1. 

FAST-UXGA PCI-X OPTION

TECHNICAL PRODUCT DESCRIPTION

30002-00192

COPYRIGHT NOTICE

Copyright © 2001-2007 by Alacron Inc.

Document Name:	Fast-UXGA PCI-X Technical Product Description	
Document Number:	30002-00192	
Revision History:	1.0	October 25, 2006
	2.0	February 26, 2007
	2.1	April 25, 2007

Trademarks:

Alacron Inc.
71 Spit Brook Road, Suite 200
Nashua, NH 03060
USA

Telephone: 603-891-2750
Fax: 603-891-2745

Web Site: http://www.alacron.com

Email: sales@alacron.com or support@alacron.com

TABLE OF CONTENTS

OTHER ALACRON MANUALS

Alacron manuals cover all aspects of FastSeries hardware and software installation and operation. Call Alacron at 603-891-2750 and ask for the appropriate manuals from the list below if they did not come in your FastSeries shipment.

30002-00148 ALFAST Runtime Software Programmer's Guide & Reference

30002-00150 FastSeries Library User's Manual

30002-00169 ALRT Runtime Software Programmer's Guide & Reference

30002-00184 FastSeries Getting Started Manual

30002-00185 FastVision Hardware Installation Manual

30002-00186 FastVision Software Installation Manual

30002-00395 FastMotion User's Manual

INTRODUCTION

The Alacron Fast-UXGA is a highly flexible, programmable and expandable video capture and processing device. The Fast-UXGA Option provides for the capture, processing, and delivery of UXGA (1600x1200) true color, and high bandwidth monochrome video to a host PC.

Figure 1 - Fast-UXGA PCI-X Board

The Fast-X USGA option attaches to a Fast-X (PCI-X) or Fast-Xe (PCI Express) card. It provides the interface to the high bandwidth analog video source. It attaches to the card on the long vertical connector seen in the image above. Note the back panel metal bracket is replaces, and the Camera link support is removed from the board with the UXGA option is used.

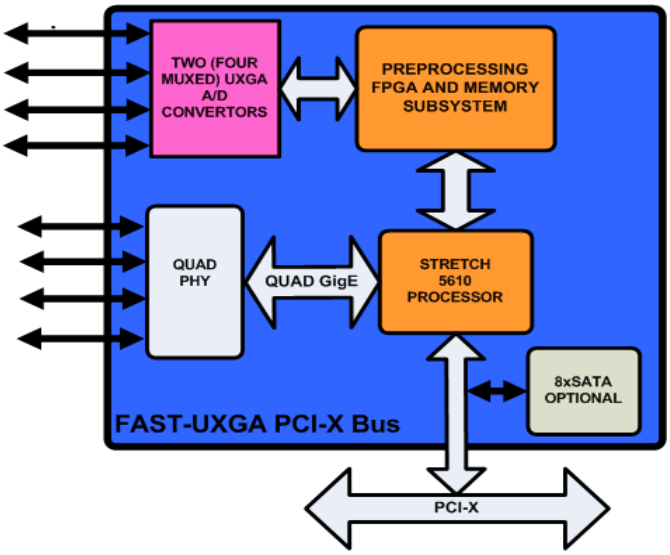


Figure 2, Fast-UXGA PCI-X Diagram

.

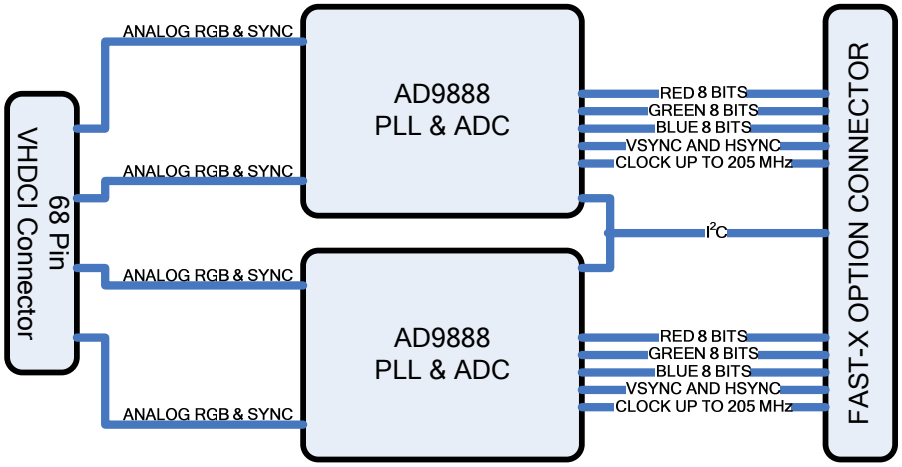


Figure 3: A block Diagram of the Fast-X UXGA option is shown below

FAST-X UXGA OPTION FEATURE SUMMARY

Board	**3"X4" board with a proprietary interface**
Video I/O Connections	**Four RGB & SYNC Inputs, five wires or sync on green. Two channels active at any time**
Video I/O Extender card	**Mezzanine expansion card with 152-pin VHDCI connector.** **Plugging-in a Video I/O Extender daughter card replaces standard video connections with a variety of analog and digital video formats**
Performance	**640x480 to 1600x1200 with a 10 MHz to 205 MHz pixel clock and Phase locked loop clock restoration**
Software Development	**Fast Motion Library for Windows** **ALRT for Stretch**
System Requirements	**Fast-X**
Temperature	**0°C (32°F) to 55°C (131°F)** **Relative Humidity: up to 95% (non-condensing)**

HARDWARE OVERVIEW

This section provides an overview of Fast-X UXGA hardware.

VIDEO INPUTS

The video inputs are designed to interface to several different video standards.

- VGA Video, or video from the output of an analog output computer display adapter. This is three wires of 1 volt video with two separate syncs as TTL levels.
- Sync on Green color video. This is three wires of analog video, with the green channel carrying a composite sync.
- Single channel of monochrome analog composite video

The video inputs are terminated to ground with 75 ohms and capacitive coupled into the ADC inputs. DC restore is provided by clamping to black.

