:

1. Use **OlFgQueryInputCaps** with the key OLC_FG_IC_DOES_FRAME_SELECT to determine if the frame grabber board supports setting the frame parameters.
2. Use **OlFgQueryInputCaps** with the key OLC_FG_IC_FRAME_HEIGHT_LIMITS to determine the range (high value, low value, granularity, and nominal value) allowed by the frame grabber board for frame height.

3. Use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_FRAME_HEIGHT to determine the current value of the frame height.
4. Use **OlFgSetInputControlValue** with the control OLC_FG_CTL_FRAME_HEIGHT to set a new value for the frame height.

Defining Timeouts

By default, the Frame Grabber SDK uses a timeout period of 10 seconds. This means that once you start an acquisition, the software waits 10 seconds for some kind of an action (such as an acquisition or trigger) to occur. If the action it expects does not complete within that time, the Frame Grabber SDK terminates the process and returns control to the application.

To change the timeout period, use **OlImgSetTimeoutPeriod** to set the amount of time, in seconds, that the frame grabber board should wait before terminating a process. To disable timeouts, set the timeout period to 0.

To determine the current timeout period setting, use **OlImgQueryTimeoutPeriod**.

Setting the Digital Camera Type

Some frame grabber boards that support digital cameras can accept 8-bit, 10-bit, 12-bit, 14-bit, or 16-bit monochrome video from one RS-422 differential digital input (single-channel mode) or 8-bit monochrome video from two RS-422 differential digital inputs (dual-channel mode). To specify the type of input provided by your camera, use **Dt3157SetDigitalCameraType**. To determine the currently defined camera type, use **Dt3157QueryDigitalCameraType**.

When using single-channel mode, the size of the image can be 2K pixels x 2K pixels, 4K pixels x 1K pixels, or 1K pixels x 4K pixels. When using dual-channel mode, the size of the image is always 1K pixels x 1K pixels. The image size is the size of the frame you want to digitize (maximum 4M pixels). For more information on establishing the size of the frame, refer to page 78.

Working with Status Codes

Most Frame Grabber SDK functions return an OLT_APISTATUS status value, which is an unsigned long value that indicates the status of the function call. Usually, if the status value is equal to OLC_STS_NORMAL, the operation was successful. Other status values can indicate the status of an operation, such as whether it has completed, or an error condition. It is recommended that you check the status value after each function call using the symbolic constants defined in the include files. You can retrieve the text string associated with a status value using **OlImgGetStatusMessage**.

Some Frame Grabber SDK functions return TRUE if the function completed successfully and FALSE if an error occurred.

For information about a specific OLT_APISTATUS status value or a returned error string, refer to the Frame Grabber SDK online help. Refer to page 9 for more information about opening the Frame Grabber SDK online help.

For an illustration of how to check the status values, refer to the C example programs (ACQ2HOST.C and PASSTHRU.MFC).

Note: Applications compiled using the large-model, C compiler automatically generate the correct interface code; other memory models require use of the include files to force FAR calls and pointers.

Releasing a Frame Grabber Board

When you finish acquiring images, release the frame grabber board using **OlImgCloseDevice**. Make sure that you close all open boards before you exit the program. Boards that are not explicitly closed can continue to use system resources (such as memory), making the resources unavailable until the system is restarted. Also, boards that are not explicitly closed may not be marked as available until the system is restarted.

Setting Up the Input Source

This section describes how to set up the following aspects of the video input signal:

- Based source mode (see the next section),
- Video signal type (see page 36),
- Video input channel (see page 39),
- Chrominance notch filter (see page 40),
- Video input signal (see page 42),
- Color definitions (see page 46),
- Sync signals (see page 49),
- Pixel clock (see page 60),
- Input look-up table (see page 62), and
- External trigger (see page 64).

Based Source Mode

Many frame grabber boards acquire image data from more than one video input source, such as a camera, simultaneously. However, the board can process incoming data from only one source at a time. If your board supports input operations, you must specify which input source (channel) to use to acquire image data.

Typically, whenever you select a new input channel, you must specify input settings for the new channel. If you want to use input settings that you have specified for another video input channel for the new channel, you can enable based source mode.

Use **OlFgEnableBasedSourceMode** both to enable based source mode and to specify the base (the channel whose input settings you want to use). The frame grabber board will use the input settings of the base for any subsequent channels you select.

To determine whether based source mode is enabled (and if so, what the base is), use **OlFgQueryBasedSourceMode**.

Note: As long as based source mode is enabled, you can access the input controls for the base input channel only. Attempting to read or modify the input controls for any other input channel results in an error until you disable based source mode.

Video Signal Type

The video signal type is typically one of the following:

- **Composite** – For monochrome frame grabber boards, the horizontal and vertical sync signals are combined into one signal. For color frame grabber boards, red, green, and blue image data are combined in one signal.
- **Variable-scan** – For monochrome frame grabber boards only. The horizontal and vertical sync signals reside on separate signals. Variable-scan is also referred to as slow-scan.
- **Y/C** – For color frame grabber boards only. Luminance (Y) and color (C) data are stored separately.
- **RGB** – For color frame grabber boards only. Red, green, and blue image data reside on separate signals.

In addition, some frame grabber boards can acquire images from nonstandard sources, such as line-scan, slow-scan, SEM, and high-resolution cameras. These nonstandard video sources provide their own control signals.

The following sections describe how to program the video signal type for monochrome and color frame grabber boards.

Monochrome Frame Grabber Boards

To determine whether your monochrome frame grabber board supports programming the video signal type, you can use **OlFgQueryInputCaps** to query the board. Table 1 lists the information about the video signal type that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 1: Getting Information About the Video Signal Type - Monochrome Frame Grabber Boards

Information	Query Key
Whether the board supports video signal types.	OLC_FG_IC_DOES_VIDEO_SELECT
The video signal types the board supports.	OLC_FG_IC_VIDEO_TYPE_LIMITS

The OLC_FG_IC_VIDEO_TYPE_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 2.

Table 2: Video Signal Type Bit Flags

Flag	Description
OLC_FG_VID_COMPOSITE	Video source is composite.
OLC_FG_VID_VARSCAN	Video source is variable-scan.

To set the video signal type for monochrome frame grabber boards, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_VIDEO_TYPE and the appropriate bit flag mask (see Table 2). To determine the current video signal type, use **OlFgQueryInputControlValue** with the same control.

If you specify a composite video signal type for a monochrome board, you must also specify the sync source (see page 50) and the sync threshold (see page 51). If you specify a variable-scan video input source, you can also modify the edge of the horizontal and vertical sync signals (see page 52).

Color Frame Grabber Boards

To determine whether your color frame grabber board supports programming the video signal type, you can query the board. Table 3 lists the information about the video signal type that you can query, the function to use to perform the query, and the controls to use to get the information.

Table 3: Getting Information About the Video Signal Type - Color Frame Grabber Boards

Information	Function	Control
Whether the board supports video signal types.	**DtColorQueryInterface**	OLC_QUERY_COLOR_INTERFACE_SIGNAL_TYPE
The video signal types the board supports.	**DtColorSignalType**	OLC_QUERY_CAPABILITY

The OLC_QUERY_COLOR_INTERFACE_SIGNAL_TYPE query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot program the signal type) or OLC_QUERY_COLOR_INTERFACE_SIGNAL_TYPE (if you can program the signal type).

The OLC_QUERY_CAPABILITY query returns either OLC_SIGNAL_UNSUPPORTED (if the specified signal type is not supported) or the constant associated with the specified signal type (if it is supported). The constants are described in Table 4.

Table 4: Video Signal Type Constants

Constant	Description
OLC_MONO_SIGNAL	A single monochrome signal is captured.
OLC_YC_SIGNAL	Luminance (Y) information and color (C) are stored separately.
OLC_COMPOSITE_SIGNAL	Red, green, and blue image data are combined into one signal.
OLC_RGB_SIGNAL	Red, green, and blue image data reside on separate signals.
OLC_TRIPLE_MONO_SIGNAL	Three monochrome signals (RVID, GVID, and BVID) are captured simultaneously.
OLC_DUAL_MONO_SIGNAL	Two monochrome signals (RVID and GVID) are captured simultaneously (progressive scan). Data from the BVID signal is ignored.

To set the video signal type for color frame grabber boards, use **DtColorSignalType** with the control OLC_WRITE_CONTROL and the appropriate constant (see Table 4). To determine the current video signal type, use **DtColorSignalType** with the control OLC_READ_CONTROL.

Video Input Channel

To determine if your board supports input operations, use **OlImgQueryDeviceCaps** with the key OLC_IMG_DC_SECTIONS. If the OLC_FG_SECTION_INPUT flag is set in the returned set of bit flags, your board supports input operations.

The number of video input channels available depends on your frame grabber board. To determine the number of video input channels available on your frame grabber board, use **OlFgQueryInputCaps** with the key OLC_FG_IC_INPUT_SOURCE_COUNT.

To specify the input channel to use to acquire video data during the current acquisition, use **OlFgSetInputVideoSource**. To determine the currently specified input channel, use **OlFgQueryInputVideoSource**.

If based source mode is disabled when you select a new video input channel, the frame grabber board reloads the default input settings. If based source mode is enabled when you select a new video input channel, the board uses the input settings of the specified base for the new input channel. For more information about based source mode, refer to page 35.

Chrominance Notch Filter

While it is possible to acquire monochrome images from color signals, the color content of these signals can cause interference patterns that degrade the image. Most monochrome frame grabber boards provide a chrominance (or notch) filter that removes color information for cleaner acquisition and more accurate analysis.

If you are using an AC-coupled video signal that has chrominance information on it, as is the case with the NTSC and PAL video formats, you can apply a chrominance notch filter to remove the chrominance information. For frame grabber boards that support filtering, the chrominance notch filter for 60 Hz is set to approximately 3.5 MHz, while the filter for 50 Hz is set to approximately 4.28 MHz. Refer to your board-specific documentation for the exact values.

If you are using a DC-coupled video signal, you must set the **DC Filter - None** flag.

To determine whether your frame grabber board supports programming the input filter, you can use **OlFgQueryInputCaps** to query the board. Table 5 lists the information about the input filter that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 5: Getting Information About the Input Filter

Information	Query Key
Whether the frame grabber board supports input filters.	OLC_FG_IC_DOES_INPUT_FILTER
The input filters the board supports.	OLC_FG_IC_INPUT_FILTER_LIMITS

The OL_FG_IC_INPUT_FILTER_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 6.

Table 6: Input Filter Bit Flags

Flag	Description
OLC_FG_FILT_AC_NONE	AC coupled, no filter.
OLC_FG_FILT_AC_50	AC coupled, 50 Hz filter.
OLC_FG_FILT_AC_60	AC coupled, 60 Hz filter.
OLC_FG_FILT_DC_NONE	DC coupled, no filter.

To select a filter (or to turn filtering off), use **OlFgSetInputControlValue** with the control OLC_FG_CTL_INPUT_FILTER and the appropriate bit flag mask (see Table 6). To determine the current filter type, use **OlFgQueryInputControlValue** with the same control.

Video Input Signal

For monochrome frame grabber boards, you can improve the resolution of your acquired image by specifying the portion of the video input signal that you want to digitize. You can do this in one of the following ways:

- For boards that support a gain of 1 only, specify a black level and a white level. For more information, refer to the next section.
- For boards that support multiple gains, specify an offset, gain, and reference value. For more information, refer to page 45.

Black and White Levels

Black level is defined as the voltage below which all other voltages are digitized to black. White level is defined as the voltage above which all other voltages are digitized to white. For ease of use, both of these voltages are measured at the camera's output.

The voltages between the black level and the white level are digitized so that they are evenly distributed throughout the range of the A/D converter. For example, in Figure 1, the image data below the black level (0.0 V) is digitized to black and the image data above the white level (1.0 V) is digitized to white.

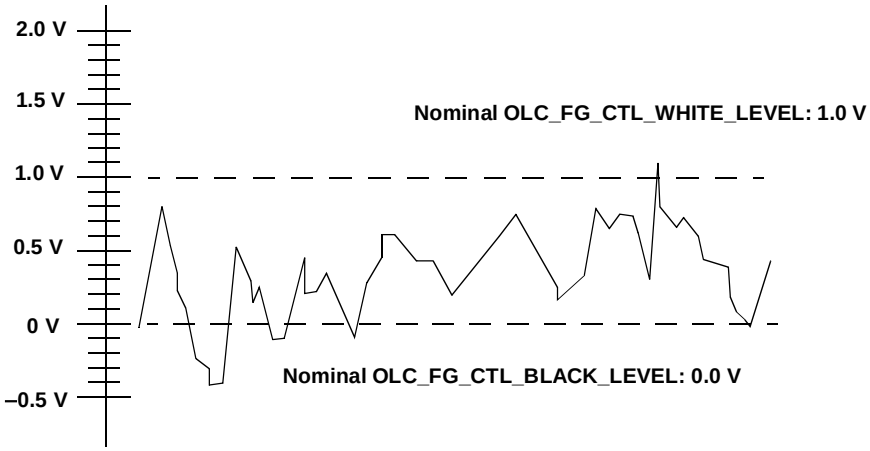


Figure 1: Original Signal

If the data that falls below the black level or above the white level is important, you can adjust the black level and white level so that the entire signal is within the digitized range. This is shown in Figure 2.

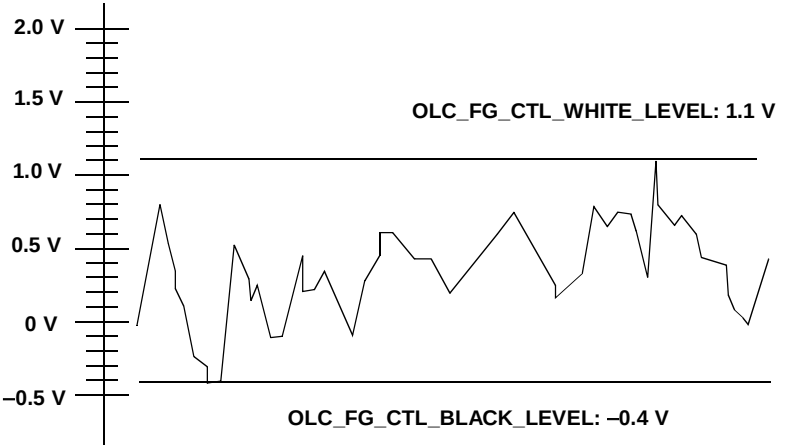


Figure 2: Use of Black Level and White Level to Increase Range

By lowering the black level voltage, the parts of the image that were digitized to black are now digitized to grayscale voltages. By raising the white level, the parts of the image that were digitized to white are also digitized to grayscale voltages. By adjusting the black level and white level voltages, you can now digitize the entire range of the video input signal.

Some frame grabber boards allow you to increase or decrease either or both the black and white levels independently; others set one level based on the other. Some boards also limit the difference allowed between the black and white levels. Refer to your board-specific documentation for more information.

To determine whether your frame grabber board supports programming the black and white levels, you can use **OlFgQueryInputCaps** to query the board. Table 7 lists the information about black and white levels that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 7: Getting Information About Black and White Levels

Information	Query Key
Whether the frame grabber board supports programming black and white levels.	OLC_FG_IC_DOES_PROG_A2D
The upper and lower black level limits the board supports.	OLC_FG_IC_BLACK_LEVEL_LIMITS
The upper and lower white level limits the board supports.	OLC_FG_IC_WHITE_LEVEL_LIMITS

To set the black level, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_BLACK_LEVEL; to set the white level, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_WHITE_LEVEL. To determine the current black and white levels, use **OlFgQueryInputControlValue** with the same controls.

Offset, Gain, and Reference

To determine whether your board supports programming the offset, gain, and reference, refer to your board-specific documentation. Offset, gain, and reference are described as follows:

- **Offset** – The voltage you apply to the minimum value of the video input signal ($-V_{min}$) to zero it. All data below 0 V is digitized as black pixels. To specify the offset voltage (from –1.075 V to 1.067 V), use **Dt3152SetInputControlValue** with the control DT3152_INPUT_CTL_OFFSET.

- **Gain** – The value you use to multiply the amplitude of the video input signal, thereby increasing or decreasing the overall range of the signal. You apply the gain after you apply the offset voltage. To specify the gain (0.5, 1, 2, or 4), use **Dt3152SetInputControlValue** with the control DT3152_INPUT_CTL_GAIN.
- **Reference** – The maximum voltage (after offset and gain have been applied) that you want to digitize. All data above the reference voltage is digitized as white pixels. To specify the reference voltage (from 0 V to 1.275 V), use **Dt3152SetInputControlValue** with the control DT3152_INPUT_CTL_REFERENCE.

To determine the current offset, gain, or reference value, use **Dt3152QueryInputControlValue** with the appropriate control.

Color Definitions

Some color frame grabber boards allow you to adjust one or more color settings. To determine whether you can adjust color settings, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_IMAGE_PARAMETER. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot adjust the color settings) or OLC_QUERY_COLOR_INTERFACE_IMAGE_PARAMETER (if you can adjust the color settings).

If **DtColorQueryInterface** indicates that you can adjust color settings, use **DtColorImageParameters** with the control OLC_QUERY_CAPABILITY and one of the constants listed in Table 8 to determine whether the particular color setting is supported. The query returns either OLC_SIGNAL_UNSUPPORTED (if the specified color setting is not supported) or the constant associated with the specified color setting (if it is supported).

Table 8: Color Constant s

Color Model	Constant	Description
HSI[a]	OLC_SET_BRIGHTNESS	The brightness level. This is a value associated with a pixel that represents its gray value.
	OLC_SET_CONTRAST	The contrast level. This is the overall range of the monochrome signal of an image. For example, a high contrast image has a large range between black and white values; a low contrast image has a small range between black and white values.
	OLC_SET_V_SAT	The V-saturation level. This is the purity of the blue and green primary colors in an image. For example, if a particular pixel has a value of 0 for green, but a value of 256 for blue, the pixel is saturated in blue.
	OLC_SET_U_SAT	The U-saturation level. This is the purity of the green and red primary colors in an image. For example, if a particular pixel has a value of 0 for green, but a value of 256 for red, the pixel is saturated in red.
	OLC_SET_HUE	The hue level. This is the intensity or shade of the color.

Table 8: Color Constants (cont.)

Color Model	Constant	Description
RGB	OLC_SET_RED_OFF OLC_SET_GREEN_OFF OLC_SET_BLUE_OFF	The offset voltage. This is the voltage that is added to the minimum full-scale voltage of your video input signal. For example, if the negative full-scale voltage of your video input signal is –53.86 mV and you specify an offset of +53.86 mV, the resulting digitized value is 0V.[b]
	OLC_SET_RED_REF OLC_SET_GREEN_REF OLC_SET_BLUE_REF	The reference voltage. This is the maximum full-scale value of your video input signal.[b]

a. Hue/Saturation/Intensity.
b. Once adjusted for offset, the frame grabber board digitizes each signal (red, green, and blue) between the adjusted negative full-scale voltage and the reference voltage. The offset value and the reference value are used together to determine the intensity of the red, green, or blue signal.

To determine the range of a particular color setting, use **DtColorImageParameters** with the control OLC_QUERY_CONTROL_MAX, OLC_QUERY_CONTROL_MIN, OLC_QUERY_CONTROL_NOMINAL, or OLC_QUERY_CONTROL_GRANULARITY and the appropriate constant (see Table 8).

To adjust a particular color setting, use **DtColorImageParameters** with the control OLC_WRITE_CONTROL and the appropriate constant. To determine the value of a particular color setting, use **DtColorImageParameters** with the control OLC_READ_CONTROL.

Sync Signals

The video input signal includes both image data and timing information. To control timing, the frame grabber board requires both horizontal and vertical sync signals. Figure 3 illustrates the process of generating the horizontal and vertical sync signals.

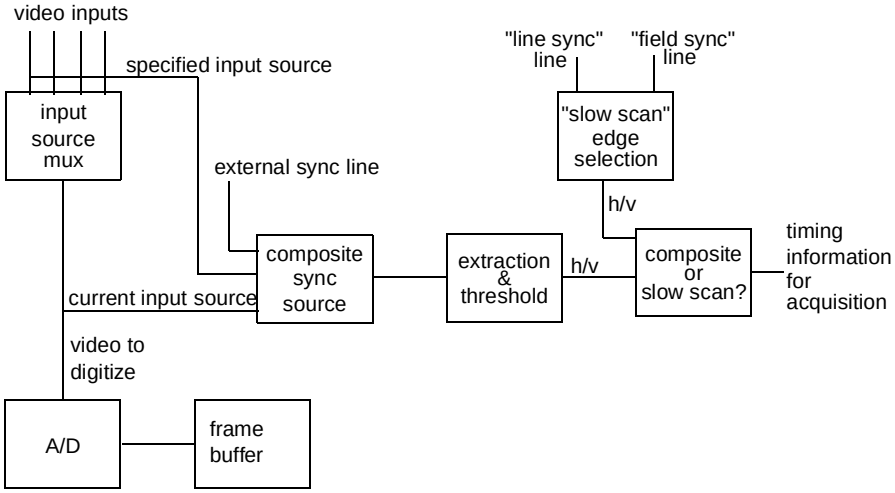


Figure 3: Horizontal and Vertical Timing

The following sections describe how to configure the following sync signal parameters:

- Sync source for composite video input signal (see the next section),
- Sync threshold for composite video input signals (see page 51),
- Horizontal and vertical sync signal transitions for variable-scan video input signals (see page 52),

- Sync Sentinel (see page 53), and
- Sync Master mode (see page 56).

Sync Source

For composite video signals, you can obtain the video sync signal from one of the following sync sources:

- The composite sync from the current input channel (the channel being digitized). In this case, the sync signal is stripped from the video signal and fed into the sync circuitry.
- The composite sync from one of the unused input channels. In this case, the sync signal from the selected channel is fed directly into the sync circuitry. For example, you could use camera 1 on channel 0 as your video input source and camera 2 on channel 1 as your composite sync source.
- The composite sync from an external sync source, such as a clock. In this case, the external sync signal is fed directly into the sync circuitry.

To determine whether your frame grabber board supports programming the sync source, you can use **OlFgQueryInputCaps** to query the board. Table 9 lists the information about the composite sync source that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 9: Getting Information About a Composite Sync Source

Information	Query Key
The composite sync sources your frame grabber board supports.	OLC_FG_IC_CSYNC_SOURCE_LIMITS
If your frame grabber board supports a composite sync source from any video input channel, the number of input channels supported.	OLC_FG_IC_INPUT_SOURCE_COUNT

The OLC_FG_IC_CSYNC_SOURCE_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 10.

Table 10: Composite Sync Source Bit Flags

Flag	Description
OLC_FG_CSYNC_CURRENT_SRC	Use the composite sync from the current video input channel only.
OLC_FG_CSYNC_SPECIFIC_SRC[a]	Use the composite sync from any of the video input channels.
OLC_FG_CSYNC_EXTERNAL_LINE	Use the composite sync from an external source.

a. Indicates the flag in the low word and the video input channel in the high word.

To set the composite sync source, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_CSYNC_SOURCE and the appropriate bit flag mask (see Table 10). To determine the current composite sync source, use **OlFgQueryInputControlValue** with the same control.

Sync Threshold

For composite video signals only, you must specify the sync threshold — the point at which the frame grabber board acknowledges the sync signal.

Typically, the sync threshold can be 50 mV, 75 mV, 100 mV, or 125 mV. To determine the upper and lower threshold limits that your board supports, use **OlFgQueryInputCaps** with the key OLC_FG_IC_CSYNC_THRESH_LIST.

To set the threshold level, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_CSYNC_THRESH. To determine the current threshold level, use **OlFgQueryInputControlValue** with the same control.

Horizontal and Vertical Sync Transition

For variable-scan video signals only, the line (horizontal) and field (vertical) sync are taken directly from the input device, and are used to generate the horizontal and vertical timing for the input section of the frame grabber board.

If you do not set a sync source and threshold, the frame grabber board assumes that you are using variable-scan video and that both the horizontal sync signal and the vertical sync signal occur on a high-to-low transition.

To specify that you want the horizontal and/or vertical sync signal to occur on a low-to-high transition, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_VARSCAN_FLAGS and the appropriate bit flag mask. The bit flags are defined in Table 11. To determine the currently specified transition use **OlFgQueryInputControlValue** with the same control.

Table 11: Variable-Scan Bit Flags

Flag	Description
OLC_FG_VS_LINE_ON_LO_TO_HI	If set, the horizontal sync occurs on a low-to-high transition; if clear, the horizontal sync occurs on a high-to-low transition.
OLC_FG_VS_FIELD_ON_LO_TO_HI	If set, the vertical sync occurs on a low-to-high transition; if clear, the vertical sync occurs on high-to-low transition.

Sync Sentinel

Some video signals, including those from VCRs, can be noisy or low quality, resulting in blurry acquisition due to missing or extraneous sync signals. The Sync Sentinel circuit (when available) corrects for missing, extraneous, or low-level sync signals by allowing for programmable control over the input sync circuitry of the frame grabber board.

When enabled, the Sync Sentinel provides a window in which a sync signal can be detected. Some frame grabber boards allow you to adjust the size of this window. On such boards, the horizontal sync search area defines the percentage of pixels in a line that should occur before the board begins to search for the horizontal sync. The board searches for the horizontal sync from that point until the horizontal sync insert position is reached. If the horizontal sync is not detected by that point, the Sync Sentinel inserts a horizontal sync to synchronize to the video signal.

The vertical sync search area defines the percentage of lines in a field that should occur before the board begins to search for the vertical sync. The board searches for the vertical sync from that point until the vertical sync insert position is reached. If the vertical sync is not detected by that point, the Sync Sentinel inserts a vertical sync to stay in sync with the video signal.

By setting the sync search area immediately before the falling edge of the sync and setting the sync insert position immediately after the falling edge of the sync, you can prevent the board from searching for syncs except where they are expected.

If you are switching among multiple input sources that are not synchronized with one another or if the sync signals occur at random intervals, you can disable the Sync Sentinel. This allows the board to wait until a sync signal actually occurs before starting the acquisition.

Note: Instead of disabling the Sync Sentinel, you can set the sync search area as far before the falling edge of the sync as possible and set the sync insert position as far after the falling edge of the sync as possible. This allows the board to search for the sync for almost the entire line and/or field.

To determine whether your frame grabber board supports programming the Sync Sentinel, you can use **OlFgQueryInputCaps** to query the board. Table 12 lists the information about Sync Sentinel that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 12: Getting Information About the Sync Sentinel

Information	Query Key
Whether the frame grabber board supports the Sync Sentinel.	OLC_FG_IC_DOES_SYNC_SENTINEL
Whether the search areas and insert positions are fixed or variable.	OLC_FG_IC_SYNC_SENTINEL_TYPE_LIMITS

The OLC_FG_IC_SYNC_SENTINEL_TYPE_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 13.

Table 13: Sync Sentinel Bit Flags

Flag	Description
OLC_FG_SYNC_SENTINEL_FIXED	The search area and insert position for the Sync Sentinel are fixed and cannot be altered through software.
OLC_FG_SYNC_SENTINEL_VARIABLE	The search area and insert position for the Sync Sentinel are variable and can be altered through software.

To enable the Sync Sentinel, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_SYNC_SENTINEL set to any nonzero value. To disable the Sync Sentinel, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_SYNC_SENTINEL set to 0.To determine the current state of the Sync Sentinel, use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_SYNC_SENTINEL.

If **OlFgQueryInputCaps** indicates that you can alter the horizontal and vertical search areas and insert positions, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 14. To determine the current value of a particular Sync Sentinel parameter, use **OlFgQueryInputControlValue** with the same controls.

