# *FASTCAMERA SERIES*

FASTCAMERA13
USER'S MANUAL

**FVM-50013**

# ***COPYRIGHT NOTICE***

## *Copyright © 2004 by FastVision, LLC.*

All rights reserved. This document, in whole or in part, may not be copied, photocopied, reproduced, translated, or reduced to any other electronic medium or machine-readable form without the express written consent of FastVision, LLC.

FastVision makes no warranty for the use of its products, assumes no responsibility for any error, which may appear in this document, and makes no commitment to update the information contained herein. FastVision, LLC. retains the right to make changes to this manual at any time without notice.

Document Name: FastCamer13 User's Manual

Document Number: 30002-50013

Revision History:

| | |
|---|---|
| 1.0 | January 20,2003 |
| 1.1 | March 26,2003 |
| 1.2 | March 30,2003 |
| 1.3 | April 9, 2003 |
| 2.0 | August 10,2004 |
| 3.0 | August 13, 2004 |
| 4.0 | August 17, 2004 |


## *Trademarks:*

**FastVision®** is a registered trademark of FastVision, LLC..
**Channel Link™** is a trademark of National Semiconductor.
**3M™** is a trademark of 3M Company
**MS DOS®** is a registered trademark of Microsoft Corporation
**SelectRAM™** is a trademark of Xilinx Inc.
**Solaris™** is a trademark of Sun Microsystems Inc.
**Unix®** is a registered trademark of Sun Microsystems Inc.
**Virtex™** is a trademark of Xilinx Inc.
**Windows™, Windows 95™, Windows 98™, Windows 2000™, Windows NT™, and Windows XP™** are trademarks of Microsoft

**All trademarks are the property of their respective holders.**

**FastVision, LLC.**
**131 Daniel Webster Highway, #529**
**Nashua, NH 03060**
**USA**

**Telephone: 603-891-4317**
**Fax: 603-891-1881**

**Web Site:**
**http://www.fast-vision.com/**
**Email:**
**sales@fast-vision.com, or support@fast-vision.com**

# *TABLE OF CONTENTS*


*1. INTRODUCTION* ____ *1-6*

*2. Features and Specifications* ____ *2-6*

**2.1. CAMERA SPECIFICATIONS** ____ **2-7**

**2.2. IMAGE SENSOR SPECIFICATIONS:** ____ **2-7**

**2.3. Physical Specifications** ____ **2-7**

**2.4. Connectors** ____ **2-9**

2.4.1. Power Connector HR10A-10R-12PB ____ 2-9

2.4.2. Data Connector J2 ____ 2-9

2.4.3. Data Connector J3 ____ 2-9

*3. Power Requirements* ____ *3-10*

*4. Timing* ____ *4-10*

*5. Trigger Modes* ____ *5-10*

*6. Camera Data Flow* ____ *6-12*

*7. The Standard Camera Functionality* ____ *7-13*

*8. Pixel Gain and Offset* ____ *8-13*

*9. Memory Option* ____ *9-13*

**9.1. FIFO memory mode** ____ **9-13**

**9.2. Circular buffer memory mode** ____ **9-14**

**9.3. Image summing memory mode** ____ **9-14**

*10. Lookup Table Option* ____ *10-14*

*11. Data Format Funnel* ____ *11-14*

*12. Internal Camera Memory* ____ *12-14*

*13. Sensor Control* ____ *13-14*

**13.1. Line Timing** ____ **13-15**

**13.2. Frame Timing** ____ **13-15**

**13.3. Exposure Modes** ____ **13-15**

**13.4. Trigger Modes** ____ **13-15**

**13.5. Trigger Options** ____ **13-16**

**13.6. Trigger Outputs** ____ **13-17**

**13.7. Reference Voltages** ____ **13-17**

**13.8. Calibration** ____ **13-17**

**13.9. Power-on Initialization 13-17**

***14. Flash Memory 14-18***

***15. Camera State Storage 15-19***

***16. Serial Control Interface 16-20***

**16.1. Encoding 16-21**

***17. Inter-FPGA Communication 17-25***

***18. Embedded Soft Processor Core 18-26***

***19. Data FPGA 19-31***

**19.1. Functional Description 19-31**

**19.2. Video Data path overview 19-33**

**19.3. Operating mode control overview 19-33**

**19.4. Camera State 19-33**

**19.5. DataFpga State controls 19-34**

19.5.1. Camera Link Readout Mode 19-34
19.5.2. Camera Link Clock Frequency setting 19-37
19.5.3. Binning Control setting 19-37
19.5.4. Binning Multiplier setting 19-37