The sync inputs are pulled up and terminated with 2K.

ADC ADI9888B

The Analog Devices AD9888B is type of ADC used on the Fast-X UXGA Option. A block diagram of the AD9888B is shown below.

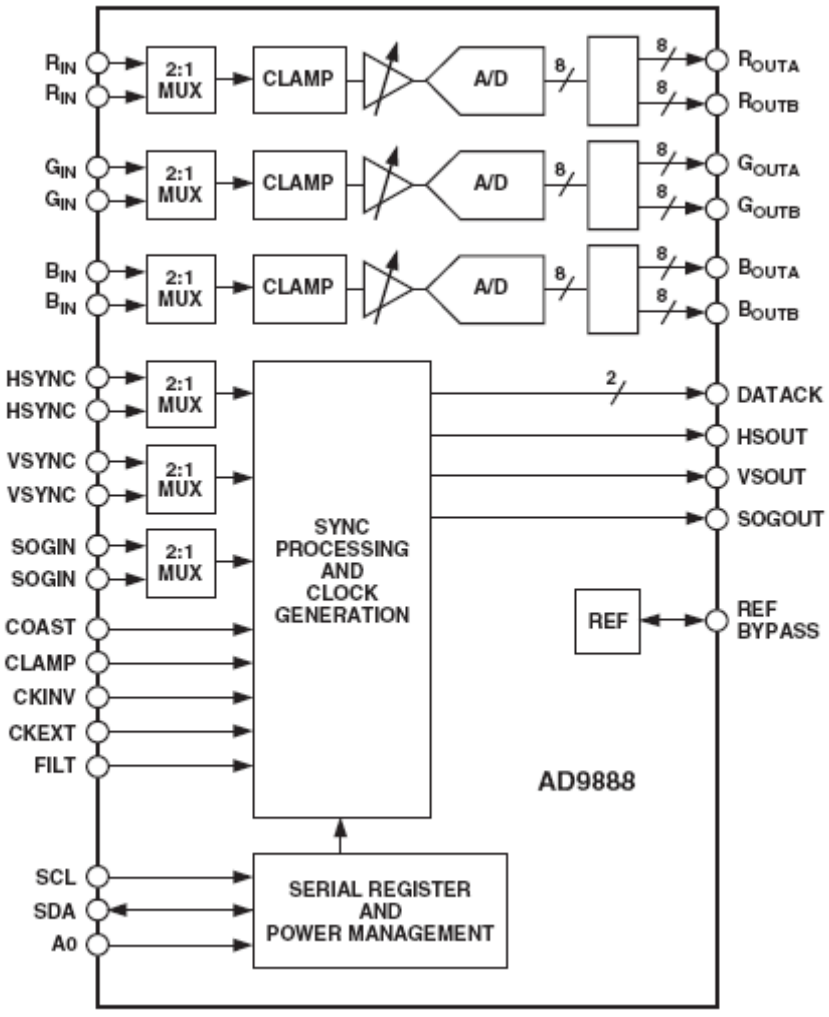


Figure 4, AD9888 internal architecture

ADC9888B Features:

- 205 MSPS Maximum Conversion Rate
- 500 MHz Programmable Analog Bandwidth
- 0.5 V to 1.0 V Analog Input Range
- Less than 450 psec p-p PLL Clock Jitter @ 205 MSPS
- 3.3 V Power Supply
- Full Sync Processing
- Sync Detect for “Hot Plugging”
- 2:1 Analog Input Mux
- 4:2:2 Output Format Mode
- Mid-scale Clamping
- Power-Down Mode
- Low Power: <1 W Typical @ 205 MSPS

The AD9888B provides for the conversion of the analog vide to digital video. The AD9888B has an input multiplexer with two inputs. The digital output of the AD9888B is provided on one or two 24 bit digital channels. At higher data rates the two outputs can be used as a double data rate interface. The AD9888B is configured via the I^2C interface from the Fast-X/Fast-Xe board.

I^2C INTERFACE

The I^2C interface exposes 32 registers to control the operation of the AD9888Bs. Each AD9888Bs has a separate slave address, Channel 1, Mux 0 and 1 are addressed at 0x98, while Channel 2, Mux 0 and 1 are addressed at address 0x9A. Please see the ADI9888B data sheet on the operation of the I^2C interfaces. The ALRT Runtime and the Fast Motion Library provide APIs that do not require knowledge of the I^2C interface.

CONNECTOR PINOUT

Table 6 – Pinout of the UXGAoption input connector

Pin	Function	Function	Pin
1	HSYNC Ch1 Mux1	GROUND	35
2	RED Ch1 Mux1	GROUND	36
3	GREEN Ch1 Mux1	GROUND	37
4	BLUE Ch1 Mux1	GROUND	38
5	VSYNC Ch1 Mux1	GROUND	39
6	NC = No Connect	GROUND	40
7	NC	GROUND	41
8	HSYNC Ch1 Mux2	GROUND	42
9	RED Ch1 Mux2	GROUND	43
10	GREEN Ch1 Mux2	GROUND	44
11	BLUE Ch1 Mux2	GROUND	45
12	VSYNC Ch1 Mux2	GROUND	46
13	NC	GROUND	47
14	NC	GROUND	48
15	HSYNC Ch2 Mux1	GROUND	49

16	RED Ch2 Mux1	GROUND	50
17	GREEN Ch2 Mux1	GROUND	51
18	BLUE Ch2 Mux1	GROUND	52
19	VSYNC Ch2 Mux1	GROUND	53
20	NC	GROUND	54
21	NC	GROUND	55
22	HSYNC Ch2 Mux2	GROUND	56
23	RED Ch2 Mux2	GROUND	57
24	GREEN Ch2 Mux2	GROUND	58
25	BLUE Ch2 Mux2	GROUND	59
26	VSYNC Ch2 Mux2	GROUND	60
27	NC	GROUND	61
28	NC	GROUND	62
29	NC	GROUND	63
30	NC	GROUND	64
31	NC	GROUND	65
32	NC	GROUND	66
33	NC	GROUND	67
34	NC	GROUND	68

SOFTWARE MODEL

The software model for the Fast-X UXGA option is very simple. From the Stretch processor which provides the I^2C interface, the AD9888Bs are controlled. As these are the only components on the UXGA board (excluding their glue circuits) the data sheet for the AD9888 provides the software model via the definition and use of the internal registers of the AD9888.