Table 14: Sync Sentinel Controls

Parameter	Control
The horizontal sync insert position, specified as a percentage of the total pixels per line.[a]	OLC_FG_CTL_HSYNC_INSERT_POS
The horizontal sync search area, specified as a percentage of the total pixels per line.[a]	OLC_FG_CTL_HSYNC_SEARCH_POS
The vertical sync insert position, specified as a percentage of the total lines per field.[a]	OLC_FG_CTL_VSYNC_INSERT_POS
The vertical sync search area, specified as a percentage of the total lines per field.[a]	OLC_FG_CTL_VSYNC_SEARCH_POS

a. The specified percentage can be greater than 100%.

Note: Enabling Sync Master mode automatically disables the Sync Sentinel. For more information about Sync Master mode, refer to the next section.

Sync Master Mode

Note: Enabling Sync Master mode automatically disables the Sync Sentinel. For more information about the Sync Sentinel, refer to page 53.

Some frame grabber boards support Sync Master mode. Sync Master mode allows you to use the sync signals generated by the board to drive one or more cameras, if desired. The video signal from the camera is then digitized as usual, using the syncs generated by the board as the sync basis (gen-locking).

For some frame grabber boards that support Sync Master mode, you can adjust the width of the horizontal and vertical sync signals, the horizontal and vertical sync frequency, and the phase between the horizontal sync and vertical sync signals.

With interlaced cameras that accept horizontal and vertical syncs, you can adjust the phase to determine the field to output. By setting up the board as noninterlaced with a sync phase of 50%, the even field is continuously selected. If the phase is 1%, the odd field is continuously selected.

You can alternate the phase at each vertical sync by setting the phase at 50% and setting up the board as interlaced. The even and odd fields are then alternately selected for a true interlaced image. Whenever the board is set up as interlaced, the phase alternates between 1% and whatever phase is specified. These different phase adjustments are illustrated in Figure 4.

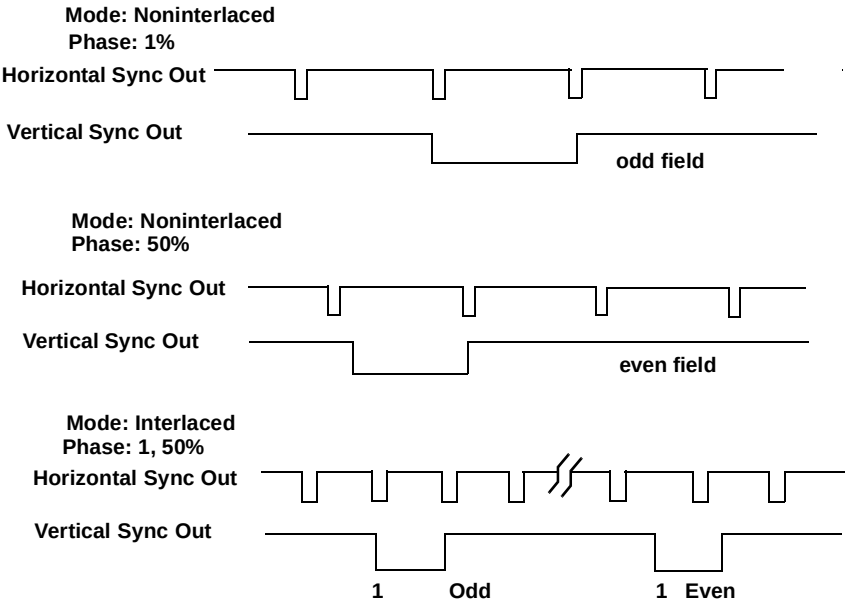


Figure 4: Sync Out Phase Adjustment

To determine whether your color frame grabber board supports Sync Master mode, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_SYNC_MASTER_MODE. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if your board does not support Sync Master mode) or OLC_QUERY_COLOR_INTEFACE_SYNC_MASTER_MODE (if your board does support Sync Master mode). To determine whether you can program the Sync Master mode parameters, refer to your board-specific documentation.

To determine whether your monochrome frame grabber board supports Sync Master mode and, if so, whether you can program the Sync Master mode parameters, refer to your board-specific documentation.

To enable and disable Sync Master mode, use either **DtColorSyncMasterMode**, **Dt3152EnableSyncMasterMode**, or **Dt3157EnableSyncMasterMode**, depending on the frame grabber board you are using.

To specify a value for one of the Sync Master mode parameters, use either **Dt3152SetSyncMasterControlValue** or **Dt3157SetSyncMasterControlValue** with the appropriate control. The controls are defined in Table 15. To determine the current value of a particular Sync Master mode parameter, use either **Dt3152QuerySyncMasterControlValue** or **Dt3157QuerySyncMasterControlValue** with the same controls.

Table 15: Sync Master Mode Controls

Control[a]	Description
DT315*x*_SYNC_CTL_HORIZ_FREQ	The horizontal frequency, in Hz. The value can range from 1 Hz to 2,000,000 Hz.
DT315*x*_SYNC_CTL_VERT_FREQ	The vertical frequency, in Hz. The value can range from 1 Hz to 200,000 Hz.
DT315*x*_SYNC_CTL_HPULSE_WIDTH	The width of the horizontal frequency's sync pulse, in ns. The value can range from 250 ns to 950,000,000 ns.
DT315*x*_SYNC_CTL_VPULSE_WIDTH	The width of the vertical frequency's sync pulse, in ns. The value can range from 250 ns to 950,000,000 ns.

Table 15: Sync Master Mode Controls

Control[a]	Description
DT315*x*_SYNC_CTL_PHASE	The amount the vertical sync is shifted relative to the horizontal sync, in percent of the total line. The value can range from 1% to 99%.

a. For DT3152 and DT3152-LS frame grabber boards, $x = 2$; for DT3157 frame grabber boards, $x = 7$.

Note: You must disable Sync Master mode before you set the values for the Sync Master mode parameters.

Pixel Clock

The pixel clock determines the video input signal digitization rate. Pixels are available to the frame grabber board at increments of PixelPeriod, which is equal to 1 / pixel clock frequency. To determine the appropriate pixel clock frequency, divide the number of pixels per line (including the active pixels and blank pixels) by the length of the horizontal line (in time).

Many cameras provide an external pixel clock; most frame grabber boards are equipped with an onboard, internal pixel clock as well. By default, the board assumes that you are using the internal pixel clock.

If you are using the internal pixel clock, you can change the frequency, if required.

If you are using an external pixel clock, the sync occurs on a high-to-low transition, by default. You can modify the sync transition, if required.

To determine whether your frame grabber board supports programming the pixel clock, you can use **OlFgQueryInputCaps** to query the board. Table 16 lists the information about the pixel clock that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 16: Getting Information About the Pixel Clock

Information	Query Key
Whether the frame grabber board supports a programmable pixel clock.	OLC_FG_IC_DOES_PIXEL_CLOCK
The pixel clock sources supported.	OLC_FG_IC_CLOCK_SOURCE_LIMITS
For an internal pixel clock, the range of frequencies supported.	OLC_FG_IC_CLOCK_FREQ_LIMITS

The OLC_FG_IC_CLOCK_SOURCE_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 17.

Table 17: Pixel Clock Source Bit Flags

Flag	Description
OLC_FG_CLOCK_INTERNAL	The frame grabber board generates the timing for acquisition.
OLC_FG_CLOCK_EXTERNAL	An external source (such as a camera) generates the timing for acquisition.

To set the pixel clock parameters, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 18. To determine the current value of the pixel clock parameters, use **OlFgQueryInputControlValue** with the same controls.

Table 18: Pixel Clock Controls

Flag	Parameter
OLC_FG_CTL_CLOCK_SOURCE[a]	The pixel clock source.
OLC_FG_CTL_CLOCK_FREQ[b]	For the internal pixel clock, the pixel clock frequency.
OLC_FG_CTL_CLOCK_FLAGS[c]	For an external pixel clock, the sync transition.

a. Use an appropriate bit flag mask. Refer to Table 17.
b. Since the range of frequencies is not linear, the frame grabber board sets the frequency using a closest approximation. Use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_CLOCK_FREQ to determine the value that the board actually sets.
c. Use the OLC_FG_CLOCK_EXT_ON_LO_TO_HI bit flag. Set the flag to indicate a low-to-high transition. Clear the flag to indicate a high-to-low transition.

Input Look-Up Table

An input look-up table (ILUT) allows you to change the value of an incoming pixel value. When the ILUT gets an input pixel value, it retrieves the output value for that particular pixel and passes the output value to the frame (region of interest). Pixel values range from 0 to 255 and can be defined for RVID, GVID, and BVID signals.

You can specify the relationship between the pixel input value and the ILUT output value by loading the ILUT with different processing setups. For example, you can pass an image unaltered (the default setting, known as identity), or you can perform pixel point operations, such as image multiplication and division, intensity correction, and reverse-video, before passing the image on.

As an example, assume that the ILUT is loaded with the identity pattern. An input value of 0 (black in monochrome mode) has an output value of 0 (black in monochrome mode). An input value of 1 has an output value of 1. An input value of 2 has an output value of 2, and so on, up to an input value of 255 (which has an output value of 255 or white in monochrome mode).

As another example, if you load the ILUT with an inverse or negative pattern, an input value of 0 has an output value of 255, an input value of 1 has an output value of 254, and so on, up to an input value of 255 (which has an output value of 0 or black in monochrome mode).

To determine whether your frame grabber board supports programming the ILUT, you can use **OlFgQueryInputCaps** to query the board. Table 19 lists the information about ILUTs that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 19: Getting Information About ILUTs

Information	Query Key
Whether the frame grabber board supports programming ILUTs.	OLC_FG_IC_DOES_INPUT_FILTER
The number of ILUTs the board provides.	OLC_FG_IC_ILUT_COUNT
The maximum number of entries allowed in a given ILUT.	OLC_FG_IC_MAX_ILUT_INDEX
The maximum value allowed in a given ILUT.	OLC_FG_IC_MAX_ILUT_VALUE

To determine the contents (full or partial) of an ILUT, use **OlFgReadInputLUT**. To change an ILUT, use **OlFgWriteInputLUT**. To specify the ILUT to use for future acquisitions, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_ILUT. To determine the currently specified ILUT, use **OlFgQueryInputControlValue** with the same control.

Note that in addition to ILUTs, passthru operations make use of passthru LUTs. For more information about passthru LUTs, refer to page 99.

External Trigger

A trigger is an external event that starts an acquisition. The external event is a signal received from a dedicated external line.

To determine whether your frame grabber board supports external triggering, you can use **OlFgQueryInputCaps** to query the board. Table 20 lists the information about the external trigger that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 20: Getting Information About the External Trigger

Information	Query Key
Whether the frame grabber board supports external triggering.	OLC_FG_IC_DOES_TRIGGER
The types of triggering supported for single-frame acquisitions.	OLC_FG_IC_TRIGGER_TYPE_LIMITS
The types of triggering supported for multiple-frame acquisitions.	OLC_FG_IC_MULT_TRIGGER_TYPE_ LIMITS
The trigger modes supported for multiple-frame acquisitions.	OLC_FG_IC_MULT_TRIGGER_MODE_ LIMITS

The OLC_FG_IC_TRIGGER_TYPE_LIMITS and OLC_FG_IC_MULT_TRIGGER_TYPE_LIMITS queries return a mask of bit flags. The bit flags are defined in Table 21.

Table 21: Trigger Type Bit Flags

Flag	Description
OLC_FG_TRIG_EXTERNAL_LINE	Externally triggered acquisition.
OLC_FG_TRIG_EVENT	Event-triggered acquisition.
OLC_FG_TRIG_ONE_EVENT_DELAY	Delayed event-triggered acquisition.

The OLC_FG_IC_MULT_TRIGGER_MODE_LIMITS query returns a mask of bit flags. The bit flags are defined in Table 22.

Table 22: Trigger Mode Bit Flags

Flag	Description
OLC_FG_MODE_START	Use a single trigger to start the acquisition of a series of multiple frames.
OLC_FG_MODE_EACH	Trigger each frame in a series of multiple frames by a separate trigger.

To set up triggering for a single-frame acquisition, use **OlFgSetTriggerInfo**. To set up triggering for a multiple-frame acquisition, use **OlFgSetMultipleTriggerInfo**. You can use these functions to specify the following:

- For either a single-frame or a multiple-frame acquisition, set the trigger type to OLC_FG_TRIGGER_EXTERNAL_LINE (to enable external triggering) or OLC_FG_TRIGGER_NONE (to disable external triggering).
- For either a single-frame or a multiple-frame acquisition, specify whether you want the trigger to occur on a low-to-high transition or on a high-to-low transition.

- For a multiple-frame acquisition only, set the trigger mode to OLC_FG_TRIGMODE_TO_START (if you want to use a single trigger to start the acquisition of a series of multiple frames) or OLC_FG_TRIGMODE_FOR_EACH (if you want to trigger each frame in a series of multiple frames by a separate trigger).

To determine the current trigger settings for a single-frame acquisition, use **OlFgQueryTriggerInfo**. To determine the current trigger settings for a multiple-frame acquisition, use **OlFgQueryMultipleTriggerInfo**.

Setting Up the Output Signals

This section describes how to set up the following output signals:

- Strobe output signal (see the next section),
- Exposure output signal (see page 69), and
- Reset output signal (see page 69).

Strobe Output Signal

Some frame grabber boards provide strobe output signals that allow you to synchronize operations between a strobing device and the frame grabber board. Strobing devices can provide a stop-motion image and allow products in motion to appear brighter, clearer, and in finer detail than without strobe lighting.

To determine whether your frame grabber board supports strobe output signals, you can use **OlFgQueryInputCaps** to query the board. Table 23 lists the information about strobe output signals that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 23: Getting Information About Strobe Output Signals

Information	Query Key
Whether the frame grabber board supports strobe output signals.	OLC_FG_IC_DOES_STROBE
When to output the strobe signal.	OLC_FG_IC_STROBE_TYPE_LIMITS
The upper and lower limits of the duration of the strobe output signal and the number of strobe output signal pulse widths that the board supports.	OLC_FG_IC_STROBE_PULSE_WIDTH_LIST_LIMITS

Table 23: Getting Information About Strobe Output Signals (cont.)

Information	Query Key
The specific strobe output signal pulse widths that the board supports.	OLC_FG_IC_STROBE_PULSE_WIDTH_LIST

To set up the strobe output signal, use **OlFgSetStrobeInfo** to specify the following:

- The length of the strobe output signal (3.3 ms, 6.6 ms, 13.3 ms, 26.6 ms, 53.3 ms, 106.6 ms, 213.3 ms, or 426.6 ms).
- The strobe type, which can be one of the following:
  - OLC_FG_STROBE_NOW – Output the strobe signal immediately.
  - OLC_FG_STROBE_FIELD_BASED – Output the strobe signal after each field is acquired (twice in a frame).
  - OLC_FG_STROBE_FRAME_BASED – Output the strobe signal after each frame is acquired (once in a frame).
- The polarity of the strobe output signal (active high or active low).
- The state of the strobe output signal (enabled or disabled).

To determine the current strobe output signal settings, use **OlFgQueryStrobeInfo**.

Exposure Output Signal

Some frame grabber boards that support digital cameras can generate an exposure output signal. To determine whether your frame grabber board supports an exposure output signal and, if so, whether you can program the pulse width and/or polarity, refer to your board-specific documentation.

To enable and disable the frame grabber board to generate an exposure pulse to the camera, use **Dt3157EnableExposureMode**. To specify the width of the exposure pulse (from 82 µs to 1.33 s) or to specify the active polarity of the exposure pulse (active high or active low), use **Dt3157SetExposure**. To determine the current pulse width or polarity, use **Dt3157QueryExposure**.

Reset Output Pulse

Some frame grabber boards that support digital cameras can send a reset output signal to the attached camera. To determine whether your frame grabber board supports a reset output signal, refer to your board-specific documentation.

To reset a digital camera, use **Dt3157ResetCamera**.

Setting Up the Video Area

The video image area is composed of pixels and lines of video. The total video area is the complete set of horizontal and vertical input lines from which you extract the active video area and the frame within the active video area that you want to digitize. The total video area includes all parts of the signal, including nonvisual portions such as horizontal and vertical blanking information and sync information.

The total video area is as wide as the total pixels per line (the entire area between two consecutive horizontal sync signals) and as tall as the total lines per field (the entire area between two consecutive vertical sync signals).

The total pixels per line can be calculated as follows:

$$\text{Total pixels per line} = \frac{\text{pixel clock frequency (MHz)}}{\text{horizontal frequency (kHz)}}$$

The total lines per field can be calculated as follows:

$$\text{Total lines per field} = \frac{\text{horizontal frequency (kHz)}}{\text{vertical frequency (Hz)}}$$

To determine whether your frame grabber board supports programming the total video area, you can use **OlFgQueryInputCaps** to query the board. Table 24 lists the information about the total video area that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 24: Getting Information About the Total Video Area

Information	Query Key
Whether the frame grabber board supports programming the total pixels per line and total lines per field.	OLC_FG_IC_DOES_ACTIVE_VIDEO
The range for total pixels per line.[a]	OLC_FG_IC_TOTAL_PIX_PER_LINE_LIMITS
The range for total lines per field.[a]	OLC_FG_IC_TOTAL_LINES_PER_FLD_LIMITS

a. Includes the upper and lower limits, granularity, and nominal value.

To set the total video area, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 25. To determine the current total video area settings, use **OlFgQueryInputControlValue** with the same controls.

Table 25: Total Video Area Controls

Parameter	Control
Total pixels per line	OLC_FG_CTL_TOTAL_PIX_PER_LINE
Total lines per field	OLC_FG_CTL_TOTAL_LINES_PER_FLD

The following sections describe the active video area and the frame within the active video area that you want to digitize.

Active Video Area

The active video area floats in the total video area. The active video area is defined as that part of the incoming signal that contains valid video data (not blanking or sync information). Therefore, the active video area consists of the visible portion of those lines containing visible pixel data. Its top is determined by the first active line, its left side is determined by the first active pixel, it is as wide as the active pixel count, and it is as tall as the active line count.

The following sections describe how to define the horizontal and vertical components of the active video area.

Horizontal Video Signal

Each line of video contains horizontal sync information, blanking information, and active video. Figure 5 shows the components of a single horizontal line of video. Pixel measurements are relative to the horizontal reference point, which is defined as the beginning of the horizontal sync.

Note that the frame is an area that you establish within the active video area. For more information about frames, refer to page 78.

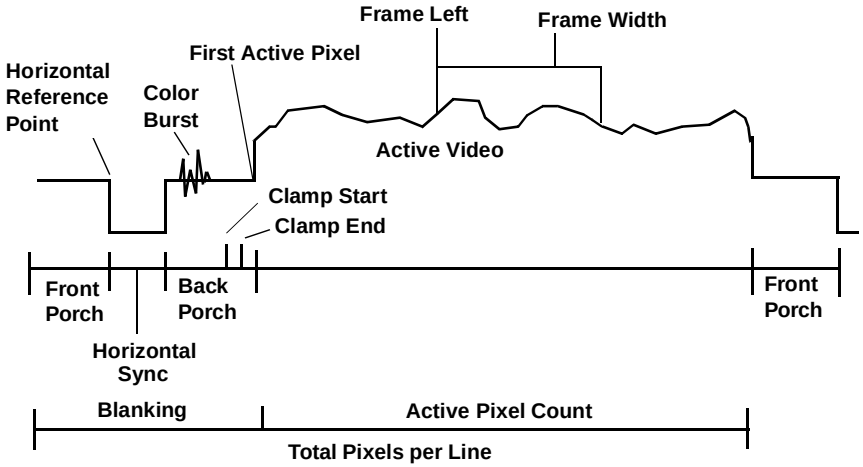


Figure 5: Horizontal Video Signal

In the horizontal video signal, blanking occurs during the horizontal sync and image border periods, which are defined by the front porch (before the horizontal sync) and back porch (after the horizontal sync).

Some frame grabber boards allow you to specify when your camera's back porch period starts, when to expect the first pixel of active video (first active pixel), and how long the camera will receive active video before the next blanking period (active pixel count).

In addition, some frame grabber boards allow you to define a period during the back porch when the board can adjust the black level from the camera. This is called a clamp period or DC restore. The board shifts the video signal so that the signal region during the clamp is at 0 V. The active video following the blanking period is then referenced to that shift.

To determine whether your frame grabber board supports programming the horizontal video signal, you can use **OlFgQueryInputCaps** to query the board. Table 26 lists the information about the horizontal video signal that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 26: Getting Information About the Horizontal Video Signal

Information	Query Key
Whether the frame grabber board supports programming the horizontal video signal.	OLC_FG_IC_DOES_ACTIVE_VIDEO
The range for back porch start.[a]	OLC_FG_IC_BACK_PORCH_START_LIMITS
The range for first active pixel.[a]	OLC_FG_IC_ACTIVE_PIXEL_LIMITS
The range for active pixel count.[a]	OLC_FG_IC_ACTIVE_WIDTH_LIMITS
The range for the clamp start position.[a]	OLC_FG_IC_CLAMP_START_LIMITS
The range for the clamp end position.[a]	OLC_FG_IC_CLAMP_END_LIMITS

a. Includes the upper and lower limits, granularity, and nominal value.

To set the horizontal video signal, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 27. To determine the current horizontal video signal settings, use **OlFgQueryInputControlValue** with the same controls.

Table 27: Horizontal Video Area Controls

Parameter	Control
Back porch start position	OLC_FG_CTL_BACK_PORCH_START
First active pixel	OLC_FG_CTL_FIRST_ACTIVE_PIXEL
Active pixel count	OLC_FG_CTL_ACTIVE_PIXEL_COUNT
Clamp start position	OLC_FG_CTL_CLAMP_START
Clamp end position	OLC_FG_CTL_CLAMP_END

Vertical Video Signal

Each field of video contains vertical sync information, blanking information, and lines of active video. Figure 6 shows the components of a noninterlaced frame or a field of an interlaced frame. Line measurements are relative to the vertical reference point, which is defined as the beginning of the vertical sync.

Note that the frame is an area that you establish within the active video area. For more information about frames, refer to page 78.

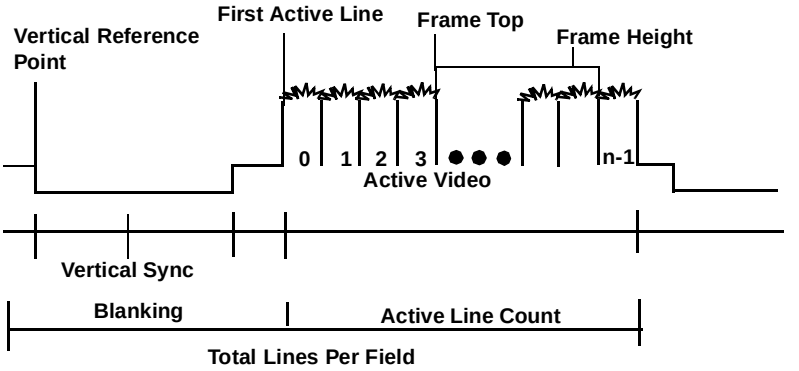


Figure 6: Vertical Video Signal

In the vertical video signal, blanking occurs during the vertical sync. Some frame grabber boards allow you to specify when your camera should expect the first line of active video (the beginning of the active video signal within the field, as a line offset from the beginning of the vertical sync) and how long the camera will receive active video before the next blanking period (active line count).

To determine whether your frame grabber board supports programming the vertical video signal, you can use **OlFgQueryInputCaps** to query the board. Table 28 lists the information about the vertical video signal that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 28: Getting Information About the Vertical Video Signal

Information	Query Key
Whether the frame grabber board supports programming the vertical video signal.	OLC_FG_IC_DOES_ACTIVE_VIDEO
The range for first active line.[a]	OLC_FG_IC_ACTIVE_LINE_LIMITS
The range for active line count.[a]	OLC_FG_IC_ACTIVE_HEIGHT_LIMITS

a. Includes the upper and lower limits, granularity, and nominal value.

To set the vertical video signal, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 29. To determine the current vertical video signal settings, use **OlFgQueryInputControlValue** with the same controls.

Table 29: Vertical Video Area Controls

Parameter	Control
First active line	OLC_FG_CTL_FIRST_ACTIVE_LINE
Active line count	OLC_FG_CTL_ACTIVE_LINE_COUNT

Frame (Region of Interest)

The frame is the portion of the active video area that you want to digitize. A frame is also referred to as the region of interest. The top of the frame is the first line of video relative to the active video area. The left side of the frame is the first pixel of video relative to the active video area. The width of the frame is the number of pixels per line of video. The height of the frame is the number of lines per field. Pixels outside this area are ignored.

The spatial relationship between the frame, the active video area, and the total video area is shown in Figure 7.

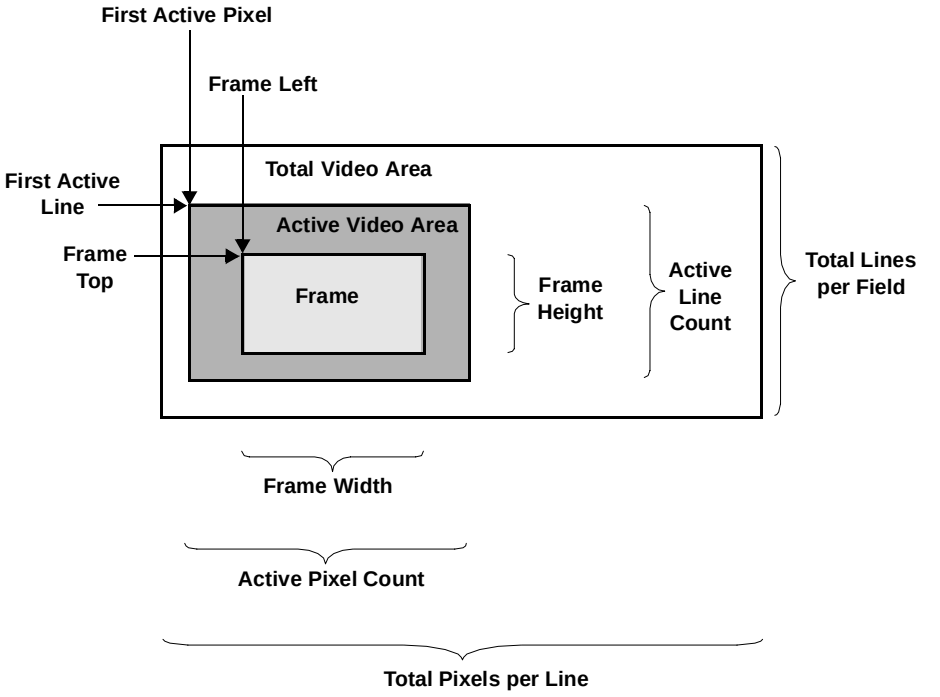


Figure 7: Spatial Relationship

Size of the Frame

To determine whether your frame grabber board supports programming the size of the frame, you can use **OlFgQueryInputCaps** to query the board. Table 30 lists the information about the size of the frame that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 30: Getting Information About the Size of the Frame

Information	Query Key
Whether the frame grabber board supports programming the size of the frame.	OLC_FG_IC_DOES_FRAME_SELECT
The range for the left side of the frame.[a]	OLC_FG_IC_FRAME_LEFT_LIMITS
The range for the width of the frame.[a]	OLC_FG_IC_FRAME_WIDTH_LIMITS
The range for the top of the frame.[a]	OLC_FG_IC_FRAME_TOP_LIMITS
The range for the height of the frame.[a]	OLC_FG_IC_FRAME_HEIGHT_LIMITS
The maximum number of pixels allowed in a frame.[b]	OLC_FG_IC_MAX_FRAME_SIZE
The maximum pixel depth (the maximum number of bytes in a pixel) for a frame.	OLC_FG_IC_PIXEL_DEPTH

a. Includes the upper and lower limits, granularity, and nominal value.
b. The frame height times the frame width must be less than or equal to the value returned.

To set the size of the frame, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 31. To determine the size of the current frame, use **OlFgQueryInputControlValue** with the same controls.