***20. Frame Rates. 20-38***

**20.1. Memory Operation. 20-38**

**20.2. DATA FPGA Technical Details. 20-38**

**20.3. 1 Tap mode 20-40**

**20.4. 2 Tap mode 20-41**

**20.5. 4 Tap mode 20-42**

**20.6. 8 Tap mode 20-43**

**20.7. Sensor Interface 20-44**

***21. USB Camera Option 21-45***

***22. Camera Control Program 22-45***

***23. Application environments 23-45***

**23.1. USB operation 23-45**

**23.2. The hardware connections are: 23-45**

**23.3. The software connections are: 23-46**

**23.4. Using a FastVision supplied framegrabber 23-46**

23.4.1. The hardware connections are: 23-47
23.4.2. The software connections are: 23-47

**23.5. Using a third-party framegrabber 23-48**

23.5.1. The hardware connections are: 23-48
23.5.2. The software connections are: 23-49

*24. TROUBLESHOOTING* _____ *24-50*

*25. FastVision TECHNICAL SUPPORT* _____ *25-50*

**25.1. Contacting Technical Support** _____ **25-50**

**25.2. Returning Products for Repair or Replacements** _____ **25-50**

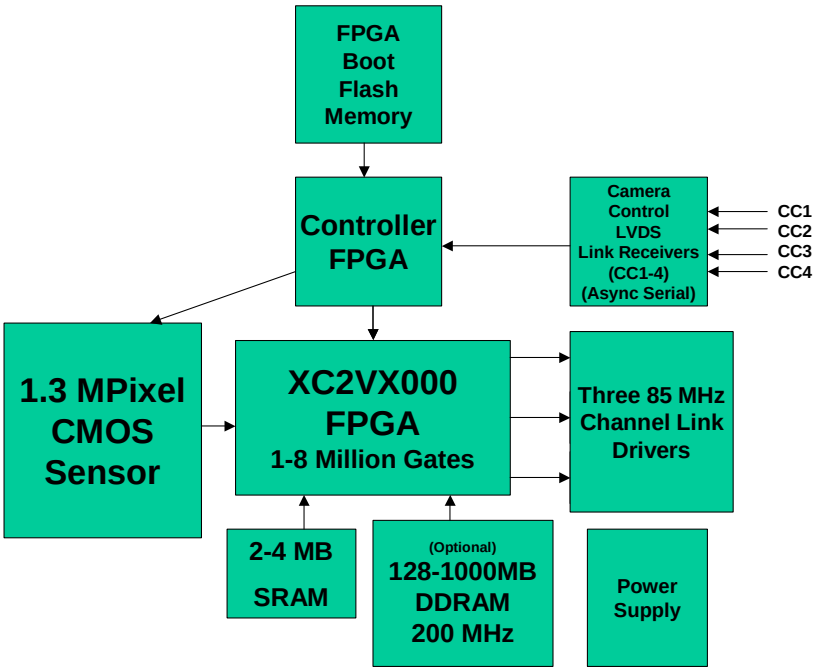
**25.3. Reporting Bugs** _____ **25-51**

# 1. INTRODUCTION

The FastCamera13 is a 1.3 megapixel CMOS camera with internal memory and FPGA's that enable it to do real-time processing. Thus it is what one would term a "smart" camera. The standard programming that is supplied with the base camera forms the basis of the most used and demanded function for data processing. It is expected that each customer may wish to customize this programming for its use. This can be achieved by purchasing additional IP from FastVision or in-house development. Inquires or discussions are always welcome.

# 2. FEATURES AND SPECIFICATIONS

The block diagram below shows the major subsystems in the camera. The camera is designed to support many different applications by customization of the programmable logic in the camera. FastVision or the customer can customize the size and content of the FPGAs and memory in the camera to contain many different features, processing algorithms, and storage schemes. This manual discusses the 'Standard' version of the cameras. Customization of the FPGAs in the camera requires significant support from FastVision. Please contact FastVision for a quote for the development tools and support.