The data passed back to the front end FPGA of the Fast-X board handles the collection of data from the two AD9888Bs and the transfer of that data to the Stretch processor. ALRT provides Stretch APIs to access the hardware, while Fast Motion Library provides the more useful interfaces as message based commands. Both software packages provide for an ASCII control file, which operates all the settings of the AD9888B.

Please see the Fast Motion Library Documentation or the Documentation for ALRT for Stretch for the details of the software.

THEORY OF OPERATION

This section describes the functionality of the firmware in the FPGA and the Stretch processor to explain how the UXGA option is used.

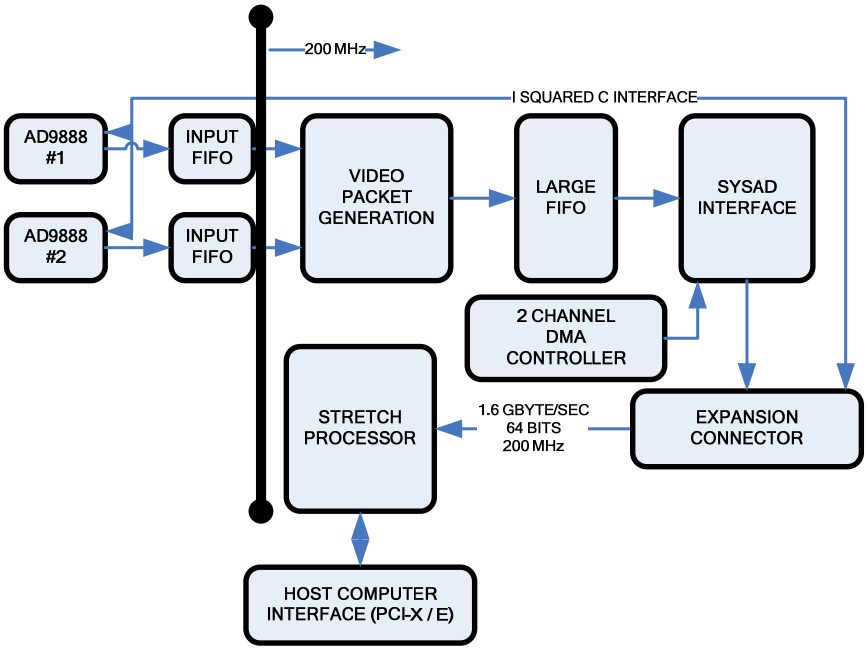


Figure 5 Data and flow control

The diagram above shows the data and control flow for the UXGA option. It shows the logical blocks in the FPGA which is on the Fast-X/e board, as well as the host interface. The AD9888s provide for the conversion of the analog video to digital form. The input FIFOs convert the digital data clock domain from the video clock speed to the 200 MHz clock used for the rest of the interface. The input FIFOs also provide a small amount of storage. Note: this limits the average video pixel rate to 200 mega pixels per second, for each video interface.

Video packets are generated as 32 byte data payload, with four bytes of address. The packets are stored in the large FIFO, until the DMA controller needs them. The large FIFO provides for any not ready conditions on the SYSAD bus to the Stretch processor. Video packets contain the target address of the buffer they go to in Stretch memory. The 2 Channel DMA controller provides for the transport of the video packets for both channels at the same time. The DMA controller is programmed by the Stretch to double buffer the data into Stretch memory.