Table 31: Frame Size Controls

Parameter	Control
Frame left (the first pixel of video, relative to the active video area, to digitize)	OLC_FG_CTL_FRAME_LEFT
Frame width (the number of pixels per line of video to digitize)	OLC_FG_CTL_FRAME_WIDTH
Frame top (the first line of video, relative to the active video area, to digitize)	OLC_FG_CTL_FRAME_TOP
Frame height (the number of lines per field of video to digitize)	OLC_FG_CTL_FRAME_HEIGHT

Note: If you modify any of these parameters after you acquire a frame, the data in the frame buffer may be invalid. If you have already acquired data and are preparing to initialize the system for the next acquisition, make sure that you first process and store any frame data that you want to keep from the previous acquisition.

For some color frame grabber boards, you can also specify the format in which to store the image in memory. To determine whether your color frame grabber board supports programming the storage mode, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_STORAGE_MODE. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot specify the storage mode) or OLC_QUERY_COLOR_INTERFACE_STORAGE_MODE (if you can specify the storage mode).

To determine whether your color frame grabber board supports a particular storage mode, use **DtColorStorageMode** with the control OLC_QUERY_CAPABILITY and one of the constants listed in Table 32.

Table 32: Storage Mode Constants

Constant	Description
OLC_IMAGE_MONO	The image is stored in 8-bit monochrome format (1 byte/pixel).
OLC_IMAGE_YUV	The image is stored in YUV (native, unfiltered RGB) format (2 bytes/pixel).
OLC_IMAGE_YUYV_422	The image is stored in 16-bit YUYV format (2 bytes/pixel).
OLC_IMAGE_RGB	The image is stored in 32-bit RGB format (4 bytes/pixel).
OLC_IMAGE_RGB_32	The image is stored in 32-bit RGB format (4 bytes/pixel).
OLC_IMAGE_RGB_24	The image is stored in 24-bit RGB format (3 bytes/pixel).
OLC_IMAGE_RGB_16	The image is stored in 16-bit RGB format (2 bytes/pixel).
OLC_IMAGE_RGB_15	The image is stored in 15-bit RGB format (2 bytes/pixel).

The OLC_QUERY_CAPABILITY query returns either OLC_IMAGE_UNSUPPORTED (if the specified storage mode is not supported) or the constant associated with the specified storage mode (if it is supported).

To set the storage mode, use **DtColorStorageMode** with the control OLC_WRITE_CONTROL. To determine the current storage mode, use **DtColorStorageMode** with the control OLC_READ_CONTROL.

Type of Frame

A frame can be either interlaced or noninterlaced, as follows:

- **Interlaced frame** – Consists of two consecutive fields, each containing the number of lines specified by the active line count, where the start of each field is identified by the falling edge of the vertical sync. These two fields are acquired to create the complete frame. The even field contains lines 0, 2, 4, and so on; the odd field contains lines 1, 3, 5, and so on.
- **Noninterlaced frame** – Consists of a single field, containing the number of lines specified by the active line count, where the start of the field is identified by the falling edge of the vertical sync.

To determine whether your frame grabber board supports programming the type of frame, use **OlFgQueryInputCaps** with the key OLC_FG_IC_FRAME_TYPE_LIMITS.

To set the frame type, use **OlFgSetInputControlValue** with the control OLC_FG_CTL_FRAME_TYPE and the appropriate bit flag mask. The bit flags are defined in Table 33. To determine the current frame type, use **OlFgQueryInputControlValue** with the same control.

Table 33: Frame Type Bit Flags

Flag	Description
OLC_FG_FRM_IL_FRAME_EVEN	Acquire interlaced frames, starting with the next even field.
OLC_FG_FRM_IL_FRAME_ODD	Acquire interlaced frames, starting with the next odd field.
OLC_FG_FRM_IL_FRAME_NEXT	Acquire interlaced frames, starting with the next field of either kind.
OLC_FG_FRM_FIELD_EVEN	Acquire even fields from interlaced frames, starting with the next even field.

Table 33: Frame Type Bit Flags (cont.)

Flag	Description
OLC_FG_FRM_FIELD_ODD	Acquire odd fields from interlaced frames, starting with the next odd field
OLC_FG_FRM_FIELD_NEXT	Starting with the next field of either kind from an interlaced frame, acquire fields of that kind
OLC_FG_FRM_NON_INTERLACED	Acquire noninterlaced frames

Scaling the Frame

In addition to setting the actual size of the frame, you can effectively scale the frame by specifying a number of lines and/or fields (in regular intervals) to skip during acquisition.

The following sections describe how to scale the frame for monochrome and color frame grabber boards.

Monochrome Frame Grabber Boards

To determine whether your monochrome frame grabber board supports scaling the frame, you can query the board using **OlFgQueryInputCaps**. Table 34 lists the information about frame scaling that is returned by **OlFgQueryInputCaps** and the keys to use to get the information.

Table 34: Getting Information about Frame Scaling

Information	Query Key
The range for the increment between sampled pixels (horizontal scale factor).	OLC_FG_IC_FRAME_HINC_LIMITS
The range for the increment between sampled lines (vertical scale factor).	OLC_FG_IC_FRAME_VINC_LIMITS

To set the frame scale factor for monochrome frame grabber boards, use **OlFgSetInputControlValue** with the appropriate control. The controls are defined in Table 35. To determine the current frame scale factor, use **OlFgQueryInputControlValue** with the same control.

Table 35: Frame Scaling Controls

Parameter	Control
Horizontal scale factor	OLC_FG_CTL_HOR_FRAME_INC
Vertical scale factor	OLC_FG_CTL_VER_FRAME_INC

Color Frame Grabber Boards

To determine whether your color frame grabber board supports scaling the frame, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_HARDWARE_SCALING. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot scale the frame) or OLC_QUERY_COLOR_INTERFACE_HARDWARE_SCALING (if you can scale the frame).

To determine the range of the frame scale factors, use **DtColorHardwareScaling** with the control OLC_QUERY_CONTROL_MAX, OLC_QUERY_CONTROL_MIN, OLC_QUERY_CONTROL_NOMINAL, or OLC_QUERY_CONTROL_GRANULARITY.

To set the frame scale factors, use **DtColorHardwareScaling** with the control OLC_WRITE_CONTROL. To determine the value of the frame scale factors, use **DtColorHardwareScaling** with the control OLC_READ_CONTROL.

Note: If you modify the scale factors after you acquire a frame, the data in the frame buffer may be invalid. If you have already acquired data and are preparing to initialize the system for the next acquisition, make sure that you first process and store any frame data that you want to keep from the previous acquisition.

Memory Allocation Operations

Before you can perform an acquisition, you must have a frame buffer available to hold the acquired data. In addition, if you intend to acquire the image to host memory, you must allocate a user buffer. The following sections describe how to

- Allocate frame buffers (see the next section), and
- Allocate a user buffer (see page 90).

Note: It is recommended that you acquire data into frame buffers, which are located in device memory, rather than user buffers, which are located in host memory. For more information about device memory and host memory, refer to page 104.

Once you have allocated a frame buffer and/or user buffer, you can perform an acquisition. For more information, refer to page 104. After you perform an acquisition, you can move the data for processing. For more information, refer to page 108.

Allocating Frame Buffers

To allocate and manage frame buffers, your frame grabber board must support memory management operations. To determine if your board supports memory management operations, use **OlImgQueryDeviceCaps** with the key OLC_IMG_DC_SECTIONS. If the OLC_FG_SECTION_MEMORY flag is set in the returned set of bit flags, your board supports memory management operations.

The following sections describe

- The memory types supported for frame buffers (see the next section),
- How to allocate a frame buffer (see page 89), and
- How to release a frame buffer (see page 90).

Memory Types

The Frame Grabber SDK allows you to create volatile and nonvolatile frame buffers, as follows:

- **Volatile memory** – A predefined memory area whose contents or format may be altered indirectly as a side-effect of other operations or directly by the software itself.
- Volatile memory is useful for image acquisitions because the frame grabber board can overwrite the data in the volatile memory with each acquisition. Volatile memory is also useful if you are acquiring images from nonstandard input devices; in this case, the board can alter the format or dimensions of the memory for any given acquisition.
- **Nonvolatile memory** – A predefined memory area whose contents and format do not change except at the explicit request of the owner of the memory area.

To determine whether your frame grabber board supports volatile and nonvolatile memory, you can use **OlFgQueryMemoryCaps** to query the board. Table 36 lists the information about volatile and nonvolatile memory that is returned by **OlFgQueryMemoryCaps** and the keys to use to get the information.

Table 36: Getting Information about Volatile and Nonvolatile Memory

Information	Query Key
The types of memory the frame grabber board supports.	OLC_FG_MC_MEMORY_TYPES
The number of volatile frame buffers provided by the frame grabber board.[a]	OLC_FG_MC_VOL_COUNT
The number of nonvolatile frame buffers provided by the frame grabber board.	OLC_FG_MC_NONVOL_COUNT

a. If your frame grabber board supports the allocation of noncontiguous frame buffers, the number of frame buffers allocated cannot be determined. However, **OlFgQueryMemoryCaps** returns a default value to ensure compatibility with existing application programs. Do not use the value returned to perform any meaningful computations.

The OLC_FG_MC_MEMORY_TYPES query returns a mask of bit flags. The bit flags are defined in Table 37.

Table 37: Memory Type Bit Flags

Flag	Description
OLC_FG_MEM_VOLATILE	The frame grabber board provides volatile memory and supports the related functions.
OLC_FG_MEM_NON_VOLATILE	The frame grabber board provides nonvolatile memory and supports the related functions.

Note: Although you can use **OlFgQueryMemoryCaps** when you are acquiring video of a nonstandard format, the data returned is not meaningful.

Allocating a Frame Buffer

To allocate a frame buffer, use **OlFgAllocateBuiltInFrame**, specifying OLC_FG_DEV_MEM_VOLATILE for a volatile frame buffer or OLC_FG_DEV_MEM_NONVOLATILE for a nonvolatile frame buffer.

You can either assign each frame buffer a unique frame ID or allow the frame grabber board to assign a frame ID to the frame buffer (guaranteeing a unique number). Allowing the board to assign the frame ID is useful if your application is not concerned with the specific location of the data, especially if you are allocating a series of frame buffers.

Frame buffers use zero-based numbering; that is, the first frame buffer allocated has a frame ID of 0, the second frame buffer allocated has a frame ID of 1, and so on.

You do not explicitly specify the size of the allocated frame buffer. The size is determined by the size of the frames you are digitizing. It is equal to the frame height times the frame width times the pixel depth. To determine the frame height, use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_FRAME_HEIGHT; to determine the frame width, use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_FRAME_WIDTH; to determine the pixel depth, use **OlFgQueryInputCaps** with the key OLC_FG_IC_PIXEL_DEPTH.

Note: For color frame grabber boards, you can determine the current format in which the image is stored in memory (storage mode) by using **DtColorStorageMode** with the control OLC_READ_CONTROL.

After you allocate a frame buffer, you can use **OlFgQueryFrameInfo** to retrieve information about the frame buffer, including the virtual base address, width, height, and bytes per sample.

Releasing a Frame Buffer

When you finish acquiring images, release each frame buffer using **OlFgDestroyFrame**. After you release a frame buffer, do not use its frame ID again in subsequent calls unless you create a new frame buffer (with **OlFgAllocateBuiltInFrame**) using that particular frame ID. If you do create a new frame buffer with the same frame ID as a previously released frame buffer, you can make no assumptions about the initial contents of the frame buffer.

Allocating a User Buffer

If you intend to acquire images to a user buffer, which is located in host memory, use the Windows function **GlobalAlloc()** (or an equivalent) to define and globally allocate a user buffer from the main heap of your process. Make sure that you specify the GPTR flag to lock the buffer in memory. This provides a valid 32-bit pointer to the device driver.

The user buffer must be at least as large as the product of the frame height times the frame width times the pixel depth. To determine the frame height, use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_FRAME_HEIGHT; to determine the frame width, use **OlFgQueryInputControlValue** with the control OLC_FG_CTL_FRAME_WIDTH; to determine the pixel depth, use **OlFgQueryInputCaps** with the key OLC_FG_IC_PIXEL_DEPTH.

Note: If you acquire data to a user buffer, you still require a single frame buffer to temporarily hold the acquired data. Any data in this temporary frame buffer is destroyed.

Due to problems that can result, it is recommended that you do not allocate user buffers on the stack under any circumstances.

Passthru Operations

This section describes the following:

- Performing a passthru operation (see the next section),
- Adjusting the source origin (see page 96),
- Scaling the passthru image (see page 98),
- Modifying the passthru LUT (see page 99),
- Taking a snapshot (see page 100), and
- Creating overlays (see page 101).

Performing a Passthru Operation

In a passthru operation, data is continually acquired to display memory (memory on your video display adapter) until some event occurs. Typically, you use passthru to view the image data (in as close to real time as possible) for the purpose of focusing or positioning the camera. The display rate depends on the speed of the CPU and graphics cards; the rate can slow down if other applications are using system resources.

The passthru image is displayed in the client region of a window. If the passthru image is larger than the window's client region, only the part of the image that is the size of the client region is displayed. If the passthru image is smaller than the window's client region, the leftover area of the window has no image displayed on it and remains unchanged.

You can choose to devote all your system resources to the passthru operation (synchronous passthru), or you can start the process and then free your system to perform other tasks while the passthru operation occurs (asynchronous passthru). You cannot interrupt a synchronous passthru operation.

Note: External triggering is automatically disabled for all passthru operations.

To determine whether your frame grabber board supports passthru operations, use **OlFgQueryPassthruCaps** with the key OLC_FG_PC_PASSTHRU_MODE_LIMITS. **OlFgQueryPassthruCaps** returns a mask of bit flags. The bit flags are defined in Table 38.

Table 38: Passthru Bit Flags

Flag	Description
OLC_FG_PASSTHRU_ASYNC_BITMAP	Asynchronous bitmap passthru operations are supported.
OLC_FG_PASSTHRU_SYNC_BITMAP	Synchronous bitmap passthru operations are supported.
OLC_FG_PASSTHRU_ASYNC_DIRECT[a]	Asynchronous direct passthru operations are supported.
OLC_FG_PASSTHRU_SYNC_DIRECT[a]	Synchronous direct passthru operations are supported.
OLC_FG_PASSTHRU_ASYNC_BITMAP_EXTENDED	Continuous-acquire passthru operations are supported.

a. Direct passthru operations were supported for Windows 3.x only. Since the current version of the Frame Grabber SDK does not support Windows 3.x, direct passthru operations are meaningless.

Typically, passthru image data in display memory is undesirable for processing. This is because, unlike an acquisition (where each image is placed into its own frame buffer), a passthru operation acquires multiple images to the same buffer in display memory. Each successive image overwrites the previous image. Additionally, the

image displayed through Windows is altered by Windows display LUTs.

If desired, you can process passthru image data in one of the following ways:

- You can acquire a single frame of passthru data (a snapshot) that is suitable for processing. For more information, refer to page 100.
- You can use continuous-acquire passthru mode to acquire multiple frames and then process the acquired data. For more information, refer to page 95.

The Frame Grabber SDK supports the following passthru modes:

- Bitmap passthru (see the next section) and
- Continuous-acquire passthru (see page 95).

Bitmap Passthru

You can perform synchronous and asynchronous bitmap passthru operations.

Note: If you are performing an asynchronous passthru operation, it is recommended that you use continuous-acquire passthru mode instead of bitmap passthru mode. For more information, refer to page 95.

For bitmap passthru mode, you must allocate a single frame buffer with **OlFgAllocateBuiltInFrame**, specifying OLC_FG_DEV_MEM_VOLATILE as the memory type, before starting the passthru operation. Note that no image data is saved in the frame buffer when the passthru operation stops. For more information about allocating frame buffers, refer to page 86.

Bitmap passthru operations use Windows functions to copy the image data to display memory. Windows functions handle obstructions to the passthru window by automatically clipping the passthru image to the visible client window region. If a window is obstructed in bitmap mode, the passthru continues. When the obstruction is removed, Windows automatically restores the correct underlying image data.

If your video board supports the Direct Draw Interface (DDI), the Frame Grabber SDK automatically substitutes direct draw for bitmapping. DDI gives the frame grabber board direct control of the video hardware. Direct draw improves passthru speed and allows you to use overlays. For more information about overlays, refer to see page 101.

To start an asynchronous bitmap passthru operation, use **OlFgStartAsyncPassthruBitmap**. To stop the operation, use **OlFgStopAsyncPassthru**.

To start a synchronous bitmap passthru operation, use **OlFgStartSyncPassthruBitmap**. To stop the operation, use a mouse or keyboard event.

Continuous-Acquire Passthru

You can perform asynchronous continuous-acquire passthru operations. Continuous-acquire passthru allows you to continually display passthru data while you acquire one or more frames of passthru data into a circular buffer in device memory. Before starting a continuous-acquire passthru operation, you must allocate an appropriate number of frame buffers with **OlFgAllocateBuiltInFrame,** specifying OLC_FG_DEV_MEM_VOLATILE as the memory type. For more information about allocating frame buffers, refer to page 86.

Continuous-acquire passthru operations use Windows functions to copy the image data to display memory. Windows functions handle obstructions to the passthru window by automatically clipping the passthru image to the visible client window region. If a window is obstructed in continuous-acquire mode, the passthru continues. When the obstruction is removed, Windows automatically restores the correct underlying image data.

To start an asynchronous continuous-acquire passthru operation, use **OlFgStartAsyncPassthruEx**. To stop the operation, use **OlFgStopAsyncPassthru**.

Note: You can also use continuous-acquire passthru to acquire image data without displaying it. This is useful if you are acquiring a large number of frames. When you acquire frames using **OlFgAsyncAcquireMultipleToDevice** (refer to page 106), you may miss one or more frames when restarting the operation after the specified number of buffers has been filled. Because continuous-acquire passthru operations use a circular buffer, no frames are lost.

Adjusting the Source Origin

Typically, the upper left corner of the display and the upper left corner of the acquired image are identical. During a passthru operation, you can select any other point in the acquired image to be the upper left corner of the display (source origin). This allows you to pan and scroll the image on the display to display part of the acquired image.

For example, if the passthru image is 640 x 480 and the source origin is 300 x 300, the point at 300 x 300 in the passthru image is placed in the upper left corner of the window. The size of the displayed image

is 340 x 180; the leftover area of the window has no image displayed on it and remains unchanged.

Figure 8 illustrates adjusting the source origin.

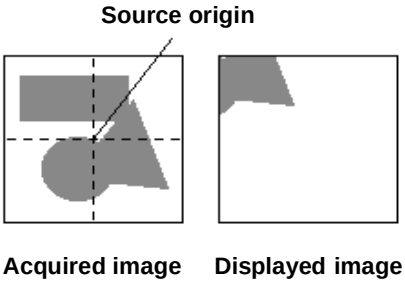


Figure 8: Source Origin Example

To determine whether your frame grabber board supports programming the source origin, you can use **OlFgQueryPassthruCaps** to query the board. Table 39 lists the information about the source origin that is returned by **OlFgQueryPassthruCaps** and the keys to use to get the information.

Table 39: Getting Information About the Source Origin

Information	Query Key
Whether the frame grabber board supports setting the source origin.	OLC_FG_PC_DOES_SOURCE_ORIGIN
The range for the x-coordinate.[a]	OLC_FG_PC_SRC_ORIGIN_X_LIMITS
The range for the y-coordinate.[a]	OLC_FG_PC_SRC_ORIGIN_Y_LIMITS

a. Includes the upper and lower limits, granularity, and nominal value.

To set the x-coordinate and y-coordinate of the source origin, use **OlFgSetPassthruSourceOrigin**. To determine the current source origin, use **OlFgQueryPassthruSourceOrigin**.

Note: Use a Windows POINT structure to define the pixels to set as the source origin before calling **OlFgSetPassthruSourceOrigin**.

Some frame grabber boards require alignment of pixels to 4-pixel, 8-pixel, or 32-pixel boundaries. The typical alignment is 4-pixel boundaries (right and left edges). Refer to your board-specific documentation for more information.

Scaling the Passthru Image

Passthru scaling allows you to reduce the size of the displayed image.

To determine whether your frame grabber board supports passthru scaling, you can use **OlFgQueryPassthruCaps** to query the board. Table 40 lists the information about passthru scaling that is returned by **OlFgQueryPassthruCaps** and the keys to use to get the information.

Table 40: Getting Information About Passthru Scaling

Information	Query Key
Whether the frame grabber board supports passthru scaling.	OLC_FG_PC_DOES_SCALING
The range for the x-coordinate (width) of the scaled passthru image.[a]	OLC_FG_PC_SCALE_WIDTH_LIMITS
The range for the y-coordinate (height) of the scaled passthru image.[a]	OLC_FG_PC_SCALE_HEIGHT_LIMITS

a. Includes the upper and lower limits.

To scale the passthru image, use **OlFgSetPassthruScaling**. Specify the height and width that you want the scaled image to be, and the frame grabber board reduces or enlarges the image as closely as possible. To determine the current size of the scaled passthru image, use **OlFgQueryPassthruScaling**.

Modifying the Passthru LUT

In addition to the ILUT (described on page 62), you can use the passthru LUT to affect the displayed image during passthru. By using the passthru LUT with the ILUT, you can change the display image without altering the ILUT itself. This allows you to display reversed or otherwise enhanced images without disrupting the underlying color settings.

When using the passthru LUT, first the values in the ILUT are applied to the image. Then the values in the passthru LUT are applied.

By default, passthru operations load the Windows system palette with 128 grayscale RGB values for display and use the default passthru LUT of 256 monotonically increasing grayscales. If you want to modify the passthru LUT so that the frame grabber board uses false coloring, use **OlFgLoadPassthruLUT**. To return to the default monochrome values, use **OlFgLoadDefaultPassthruLUT**.

The passthru LUT is filled with RGBTRIPLEs. For each entry in the passthru LUT, the index of the closest matching RGB value in the Windows system palette is used. If you want an exact RGBTRIPLE to display during passthru, use **OlFgExtendPassthruPalette** to add that RGBTRIPLE to the system palette. This guarantees that the color is available when the frame grabber board attempts to find the closest match in the Windows system palette.

The Frame Grabber SDK keeps the added colors updated in response to WM_PALETTECHANGED and WM_QUERYNEWPALETTE messages. If you want more additional colors than your frame grabber board supports, you can create a Windows palette and add

that palette to the system palette. Note that if you do create a Windows palette, you must maintain the palette in response to WM_PALETTECHANGED and WM_QUERYNEWPALETTE window messages.

To determine whether your frame grabber board supports programming the passthru LUT, you can use **OlFgQueryPassthruCaps** to query the board. Table 41 lists the information about the passthru LUT that is returned by **OlFgQueryPassthruCaps** and the keys to use to get the information.

Table 41: Getting Information About the Passthru L UT

Information	Query Key
Whether the frame grabber board supports programming the passthru LUT.	OLC_FG_PC_DOES_PASSTHRU_LUT
The number of the maximum entry in the passthru LUT.	OLC_FG_PC_MAX_PLUT_INDEX
The maximum value for a red, green, or blue element in an RGBTRIPLE in the passthru LUT.	OLC_FG_PC_MAX_PLUT_VALUE
The maximum number of entries that can be added to the system palette array.	OLC_FG_PC_MAX_PALETTE_INDEX
The maximum value for a red, green, or blue element in an RGBTRIPLE in the system palette.	OLC_FG_PC_MAX_PALETTE_VALUE

Taking a Snapshot

Some frame grabber boards allow you to capture one frame of an asynchronous passthru operation without stopping the passthru operation. This is called taking a snapshot. To determine if your board supports taking a snapshot, use **OlFgQueryPassthruCaps** with the key OLC_FG_PC_DOES_PASSTHRU_SNAPSHOT.

Before taking a snapshot, use **OlFgAllocateBuiltInFrame** to allocate a frame buffer that is large enough to hold the snapshot.

While the asynchronous passthru is in progress, use **OlFgPassthruSnapShot** to capture the single passthru image. The board writes the captured image to the specified frame buffer. Note that you cannot use **OlFgPassthruSnapShot** to capture overlays on the passthru display. To capture both the display and the overlay, call **OlFgAddOverlayToFrame** immediately after calling **OlFgPassthruSnapShot**.

After you acquire the snapshot, you can process the data as you would any other acquired frame. For more information, refer to page 108.

Note: Snapshots are not supported in continuous-acquire passthru operations.

Creating Overlays

Overlays allow to you to place or overlay an image on top of another image. You can create and work with overlays both before and during a passthru operation.

To create and manipulate overlays, both your system and your frame grabber board must support DDI functions. To determine if your board supports DDI, use **OlImgQueryDeviceCaps** with the key OLC_IMG_DC_SECTIONS. If the OLC_FG_SECTION_DDI flag is set in the returned set of bit flags, the board supports DDI functions.

Before you start working with overlays, use **OlFgEnableOverlays** to enable overlays.

The first step in creating an overlay is to allocate a surface buffer (or surface) with **OlFgCreateSurface**. The surface must be large enough to overlay the entire frame area. An overlay surface cannot be smaller than its entire underlying image. Each overlay must have its own surface.

After you allocate a surface buffer, get a handle to a device context (DC) for the surface buffer with **OlFgGetSurfaceDC**. DCs are Windows Graphical Device Interface (GDI) objects. Refer to your Windows SDK documentation for more information about DCs and GDIs.

After getting a handle to a DC, use standard GDI functions to draw onto the DC as you would in a window or in another DC (such as a printer DC). This provides you with a graphical interface (Windows API) to render shapes onto overlay surfaces. You can erase the contents of a surface at any time with **OlFgEraseSurface**.

When you have finished working with the DC, release the surface DC with **OlFgReleaseSurfaceDC**. Repeat the **OlFgGetSurfaceDC**/edit/**OlFgReleaseSurfaceDC** process for each surface buffer you allocate.

Note: Do not retain a handle to a DC for any longer than necessary. When you call **OlFgGetSurfaceDC**, the frame grabber board locks the surface buffer and other functions cannot create new DCs for the surface buffer. In addition, a deadlock condition can occur if one function retains a handle while another function is waiting for it to be released.

Use **OlFgSetVisibleSurface** to specify the overlay to make visible. Overlays that you can see and that allow you to see the image underneath at the same time are called translucent. Translucent overlays are useful when you want to align overlays with the

underlying image. To specify whether you want the current overlay to be translucent or opaque, use **OlFgSetTranslucentOverlay**.

You can also use **OlFgSetOverlayColorKey** to specify the source keying color (the color that is replaced by the passthru image) for the active surface. Specify the color with a WIN32 RGBTRIPLE structure. Refer to the Windows SDK documentation for information about the RGBTRIPLE data type.

For animation, you must create two surface buffers. One surface buffer is used as the front buffer (the displayed buffer); the other surface buffer is the back buffer (the buffer you are drawing to). To specify the front buffer (the surface to overlay on the next refresh), use **OlFgSetVisibleSurface**.

You can synchronize the refresh routine of the passthru operation with a WIN32[event] synchronization object and **OlFgGetPassthruSyncEvent**. You can use this event to make a thread sleep and wait for the event to be signaled. The passthru refresh routine signals the event after each refresh.

When you finish using overlays, use **OlFgEnableOverlays** to disable overlays, and then use **OlFgDestroySurface** to destroy each surface buffer that you allocated. This frees memory and prevent memory errors during subsequent operations.

Acquisition Operations

Once you have established the frames that you want to digitize (see page 78) and allocated frame buffers to store the frames (see page 86), you can acquire the frames.
You can acquire frames to either host memory or device memory, as follows:

- **Host memory** – A block of system memory that can come from anywhere in your process address space, such as from the operating system heaps, a global array, or any other memory management system you may use. You cannot manage host memory with Frame Grabber SDK functions. You cannot use host memory to perform a multiple-frame acquisition.
- **Device memory** – System memory that is allocated and managed by the device driver. You define the size of this memory location when you configure the device driver. You can manage device memory with Frame Grabber SDK functions.