## 2.1. CAMERA SPECIFICATIONS

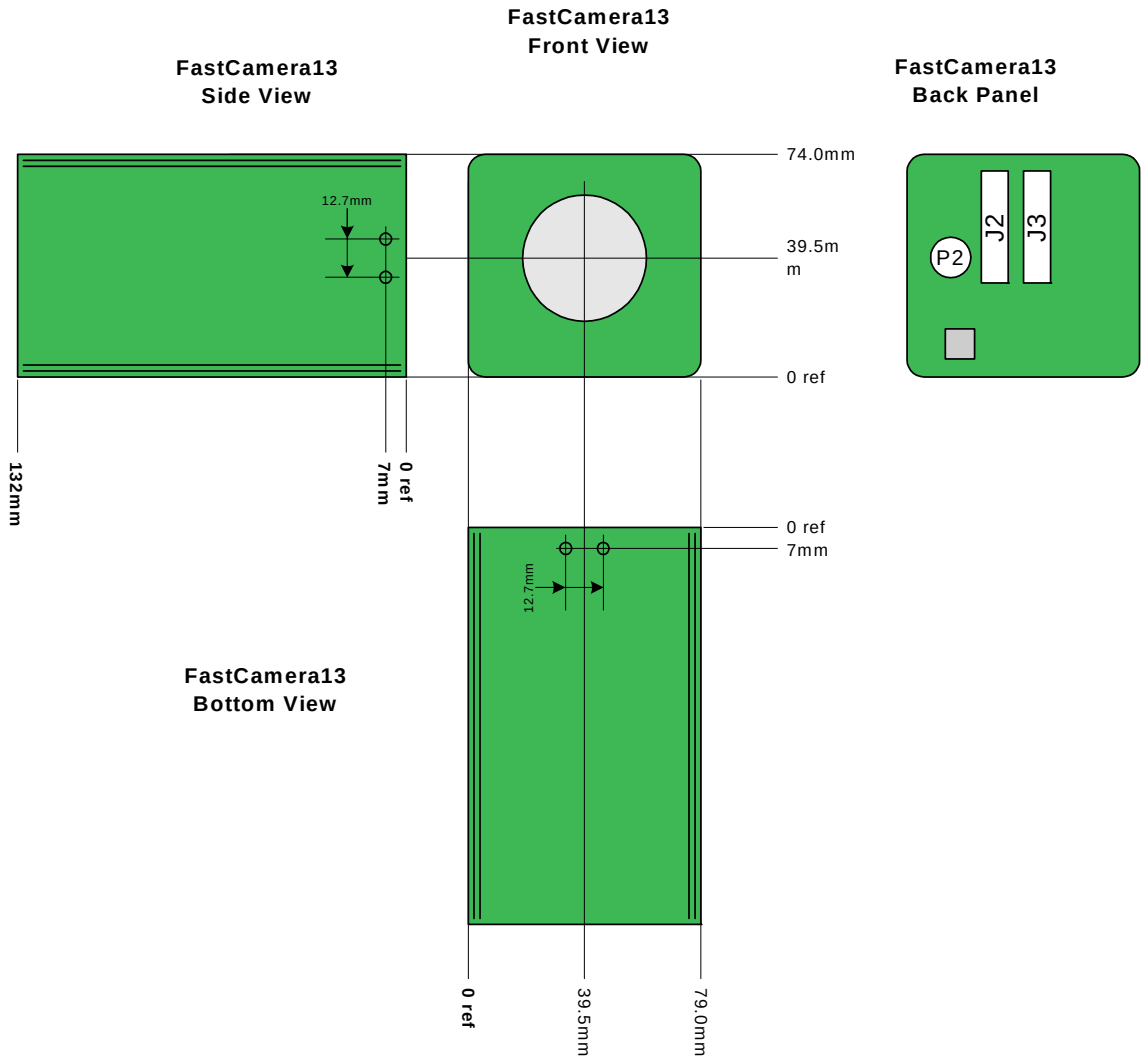
- The FastCamera13 uses a 1280H x 1024V (1.3 megapixel) CMOS digital image sensor capable of 500 frames/second operation at full resolution
- 1280H x 1024V image resolution
- 12-micron-square active-pixel photodiodes
- 500+ frames per second, progressive-scan
- At full resolution, frame rate can go up to 500,000 fps at 1 x 1280pixels
- Monochrome or color (Bayer Pattern)
- Ten (10) parallel output ports
- Photobit® TrueColorTM Image Fidelity
- On-chip TrueBit® Noise Cancellation
- On-chip 10-bit analog-to-digital converters
- FPGA and memory-based configurable interface formats and onboard processing
- Supported by a full range of software tools
- Binning in order to achieve increased sensitivity at full frame rates
- Optional SRAM for ultra fast processing
- Optional additional DDRAM and increased FPGA size for additional processing capability
- Trigger-able global electronic shutter (sync./async. modes)
- C holder mount (F with adaptor)