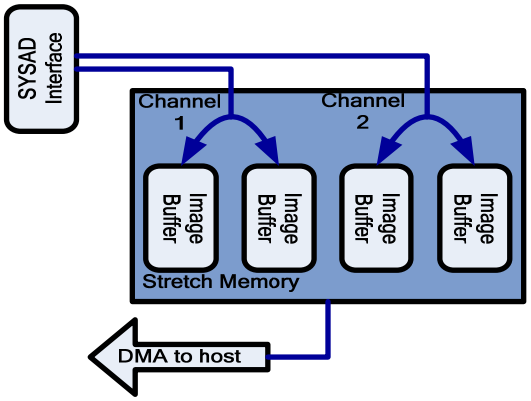


Figure : Stretch memory flow

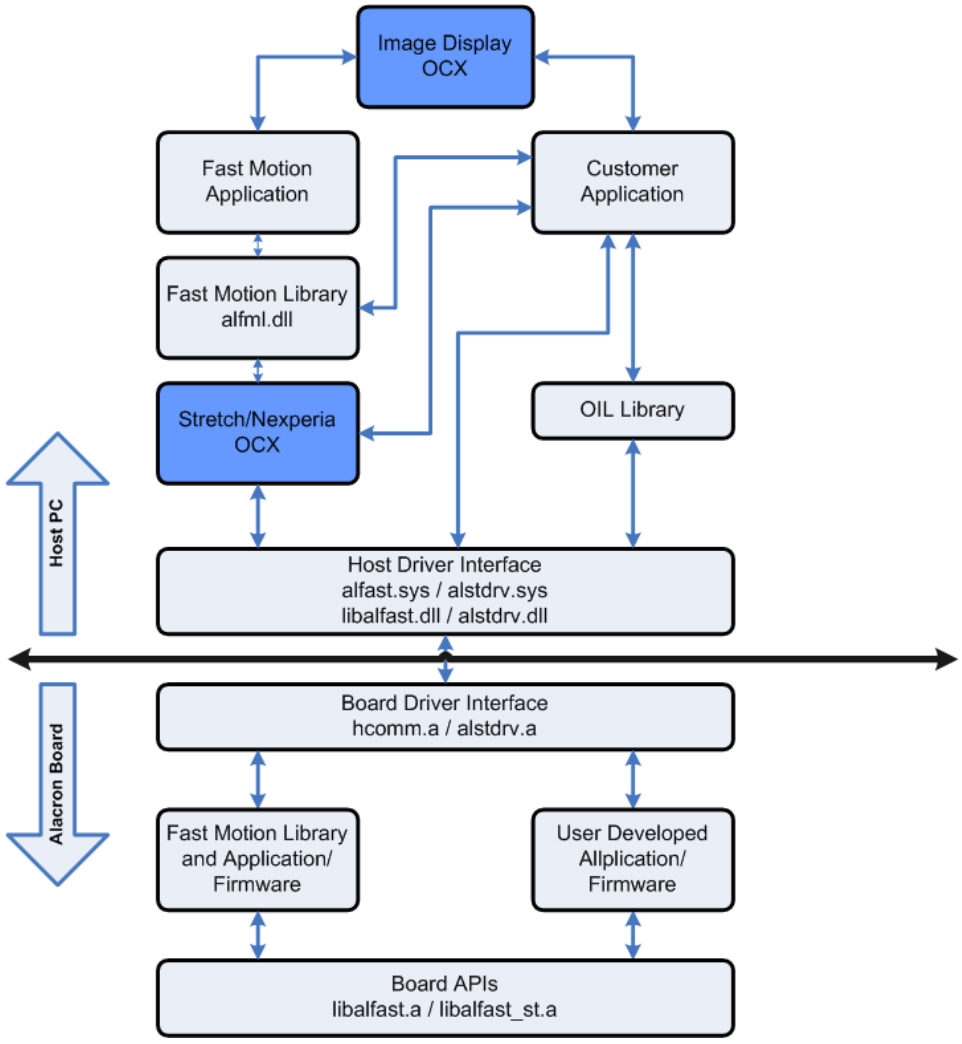
The sequence of operation of the system is as follows. (1) The Stretch programs the AD9888s over the I-squared-C bus to digitize the input video. (2) The Stretch programs the input FIFOs to accept data from the AD9888s (essentially just an enable). (3) The Stretch programs the DMA controller, to use two buffers in Stretch memory for each channel. (4) The Stretch starts the DMA controllers which enables the data flow. From this point video will be presented to the Stretch processor in a buffer in memory. The Stretch processor is notified of the buffer being ready (full) with an interrupt. For most applications the Stretch does not need to reprogram the interface while it is operating, except to switch the input multiplexers in the AD9888s.

Please see the Fast Motion Library manual for the details of the data flow, as well as the ALRT for Stretch manual for the details of the operation of the option boards.

The FPGA that implements the above data flow can be customized to provide different types of functionality. Please contact Alacron should you require modification to the FPGA firmware.

INTRODUCTION TO THE ALACRON OCX MODEL

When developing under Windows Alacron provides an OCX pathway to speed application development. These OCXs are also used in the canned Alacron applications and Libraries such as FastMotion Application and FastMotion Library that support simplified custom development. Alacron's development pathways are outlined in the table below.



Thus this section will concentrate on the documentation and use cases of the Alacron Imaging and the Alacron Stretch/Nexperia OCXs.

ALACRON IMAGING OCX

ALPICTURE OCX

INTRODUCTION

The Alacron Picture OCX (Alpicture.ocx) provides an easy way for a user to display an image from his application. It is not intended to be a high performance display object, but rather a load and examine the image tool. It can display several images a second, and is depends on the speed of the processor to do this. It does not take advantage of DirectX or OpenGL.

DEFINITION

The Alpicture.ocx can be defined by it Interfaces Description which we present in the IDL used to describe it. This OCX uses the MFC class library to leverage the predefined classes in MFC. It can not claim to be a light weight control, but rather a multipurpose control.

ALPICTURE TYPE LIBRARY IDL

```
 [uuid(F3D73419-0BEE-438C-BA84-264B37A829C5), version(2.0),
  helpfile("ALPicture.hlp"),
  helpstring("ALPicture 2.0 ActiveX Control module"),
  control ]
[ uuid(B6C7F46D-FF0B-4B3B-9FD0-5CD704A74493),
  helpstring("Dispatch interface for ALPicture Control 2.0"), hidden ]
dispinterface _DALPicture
{
properties:
[id(DISPID_HWND)] OLE_HANDLE hWnd;
[id(DISPID_READYSTATE), readonly] long ReadyState;
[id(0), readonly] long _ReadyState;
[id(1)] float Zoom;
[id(2)] long XOrigin;
[id(3)] long YOrigin;
[id(4)] boolean RedEnable;
[id(5)] boolean BluEnable;
[id(6)] boolean GrnEnable;
[id(7)] boolean AutoZoomAndPan;
[id(14)] boolean Rotated;
// 2.0 feature
[id(18), readonly] float Compression;
methods:
[id(8)] SCODE Update(long* Image, short Shift);
```

```
[id(9)] SCODE Create(long* Image, short Shift);
[id(10)] SCODE Move(long X, long Y);
[id(11)] SCODE ClientToImage(long X, long Y, long* iX, long* iY);
[id(12)] SCODE ImageToClient(long X, long Y, long* cX, long* cY);
[id(13)] SCODE UpdateROI(long* Image, long X, long Y,
                  long sX, long sY,short Shift);
[id(15)] SCODE ClearROI(long X, long Y,
                  long sX, long sY, long value);
[id(16)] SCODE MoveI(long x, long y);
// 2.0 feature
[id(17)] SCODE SwapColors (long a, long b);
[id(DISPID_ABOUTBOX)] void AboutBox();

//  Event dispatch interface for CALPictureCtrl
[ uuid(C0EC6F6F-7393-4754-B39A-7FBF55CA0C1C),
  helpstring("Event interface for ALPicture Control 2.0") ]
dispinterface _DALPictureEvents

properties:
//  Event interface has no properties
methods:
[id(DISPID_READYSTATECHANGE)] void ReadyStateChange();
[id(DISPID_MOUSEMOVE)] void MouseMove(short Button, short Shift,
                       OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);
[id(DISPID_MOUSEUP)] void MouseUp(short Button, short Shift,
                       OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);
[id(DISPID_DBLCLICK)] void DblClick(short Button, short Shift,
                       OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);

//  Class information for CALPictureCtrl
[ uuid(E582F61D-81DF-406C-9F49-FA9332DA6496), licensed,
        helpstring("ALPicture Control 2.0"), control ]
coclass ALPicture
{
      [default] dispinterface _DALPicture;
      [default, source] dispinterface _DALPictureEvents;
};
```