Generally, you should acquire data to device memory. Acquire data to host memory only if the amount of system memory reserved for device memory is too small for your purposes and reserving more system memory for device memory is undesirable.
You can acquire frames either synchronously or asynchronously, as follows:

- **Synchronous acquisition** – All your system resources are devoted to the acquisition. You cannot perform another operation until the synchronous acquisition completes.
- **Asynchronous acquisition** – The operation starts and then returns control to you immediately, allowing you to perform other operations while data is acquired.

Once the operation is complete, you can move the data for processing. For more information, refer to page 108.
The following sections describe how to

- Acquire a single frame (see page 105), and
- Acquire multiple frames (see page 106).

Acquiring a Single Frame

When acquiring a single frame, you can do one of the following:

- Acquire the frame asynchronously to device memory using **OlFgAsyncAcquireFrameToDevice**. The frame grabber board acquires the image to the frame buffer you allocated.

  During the process, use **OlFgIsAsyncAcquireJobDone** to determine whether the acquisition has completed. If the acquisition has completed, **OlFgIsAsyncAcquireJobDone** returns TRUE; otherwise, **OlFgIsAsyncAcquireJobDone** returns OLC_STS_PENDING if the job has not yet started or OLC_STS_ACTIVE if the job has started but is not complete. To cancel the acquisition, use **OlFgCancelAsyncAcquireJob**.

  Note: If you want to acquire a single frame asynchronously to device memory while you display the image on a window, you can perform a continuous-acquire passthru operation. For more information, refer to page 95.

- Acquire the frame asynchronously to host memory using **OlFgAsyncAcquireFrameToHost**. The frame grabber board acquires the image to the frame buffer you allocated and then transfers it to the user buffer.

During the process, use **OlFgIsAsyncAcquireJobDone** to determine whether the acquisition has completed. If the acquisition has completed, **OlFgIsAsyncAcquireJobDone** returns TRUE; otherwise, **OlFgIsAsyncAcquireJobDone** returns OLC_STS_PENDING if the job has not yet started or OLC_STS_ACTIVE if the job has started but is not complete. To cancel acquisition, use **OlFgCancelAsyncAcquireJob**.

- Acquire the frame synchronously to device memory using **OlFgAcquireFrameToDevice**. The frame grabber board acquires the image to the frame buffer you allocated.
- Acquire the frame synchronously to host memory using **OlFgAcquireFrameToHost**. The frame grabber board acquires the image to the frame buffer you allocated and then transfers it to the user buffer.

Acquiring Multiple Frames

When acquiring multiple frames, you can do one of the following:

- Acquire the frames asynchronously to device memory using **OlFgAsyncAcquireMultipleToDevice**. In **OlFgAsyncAcquireMultipleToDevice**, specify the number of previously allocated frame buffers. The frame grabber board acquires and stores the first image in the first frame buffer you allocated, the second image in the second frame buffer you allocated, and so on, until the count you specified has been reached.

  During the process, use **OlFgIsAsyncAcquireJobDone** to determine whether the acquisition has completed. If the acquisition has completed, **OlFgIsAsyncAcquireJobDone** returns TRUE; otherwise, the **OlFgIsAsyncAcquireJobDone** returns OLC_STS_PENDING if the job has not yet started or OLC_STS_ACTIVE if the job has started but is not complete. To cancel acquisition, use **OlFgCancelAsyncAcquireJob**.

Note: If you want to acquire multiple frames asynchronously to device memory while you display the images on a window, you can perform a continuous-acquire passthru operation. For more information, refer to page 95.

Continuous-acquire passthru is also useful if you are acquiring a large number of frames. When you acquire frames using **OlFgAsyncAcquireMultipleToDevice**, you may miss one or more frames when restarting the operation after the specified number of buffers has been filled. Because continuous-acquire passthru operations use a circular buffer, no frames are lost.

- Acquire the frames synchronously to device memory using **OlFgAcquireMultipleToDevice**. In **OlFgAcquireMultipleToDevice**, specify the number of previously allocated frame buffers. The frame grabber board acquires and stores the first image in the first frame buffer you allocated, the second image in the second frame buffer you allocated, and so on, until the count you specified has been reached.

Image Processing Operations

After performing an acquisition, one or more frame buffers of data exist. If you performed a single-frame acquisition, the frame buffer resides in host memory or in device memory. If you performed a multiple-frame acquisition or a continuous-acquire acquisition, all the frame buffers reside in device memory.

If your frame grabber board supports memory management operations, you can move data among user buffers, frame buffers, and applications for processing. To determine whether your board supports memory management operations, use **OlImgQueryDeviceCaps** with the key OLC_IMG_DC_SECTIONS. If the OLC_FG_SECTION_MEMORY flag is set in the returned set of bit flags, your board supports memory management operations.
The remainder of this section describes the following image processing operations:

- **Display operations** – Allow you to display image data in a window.
- **Read operations** – Allow you to move data from a frame buffer in device memory to a user buffer in host memory (see the next section).
- **Write operations** – Allow you to move data from a user buffer in host memory to a frame buffer in device memory (see page 111).
- **Copy operations** – Allow you to move data either from one frame buffer to another in device memory or from one part of a frame buffer to another part of the same frame buffer in device memory (see page 112).
- **Map operations** – Allow a software application other than the Frame Grabber SDK to access data in a frame buffer in device memory (see page 112).

Display Operations

For monochrome frame grabber boards, you can display an acquired image in a window using **OlFgDrawAcquiredFrame** or **OlFgDrawAcquiredFrameEx**. **OlFgDrawAcquiredFrameEx** allows you to display part of an acquired image, if desired, and is faster than **OlFgDrawAcquiredFrame**, making it more appropriate for surveillance purposes.

To determine whether your monochrome frame grabber board supports displaying an acquired image with **OlFgDrawAcquiredFrame**, use **OlFgQueryInputCaps** with the key OLC_FG_IC_DOES_DRAW_ACQUIRED_FRAME. To determine whether your frame grabber board supports displaying an acquired image with **OlFgDrawAcquiredFrameEx**, refer to your board-specific documentation.

For color frame grabber boards, you can display an entire acquired image in a window using **DtColorDrawAcquiredFrame**, or you can display the contents of a single color plane in a window using **DtColorDrawBuffer**. Note that the color plane must have been previously extracted using **DtColorExtractFrametoBuffer**.

To determine whether your color frame grabber board supports displaying an acquired image with **DtColorDrawAcquiredFrame**, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_DRAW_ACQUIRED_FRAME. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot display an acquired image) or OLC_QUERY_COLOR_INTERFACE_DRAW_ACQUIRED_FRAME (if you cannot display an acquired image).

To determine whether your color frame grabber board supports displaying a single color plane, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_DRAW_BUFFER. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if

you cannot display a single color plane) or OLC_QUERY_COLOR_INTERFACE_DRAW_BUFFER (if you can display a single color plane).

To determine whether your color frame grabber board supports extracting color planes, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_EXTRACT_FRAME. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot extract color planes) or OLC_QUERY_COLOR_INTERFACE_EXTRACT_FRAME (if you can extract color planes).

Read Operations

Before performing a read operation, make sure that you have allocated a user buffer in host memory. For more information, refer to page 90.

When transferring data forom a frame buffer in device memory to a user buffer in host memory, you can do one of the following:

- Transfer data from a rectangular region of the frame buffer using **OlFgReadFrameRect**. This is useful if you want to copy a specific rectangular region of a frame. Specify the upper left pixel of the rectangle and the rectangle's width and height. The frame grabber board transfers the data contiguously.
- Transfer data from a contiguous section of the frame buffer using **OlFgReadContiguousPixels**. This is useful if you want to copy an entire section of a frame. For example, if you want to work with the bottom third of a 1200-pixel frame, you can transfer 400 pixels, starting with pixel 801. Specify the upper left pixel of the rectangle and the number of pixels to transfer from that point.
- Transfer selected pixel data from the frame buffer. This is useful if you want to transfer specific points in a frame or if you want to transfer an oddly shaped section of a frame. Define the points

you want to transfer using a Windows POINT structure and then use **OlFgReadPixelList** to make the transfer. The board stores the pixels linearly in the user buffer.

Make sure that the user buffer is big enough to hold all the data you want to transfer. In addition, make sure that the dimensions of the data you transfer do not exceed the dimensions of the frame buffer. To determine the dimensions of the frame buffer (in pixels) and the number of bytes used per pixel, use **OlFgQueryFrameInfo**.

Write Operations

Before performing a write operation, make sure that you have allocated a frame buffer in device memory. For more information, refer to page 86.

When transferring data from a user buffer in host memory to a frame buffer in device memory, you can do one of the following:

- Transfer data from a rectangular region of the user buffer using **OlFgWriteFrameRect**. Specify the upper left pixel of the rectangle and the rectangle's width and height. The frame grabber board transfers the data contiguously.
- Transfer data from a contiguous section of the user buffer using **OlFgWriteContiguousPixels**. Specify the upper left pixel of the rectangle and the number of pixels to transfer from that point.
- Transfer selected pixel data from the user buffer. Define the points you want to transfer using a Windows POINT structure and then use **OlFgWritePixelList** to make the transfer.

Make sure that the frame buffer is large enough to hold all the data you want to transfer. In addition, make sure that the dimensions of the data you transfer do not exceed the dimensions of the frame buffer. To determine the dimensions of the frame buffer (in pixels) and the number of bytes used per pixel, use **OlFgQueryFrameInfo**.

Copy Operations

To transfer data from one frame buffer to another frame buffer or from one part of a frame buffer to another part of the same frame buffer, use **OlFgCopyFrameRect**. Define the rectangle to copy by specifying the upper left pixel of the rectangle and the rectangle's width and height, and specify the pixel location where you want the upper left corner of the rectangle to be placed in the destination frame buffer. The frame grabber board transfers the data contiguously.

For example, you can copy a 500 x 300 rectangle from the upper left part of one frame buffer and transfer it to a 500 x 300 rectangle in the center of another frame buffer.

Make sure that if you place more than one rectangle in a single frame buffer, the rectangles do not overlap; if they do, the board may return an error. Even if the operation completes successfully, the data in the rectangles may be incorrect.

Make sure that the entire rectangle fits fully within the destination frame buffer; if it does not, the board returns an error. The board does not clip the rectangle to make it fit within the frame. To determine the size of a frame buffer, use **OlFgQueryFrameInfo**.

Map Operations

If a frame buffer can be mapped, you can enhance and/or analyze the data in the frame buffer using an editing software package. To determine whether you can map a frame buffer, use **OlFgQueryFrameInfo**. If the OLC_FG_FRAME_CAN_MAP flag is set in the returned set of bit flags, you can map the frame buffer.

To map a frame buffer, use **OlFgMapFrame**. After you have finished editing the data, use **OlFgUnmapFrame**. Until you unmap the frame buffer, the frame grabber board cannot access the frame buffer.

Digital I/O Operations

Some frame grabber boards provide digital input lines and/or digital output lines. The digital I/O signals are simple register-driven, TTL-level signals that you can use for any purpose, such as controlling or actuating external devices.

To determine whether your color frame grabber board supports programming the digital I/O lines, use **DtColorQueryInterface** with the constant OLC_QUERY_COLOR_INTERFACE_DIGITAL_IO. The query returns either OLC_QUERY_INTERFACE_UNSUPPORTED (if you cannot program the digital I/O lines) or OLC_QUERY_COLOR_INTERFACE_DIGITAL_IO (if you can program the digital I/O lines).

To determine whether your monochrome frame grabber board supports programming the digital I/O lines, refer to your board-specific documentation.

The following sections describe how to

- Configure digital lines as input or output (see the next section),
- Write data to a digital output line (see page 114), and
- Read data from a digital input line (see page 115).

Digital I/O Configuration

For most frame grabber boards, the configuration of the digital I/O lines is fixed. To determine whether your frame grabber board supports configuring the digital I/O lines, refer to your board-specific documentation. To configure the digital I/O lines, use either **DtColorDigitalIOControl** with the control OLC_CONFIGURE_CONTROL or **Dt3157SetDigitalIOConfiguration**.

To configure the digital I/O lines, create a bit mask in which each bit represents a different line, interpreted from the low order bit (bit 0 represents line 0, bit 1 represents line 1, and so on). When a bit is set to 1, the corresponding line is configured as an output. When a bit is set to 0, the corresponding line is configured as an input.

To determine the current digital I/O configuration, use either **DtColorDigitalIOControl** with the control OLC_QUERY_CONFIGURATION or **Dt3157QueryDigitalIOConfiguration**.

Digital Output

You can use the Frame Grabber SDK to write data to the digital output lines on your frame grabber board. To determine the number of digital output lines on your frame grabber board, use **OlFgQueryCameraControlCaps** with the key OLC_FG_CC_DIG_OUT_COUNT.

To write data to a digital output line, use either **OlFgSetDigitalOutputMask**, **DtColorDigitalIOControl** with the control OLC_WRITE_CONTROL, or **Dt3157SetDigitalIO**.

To write to a digital output line, create a bit mask correlating to the high/low settings of each of the output lines. Digital output lines are interpreted starting from the low order bit of the mask (bit 0 represents digital output line 0, bit 1 represents digital output line 1, and so on).

Set a bit to 1 to write a high-level TTL signal to the corresponding digital output line; set a bit to 0 to write a low-level TTL signal to the corresponding digital output line. For example, assume that your frame grabber board supports eight digital I/O lines, that lines 0, 1, 2, and 3 are configured for output, and that lines 4, 5, 6, and 7 are configured for input (0000 1111). If the bit mask equals 5h (0000 0101), the output from lines 0 and 2 is high and the output from lines 1 and 3 is low. Values written to input lines 4, 5, 6, and 7 are ignored.

Digital Input

If your frame grabber board contains digital input lines, you can use the Frame Grabber SDK to read data from the lines.

To read data from a digital input line, use either **DtColorDigitalIOControl** with the control OLC_READ_CONTROL or **Dt3157QueryDigitalIO**. The return value correlates to the high/low values of each of the input lines. If a bit is set to 1, the input is high; if a bit is set to 0, the input is low.

For example, assume that your frame grabber board supports eight digital I/O lines, that lines 0, 1, 2, and 3 are configured for input, and that lines 4, 5, 6, and 7 are configured for output (1111 0000). If the return value is 5h (0000 0101), the input at lines 0 and 2 is high and the input at lines 1 and 3 is low. Values returned from output lines 4, 5, 6, and 7 are the last values written to the lines.

Line-Scan Operations

Some frame grabber boards allow you to perform line-scan operations. These boards acquire lines (rows of pixels) only and create one-dimensional images. Only horizontal (line) sync signals are accepted from the video source. Vertical syncs are ignored.

Use **OlFgEnableLsMode** to enable and disable line-scan mode. When line-scan mode is enabled, the following features of the Frame Grabber SDK are either unsupported or perform in a different way:

- Composite and variable-scan video signal are not supported.
- A chrominance notch filter is not supported.
- The Sync Sentinel is not supported.
- Sync Master mode is not supported.
- Triggering a multiple-frame acquisition is not supported.
- Line-scan frame grabber boards support unique output signals. For more information, refer to page 117.
- Programming the total video area (total pixels per line and total lines per field) is not supported.
- When programming the horizontal video area, you cannot set the back porch start position or the active pixel count. You can set only the clamp start position, the clamp end position, and the first active pixel.
- Programming the vertical video area (first active line and active line count) is not supported.
- When programming the size of a frame, you cannot set the frame top or the frame left. You can set only the frame height and the frame width.
- Programming the type of frame is not supported.
- Scaling a frame is not supported.

- Acquiring image data into frame buffers is not supported. You must acquire data into a user buffer; for more information, refer to page 119.
- For passthru operations, the only supported mode is continuous-acquire line-scan passthru mode; for more information, refer to page 119. You cannot scale the passthru image, modify the passthru LUT, take a snapshot, or create overlays. You can, however, adjust the source origin.
- Acquiring line-scan data requires unique line-scan functions. For more information, refer to page 122.
- Displaying acquired line-scan data requires a unique line-scan function. For more information, refer to page 122.
- Performing digital I/O operations requires unique line-scan functions. For more information, refer to page 123.

Note: All other features of the Frame Grabber SDK perform in the same way whether line-scan mode is enabled or disabled.

Line-Scan Output Signals

Some line-scan frame grabber boards can generate the following output signals:

- **Integration output signal** – The integration output pulse controls the exposure time of the camera. You can use **OlFgSetLsIntegration** to specify the number of internal pixel clock pulses that you want to occur before the specified edge (rising or falling) of the integration output pulse becomes active.

  For example, assume that you use **OlFgSetLsIntegration** to set the active edge to low-to-high and the pixel clock value to 4095 and then use **OlFgSetLsIntegration** again to set the active edge to high-to-low and the pixel clock value to 5000. The integration

output signal goes high at pixel count 4095 and goes low at pixel count 5000.

If the maximum pixel count does not allow a long enough period, you can use **OlFgSetLsDriveClkDiv** to specify a clock divider to change the period of the integration output pulse.

To determine the current pulse count and polarity, use **OlFgGetLsIntegration**. To determine the current value of the clock divider, use **OlFgGetLsDriveClkDiv**.

- **Line-sync output signal** – The line-sync output pulse drives the line-sync camera and controls the camera's line rates. The video signal from the camera is then digitized as usual, using the syncs returned by the camera as the sync basis. You can use **OlFgSetLsLineDrive** to specify the number of internal pixel clock pulses that you want to occur before the specified edge (rising or falling) of the line-sync output pulse becomes active.

  For example, assume that you use **OlFgSetLsLineDrive** to set the active edge to low-to-high and the pixel clock value to 5127 and then use **OlFgSetLsLineDrive** again to set the active edge to high-to-low and the pixel clock value to 6000. The line-sync output pulse goes high at pixel count 5127 and goes low at pixel count 6000.

  If the maximum pixel count does not allow a long enough period, you can use **OlFgSetLsDriveClkDiv** to specify a clock divider to change the period of the line-sync output pulse.

  To determine the current pulse count and polarity, use **OlFgGetLsLineDrive**. To determine the current value of the clock divider, use **OlFgGetLsDriveClkDiv**.

With the falling edge of the line-sync output pulse, another line starts and the pixel clock counter is reset to 0.

Line-Scan Memory Allocation Operations

For line-scan operations, you must acquire data into a user buffer. For more information on allocating a user buffer, refer to page 90.

Note: For a line-scan operation, a frame is defined by successive line acquisitions. Since no predefined frame height exists, the frame can be as large as the available system memory allows.

If you experience buffer overflows during line-scan operations, either increase the amount of system memory available or increase the amount of transfer memory in the device driver configuration. For more information, refer to your board-specific documentation.

Line-Scan Passthru Operations

Frame grabber boards that support line-scan operations typically support continuous-acquire line-scan passthru operations. A continuous-acquire line-scan passthru operation allows you to store line data in a user buffer while you display the data in a window. You can display the data without storing it, store the data without displaying it, or display and store the data.

Use **OlFgStartAsyncLsPassthru** to start a continuous-acquire line-scan passthru operation. Use **OlFgStopLsPassthru** to stop a continuous-acquire line-scan passthru operation.

Before you perform a continuous-acquire line-scan passthru operation, you must allocate a user buffer. For more information, refer to page 90.

In addition, you must globally allocate

- A list of pointers to the user buffers (contains one pointer per user buffer), and
- A list of buffer-done flags (contains one flag per user buffer).

Refer to Figure 9 for an example of using these buffer lists.

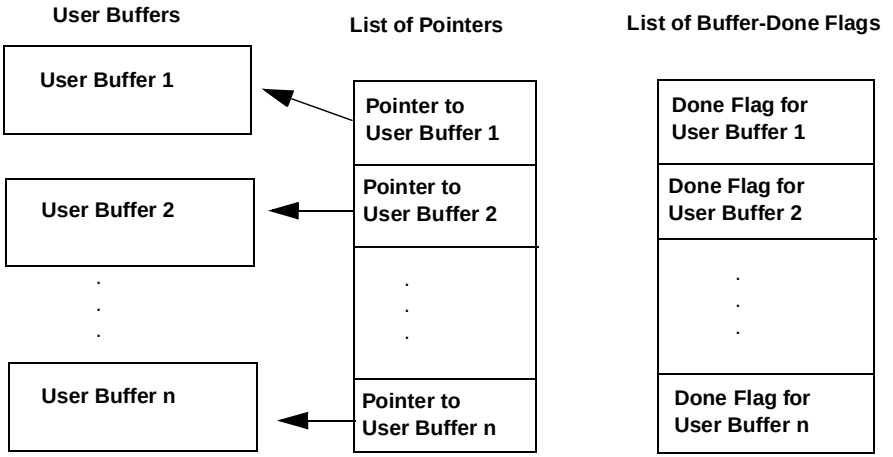


Figure 9: Buffers Needed in Continuous-Acquire Line-Scan Passthru Mode

To globally allocate a list of pointers and a list of buffer-done flags from the main heap of your process, use the WIN32 function **GlobalAlloc()**; ensure that you specify the GPTR flag to lock the buffer lists in memory. The size of each buffer list must be at least as large as the number of user buffers multiplied by the data type of the argument (LPVOID for the list of pointers; DWORD for the list of buffer-done flags).

When a user buffer has been filled with data, the corresponding buffer-done flag is set to TRUE; otherwise, the buffer-done flag is FALSE.

You can use the WIN32 event synchronization object that was supplied to you when you started the continuous-acquire line-scan passthru operation to synchronize your application with your frame grabber board. For example, you can call the WIN32 function **WaitForSingleObject()**, where the object is the WIN32 event, to put the calling thread to sleep until a new user buffer is ready. When the new user buffer has been filled, the driver sets the corresponding buffer-done flag to TRUE, and pulses the WIN32 event synchronization object. The operating system then allocates CPU time to the sleeping threads.

Once the buffer-done flag is set to TRUE, you can process the data. To reuse the user buffer, you must reset the buffer-done flag to FALSE. If you do not reset the buffer-done flags and the LS_PASS_STOP_ON_OVERFLOW flag is set in **OlFgStartAsyncLsPassthru**, the continuous-acquire line-scan passthru operation stops when all the buffer-done flags are set to TRUE. If the LS_PASS_DONT_STOP_ON_OVERFLOW flag is set, the data in the user buffers is overwritten and the reference count of the buffer-done flag is increased by one.

Refer to the LS-Acquire source code for an example of how to allocate user buffers and buffer lists, how to use the WIN32 event synchronization object, and how to deal with the buffer-done flags.

Line-Scan Acquisition Operations

Note: Before you can acquire lines, you must allocate a user buffer to hold the data. For information about allocating a user buffer, refer to page 90.

After allocating the user buffer, use **OlFgAcquireLines** to acquire the lines of data to the user buffer either synchronously or asynchronously. If you specified an asynchronous acquisition, use **OlFgIsAcquireLinesDone** to determine whether the acquisition has completed. If the acquisition has completed, **OlFgIsAcquireLinesDone** returns TRUE; otherwise, **OlFgIsAcquireLinesDone** returns the number of lines acquired in the user buffer and the granularity of the value.

After you acquire line-scan data, you can use **OlFgDrawAcquiredLines** to draw the contents of the user buffer to a window.

Line-Scan Display Operations

For monochrome frame grabber boards, you can use **OlFgDrawAcquiredLines** to draw the contents of the user buffer containing the acquired lines to a window. To determine whether your frame grabber board supports displaying an acquired image with **OlFgDrawAcquiredLines**, refer to your board-specific documentation.

Line-Scan Digital I/O Operations

To write data to a digital output line, use **OlFgSetLsDigIo**. For more information about digital output operations, refer to page 114. To read data from a digital input line, use **OlFgGetLsDigIo**. For more information about digital input operations, refer to page 115.

Programming Flowcharts

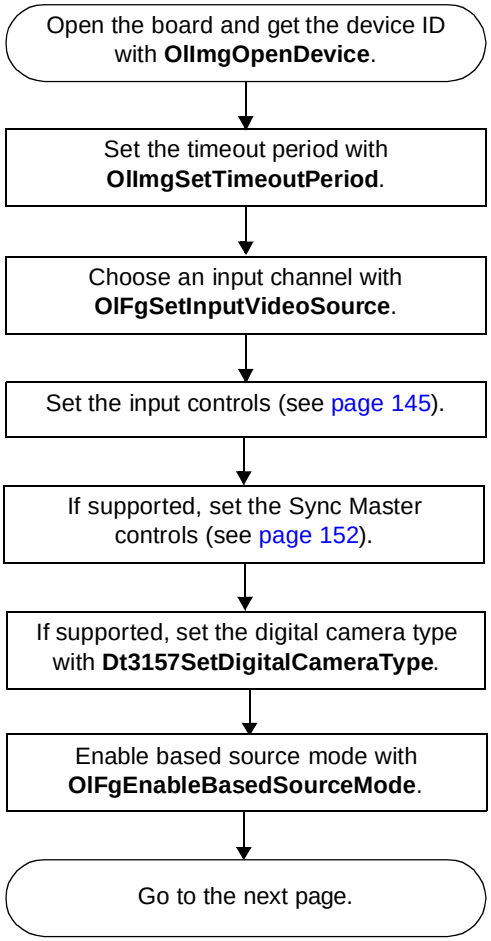
This chapter provides a series of flowcharts showing an overview of the steps required to perform each of the operations supported by the Frame Grabber SDK.

Note that query functions are not included in the flowcharts. If you are unfamiliar with the capabilities of your frame grabber board, query the board as follows:

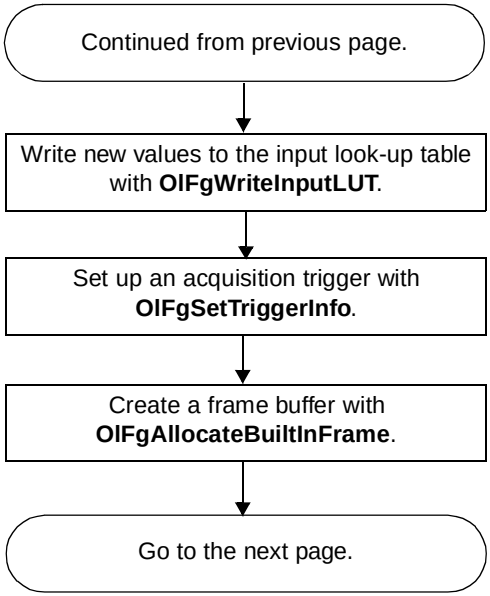
- To determine the number of DT-Open Layers frame grabber boards in your system, use **OlImgGetDeviceCount**.
- To determine the alias, Data Translation product name, and board type for a closed DT-Open Layers frame grabber board, use **OlImgGetDeviceInfo**. To determine this information along with the general capabilities of an open board, use **OlImgQueryDeviceCaps**.
- To determine whether your frame grabber board supports volatile and/or nonvolatile memory, use **OlFgQueryMemoryCaps**.
- To determine whether your frame grabber board supports setting input values for black and white levels, input video source, active video area, Sync Sentinel, digitized frame, pixel clock, trigger, input filter, and strobe output signal, use **OlFgQueryInputCaps**.
- To determine the number of digital output lines your frame grabber board supports, use **OlFgQueryCameraControlCaps**.
- To determine whether your frame grabber board supports passthru operations, use **OlFgQueryPassthruCaps**.
- To determine whether your frame grabber board supports overlays, use **OlFgQueryDDICaps**.
- To determine whether your frame grabber board supports setting color parameters, use **DtColorQueryInterface**, **DtColorHardwareScaling**, **DtColorImageParameters**, **DtColorSignalType**, **DtColorSyncMasterMode**, and **DtColorStorageMode**.

Although the flowcharts do not show error/status checking, it is recommended that you check for error/status messages after calling each function.