## 2.2. IMAGE SENSOR SPECIFICATIONS:

- Uses Micron Imaging's MI-MV13 sensor
- 1280 x 1024 x 8 bits @ 500 fps (10 bits 400 fps)
- 15.36 mm x 12.29 mm active area
- 12-micron square active pixels
- 40% Fill Factor
- Monochrome or color (Bayer Pattern)
- On-chip Noise Cancellation
- Dynamic range 59 db
- Monochrome: 1000 bits per lux-second @ 550 nm
- Shutter 99.9% efficiency
- Noise 58 db (10 bit mode lowest sensor gain setting, nominal pixel of 512 counts)

## 2.3. Physical Specifications

## FASTCAMERA13 CASE AND MOUNTING DIMENSIONS



## *2.4. Connectors*

### 2.4.1. Power Connector HR10A-10R-12PB

| Pin | Function |
|---|---|
| 1 | Ground |
| 2 | +5 Volts |
| 3 | Ground |
| 4 | Reserved for application dependent I/O |
| 5 | Ground |
| 6 | +5 Volts |
| 7 | Reserved for application dependent I/O |
| 8 | Ground |
| 9 | +5 Volts |
| 10 | Reserved for application dependent I/O |
| 11 | +5 Volts |
| 12 | Reserved for application dependent I/O |

### 2.4.2. Data Connector J2

| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | Ground | 14 | Ground |
| 2 | CL1_TXOUT0N | 15 | CL1_TXOUT0P |
| 3 | CL1_TXOUT1N | 16 | CL1_TXOUT1P |
| 4 | CL1_TXOUT2N | 17 | CL1_TXOUT2P |
| 5 | CL1_TXCLKOUTN | 18 | CL1_TXCLKOUTP |
| 6 | CL1_TXOUT3N | 19 | CL1_TXOUT3P |
| 7 | SERTCP | 20 | SERTCN |
| 8 | CC1N | 21 | SERTFGP |
| 9 | CC2P | 22 | CC1P |
| 10 | CC3N | 23 | CC2N |
| 11 | CC4P | 24 | CC3P |
| 12 | SERTFGN | 25 | CC4N |
| 13 | Ground | 26 | Ground |

### 2.4.3. Data Connector J3

| Pin | Signal | Pin | Signal |
|---|---|---|---|
| 1 | Ground | 14 | Ground |
| 2 | CL2_TXOUT0N | 15 | CL2_TXOUT0P |
| 3 | CL2_TXOUT1N | 16 | CL2_TXOUT1P |
| 4 | CL2_TXOUT2N | 17 | CL2_TXOUT2P |
| 5 | CL2_TXCLKOUTN | 18 | CL2_TXCLKOUTP |
| 6 | CL2_TXOUT3N | 19 | CL2_TXOUT3P |
| 7 | Application Specific I/O (P) | 20 | Application Specific I/O (N) |
| 8 | CL3_TXOUT0N | 21 | CL3_TXOUT0P |
| 9 | CL3_TXOUT1N | 22 | CL3_TXOUT1P |
| 10 | CL3_TXOUT2N | 23 | CL3_TXOUT2P |
| 11 | CL3_TXCLKOUTN | 24 | CL3_TXCLKOUTP |
| 12 | CL3_TXOUT3N | 25 | CL3_TXOUT3P |
| 13 | Ground | 26 | Ground |

# 3. POWER REQUIREMENTS

Power requirements are a strong function of the specific application; the camera is 15 Watts worst case, 5 to 10 Watts typical. Low noise +5 Volt input recommended. Internally the camera has high frequency switching supplies that convert the 5 volt input to 3.3, 2.5 and 1.8 volts.

# 4. TIMING

- Camera Clock is configurable from 25 to 85 MHz depending on the application.
- Pixel data is valid when LVAL, FVAL and DVAL are true.
- Minimum two clocks LVAL and FVAL inactive between lines and frames.
- CC4 reserved for application specific needs.

# 5. TRIGGER MODES

Trigger modes are selected by serial commands.

Free Running, camera generates frames without triggers.

CC1 Positive edge triggered.

CC1 When active, expose, falling edge reads out the sensor.

Sensor readout can be in parallel with exposure.

The camera can be read out with 1,2, or 3 camera links, the serial port (for the really patient), or via the USB port. Only image data from the selected ROI is sent. The camera can be configured to read out when the image is taken, to read out from memory on serial command or the CC2 positive edge.