ALPICTURE PROPERTIES

id(DISPID_HWND) OLE_HANDLE hWnd;

This is a stock property provided by MFC for all controls. It is the handle to the window which is the face of the control.

id(DISPID_READYSTATE) long ReadyState;

This is a stock property provided by MFC for all controls. It is set to True and stays true.

id(1) float Zoom;

This variable controls how the images are displayed on the controls surface. When set to 1.0 the pixels of the image are display as pixels on the screen, this is some time called 1:1 viewing. If set to less than one a larger piece of the image is displayed (zoom out). If set larger than one a small portion of the image is displayed (zoom in).

id(2) long XOrigin;

This property controls which pixel is displayed in the upper left corner of the OCX display surface. Large values display pixels to the right.

id(3) long YOrigin;

This property controls which line is displayed at the top of the OCX display surface. Larger values move down the image.

id(4) boolean RedEnable;

If the image is color, and the RedEnable is true, then the first plane of the image is displayed. If it is false the first plane is not displayed. For non-RGB formats for example YUV 4:4:4 this refers to the Y part of the image. All images are converted from their native format to RGB when input to the control. Typically no color space conversion is done.

id(5) boolean BluEnable;

If the image is color, and the BlueEnable is true, then the last plane of the image is displayed. If it is false the last plane is not displayed. For non-RGB formats for example YUV 4:4:4 this refers to the V part of the image. All images are converted from their native format to RGB when input to the control. Typically no color space conversion is done.

id(6) boolean GrnEnable;

If the image is color, and the GrnEnable is true, then the middle plane of the image is displayed. If it is false the middle plane is not displayed. For non-RGB formats for example YUV 4:4:4 this refers to the U part of the image. All images are converted from their native format to RGB when input to the control. Typically no color space conversion is done.

id(7) boolean AutoZoomAndPan;

This flag controls whether the use of the control can pan and zoom the control view with the mouse and the shift key.

Normally the mouse just moves the cursor over the image. If the shift key is held down (either one), the image will pan. The panning area is the central portion of the display area, and is scaled so you can pan of the whole image without double dragging.

id(14) boolean Rotated;

This flag allows one to rotate the image 90 degrees, before displaying.

id(18) readonly] float Compression;

This property is set to the compression level of a JPEG image if one is passed to the control.

ALPICTURE METHODS

id(8) Update(long* Image, short Shift);

This method is used to provide a pointer to the image structure containing the image. The shift factor is applied to the pixel planes before display. The image structure is defined below. A long pointer is used to avoid complexity in the IDL of little benefit. As the pointer is being passed into the control, this method can not be used across processes. The Update API is canned by the Create API and is typically not used by a user. The Update method assumes that the image surface has already been setup in memory to match the display characteristics of the display mode in effect.

id(9) Create(long* Image, short Shift);

This method is used to provide a pointer to the image structure containing the image. The shift factor is applied to the pixel planes before display. The image structure is defined below. A long pointer is used to avoid complexity in the IDL of little benefit. As the pointer is being passed into

the control, this method can not be used across processes. The Update API is canned by the Create API and is typically not used by a user. The Create API builds an image surface in memory that matches the input image and then passes it to the Update method.

id(10) Move(long X, long Y);

This method moves the display window around on the image surface. The X and Y parameters are in display window space, X being the display pixel (column) and Y being the display line (row) of the image. The display window will move so X,Y is in the center of the display area.

id(11) ClientToImage(long X, long Y, long* iX, long* iY);

The method will take a point on the display surface of the image, and return its image coordinates.

id(12) ImageToClient(long X, long Y, long* cX, long* cY);

This method will take a point on the image and return its coordinates on the display surface. Note the returned values are not clipped, and could be out side the display area of the OCX.

id(13) UpdateROI(long* Image, long X, long Y, long sX, long sY,short Shift);

This method allows you to update a portion of an image. As big images can take a long time to be processed into the control. This API allows a smaller are of an image to be updated. It can be used to combine images on the same display surface. It uses the copy raster op to update the image surface, the pixels of the ROI that are on the display surface are replaced with those in the input image.

id(15) ClearROI(long X, long Y, long sX, long sY, long value);

This method updates an ROI to black, or all zero.

id(16) MoveI(long x, long y);

This method moves the display window so that the point (x,y) in the image is in the center of the display window. X and y are in image coordinates.

id(17) SwapColors (long a, long b);

This method allows the user to swap the display planes of the image. The parameters are the plane indexes (0=R,1=G,2=B etc.) to swap. It supports three cases (which are all there are):

Swap 0 with 1

Swap 0 with 2

Swap 1 with 2

id(DISPID_ABOUTBOX) void AboutBox();

This method will display the about box.