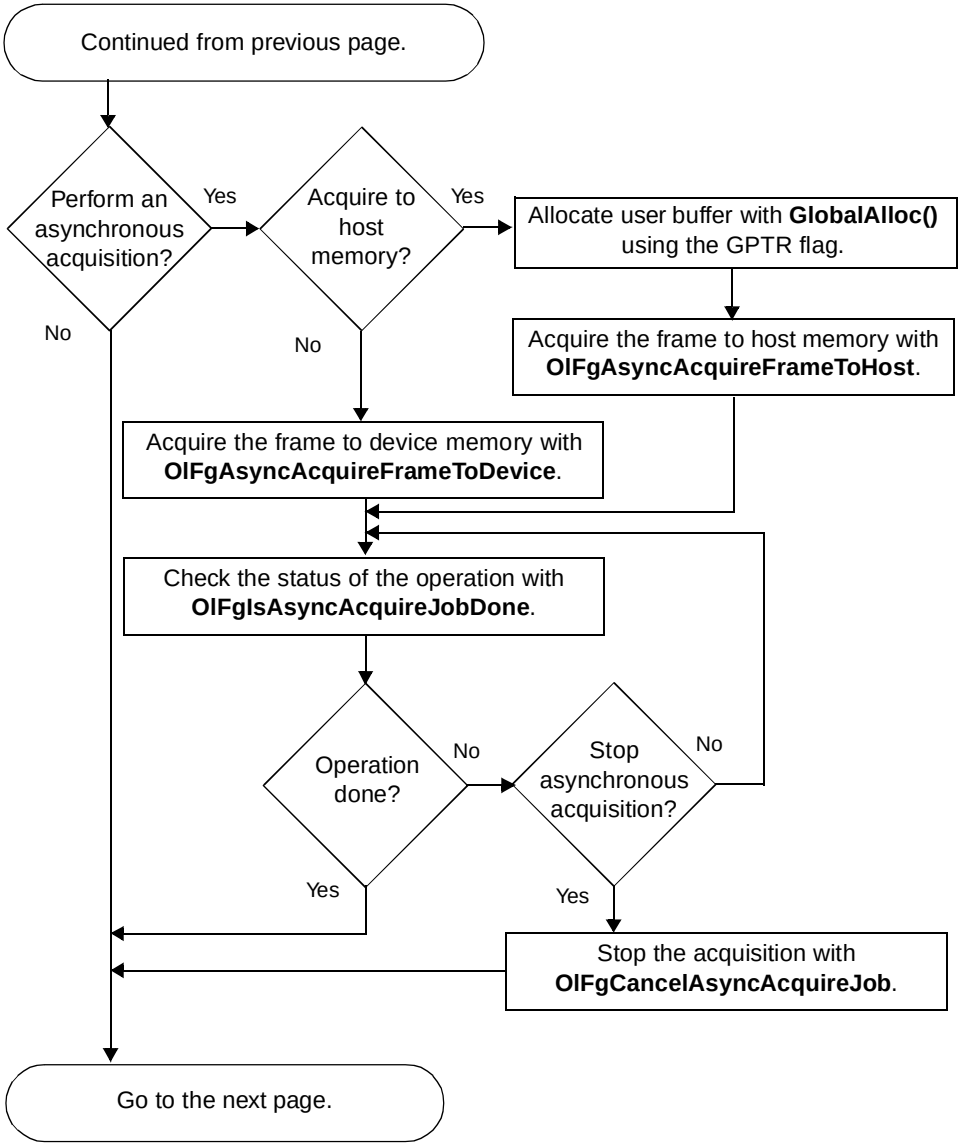
Single-Frame Acquisition



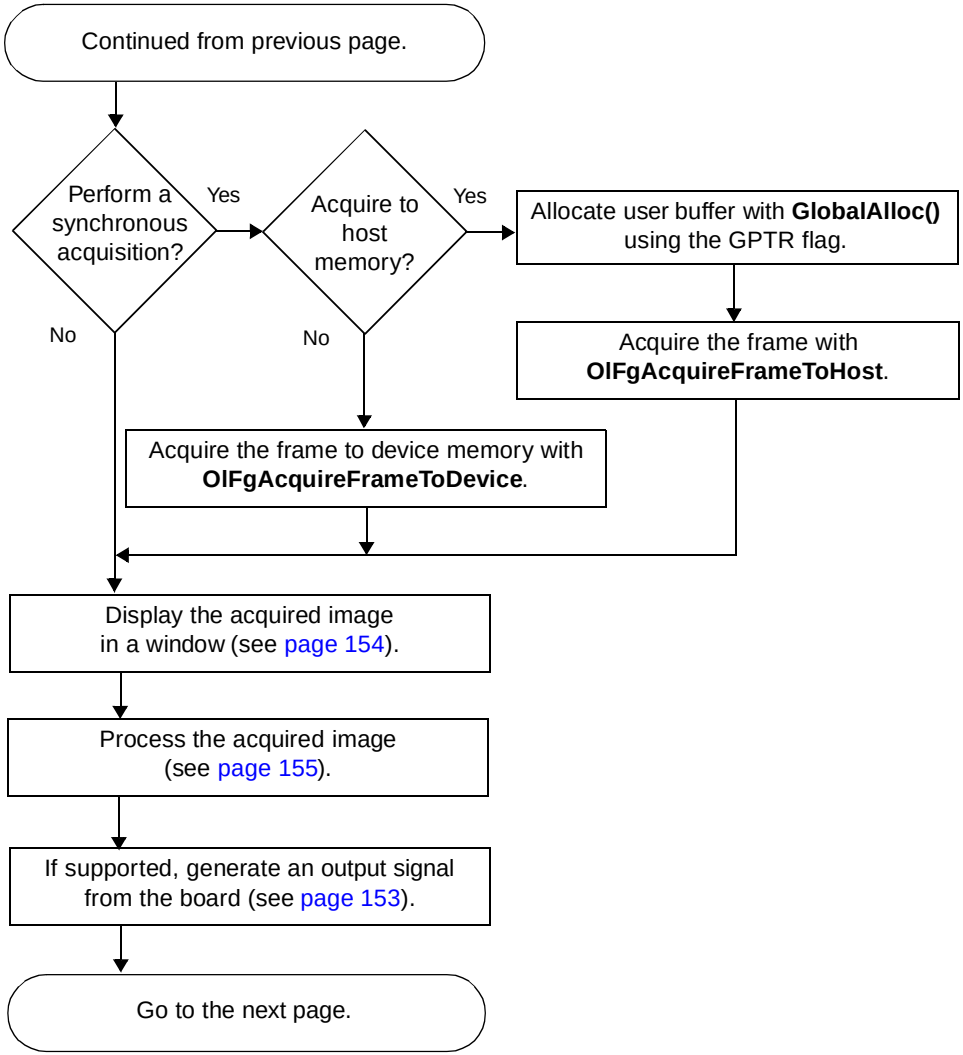
Single-Frame Acquisition (cont.)



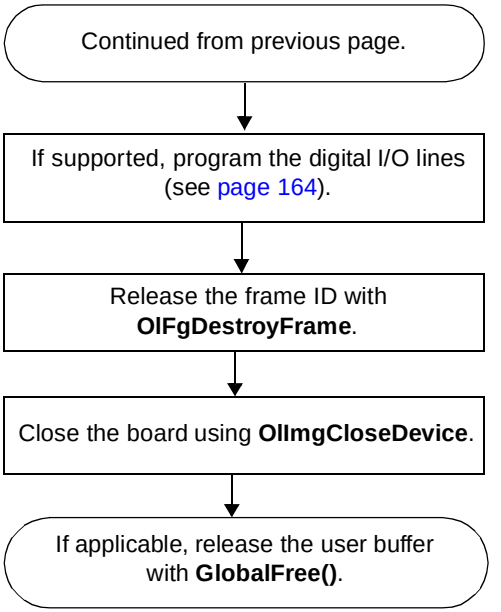
Single-Frame Acquisition (cont.)



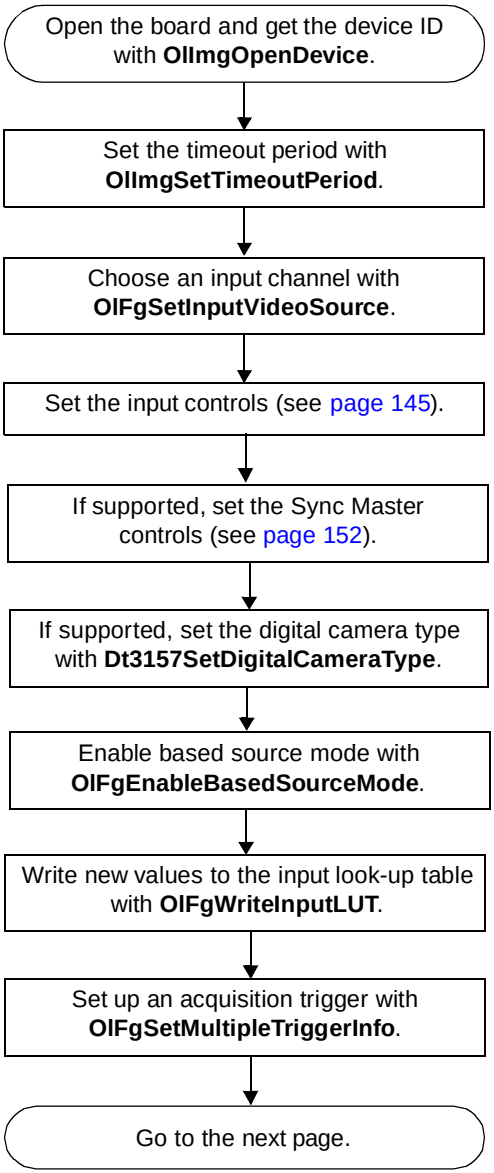
Single-Frame Acquisition (cont.)



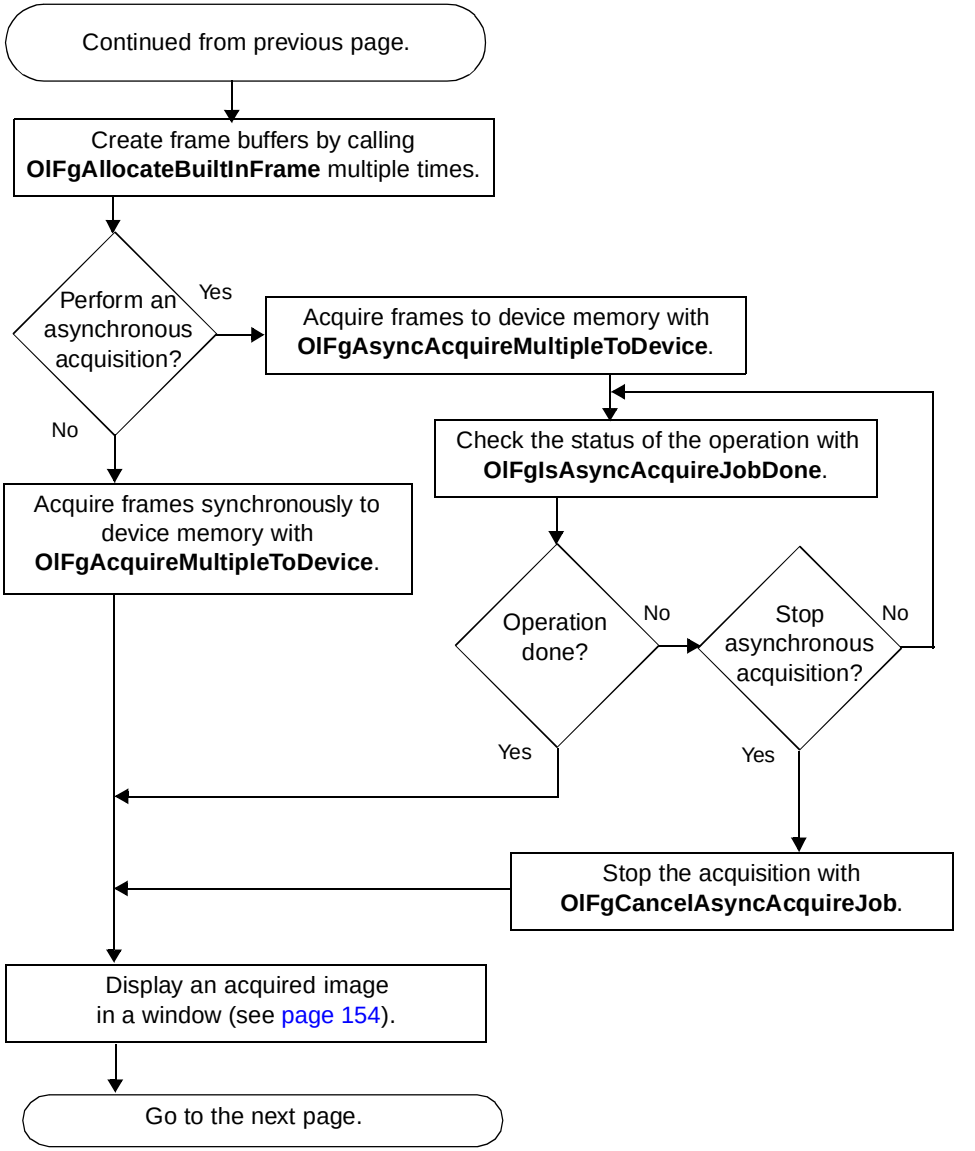
Single-Frame Acquisition (cont.)



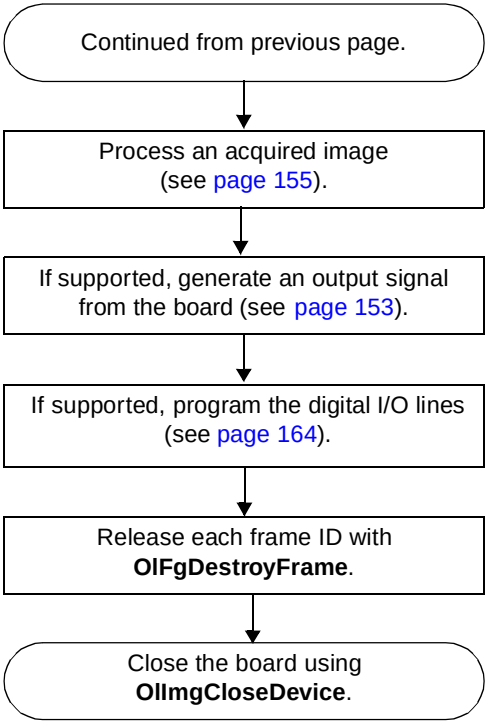
Multiple-Frame Acquisition



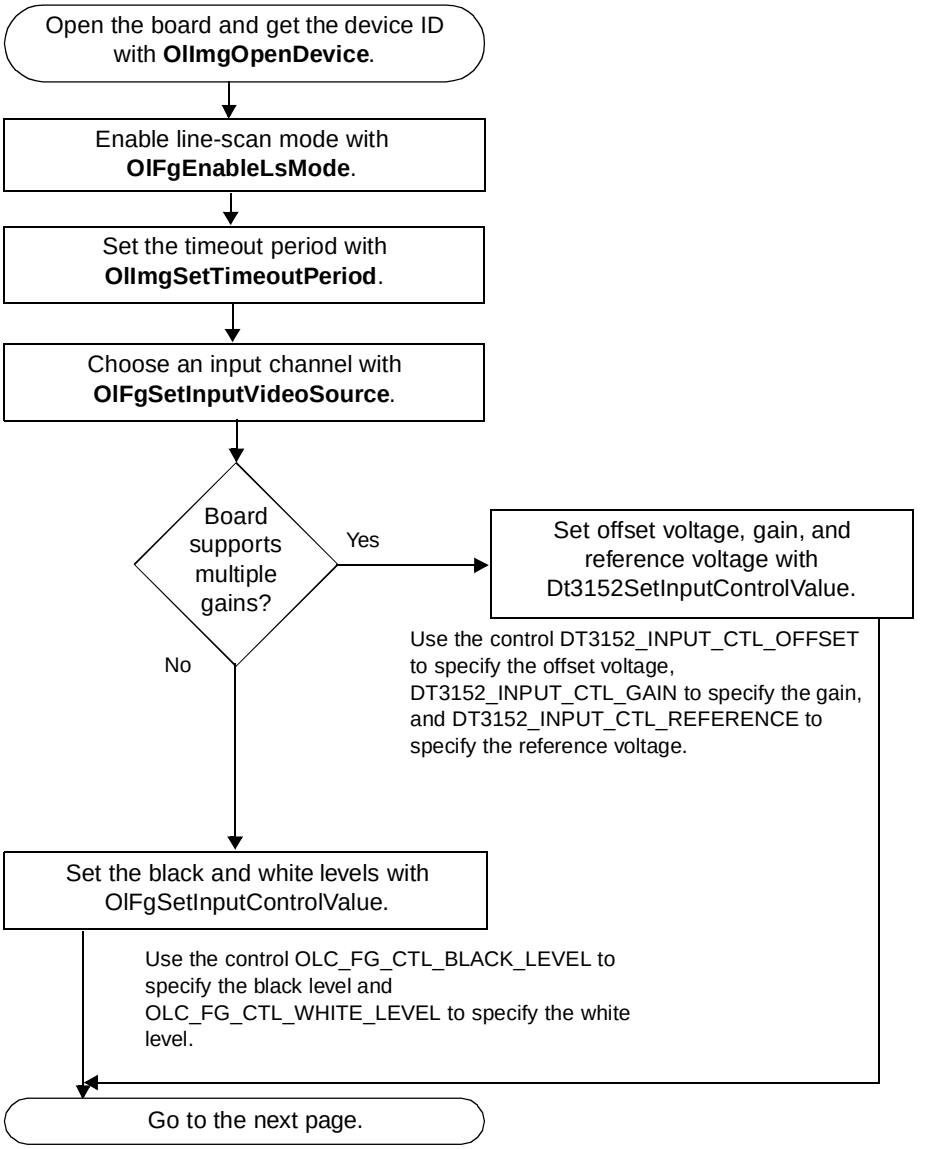
Multiple-Frame Acquisition (cont.)



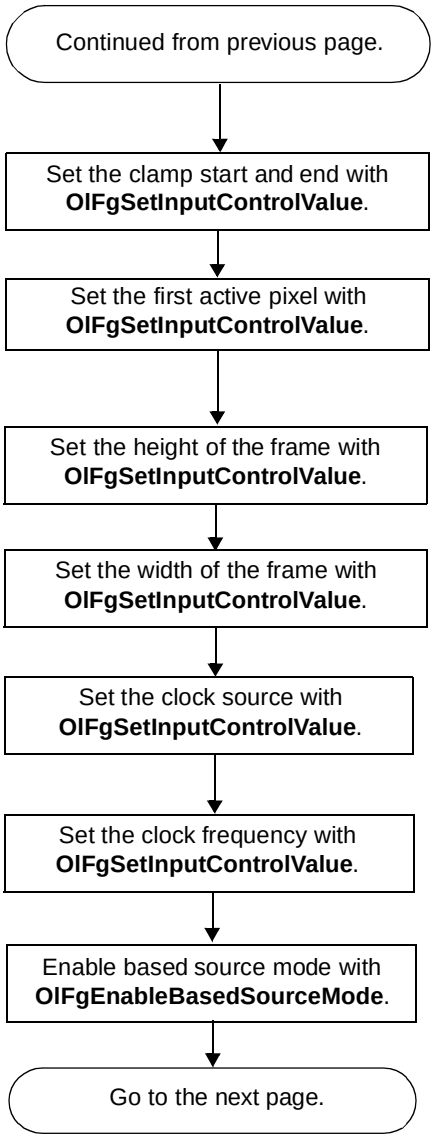
Multiple-Frame Acquisition (cont.)



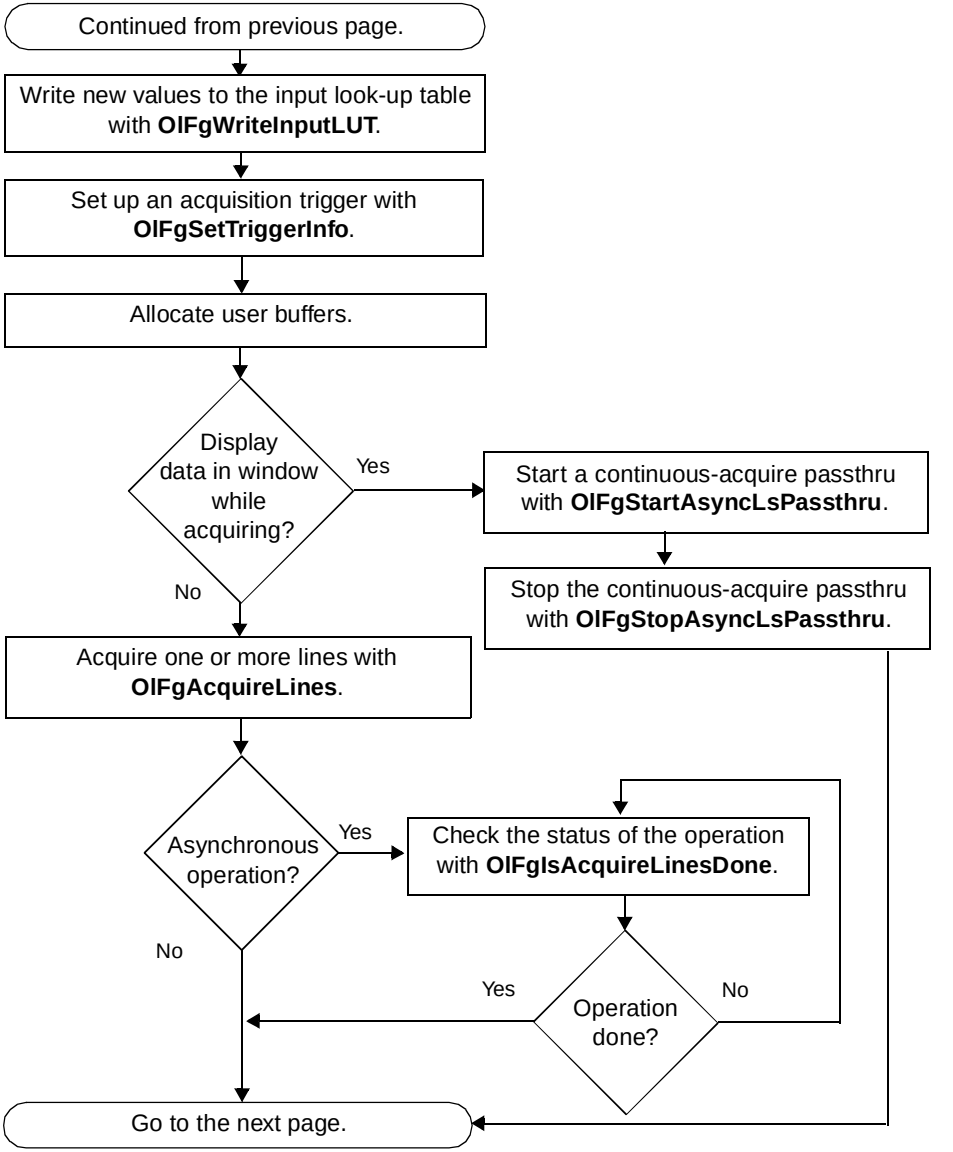
Line-Scan Acquisition



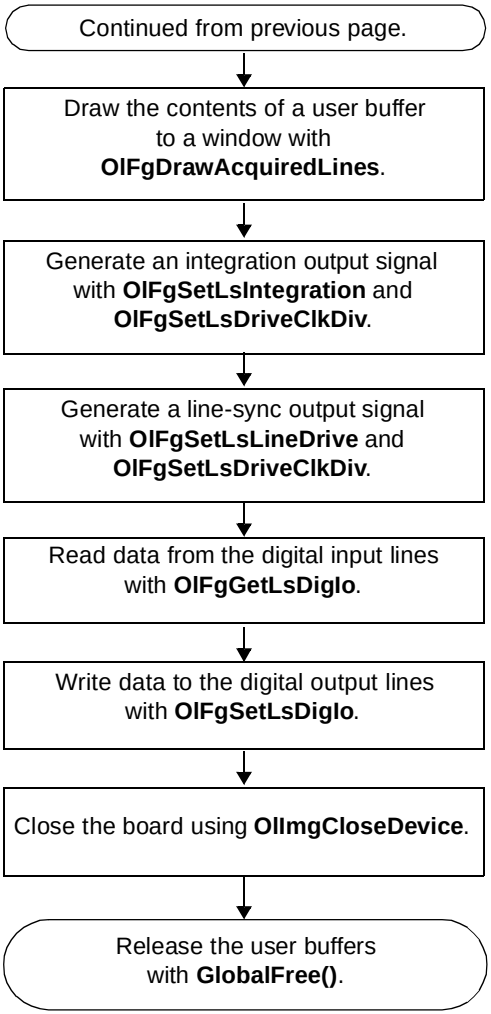
Line-Scan Acquisition (cont.)



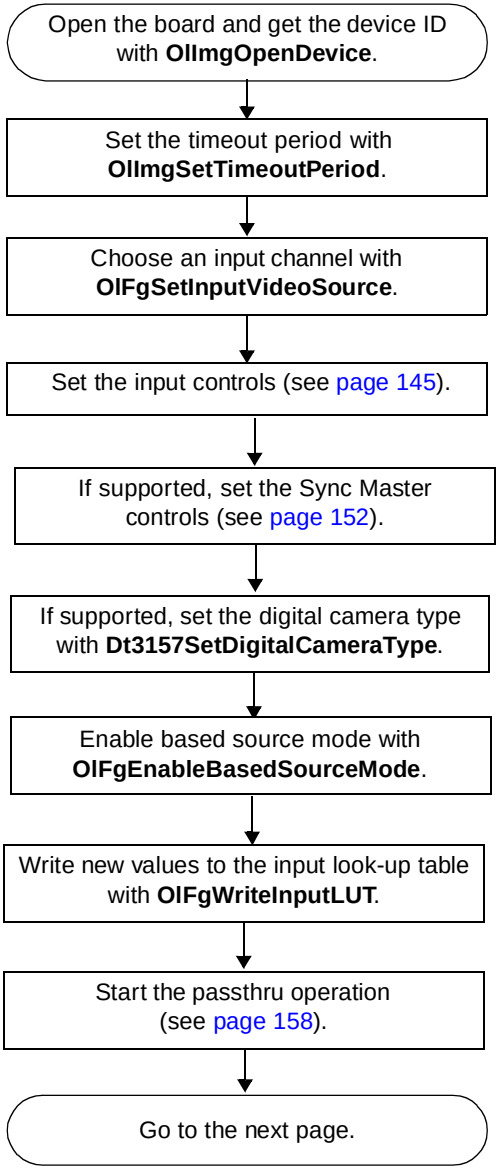
Line-Scan Acquisition (cont.)



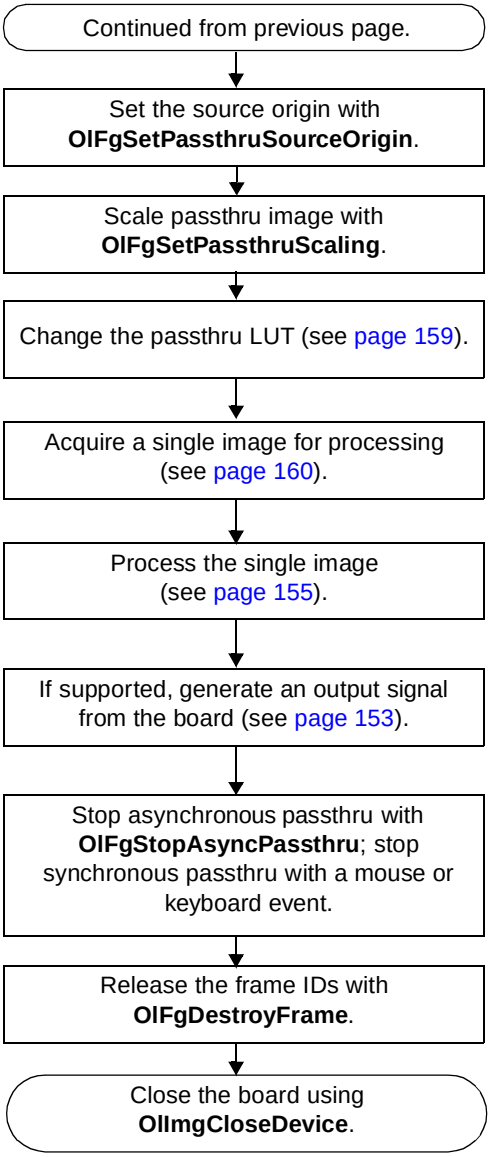
Line-Scan Acquisition (cont.)