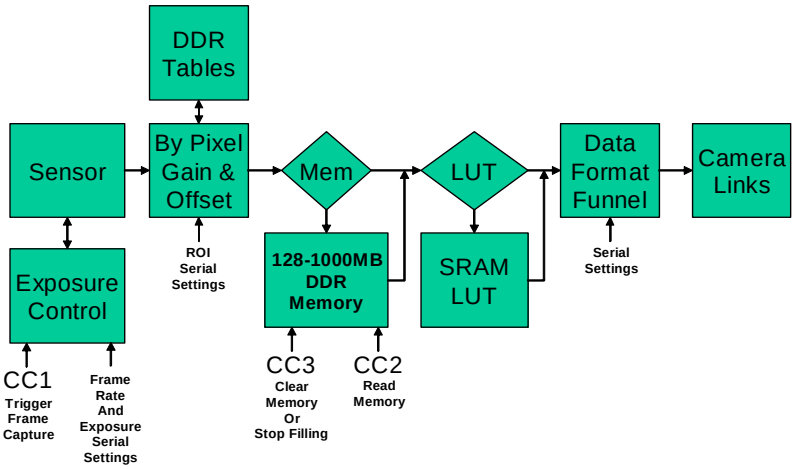
#### 1.1.1.1 Data formats supported:

| Mode | Camera Link Format<br>MSB ---------- LSB | Read Out Mode |
|---|---|---|
| 0 | CL1_A[7:0] | Single tap 8 bits (Basic) (default) |
| 1 | CL1_B[1:0],CL1_A[7:0] | Single tap 10 bits (Basic) |
| 2 | CL1_A[7:0] even pixels (0,2,...)<br>CL1_B[7:0] odd pixels (1,3,...) | Two Taps 8 bits (Basic) |
| 3 | CL1_B[1:0],CL1_A[7:0] even<br>CL1_B[5:4],CL1_C[7:0] odd | Two Taps 10 bits (Basic) |
| 4 | CL1_A[7:0] Red, Y<br>CL1_B[7:0] Green, U<br>CL1_C[7:0] Blue, V | Three Taps 8 bits (Color) (Basic) |
| 5 | CL1_B[1:0],CL1_A[7:0] Red, Y<br>CL1_B[5:4],CL1_C[7:0] Green, U<br>CL2_D[1:0],CL2_C[7:0] Blue, V | Three Taps 10 bits (Color) (Medium) |
| 6 | CL1_B[7:0],CL1_A[7:0]<br>CL1_A[4:0] Red, U<br>CL1_B[2:0],CL1_A[7:5] Green, Y<br>CL1_B[7:3] Blue V | One tap 16 Bits RGB565 (Basic) |
| 7 | CL1_A[7:0] pixels 4*n+0<br>CL1_B[7:0] pixels 4*n+1 | Four Taps 8 bits (Medium) |

| | | |
|---|---|---|
| | CL1_C[7:0] pixels 4*n+2<br>CL2_A[7:0] pixels 4*n+3 | |
| 8 | CL1_B[1:0],CL1_A[7:0] 4*n+0<br>CL1_B[5:4],CL1_C[7:0] 4*n+1<br>CL2_C[1:0],CL2_B[7:0] 4*n+2<br>CL2_C[4:5],CL2_A[7:0] 4*n+3 | Four Taps 10 bits (Medium) |
| 9 | CL1_A[7:0] pixels 5*n+0<br>CL1_B[7:0] pixels 5*n+1<br>CL1_C[7:0] pixels 5*n+2<br>CL2_A[7:0] pixels 5*n+3<br>CL2_B[7:0] pixels 5*n+4 | Five Taps 8 bits (Medium) |
| 10 | CL1_B[1:0],CL1_A[7:0] 5*n+0<br>CL1_B[5:4],CL1_C[7:0] 5*n+1<br>CL2_C[1:0],CL2_B[7:0] 5*n+2<br>CL2_C[4:5],CL2_A[7:0] 5*n+3<br>CL3_C[1:0],CL3_B[7:0] 5*n+4 | Five Taps 10 bits (Full) |
| 11 | CL1_A[7:0] R even<br>CL1_B[7:0] G even<br>CL1_C[7:0] B even<br>CL2_A[7:0] R odd<br>CL2_B[7:0] G odd<br>CL2_C[7:0] B odd | Six Taps 8 bits (Color) (Medium) |
| 12 | CL1_B[1:0],CL1_A[7:0] R even<br>CL1_B[5:4],CL1_C[7:0] G even<br>CL2_C[1:0],CL2_B[7:0] B even<br>CL2_C[4:5],CL2_A[7:0] R odd<br>CL3_C[1:0],CL3_B[7:0] G odd<br>CL3_C[5:4],CL3_A[7:0] B odd | Six Taps 10 bits (Color) (Full) |
| 13 | CL1_A[7:0] 8*n+0<br>CL1_B[7:0] 8*n+1<br>CL1_C[7:0] 8*n+2<br>CL2_A[7:0] 8*n+3<br>CL2_B[7:0] 8*n+4<br>CL2_C[7:0] 8*n+5<br>CL3_A[7:0] 8*n+6<br>CL3_B[7:0] 8*n+7 | Eight Taps 8 bits (Full) |
| 14 | CL1_B[1:0],CL1_A[7:0] 8*n+0<br>CL1_B[5:4],CL1_C[7:0] 8*n+1<br>CL2_B[1:0],CL2_A[7:0] 8*n+2<br>CL2_B[5:4],CL2_C[7:0] 8*n+3<br>CL3_B[1:0],CL3_A[7:0] 8*n+4<br>CL3_B[5:4],CL3_C[7:0] 8*n+5<br>CL2_FVAL,CL2_LVAL,CL2_SP,CL1_SP,<br>CL1_B[7:6],CL1_B[3:2] 8*n+6<br>CL3_FVAL,CL3_LVAL,CL1_DVAL,CL3_SP,<br>CL2_B[7:6],CL2_B[3:2] 8*n+7 | Eight Taps 10 bits (Full) |
| 254 | NA | Use Serial Port |
| 255 | NA | Use USB |