THE IMAGE STRUCTURE

The (long *) used in the methods above point to a structure called an image structure which is used in Alacron's OIL library and most other places in the Alacron software. It has the following definition:

```
typedef struct image_struct {
      int nr;
      int nc;
      int st;
      itype_t itype;
      int bpp;
      void *data;
      void (*pfree) (struct image_struct *pi);
      void *extension;
```

```
memclass_t memclass;
int ncomp;
struct image_struct *next;
int id;

} image_t;
```

Most of these fields are clear. 'nr' number of rows (lines) in the image, sometimes also called the height of the image. 'nc' is the number of columns, pixels,width of the image. 'st' is the image stride, or the number of bytes from the left most pixel in a line to the left most pixel in the next line. Note: It can be negative. 'data' points to the actual image data. The size of this data area is at least nr*st bytes. 'pfree' points to a function which can free the image data and the structure. This function is used by the OIL API ImageFree. 'extension' is a pointer to data that some how extends the image (like a color map). 'memclass' describes the type of memory that the image data uses. Normally this is local memory. 'ncomp' reflects the number of additional image structures linked on the 'next' field. For planer images each plane is a separate image linked in a list on the 'next' field. 'id' is an application specific value.

The OCX only uses nr,nc.st.itype,bpp,data, and ncomp. The other fields are not read.

What is passed to the OCX APIs as a long pointer is a pointer to the image structure in memory.

The two enum types are:

```
typedef enum {IMAGE_CLASS_LOCAL=1, IMAGE_CLASS_PCI} memclass_t;
```

This type indicates the kind of memory being used, which for the host will always be IMAGE_CLASS_LOCAL. The OCX does not use this field.

```
typedef enum {
            IMAGE_TYPE_GRAY8 = 1,
            IMAGE_TYPE_GRAY16,
            IMAGE_TYPE_GRAY32,
            IMAGE_TYPE_YUV422,
            IMAGE_TYPE_YUV420,
            IMAGE_TYPE_RGBPLANER,
            IMAGE_TYPE_RGB15,
            IMAGE_TYPE_RGB16,
            IMAGE_TYPE_RGB24,
            IMAGE_TYPE_RGB32,
            IMAGE_TYPE_GRAY32F,
            IMAGE_TYPE_BINARY,
            IMAGE_TYPE_RGBPLANER16,
            IMAGE_TYPE_RGBPLANER32,
            IMAGE_TYPE_RGB48,
            IMAGE_TYPE_BINARYR,
            IMAGE_TYPE_RGB8_332,
```

```
        IMAGE_TYPE_RGB8_233,
        IMAGE_TYPE_UYVY,
        IMAGE_TYPE_BAYER8,
        IMAGE_TYPE_BAYER16,
        IMAGE_TYPE_BAYERLINER8,
        IMAGE_TYPE_BAYERLINER16,
        IMAGE_TYPE_YUV24, //YUV444 in 24 bit pixels
        IMAGE_TYPE_YUV32, //YUV444 in 32 bit pixels
    } itype_t;
```

This enum defines the type of data in memory.

ALPICTURE EVENTS

This OCX supports these events:

void ReadyStateChange();

This event is trigger if the OCX changes ready state, which it never does.

void MouseMove(short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);

This event allows the user to trap mouse movements over the control.

void MouseUp(short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);

This event is triggered when a mouse button is released in the OCX. This is used as a click indication

void DblClick(short Button, short Shift, OLE_XPOS_PIXELS x, OLE_YPOS_PIXELS y);

This event is triggered if the mouse is double clicked within the OCX display window.

USING THE CONTROL

Be sure you have the file 'alpicture.lic' in the same directory and the OCX if you are developing software. This file is needed to enable the control to be inserted in a new application.

The control as provided will need to be Com registered as all OCXs are. The control contains code to do that for you. This can be done using the program 'regsvr32.exe' which is in the Windows System32 directory. You can drag and drop the OCX on the file name or pass the name of the control to the program on the command line.

If the control does not register, the likely cause is a DLL it uses is not found.

The DLLs used by the control are:

GDIPLUS.DLL	Microsoft
LIBBASICFF.DLL	Alacron Inc.
MFC71.DLL	Microsoft
MSVCR71.DLL	Microsoft

All the other DLLs used are provided with the Windows installation.

ALACRON'S STRETCH/NEXPERIA OCX

Many of the Alacron Stretch applications are based on an ActiveX Control (OCX). The control masks the details of using ALRT to load and communicate with a processor. The interface is based on a callback scheme which provides information to the user, and an interface definition which performs all the functions provided by the ALRT runtime environment.

The OCX does not provide support for abstractions such as images and ring buffers it only provides a thin layer over the ALRT driver and drive library, while simplifying their use. In addition it does not provide support for the old legacy interfaces, which do not perform well in any case, and to a degree can not be supported due to the way the Stretch provides access to its memory.

If you do not know what an OCX is, the short definition is it is a component object (Com Object), which supports a number of standard interfaces required for it to be instantiated into a host program in a standard control site in an application.

The StretchBoard.OCX is implemented with Microsoft ATL.

One possible drawback of the OCX is it does not use 'standard' events to signal events to the user. It uses a callback which is higher performance, but will not work with older versions of Visual Basic.

The OCX was not created for Visual Basic users and does not have a lot of testing time in that local. It is used by Alacron in C++ applications.