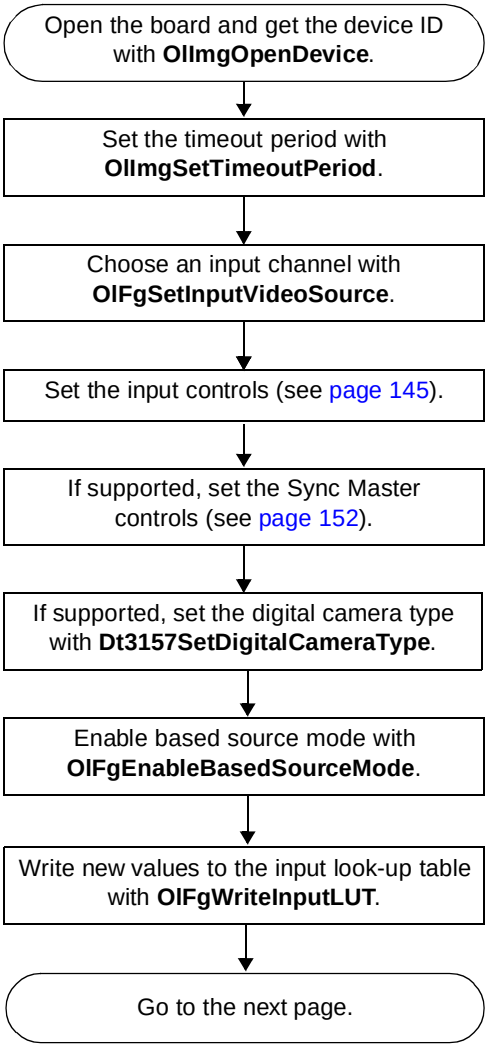
Passthru without Overlays



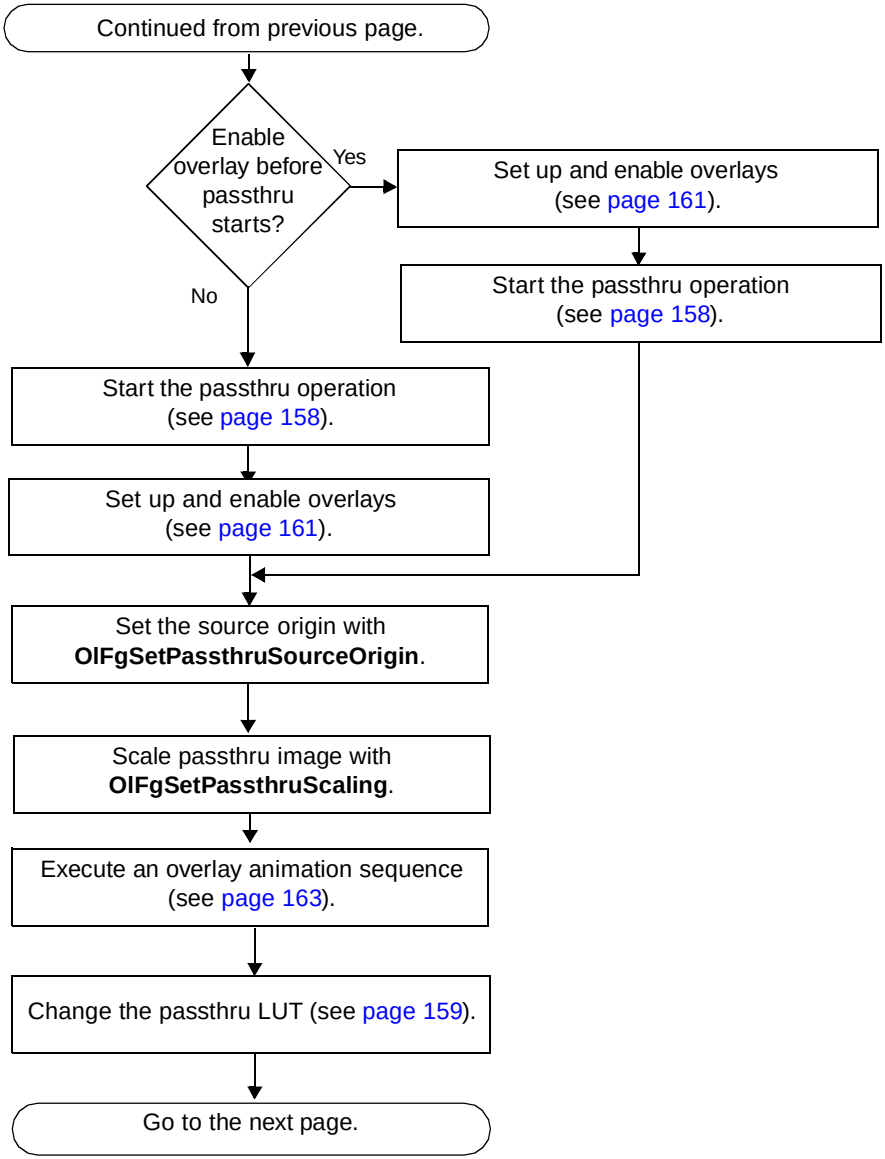
Passthru without Overlays (cont.)



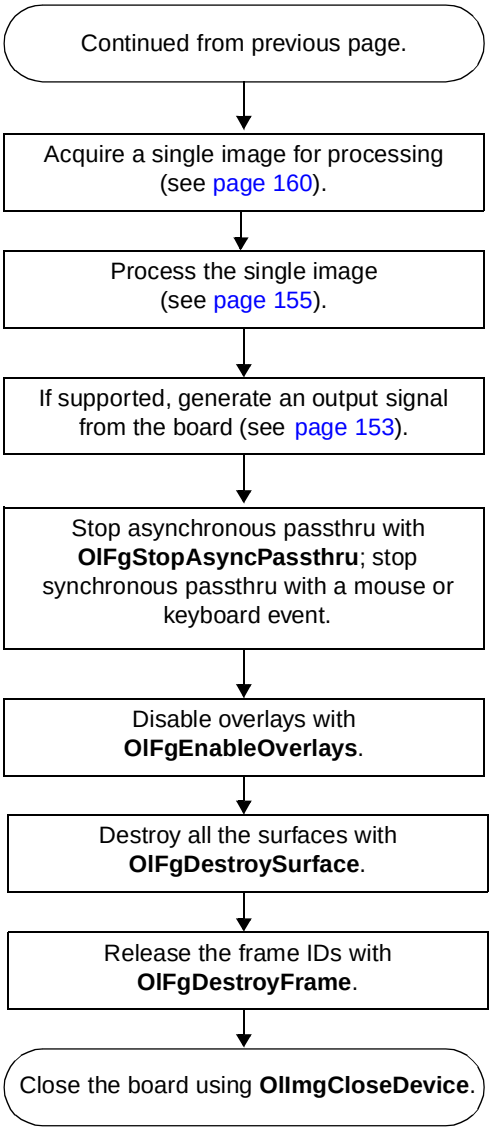
Passthru with Overlays



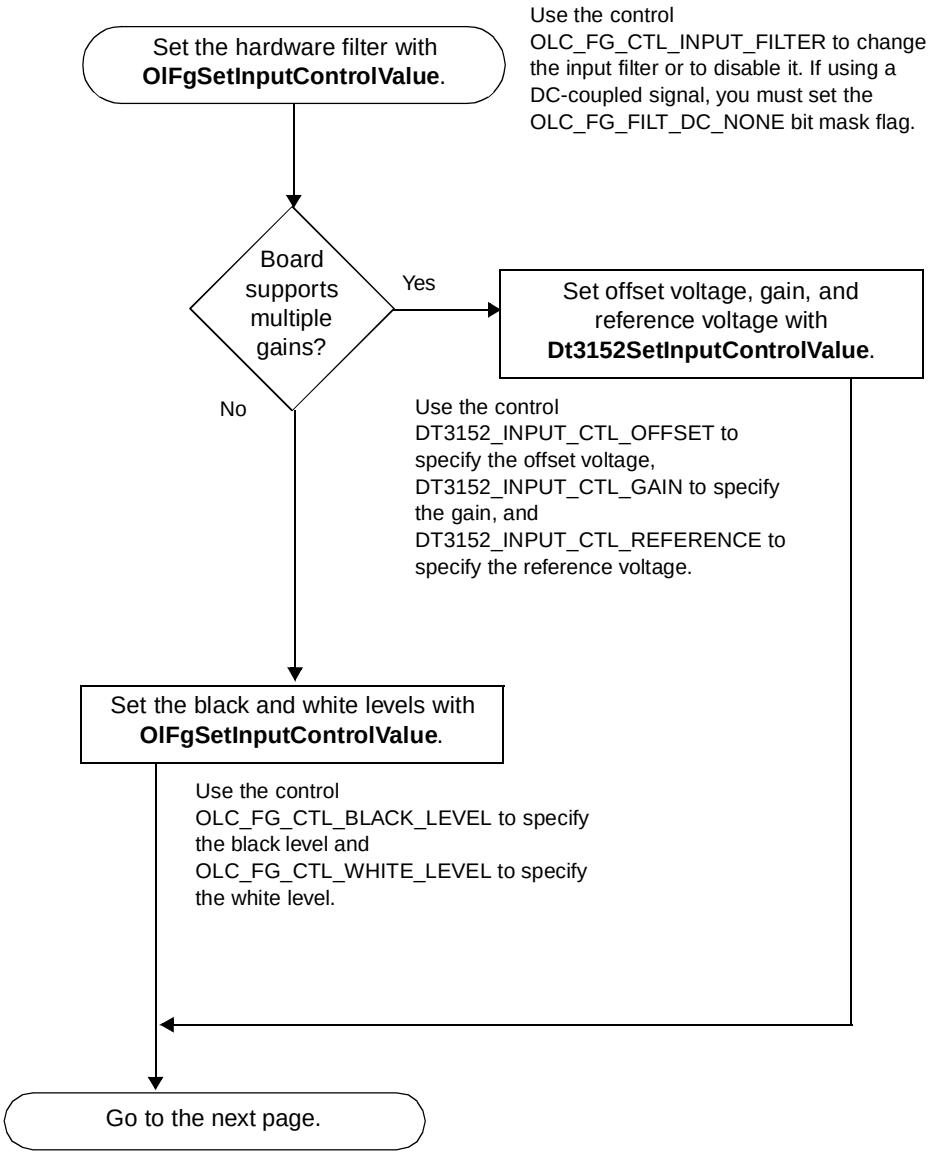
Passthru with Overlays (cont.)



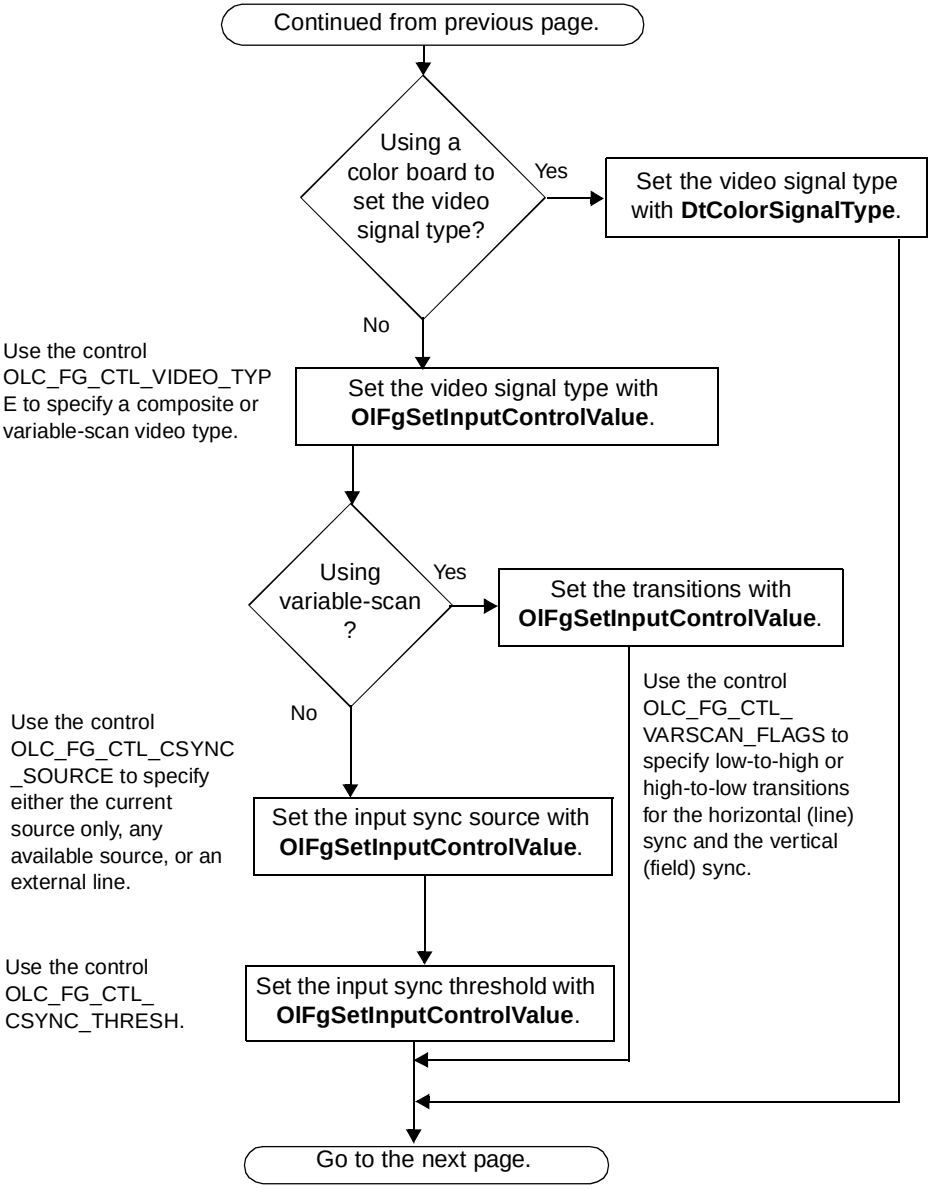
Passthru with Overlays (cont.)



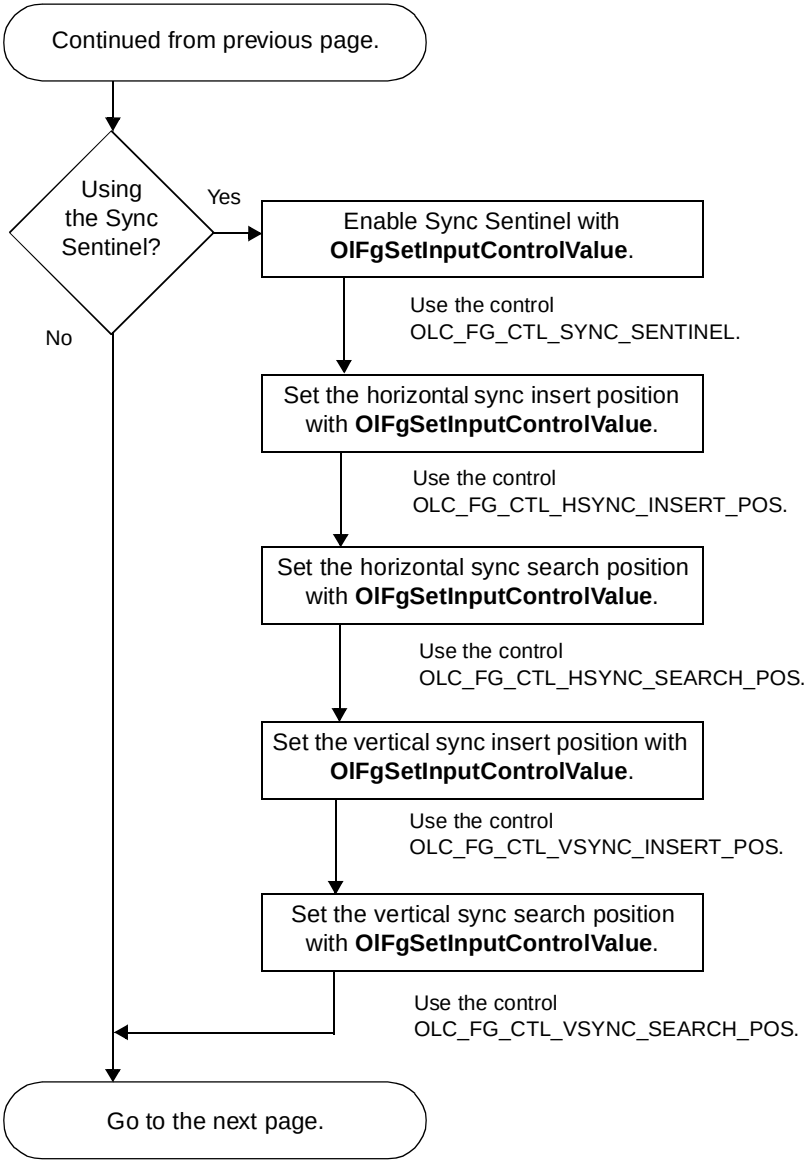
Set the Input Controls



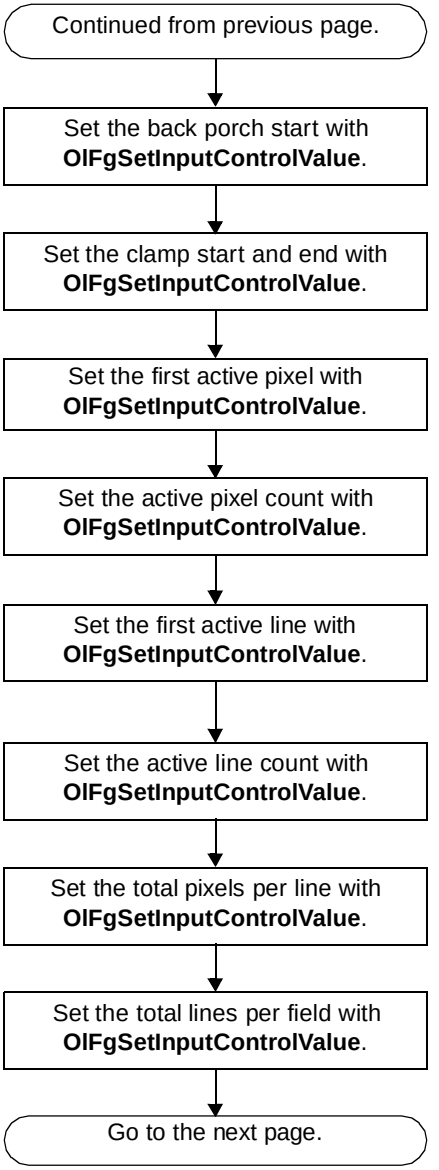
Set the Input Controls (cont.)



Set the Input Controls (cont.)



Set the Input Controls (cont.)



Use the control OLC_FG_CTL_BACK_PORCH_START to specify the back porch start position in the horizontal video signal, in pixels.

Use the control OLC_FG_CTL_CLAMP_START to specify the clamp start position and OLC_FG_CTL_CLAMP_END to set the clamp end position, in pixels.

Use the control OLC_FG_CTL_FIRST_ACTIVE_PIXEL to specify the offset from the beginning of the horizontal sync, in pixels.

Use the control OLC_FG_CTL_ACTIVE_PIXEL_COUNT to specify the number of pixels per line in the active video area, in pixels.

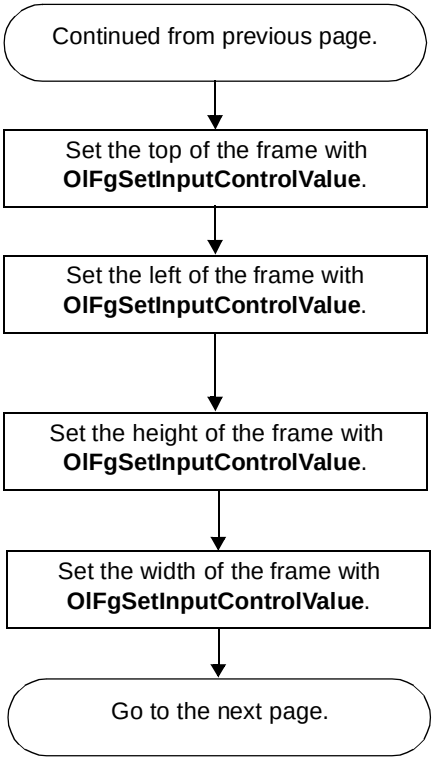
Use the control OLC_FG_CTL_FIRST_ACTIVE_LINE to specify the offset from the beginning of the vertical sync, in pixels.

Use the control OLC_FG_CTL_ACTIVE_LINE_COUNT to specify the number of lines per field (or noninterlaced frame) in the active video area.

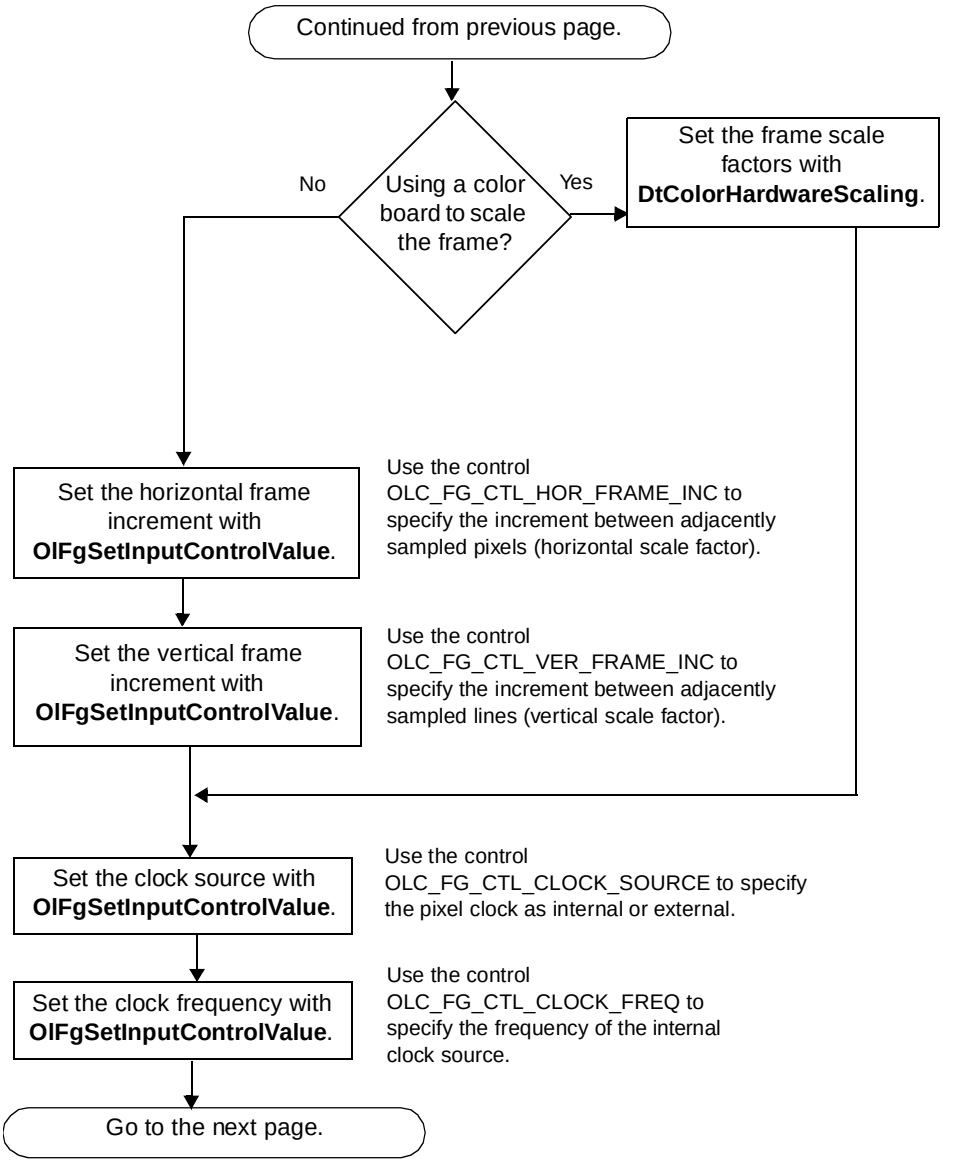
Use the control OLC_FG_CTL_TOTAL_PIX_PER_LINE to specify the total number of pixels in a single horizontal line of video.

Use the control OLC_FG_CTL_TOTAL_LINES_PER_FLD to specify the total number of lines between two consecutive vertical syncs.

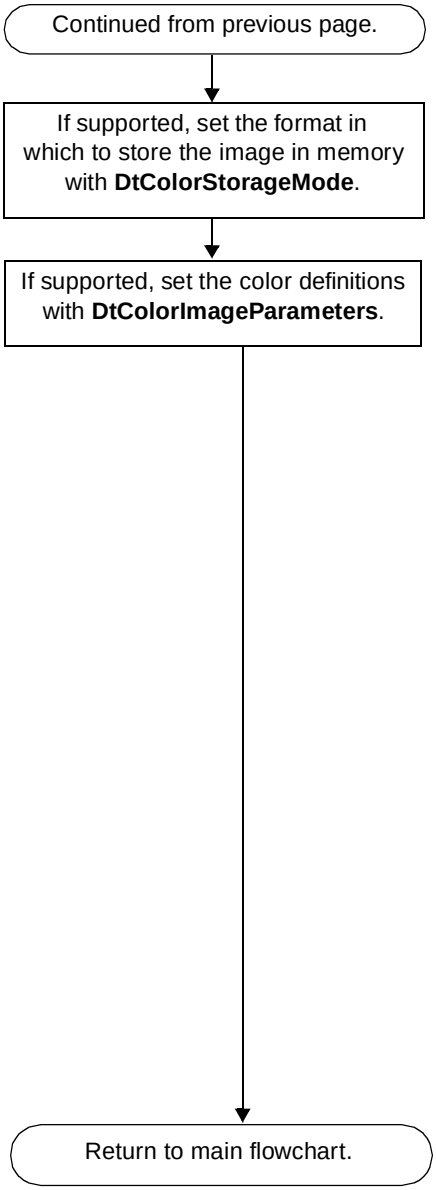
Set the Input Controls (cont.)