Other modes are possible with custom FPGA configurations purchased from FastVision or developed in-house.

## *6. CAMERA DATA FLOW*




DDR Memory can be organized as a FIFO or as a circular buffer. CC3 is used to clear the memory (FIFO) or stop filling the memory (circular buffer). CC2 is used to read the top most images from memory (FIFO) or to read the oldest non-read image from memory (Circular buffer).

The LUT converts the input pixels (10 Bits) to the output pixel size (8 or 10 Bits). These tables can also be used to improve the linearity of the sensor, or to set the gamma of the camera.

DDR Memory can be used for image averaging. In this mode Images are summed into a 32 bit per pixel buffers, which can be read out at any time (CC2) and/or cleared (CC3).

CC1,CC2, and CC3 can be replaced by Serial Commands, or can be always enabled or disabled, for free running operation.

# 7. THE STANDARD CAMERA FUNCTIONALITY

The 'Standard' Camera is a set of FPGAs designed to support most of the typical uses of the camera. It is a good starting point for modifications. This design is copyrighted IP from FastVision and forms the reference design, which is available from FastVision for use only with the FastCamera13.

# 8. PIXEL GAIN AND OFFSET

The offset and gain of each pixel can be calibrated by setting the pixel gain and offset arrays in the DDR memory of the camera. It is a good idea to do this as the large number of converters, need to be calibrated for best results. Note: only the pixels in the selected ROI are processed (see serial commands 0x02 and 0x03).

The offset table is set by taking several images with no light input, averaging them and uploading them to the offset (or dark field) table. These values are subtracted from the sensor pixel values (clipped so less than zero values are zero), before applying the gain table values. The gain table values are obtained by taking a defocused image of a uniform object, with enough light to get to 75% of saturation at most on the brightest point in the image. Using the resulting image gain values are computed (in the simplest case the pixel values and the ratio with 75% of full scale (191 (8 bit) or 768 (10 bits)) determines the gain.)

***Note:*** It is assumed that you are trying for a uniformly lit field of view (i.e. a Flat Field). It is important that you have as nearly as is practical, a uniform field of light, as large or small gain values can introduce significant artifacts in your images.

The gain table contains 8 bit values, which are formatted unsigned 1.7 format, that is one binary digit followed by 7 fractional digits. This makes the range of gains 1.992 down to 0.00390625.

The equation for the pixel gain and offset is:

CP[i,j] = Saturate((2*Gain[i,j]/256)* Clip (P[i,j] – Offset[i,j]))

CP = Corrected Pixel

P = Raw sensor Pixel

Clip = 0 if P[i,j]-Offset[i,j] is negative, otherwise its P[i,j]-Offset[i,j].

Saturate = FullScale if its input is greater than or equal to FullScale, other wise it is it's input alue.

FullScale = 255 for 8 bit pixels 1023 for 10 bit pixels.

# 9. MEMORY OPTION

The camera has 120 MB (or more) memory, which can be used to store images, and do averaging. Each mode of operation is explained in the following sections. If the memory option is enabled, then sensor data goes to memory instead of down stream.

## 9.1. FIFO memory mode

In this mode memory is used as a first in first out memory (FIFO). The memory fills with images, until it is full, and then stops filling. At any time the user may request an image from the FIFO with CC2, or via a serial command. This will make room for a new image, which will be filled as soon as one is collected. This mode is typically used by systems that can accept images in bursts, but can not accept a continuous stream of images. For example your system might take 8 images very fast and then read them out at a slower rate (via a single camera link for example), your average frame rate is slow, but your peak rate is high. If the host activates CC2 (or sends the serial command), it can cause the next image to be read out when it comes in (pre-trigger the readout).