INTERFACES

ISTRETCHBOARD

The definition of the interface is:

```
[
      object,
      uuid(D5D4FE04-5254-430E-9E19-03313AF769A6),
      dual,
      helpstring("IStretchBoard Interface"),
      pointer_default(unique)
]
__interface IStretchBoard : public IDispatch
{
      [propput, bindable, requestedit, id(DISPID_VALID)]
      HRESULT Valid([in]VARIANT_BOOL vbool);
      [propget, bindable, requestedit, id(DISPID_VALID)]
      HRESULT Valid([out, retval]VARIANT_BOOL* pbool);

      [propput, bindable, requestedit, id(DISPID_READYSTATE)]
      HRESULT ReadyState([in]long state);
      [propget, bindable, requestedit, id(DISPID_READYSTATE)]
      HRESULT ReadyState([out, retval]long* pstate);

      [propget, id( 1), helpstring("property pid")]
      HRESULT pid ([out, retval] LONG* pVal);

      [propget, id( 2), helpstring("property rval")]
```

```
      HRESULT rval ([out, retval] LONG* pVal);

      [propget, id( 3), helpstring("property ResetOnRelease")]
      HRESULT ResetOnRelease ([out, retval] BOOL* pVal);
      [propput, id( 3), helpstring("property ResetOnRelease")]
      HRESULT ResetOnRelease ([in] BOOL newVal);

      [propget, id( 4), helpstring("property FlagRegister")]
      HRESULT FlagRegister ([in] ULONG index, [out,retval] ULONG
*pVal);
      [propput, id( 4), helpstring("property FlagRegister")]
      HRESULT FlagRegister ([in] ULONG index, [in] ULONG newVal);

      [propget, id( 5), helpstring("property AlfastVersion")]
      HRESULT AlfastVersion ([out,retval]CHAR** pVer);

      [propget, id( 6), helpstring("property PCIProbeVersion")]
      HRESULT PCIProbeVersion ([out,retval]CHAR** pVer);

      [propget, id( 7), helpstring("property MMIO")]
      HRESULT MMIO ([in] ULONG reg, [out,retval]ULONG *pVal);
      [propput, id( 7), helpstring("property MMIO")]
      HRESULT MMIO ([in] ULONG reg, [in]ULONG newVal);

      [propget, id(8), helpstring("property SRAM Base")]
      HRESULT SRAM ([out,retval]CHAR **pVal);

      [id(50), helpstring("method Open")]
      HRESULT Open ([in] ULONG pid, [in] CHAR* CapFileName);

      [id(51), helpstring("method CapValue")]
      HRESULT CapValue ([in] CHAR* key, [in] CHAR* subkey,
[out,retval] VARIANT* value);

      [id(54), helpstring("method SDRAMQuery")]
      HRESULT SDRAMQuery ([out] ULONG *Size, [out]ULONG *paddr,
[out,retval] CHAR ** pVaddr);

      [id(55), helpstring("method SendMsg")]
      HRESULT SendMsg ([in] LONG id, [in] LONG len, [in] CHAR * msg);

      [id(56), helpstring("method ShmQuery")]
      HRESULT ShmQuery ([out] ULONG * size, [out] ULONG *paddr, [out]
CHAR ** ppVaddr);

      [id(57), helpstring("method Advise")]
```

```
        HRESULT Advise ([in] IUnknown *pEvent, [out] ULONG *Cookie);
};
```

PROPERTIES

BOOL VALID

Valid is the standard property provided by ActiveX Controls. It is get/put able.

BOOL READYSTATE

Ready is the standard property provided by ActiveX Controls. It is get/put able.

INT PID ID(1)

PID is the processor number for this object. It is only getable.

INT RVAL ID(2)

'rval' is the last return value for an ALRT method called. It is only getable.

BOOL RESETONRELEASE ID(3)

ResetOnRelease if set t true will cause the OCX to reset the processor when it is released (reference count goes to zero). It is get/put able.

ULONG FLAGREGISTER[0-9] ID(4)

FlagRegister is an ulong array of 10 values which are mutually accessible by the host and the Stretch processor. It is uncached. It is get/put able. It resides in SRAM inside the Stretch processor.

*CHAR *ALFASTVERSION ID(5)*

AlfastVersion returns a pointer to the version string of the under laying ALRT. It is get able. The returned point is to an internal const string. The pointer is only valid in the current process.

*CHAR * PCIPROBEVERSION ID(6)*

This attribute is obsolete and returns not implemented.

ULONG MMIO(REG) ID(7)

This attribute is not implemented in the control and returns not implemented.

*CHAR *SRAM ID(8)*

This attribute returns a pointer (virtual address) to the SRAM inside the Stretch. It provides the base address. It is an application problem to make this useful if it's needed.

```
RESULT SRAM ([out,retval]CHAR **pVal);
```

METHODS

*OPEN(PID, CHAR *FILENAME) ID(50)*

This method attaches to a processor (Stretch Board), and uses the content of the Cap File, to load the processor with some code. It typically fails for missing Cap File, Missing processor code, or missing FPGA files.

*VARIANT CAPVALUE (CHAR *KEY, CHAR *SUBKEY) ID(51)*

This method provides access to the values in the cap file which opened this processor. The values are returned as a Variant, so that string, float, and int values are supported by the method. There is no write method. Cap files are read only.

ID(52) ID(53) ARE NOT USED.

SDRAMQUERY ID(54)

This method is not implemented on the Stretch.

*SENDMSG (INT COMMAND, INT LEN, CHAR*DATA) ID(55)*

This method sends a message to its processor. Parameter data is limited to less than 128 bytes.

SHMQUERY (ULONG SIZE, ULONG* PADDR, CHAR **PPVADDR) ID(56)*

This method returns the size, physical address, and a virtual pointer to the start of shared memory in host memory. This memory is not otherwise used by the OCX. It is up to the application to implement the meaning of 'shared'.

ADVISE ID(57)

This interface provides the OCX with the IUnknown interface of the com object that supports the call back function via the IStretchEvent interface. The provided interface is not reference counted or released by the method. The returned cookie is used in the unadvised call to the Advise method.

After a successful advise call, the Event method of the IStretchEvent interface will be called for every event detected by the OCX. All processors registered with any OCX in this process are notified. See below.

An un-advice is performed with the same interface, with a null pointer for the first parameter, and a pointer to the cookie returned in the successful advise call. If the method works, no more calls to the Event method will be made, and the com object will be released.

ISTRETCHBOARDEVENT

IStretchBoardEvent is an outgoing interface implemented by the application which is using the OCX. Its signature is

```
HRESULT Event([in] LONG type, [in] LONG Length, [in] BYTE * Msg);
```

The type parameter is dual purpose. The upper 8 bits of the value is the processor number (aka pid). The remaining 24 bits indicate the reason for the call.

A type == 1 call indicates an ASCII message from the processor is pointed to by MSG, whose length is Length. These messages are application dependent but do not typical require any kind of response.