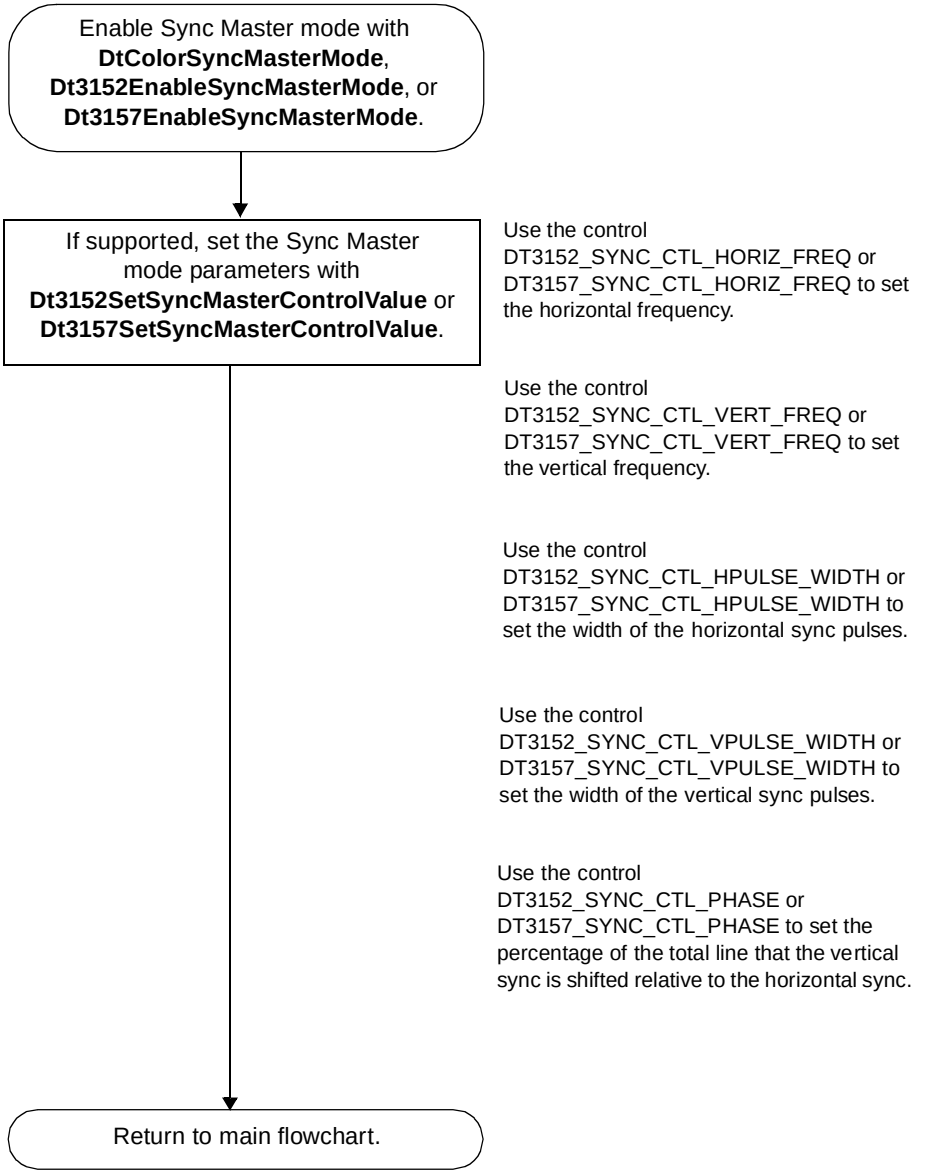
Set the Input Controls (cont.)



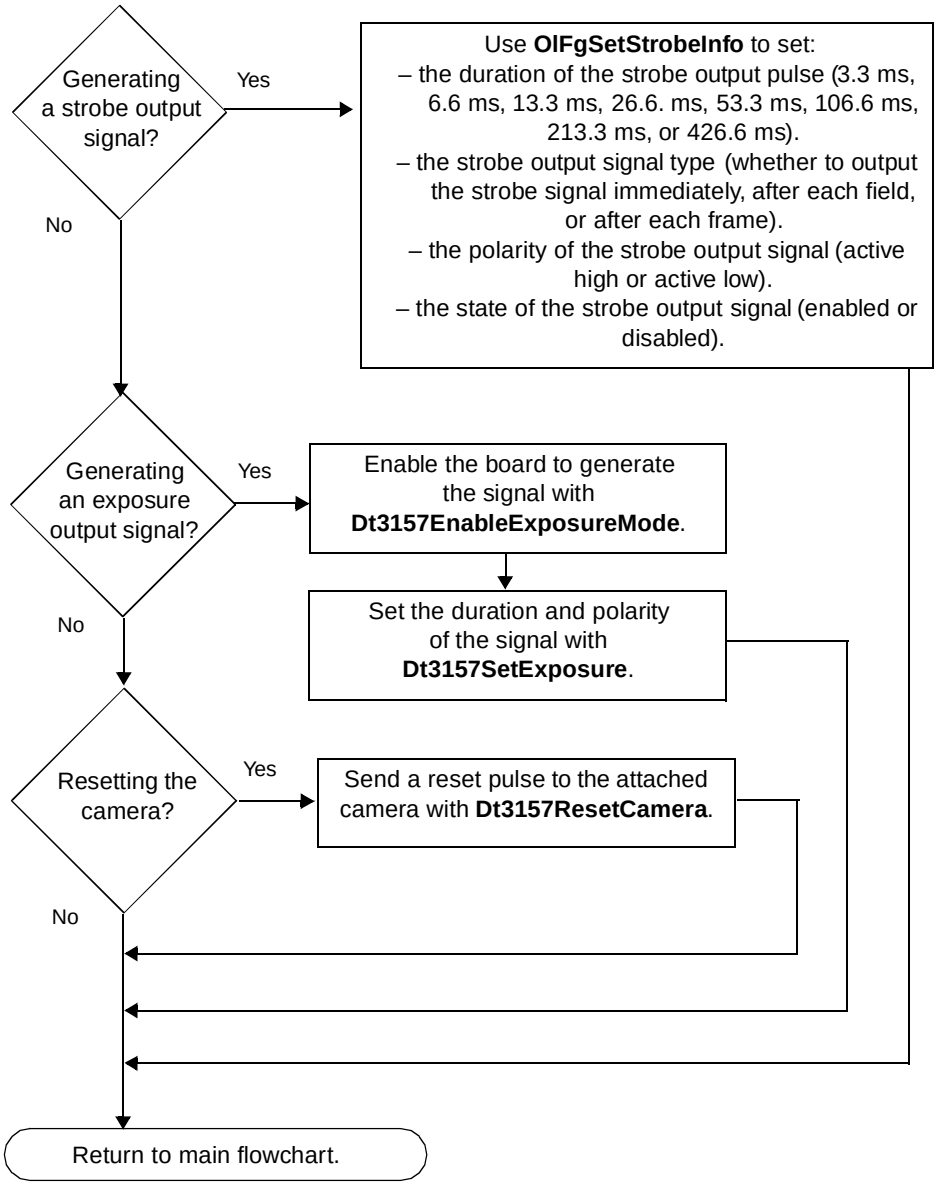
Set the Input Controls (cont.)



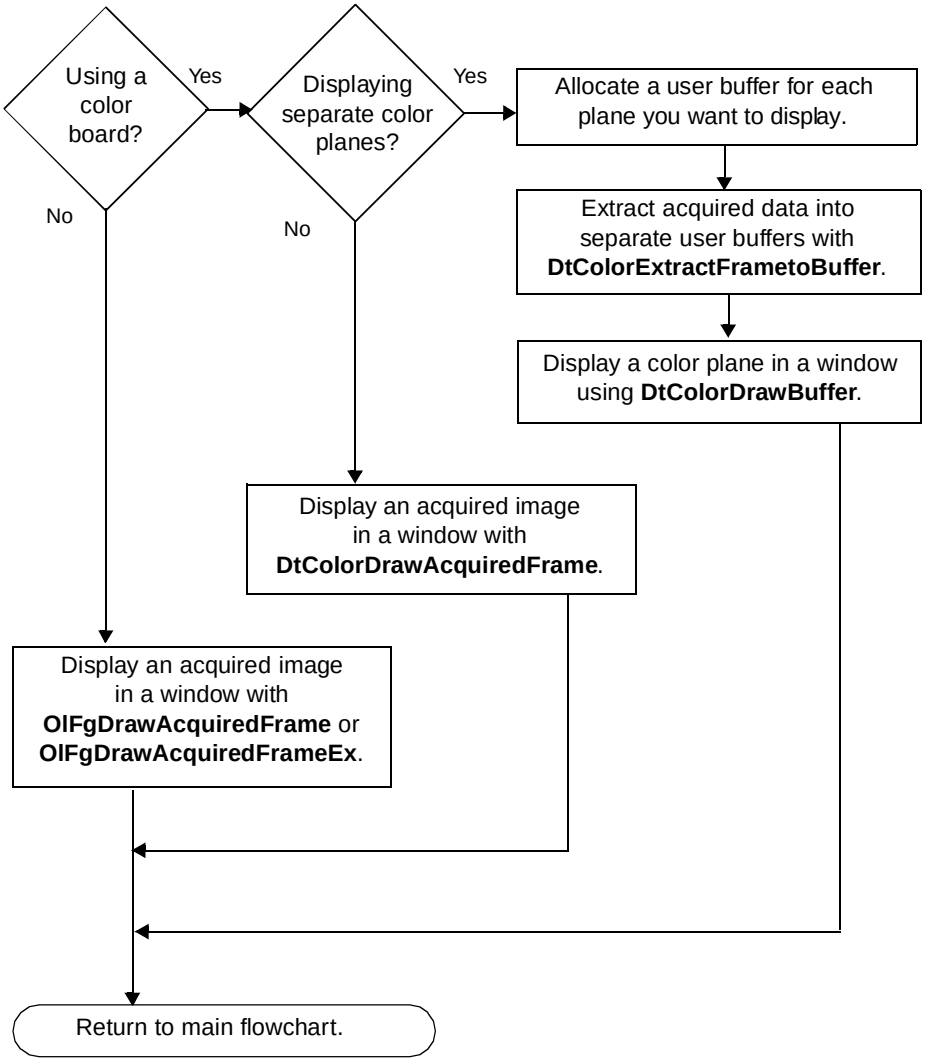
Set the Sync Master Controls



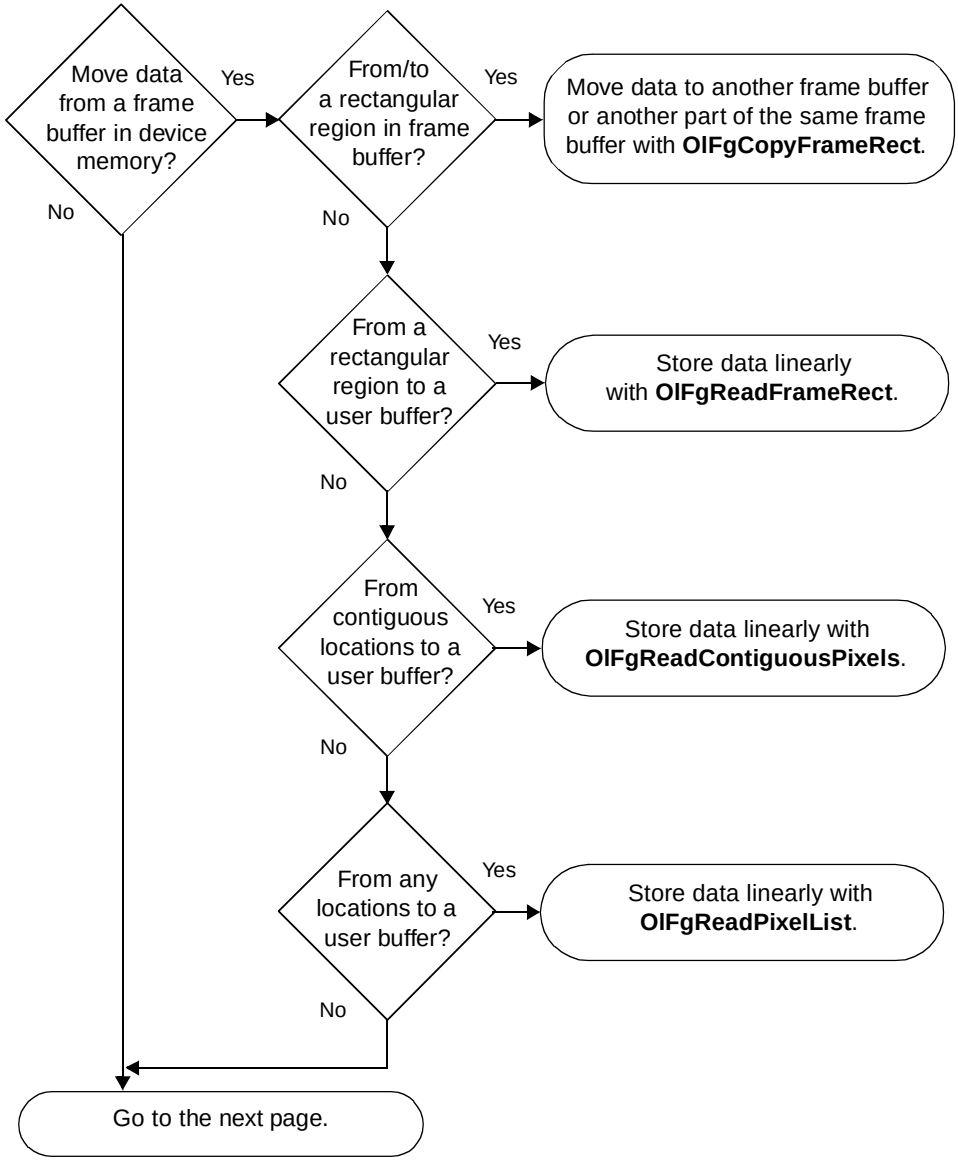
Generate an Output Signal



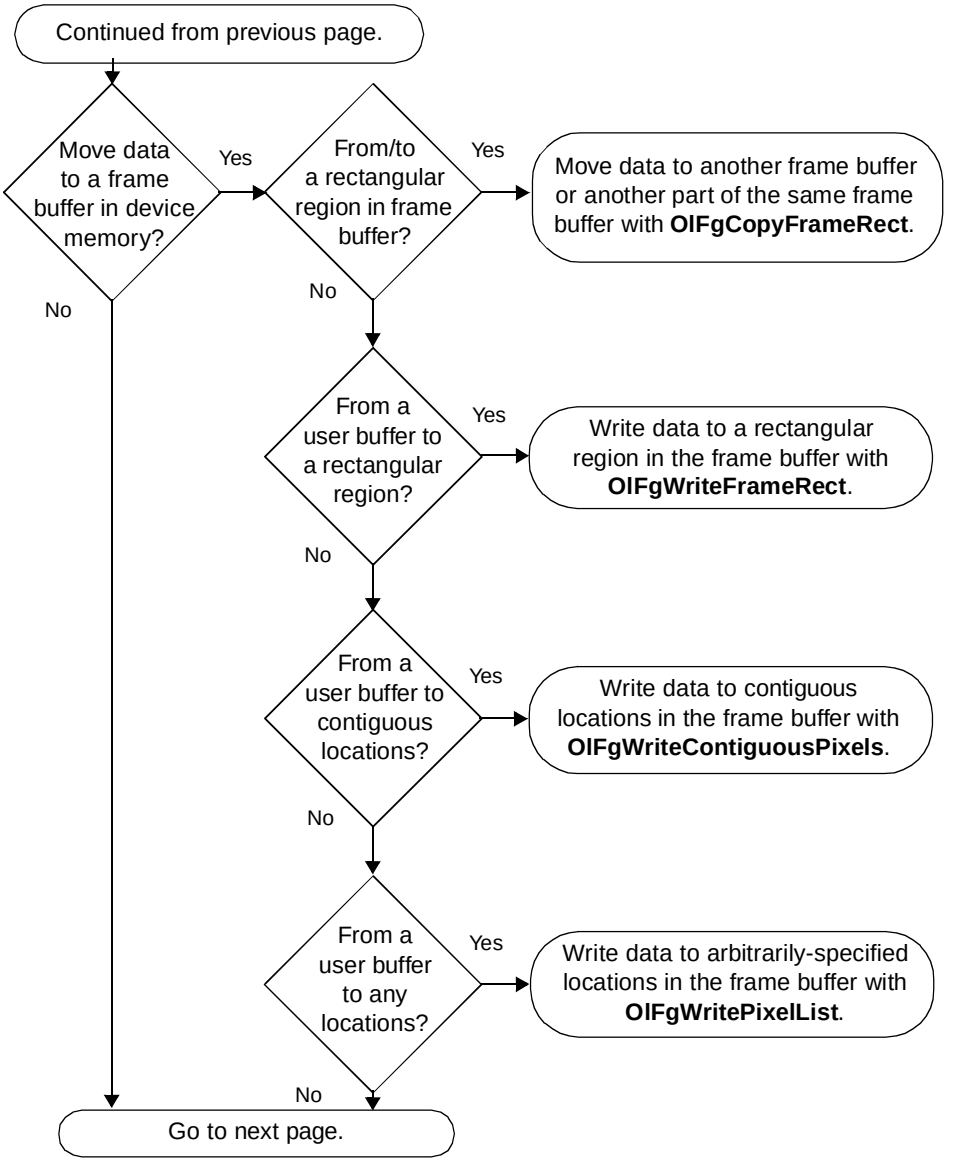
Display an Acquired Image



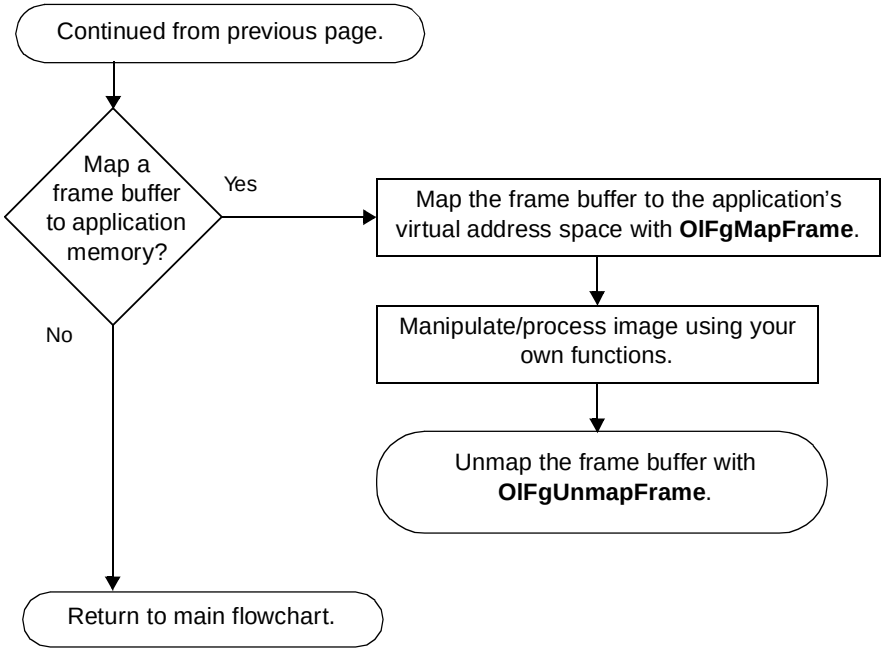
Process an Acquired Image



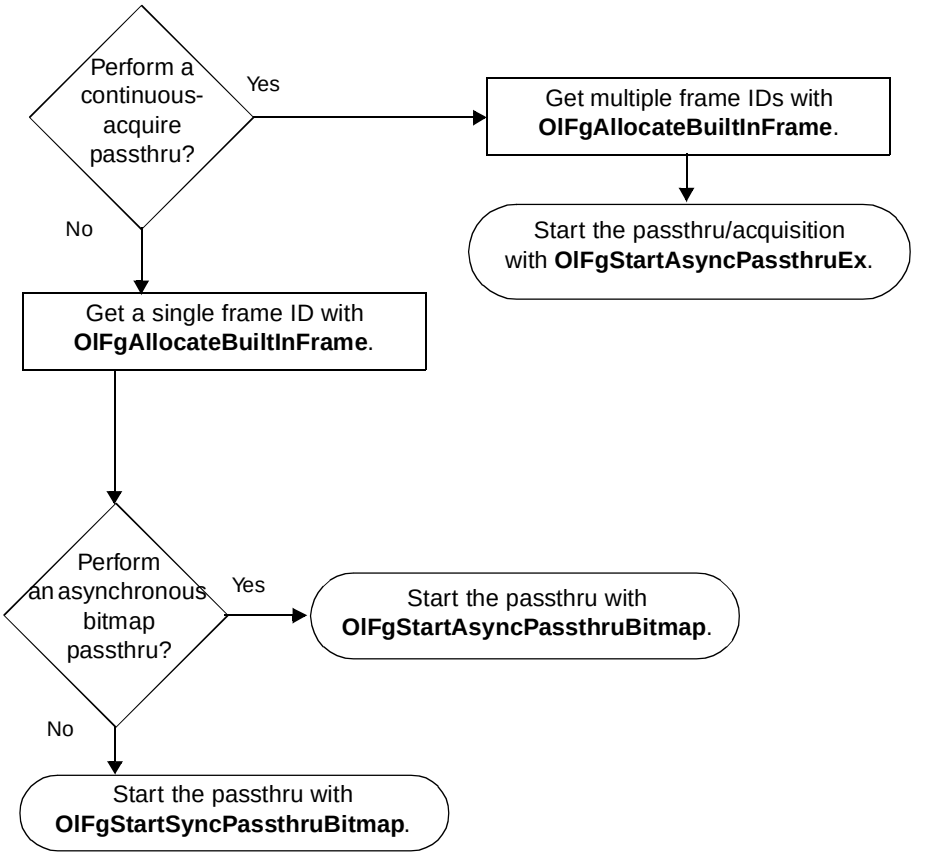
Process an Acquired Image (cont.)



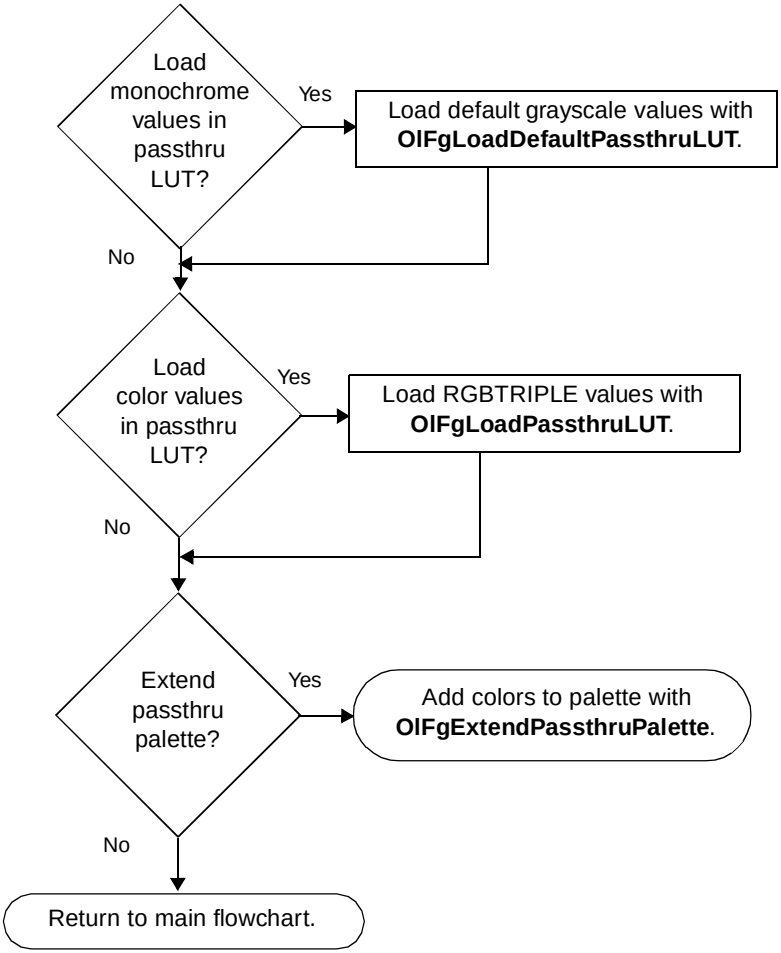
Process an Acquired Image (cont.)