### 9.2. Circular buffer memory mode

In this mode the memory is used as a circular buffer. Each time an image is presented to the memory it is stored over-writing the oldest image in the buffer. When CC2 goes active (or via a serial command), the filling operation stops, after a selected number of frames are added to the memory buffer (delayed trigger). After the memory image collection stops, each time CC2 toggles (or by serial command) the oldest remaining image is sent to the host. After the first CC2 is sent the host may send the next CC2 right away it does not have to wait for the filling to finish. Each image will be sent to the host as it becomes available. After the memory is full the camera will not accept any more images until it is read out or reset via CC3 (or serial command). If the host is draining images as fast as they are coming in the image capture will never stop. A reset (CC3) is required to re-arm the trigger and delay process.

### 9.3. Image summing memory mode

In this mode the memory is divided up into buffers the size of the selected ROI, but with 32 bit values. Each time an image is added to memory, it is added to the current average buffer. When a programmed count of images is exceeded, the system advances to the next buffer. When all the buffers are full, the system stops until it is read out (CC2) or reset (CC3). Note: as each buffer is read out (CC2) it is reset. If the host reads out the buffers faster than they are collected, image collection will not stop. For example you can program the camera to total 10 images in each buffer, before passing it to the host. The host triggers the readout via CC2 (or serial command).The CC2 trigger may be sent at any time, the camera will provide a total buffer when it becomes available.

## 10. LOOKUP TABLE OPTION

If the lookup table option is enabled, the image data from up stream is passed through a lookup table. This applies a point transformation to each pixel value, producing 10 bit results, from the 8 or 10 bit input. 32 Bit totals bi-pass the block. (Or better you should not enable this block if you are doing image totaling, as it will only operate on the lower 10 bits of the total.)

## 11. DATA FORMAT FUNNEL

The data format funnel takes the image data and parcels it out the camera links (or USB) as selected by the Format Funnel mode. If 8 bit data is selected the upper 8 bits of each pixel is sent. (Note if you don't like that use the LUT to change this behavior).

32 Bit data total buffers are sent as 8 bit data in little endian (that is least significant byte first).

## 12. INTERNAL CAMERA MEMORY

The camera contains at least 128 MB of internal memory which can be expanded to 1000 MB, which can be used to FIFO the input images, to allow burst exposure, and slow readout operation. Only the image data in the selected ROI is stored. A serial command or the positive edge of CC3 can be used to clear FIFO memory.

In addition the Camera contains LUTs for conversion from 8 to 10 bit data which can be uploaded via the serial port.

## 13. SENSOR CONTROL

## 13.1. Line Timing

Sensor line and frame rates are controlled by the Camera State settings. The minimum line period depends on the sensor type and the ROI width. When the Line Period setting exceeds the minimum period, extra clocks are inserted between lines. The Line Valid period only depends on the ROI width and sensor type. The number of columns read from the sensor is always a multiple of 10 for the MV13 and 16 for the MV40 sensor. Since the Data FPGA also has access to the ROI settings it is possible for the actual ROI width (as sent to the framegrabber) to start and end on any pixel, however this is not implemented. The Control FPGA ensures that the starting and ending pixels of the ROI are read from the sensor. For the MV13 this means that as many as 9 pixels before and 9 pixels after the ROI could be read from the sensor, depending on the starting and stopping pixel values modulo 10. For the MV40 sensor as many as 15 pre- and post-ROI pixels could be read from the sensor, depending on the starting and stopping pixel values modulo 16. The software GUI is responsible for determining the actual number of pixels per line based on the sensor and readout mode.

Camera-Link or USB readout rate can be the limiting factor for the line rate in non-memory modes. In this case it is the GUI's responsibility to maintain the Line Period setting large enough to prevent FIFO overrun in the Data FPGA.

## 13.2. Frame Timing

In free-running mode the minimum frame period depends on the ROI height and the line period. When either the frame period or exposure time setting exceeds the minimum period, extra clocks are inserted between frames. A 32-bit frame time counter allows frame periods from the minimum up to 64 seconds in increments of 15 nS.

In triggered modes the frame period is also affected by the exposure delay. In multiple trigger mode the exposure delay only affects the time until the first readout of the trigger sequence and subsequent frames run at the free-running rate as specified. In single-trigger mode with external exposure control, applying the trigger faster than the maximum rate will cause the camera to skip triggers and run at a submultiple of the trigger frequency. In multiple trigger mode and internal timed trigger mode, applying the trigger faster than the maximum rate will result in the camera running at the specified free-run rate.