A type==0 call indicates the reception of a message from the processor. The length parameter is the size of the msg structure, the pointer Msg is a pointer to the message structure. The pointer is only valid within the Event method, it is a stack object which will be popped on return.

A type==10 call indicates the reception of an event trigger from the processor. The length field contains the index of the event (0 to 9). It is application dependent what these events mean.

A type==666 call indicates there was no activity from the processor for one second.

TROUBLESHOOTING

There are several things you can try before you call Alacron Technical Support for help.

_____ Make sure the computer is plugged in. Make sure the power source is on.

_____ Go back over the hardware installation to make sure you didn't miss a page or a section.

_____ Go back over the software installation to make sure you have installed all necessary software.

_____ Run the Installation User Test to verify correct installation of both hardware and software.

_____ Run the user-diagnostics test for your main board to make sure it's working properly.

_____ Insert the Alacron CD-ROM and check the various Release Notes to see if there is any information relevant to the problem you are experiencing.

The release notes are available in the directory: **\usr\alacron\alinfo**

_____ Compile and run the example programs found in the directory: **\usr\alacron\src\examples**

_____ Find the appropriate section of the Programmer's Guide & Reference or the Library User's Manual for the particular library and problem you are experiencing. Go back over the steps in the guide.

_____ Check the programming examples supplied with the runtime software to see if you are using the software according to the examples.

_____ Review the return status from functions and any input arguments.

_____ Simplify the program as much as possible until you can isolate the problem. Turning off any operations not directly related may help isolate the problem.

_____ Finally, first **save your original work**. Then remove any extraneous code that doesn't directly contribute to the problem or failure.

ALACRON TECHNICAL SUPPORT

Alacron offers technical support to any licensed user during the normal business hours of 9 a.m. to 5 p.m. EST. We offer assistance on all aspects of processor board and PMC installation and operation.

CONTACTING TECHNICAL SUPPORT

To speak with a Technical Support Representative on the telephone, call the number below and ask for Technical Support:

Telephone: **603-891-2750**

If you would rather FAX a written description of the problem, make sure you address the FAX to Technical Support and send it to:

Fax: **603-891-2745**

You can email a description of the problem to ***support@alacron.com***

Before you can contact technical support have the following information ready:

_____ Serial numbers and hardware revision numbers of all of your boards. This information is written on the invoice that was shipped with your products.

_____ Also, each board has its serial number and revision number written on either in ink or in bar-code form.

_____ The version of the ALRT, or FastMotion software that you are using.

_____ You can find this information in a file in the directory: **\usr\alfast\alinfo**

_____ The type and version of the host operating system, i.e., Windows XP.

_____ Note the types and numbers of all your software revisions, daughter card libraries, the application library and the compiler

_____ The piece of code that exhibits the problem, if applicable. If you email Alacron the piece of code, our Technical-Support team can try to reproduce the error. It is necessary, though, for all the information listed above to be included, so Technical Support can duplicate your hardware and system environment.

RETURNING PRODUCTS FOR REPAIR OR REPLACEMENT

Our first concern is that you be pleased with your Alacron products.

If, after trying everything you can do yourself, and after contacting Alacron Technical Support, you feel your hardware or software is not functioning properly, you can return the product to Alacron for service or replacement. Service or replacement may be covered by your warranty, depending upon your warranty.

The first step is to call Alacron and request a "Return Materials Authorization" (RMA) number.

This is the number assigned both to your returning product and to all records of your communications with Technical Support. When an Alacron technician receives your returned hardware or software he will match its RMA number to the on-file information you have given us, so he can solve the problem you've cited.

When calling for an RMA number, please have the following information ready:

_____ Serial numbers and descriptions of product(s) being shipped back

_____ A listing including revision numbers for all software, libraries, applications, daughter cards, etc.

_____ A clear and detailed description of the problem and when it occurs

_____ Exact code that will cause the failure

_____ A description of any environmental condition that can cause the problem

All of this information will be logged into the RMA report so it's there for the technician when your product arrives at Alacron.

Put boards inside their anti-static protective bags. Then pack the product(s) securely in the original shipping materials, if possible, and ship to:

Alacron Inc.
71 Spit Brook Road, Suite 200
Nashua, NH 03060
USA

Clearly mark ***the outside of your package:***

Attention **RMA #80XXX**

Remember to include your return address and the name and number of the person who should be contacted if we have questions.

REPORTING BUGS

We at Alacron are continually improving our products to ensure the success of your projects. In addition to ongoing improvements, every Alacron product is put through

extensive and varied testing. Even so, occasionally situations can come up in the fields that were not encountered during our testing at Alacron.

If you encounter a software or hardware problem or anomaly, please contact us immediately for assistance. If a fix is not available right away, often we can devise a work-around that allows you to move forward with your project while we continue to work on the problem you've encountered.

It is important that we are able to reproduce your error in an isolated test case. You can help if you create a stand-alone code module that is isolated from your application and yet clearly demonstrates the anomaly or flaw.

Describe the error that occurs with the particular code module and email the file to us at:

support@alacron.com

We will compile and run the module to track down the anomaly you've found.

If you do not have Internet access, or if it is inconvenient for you to get to access, copy the code to a disk, describe the error, and mail the disk to Technical Support at the Alacron address below.

If the code is small enough, you can also:

FAX the code module to us at **603-891-2745**

If you are faxing the code, write everything large and legibly and remember to include your description of the error.

When you are describing a software problem, include revision numbers of all associated software.

For documentation errors, photocopy the passages in question, mark on the page the number and title of the manual, and either FAX or mail the photocopy to Alacron.

Remember to include the name and telephone number of the person we should contact if we have questions.

Alacron Inc.
71 Spit Brook Road, Suite 200
Nashua, NH 03060
USA

Telephone: 603-891-2750
Fax: 603-891-2745
Web site
http://www.alacron.com/
Electronic Mail
sales@alacron.com
support@alacron.com