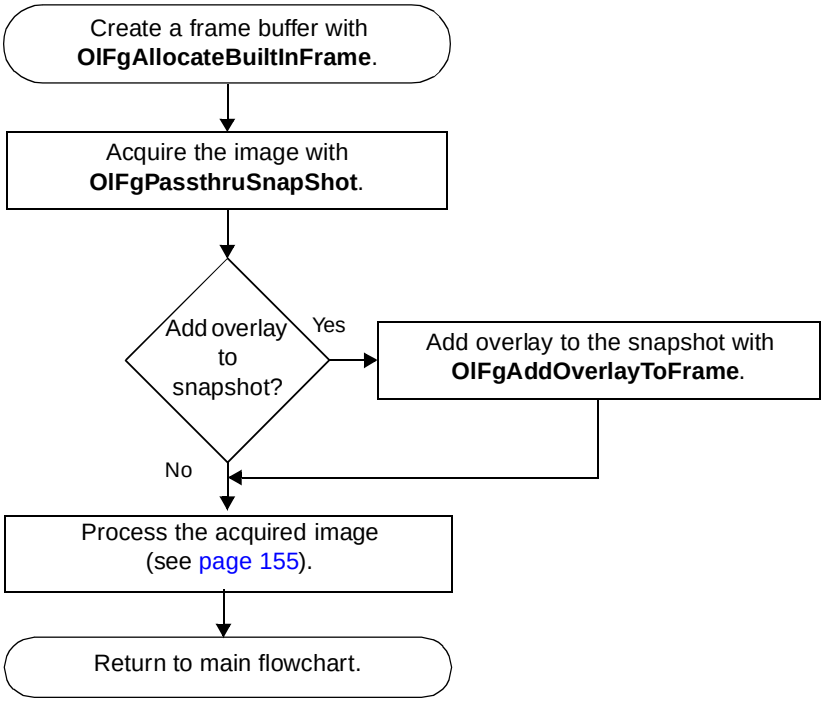
Start a Passthru Operation



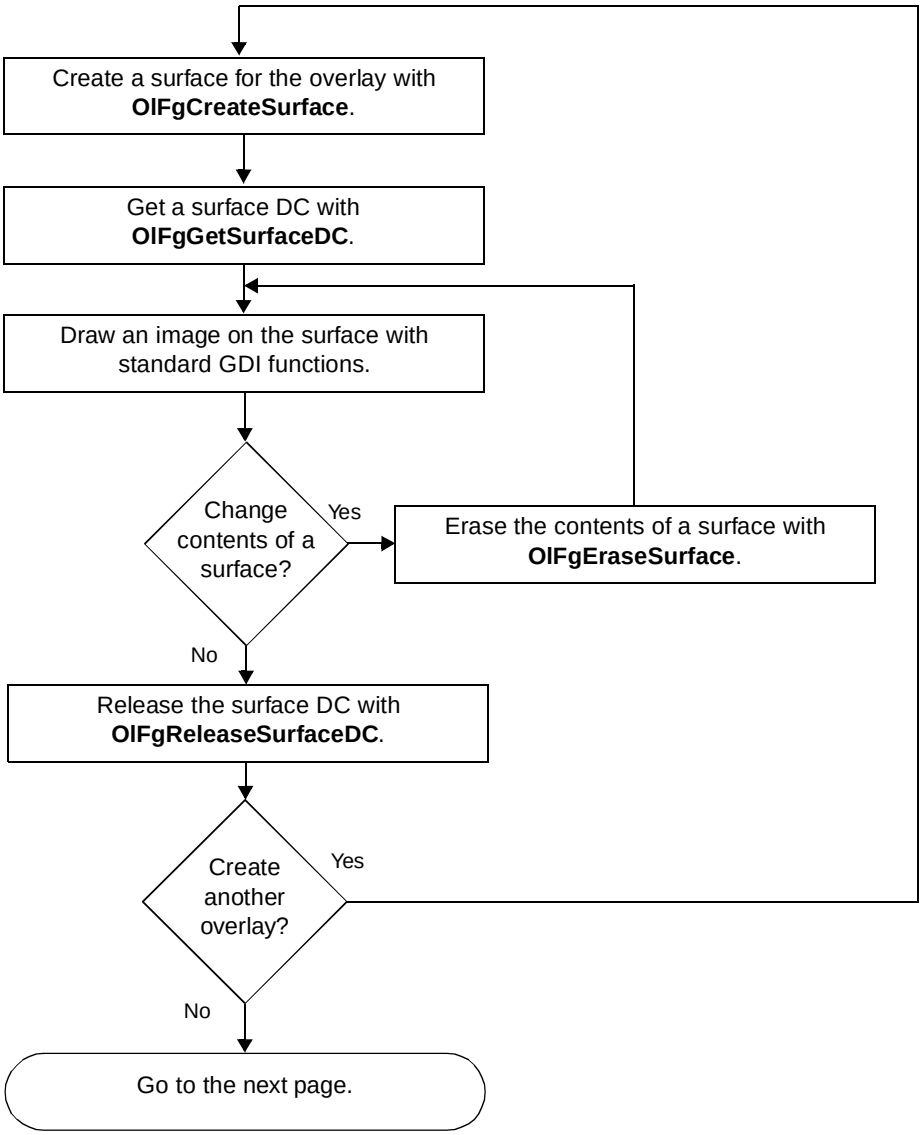
Change the Passthru LUT



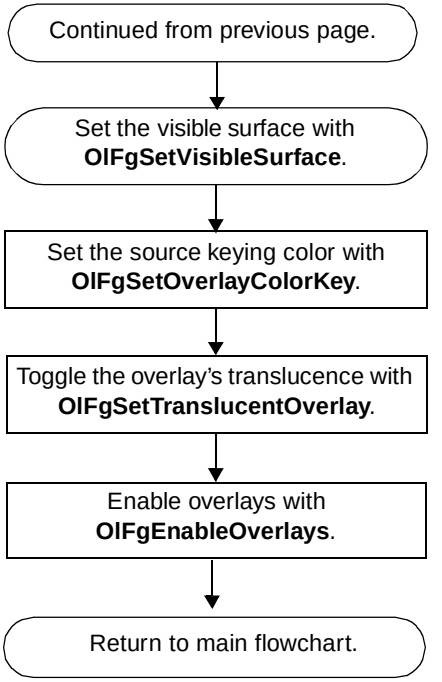
Take a Snapshot



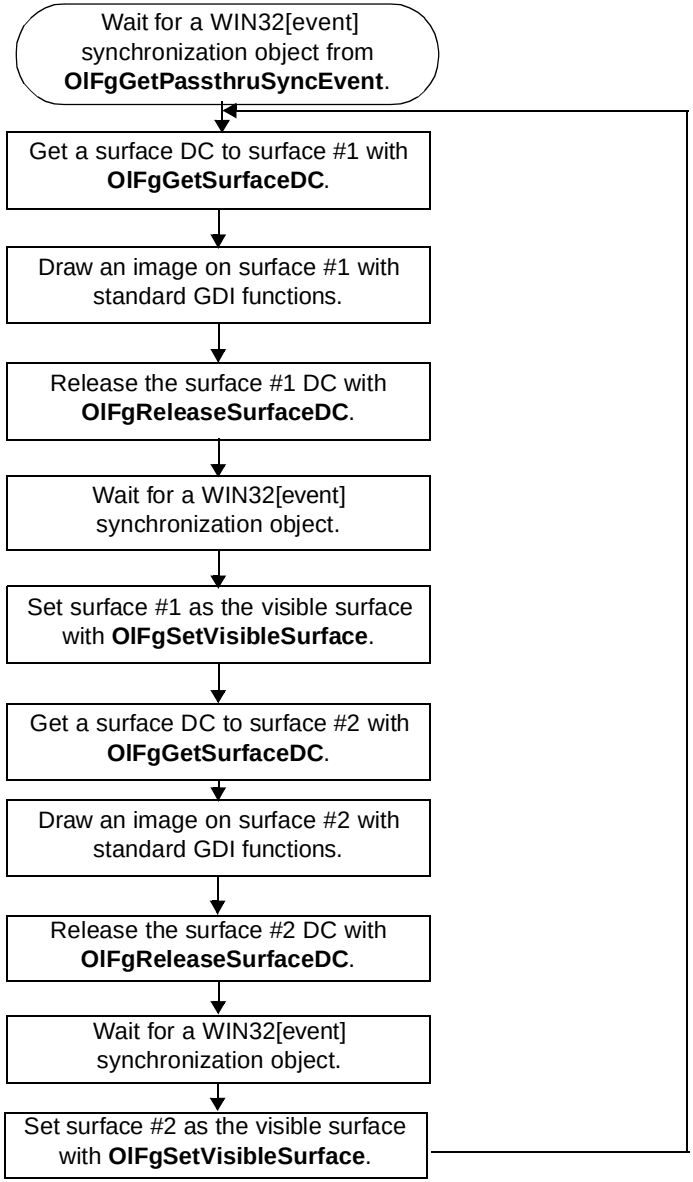
Set up and Enable an Overlay



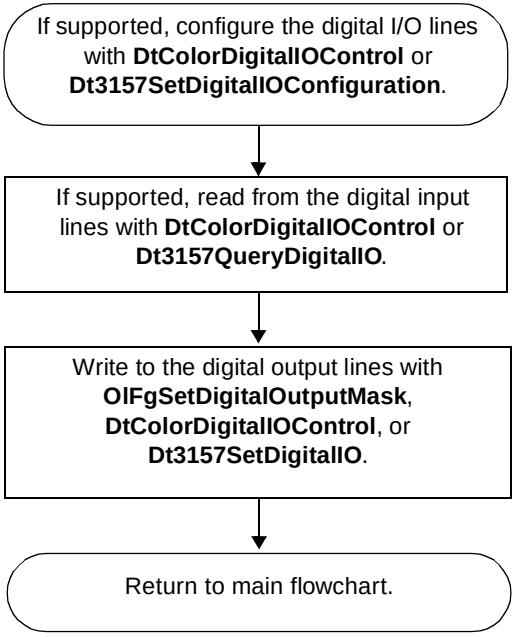
Set up and Enable an Overlay (cont.)



Execute an Overlay Animation Sequence



Program the Digital I/O Lines



Product Support

General Checklist

Should you experience problems using the Frame Grabber SDK, follow these steps:

1. Read all the appropriate sections of this manual. Make sure that you have added any "Read This First" information to your manual and that you have used this information.
2. Check your distribution CD-ROM for a README file. If present, read this file for the latest installation and configuration information.
3. Check that you have installed and configured your software properly. For information, refer to the instructions on page 5.
4. Check that you have installed your frame grabber board properly. For information, refer to your board-specific documentation.
5. Check that you have installed the device driver for your board properly. For information, refer to your board-specific documentation.

Note: If you are still having problems, follow the instructions provided in the next section.

Service and Support

If you have difficulty using the Frame Grabber SDK, Data Translation's Technical Support Department is available to provide prompt technical assistance. Support upgrades, technical information, and software are also available.

All customers can always obtain the support needed. The first 90 days are complimentary, as part of the product's original warranty, to help you get your system running. Customers who call outside of this time frame can either purchase a support contract or pay a nominal fee (charged on a per-incident basis).

For "priority support," purchase a support contract. Support contracts guarantee prompt response and are very affordable; contact your local sales office for details.

Refer to the Data Translation Support Policy located at the end of this manual for a list of services included and excluded in our standard support offering.

Telephone Technical Support

Telephone support is normally reserved for original warranty and support-contract customers. Support requests from non-contract or out-of-warranty customers are processed after requests from original warranty and support-contract customers.

For the most efficient service, please complete the form on page 169 and be at your computer when you call for technical support. This information helps to identify specific system and configuration-related problems and to replicate the problem in house, if necessary.

You can reach the Technical Support Department by calling (508) 481-3700 x1401.

If you are located outside the USA, call your local distributor. The name and telephone number of you nearest distributor are provided in your Data Translation catalog.

If you are leaving a message to request a support call, please include the following information:

- Your name (please include proper spelling),
- Your company or organization (please include proper spelling),
- A phone number,
- An email address where you can be reached,
- The hardware/software product you need help on,
- A summary of the issue or question you have,
- Your contract number, if applicable, and
- Your product serial number or purchase date.

Omitting any of the above information may delay our ability to resolve your issue.

Information Required for Technical Support

Name:_____Phone_____
Contract Number: _____
Address: _____
_____

Data Translation hardware product(s): _____
serial number: _____
configuration: _____
Data Translation device driver - SPO number: _____ _____
_____ version: _____
Data Translation software - SPO number:_____ _____
serial number: _____ version:_____
PC make/model: _____
operating system: _____ version:_____
Windows version: _____
processor: _____ speed:_____
RAM:_____ hard disk space:_____
network/number of users: _____ disk cache:_____
graphics adapter: _____ data bus:_____

I have the following boards and applications installed in my system:_____
_____
_____
I am encountering the following problem(s): _____
_____
_____
_____
and have received the following error messages/codes: _____
_____
_____
I have run the board diagnostics with the following results: _____
_____
You can reproduce the problem by performing these steps:
1. _____
_____
2. _____
_____
3. _____
_____

E-Mail and Fax Support

You can also get technical support by e-mailing or faxing the Technical Support Department:

- **E-mail**: You can reach Technical Support at the following address: tsupport@datx.com

  Ensure that you provide the following minimum information:

  – Your name,

  – Your company or organization,

  – A phone number,

  – An email address where you can be reached,

  – The hardware/software product you need help on,

  – A summary of the issue you are experiencing,

  – Your contract number, if applicable, and

  – Your product serial number or purchase date.

  Omitting any of the above information may delay our ability to resolve your issue.

- **Fax**: Please photocopy and complete the form on page 169, then fax Technical Support at the following number: (508) 481-8620.

Support requests from non-contract and out-of-warranty customers are processed with the same priority as telephone support requests.

World-Wide Web

For the latest tips, software fixes, and other product information, you can always access our World-Wide Web site free of charge at the following address: http://www.datatranslation.com

Example Programs

About the Example Programs

The Frame Grabber SDK is shipped with two example programs (ACQ2HST and PASSTHRU). You can use these programs to help you understand the functions in the Frame Grabber SDK, or you can modify these programs to create your own custom application programs. The source code for these example programs is located in the FGSDK32\EXAMPLES directory.

The following sections describe the example programs. For more information, refer to EXAMPLES.TXT.

Acquire to Host Example Program

The Acquire to Host example program (ACQ2HST.C) illustrates the required programming sequence when using Frame Grabber SDK standard functions. It shows you how to perform queries and then enable/disable user options and/or change input controls based on the supported input capabilities of the frame grabber board (indicated by the query responses). The program then acquires data to host memory and displays the data as a Device-Independent Bitmap (DIB) in the client area of the application. Note that the program uses only Frame Grabber SDK standard functions; no extension functions are used.

The files associated with the ACQ2HST.C example program are located in the \FGSDK32\EXAMPLES\ACQ2HST directory.

Assumptions

The ACQ2HST.C example program does not provide support for external triggers, event counting, camera control operations, linear memory operations, or onboard memory operations and, therefore, does not query for them.

To extend the program to support other capabilities, add queries for these capabilities to determine whether the frame grabber board supports them. For example, to determine whether the board supports single-frame acquisition to device memory, use **OlFgQueryInputCaps** with the key OLC_FG_IC_SINGLE_FRAME_OPS.

Compiling and Linking Your Application

Ensure that you update the directory paths for the library and include files before you compile this example program in Microsoft C.

Notes

It is recommended that you make a query before using any capability. Refer to the Frame Grabber SDK online help for more information about the query functions and query keys to use.

Querying for specific board capabilities and extensive error checking make this example program somewhat large. Note, however, that once the initialization is performed, the program flows like any other Windows application program.

If a frame grabber board is set up with a sync source that is different from the video input source, the Sync Sentinel may not be enabled (even though it is shown as ON). To correct this, either make sure that the video input source and the sync source are the same or manually set the Sync Sentinel to ON. Other changes to the sync source or video input source operate normally.

Initialization Procedure

The following procedure illustrates the steps required to comply with the DT-Open Layers specification:

1. Get the number of DT-Open Layers frame grabber boards.
   The following code fragment shows how to determine the number of boards in the system.

```
/* Find the number of devices in the system */
/* How many aliases do we have */
if ((Status = OlImgGetDeviceCount(&iCount))!=
        OLC_STS_NORMAL)
   {
      //Display error message
      return;
   }
```

2. Get the device information.
If iCount *from the previous step is greater than 0, at least one DT-Open Layers-compliant frame grabber board exists in the system. The following code fragment assumes this to be true and attempts to retrieve board information. This example retrieves information for one board at a time.*

```
/* Allocate space for info */
hDevInfoList = GlobalAlloc
         (GHND, sizeof(OLT_IMGDEVINFO));
if ( !hDevInfoList )
   {
      //Display error message
      return;
   }

lpDevInfoList = (LPOLT_IMGDEVINFO)
         GlobalLock(hDevInfoList);
if ( !lpDevInfoList )
   {
      //Display error message
      return;
   }
/* Fill in the device info */
lpDevInfoList->StructSize = sizeof(OLT_IMGDEVINFO);
if(Status= OlImgGetDeviceInfo(lpDevInfoList,
   sizeof(   OLT_IMGDEVINFO)))
   {
      //Display error message
      return;
   }
```

3. Open the frame grabber board.
   Assume a board alias was returned from the previous step and pointed to by lpcszAlias. *You must use this alias to open this board. The following code fragment demonstrates this process:*

```
/* Try to open the frame grabber */
if ( OlImgIsOkay( OlImgOpenDevice(lpcszAlias,
     &DevId) ) )
   CurDevInfo.DevId = DevId;
else
   {
      //Display error message
      return;
   }
```

4. Determine the frame grabber board capabilities.
   Assume that the board was opened successfully in the previous step. The following code fragment queries the board to determine which DT-Open Layers sections (capability categories) it supports.

```
/* Query DEVICE capabilities */
Status = OlImgQueryDeviceCaps
         (DevId, OLC_IMG_DC_SECTIONS, &Result,
            sizeof(ULNG));

if ( ! OlImgIsOkay(Status)
   {
      //Display error message
      return;
   }
```

5. Determine the controls for each supported device capability.
   The following code fragment checks to see if the Input Capability section of the Device Capabilities is supported. If it is, then specific Input Capability queries must be performed. The Result *variable was set by the previous* ***OlImgQueryDeviceCaps()*** *call (see step 4). Although this code fragment demonstrates querying only one input capability (the size of the frame to acquire), you can determine other input capabilities similarly. For information, refer to the* ***NewDevice()*** *function in the ACQ2HST.C module.*

```
/* Check if the frame grabber has an INPUT */
/* capabilities section */
if (Result & OLC_FG_SECTION_INPUT)
     {

     /*  now query all the capabilities */
   .
   .
   .
   /* Query frame size stuff */

   Status = OlFgQueryInputCaps
         (DevId, OLC_FG_IC_DOES_FRAME_SELECT,
         &CurDevCaps.DoesFrameSize, sizeof(BOOL));

   if ( ! OlImgIsOkay(Status)  &&
         (Status != OLC_STS_UNSUPKEY)
   {
      //Display error message
      return;
   }
} /* end query Input Capability section */

/* Check for remaining capabilities sections */
/* (MEMORY, CAMCTL, LINEAR) */
```

```
if (Result & OLC_FG_SECTION_MEMORY)
      {
   .
   .
   .
      }  /* end Memory Capabilities section */
```

6. Set values for the supported controls.
The following code fragment checks the result of the query with the OLC_FG_IC_DOES_FRAME_SELECT *key to determine if the frame select controls are supported. If so, the example sets the related frame select controls within the limits specified by the frame select limits queries.*

```
if (CurDevCaps.DoesFrameSize)
   {
Status = OlFgQueryInputCaps
         (DevId, OLC_FG_IC_FRAME_TOP_LIMITS,
         LPFRAMEAREAINFO->TopRange,
         sizeof(OLT_LNG_RANGE));

if (Status)
     {
   //Display error message
   return;
      }
/* Get limits for LEFT, HEIGHT WIDTH as */
/* displayed above */
/* Get input from dialog box and verify it is */
/* within the legal range limits.*/

/* Now set the Frame Top control, Frame Top */
/* value contained in lResult */
Status = OlFgSetInputControlValue
         (CurDevInfo.DevId, CurDevInfo.InputSource,
         OLC_FG_CTL_FRAME_TOP, lResult,
         &ulOldData);
```

```
if ( !OlImgIsOkay(Status) )
{
   //Display error message
   return;
}

/* Now set Left, Height, Width controls as */
/* displayed above */
  }  /* end if DoesFrameSize */
```

7. Set up and perform the acquisition.

```
   /* Query for Height, Width and Pixel Depth */
   /* of the frame */
   (void) OlFgQueryInputControlValue
            (DevInfo.DevId, DevInfo.InputSource,
            OLC_FG_CTL_FRAME_HEIGHT, &ulHeight);
   (void) OlFgQueryInputControlValue
            (DevInfo.DevId, DevInfo.InputSource,
            OLC_FG_CTL_FRAME_WIDTH, &ulWidth);
   (void) OlFgQueryInputCaps
            (DevInfo.DevId, OLC_FG_IC_PIXEL_DEPTH,
            &ulPixelDepth, sizeof(ULNG));
   /* calculate minimum size of buffer in bytes */
   ulMinBufSize = ulHeight * ulWidth *
            ulPixelDepth;

   hAcquireBuf = GlobalAlloc(GHND, ulMinBufSize);
   if ( !hAcquireBuf )
      {
         /* unable to allocate memory */
         char msg[OLC_MAX_STATUS_MESSAGE_SIZE+60];
         wsprintf(msg, "Unable to allocate enough
               memory for acquire.");
         MessageBox(NULL, msg, "Acquire Error",
               MB_ICONSTOP | MB_OK);
         return;
      }
```

```
hpAcquireBuf = (HPUCHR) GlobalLock(hAcquireBuf);
if ( !hpAcquireBuf)
   {
      /* unable to lock memory */
      char msg[OLC_MAX_STATUS_MESSAGE_SIZE+60];
      wsprintf(msg, "Unable to lock memory for
             acquire.");
      MessageBox(NULL, msg, "Frame Size Error",
             MB_ICONSTOP | MB_OK);
      return;
   }

Status = OlFgAllocateBuiltInFrame
         (DevInfo.DevId,
         OLC_FG_DEV_MEM_VOLATILE,
         OLC_FG_NEXT_FRAME, &FrameId);
if ( !OlImgIsOkay(Status))
{
   PrintStatus(NULL, Status, "Unable to
            Allocate Frame.", "Acquire Error");
}

/* Have the memory, attempt the acquire */
Status = OlFgAcquireFrameToHost
         (DevInfo.DevId, FrameId, hpAcquireBuf,
         ulMinBufSize);
if ( !OlImgIsOkay(Status))
{
   PrintStatus(NULL, Status, "Unable to
         acquire.", "Acquire Error");
   (void) OlFgDestroyFrame
         (DevInfo.DevId, FrameId);
   GlobalUnlock(hAcquireBuf);
   GlobalFree(hAcquireBuf);
   return;

}
```

```
/* create a DIB and display it in the client area */

   if(RGBDataFlag)
   {
      DrawRGBFrame(hWnd,hpAcquireBuf,
            ulWidth,ulHeight);

   }
   else
   {
      CreateDIBDisplay(hpAcquireBuf, ulHeight,
             ulWidth, ulPixelDepth);
      GlobalUnlock(hAcquireBuf);
      GlobalFree(hAcquireBuf);
   }
   /* we no longer need the frame, free it */
   Status = OlFgDestroyFrame( DevInfo.DevId,
            FrameId);
   if (!OlImgIsOkay(Status))
   {
      PrintStatus(NULL, Status, "Unable to free
         frame after acquire.", "Acquire Error");
   }
```

8. Close the frame grabber board.
The following code fragment demonstrates how to close the device. The ACQ2HST.C example allows only one opened device at a time, so a call to close device must be made before you can open another device. This code fragment uses the device ID returned from the call to ***OlImgOpenDevice()*** *(see step 3).*

```
/* Close current device */
if (CurDevInfo.DevId != NULL)
   (void) OlImgCloseDevice(CurDevInfo.DevId);
```

Passthru Example Program

The passthru example program (PASSTHRU.MFC) illustrates the use of the passthru functions, the supporting Windows functions, and associated Windows resources such as palettes, bitmaps, and scrollbars. It allows you to open a device, select synchronous or asynchronous passthru, and run the passthru operation. If the desired passthru mode is supported by DDI, this mode is used; otherwise, bitmap passthru is used.

The files associated with the PASSTHRU.MFC example program are located in the \FGSDK32\EXAMPLES\PASSTHRUMFC directory.

Ensure that you update the directory paths for the library and include files before you compile this example program in Microsoft C.

Notes: Passthru is not supported by all frame grabber boards. To determine if your board supports passthru operations, use **OlFgQueryPassthruCaps** with the key OLC_FG_PC_DOES_PASSTHRU.

See also PASSTHRUEX.MFC in the \FGSDK32\EXAMPLES\PASSTHRUEXMFC directory for an example of performing a continuous-acquire passthru operation.

If you want to see a C example instead of an MFC example, a PASSTHRU.C example is located in the \FGSDK32\EXAMPLES\PASSTHRU directory.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

Data Translation Support Policy

Data Tr**an**slation, Inc. (Data Translation) offers support upon the following terms and conditions at prices published by Data Translation from time to time. Current price information is available from Data Translation, or its authorized distributor. If Licensee elects to obtain support services from Data Translation, Licensee must complete the Support Order Form attached hereto and submit to Data Translation the completed form, along with Licensee's purchase order for support. Support will only be provided for all (not less than all) Licensed Processors (as defined in the Data Translation Software License Agreement).

1. DEFINITIONS. Capitalized terms used herein and not otherwise defined shall have the meanings assigned thereto in the applicable Data Translation Software License Agreement (the Agreement). The following terms have the meanings set forth below:

Enhanced Release means a new release of any Product that contains new features and may contain corrections to previously identified errors. Enhanced Releases are designated in the tenths digit of the release designation (e.g., 1.2 is an Enhanced Release from 1.1.x).

Maintenance Release means a new release of any Product that contains corrections to previously identified errors. Maintenance Releases are designated in the hundredths digit of the release designation (e.g., 1.2.2 is a Maintenance Release from 1.2.1).

Major Release means a new version of any Product that involves major feature changes. Major Releases are designated in the ones digit of the release designation (e.g., 2.0, 3.0, etc., are Major Releases).

2. DATA TRANSLATION'S OBLIGATIONS. Subject to the terms of the Agreement, and this Support Policy, Data Translation will provide the following support services (Support Services) for the Products comprising the Software, as they may be used with the Licensed Processors:

(a) problem reporting, tracing and monitoring by internet electronic mail; (b) telephone support for problem determination, verification and resolution (or instruction as to work-around, as applicable) on a call-back basis during Data Translation's normal weekday business hours of 8:30 a.m. to 5 p.m. Eastern Time, excluding holidays; (c) one (1) copy of each Maintenance Release for the Products comprising the Software; (d) commercially reasonable efforts to diagnose and resolve defects and errors in the Software and Documentation; and (e) furnishing of the maintenance and technical support described above, for the current release and the immediately previous Enhanced Release of the Software. Support Services will be delivered in English. Enhanced Releases and Major Releases can be purchased by Licensee at a discount of twenty five percent (25%) off the then-current list prices for such releases.

3. EXCLUSIONS. Support Services do not include: (a) the provision of or support for Products other than those identified in the Agreement as to which the applicable license and support fees shall have been paid, including without limitation, compilers, debuggers, linkers or other third party software or hardware tools or components used in conjunction with any Product; (b) services required as a result of neglect, misuse, accident, relocation or improper operation of any Product or component thereof, or the failure to maintain proper operating and environmental conditions; (c) support for processors other than Licensed Processors or for Products modified by or on behalf of Licensee; (d) repair or restoration of any Software arising from or caused by any casualty, act of God, riot, war, failure or interruption of any electrical power, air conditioning, telephone or communication line or any other like cause.

Data Translation Support Policy

It is Licensee's responsibility to have adequate knowledge and proficiency with the use of the compilers and various software languages and operating systems used with the Products, and this Support Policy does not cover training of, or detailed direction on the correct use of these compilers, operating systems, or components thereof. On-site assistance shall not be provided hereunder, but may be available on a per call basis at Data Translation's then current rates (Specialized Application Support Charges) for labor, travel time, transportation, subsistence and materials during normal business hours, excluding holidays observed by Data Translation. The troubleshooting of faulty Licensee programming logic may also be subject to Specialized Application Support Charges and is not covered under this Support Policy. Direct authoring or development of customized application code is not provided hereunder but may be available on a per call basis upon payment of Specialized Application Support Charges.

4. LICENSEE'S OBLIGATIONS. Licensee agrees: (a) that the Designated Contact persons identified on the Support Order Form (or such other replacement individuals as Licensee may designate in writing to Data Translation) shall be the sole contacts for the coordination and receipt of the Support Services set forth in Section 2 of this Support Policy; (b) to maintain for the term of the support, an internet address for electronic mail communications with Data Translation; (c) to provide reasonable supporting data (including written descriptions of problems, as requested by Data Translation) and to aid in the identification of reported problems; (d) to install and treat all software releases delivered under this Support Policy as Software in accordance with the terms of the Agreement; and (e) to maintain the Agreement in force and effect.

5. TERM AND TERMINATION.

5.1 Term. For each Product comprising the Software, Support Services will begin on the later of the date the Software warranty granted in the Agreement expires or the date of Licensee's election to obtain Support Services and will apply to such Product for an initial term of one (1) year, unless an alternative commencement date is identified in the Support Order Form. The initial term will automatically be extended for additional terms of one (1) year unless Support Services are terminated at the expiration of the initial term or any additional term, by either party upon thirty (30) days prior written notice to the other party.

5.2 Default. If Licensee is in default of its obligations under the Agreement (except for Licensee's obligation to maintain valid licenses for the Software, in which case termination is immediate) and such default continues for thirty (30) days following receipt of written notice from Data Translation, Data Translation may, in addition to any other remedies it may have, terminate the Support Services.

6. CHARGES, TAXES AND PAYMENTS.

6.1 Payment. The Support Fee in respect of the initial term, and, as adjusted pursuant to Section 5.2 in respect of additional terms, is payable in full prior to the commencement of the initial term or any additional term, as applicable.

6.2 Changes From Term to Term. The Support Fee and the terms and conditions of this Support Policy may be subject to change effective at the end of the initial term or any additional term by giving Licensee at least sixty (60) days prior written notice.

Data Translation Support Policy

6.3 Taxes. The charges specified in this Support Policy are exclusive of taxes. Licensee will pay, or reimburse Data Translation, for all taxes imposed on Licensee or Data Translation arising out of this Support Policy except for any income tax imposed on Data Translation by a governmental entity. Such charges shall be grossed-up for any withholding tax imposed on Data Translation by a foreign governmental entity.

6.4 Additional Charges. Licensee agrees that Data Translation or its authorized distributor will have the right to charge in accordance with Data Translation's then-current policies for any services resulting from (a) Licensee's modification of the Software, (b) Licensee's failure to utilize the then-current release, or the immediately previous Enhanced Release, of the Software, (c) Licensee's failure to maintain Data Translation Support Services throughout the term of the Agreement, (d) problems, errors or inquiries relating to computer hardware or software other than the Software, or (e) problems, errors or inquiries resulting from the misuse or damage or of the Software or from the combination of the Software with other programming or equipment to the extent such combination has not been authorized by Data Translation. Pursuant to Section 2.4 of the Agreement, the Support Fee will also be adjusted in accordance with Data Translation's then current fee schedule as additional Licensed Processors are added. Support Fees do not include travel and living expenses or expenses for installation, training, file conversion costs, optional products and services, directories, shipping charges or the cost of any recommended hardware, third party software, or third party software maintenance fees or operating system upgrade.

7. WARRANTY LIMITATION. EXCEPT AS EXPRESSLY STATED IN THIS SUPPORT POLICY, THERE ARE NO EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO THE SUPPORT SERVICES PROVIDED HEREUNDER (INCLUDING THE FIXING OF ERRORS THAT MAY BE CONTAINED IN THE APPLICABLE DATA TRANSLATION SOFTWARE), INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE WARRANTIES AND REMEDIES SET FORTH IN THIS SUPPORT POLICY ARE EXCLUSIVE, AND ARE IN LIEU OF ALL OTHER WARRANTIES WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED.

8. GENERAL PROVISIONS. Upon the election by Licensee to obtain Support Services, the terms of this Support Policy shall be governed by and are made a part of the Agreement.