Trigger pipelining is provided so that a trigger event which occurs during exposure or readout (just during readout in external exposure mode) will generate another frame.

## 13.3. Exposure Modes

The MV13 sensor is normally used in TrueSNAP shutter mode. The sensor has analog frame storage and a transfer gate that allows exposure to overlap readout while exposing all lines together. The MV40 sensor does not have analog storage and can only be exposed in a rolling shutter mode. Thus the individual lines have different exposure windows which may or may not overlap depending on the line rate and exposure time settings. This is the equivalent of using the MV13 sensor with the transfer gate on all the time.

When viewing high-speed motion, the MV13 works best if the lines are read out at the maximum possible rate while additional time for exposure or frame period is inserted between frames. This maximizes the overlap in the line exposure times. Unfortunately in the non-memory readout modes with reduced output rates just the opposite must be done. Long interline delays in these modes causes excessive temporal distortion in the moving image. This can appear as stretching or shrinking of vertically moving objects or trapezoidal distortion of horizontally moving objects.

## 13.4. Trigger Modes

There are four basic trigger modes:

- Free-running mode ignores trigger inputs and reads out the sensor continuously at a programmed frame rate with programmed exposure timing.
- Multi-frame edge-triggered mode activates a programmed number of exposures after a programmed delay at a programmed frame rate with programmed exposure timing.
- Single edge-triggered mode activates a programmed exposure after a programmed delay.
- External exposure mode exposes during the active trigger period.

Free-running mode allows continuous readout at rates ranging from about one frame per minute to the maximum capability of the sensor. Exposure can be programmed in increments of 15 nS up to the frame period. There are gaps in the allowable exposure times due to interaction between the readout circuitry and pixel reset in the MV13 sensor. In the MV40 sensor the exposure time is even more restricted due to the rolling shutter mode of operation. The Control FPGA will generate the exposure timing as close as possible to the requested value.

Multi-frame edge-triggered mode effectively starts the camera into free-running mode for a programmed number of frames after a delay of zero to about 64 seconds programmable in 15 nS increments. The active edge of the trigger activates the time delay and the first exposure starts after this delay. The camera can be re-triggered during active readout. If re-trigger event occurs too soon to start the subsequent exposure after the programmed delay, the next exposure will start as soon as possible thereafter. This prevents the effect of running at submultiples of the trigger rate when the trigger rate exceeds the capability of the camera.

Single edge-triggered mode is the same as asynchronous multi-frame edge-triggered mode with the frame count set to 1.

External exposure mode starts exposure as soon as possible after the active-going edge of the trigger and stops exposure as soon as possible after the inactive-going edge of the trigger. For the MV13, this provides accurate exposure timing based on the trigger pulse width. For the MV40, limitations of the rolling shutter limit the exposure timing resolution to the line readout rate unless the exposure time exceeds the readout time. Subsequent exposures can overlap readout of the current frame in this mode. This can place further restrictions on the exposure timing resolution in both sensor types.

The MV13 camera has “synchronous” and “asynchronous” modes of exposure. These only apply to free-running and multi-frame modes. Single-frame and external modes are always “asynchronous.” When the synchronous trigger mode bit is set, the exposure timing is linked to the readout timing to avoid exposure period jitter that can occur when the readout overlaps the exposure. In multi-frame triggered mode, the synchronous mode also creates an additional delay of one readout time that must be added to the programmed delay to find the actual delay to the first exposure.

## *13.5. Trigger Options*

There are four trigger sources - Camera Link CC1, TTL trigger input, Serial, and USB (via I2C).

- The CC1 and TTL sources can be enabled or disabled. They can also be active high (positive-going edge for edge-triggering or high level for external exposure) or active low (negative-going edge for edge-triggering or low level for external exposure).
- The Serial source uses the "O" command. This cannot be disabled in the Control FPGA. It must be disabled by the GUI if desired.
- The USB source uses writes to subaddress TBD. In external exposure mode two writes are required. Writes to subaddress TBD with bit 0 high start the exposure, and writes to subaddress TBD with bit 0 low end the exposure. This cannot be disabled in the Control FPGA. It must be disabled by the GUI if desired.

## *13.6. Trigger Outputs*

There are two trigger outputs, both TTL level on the P2 power connector. One output is high whenever the sensor is exposing. The other is high during the programmed delay period in the two edge-triggered modes.

## *13.7. Reference Voltages*

All sensor reference voltages are programmable and generated by two LTC1660 octal D/A converters. At power-up the DAC's are loaded with default values hard-coded into the Control FPGA. After the Data FPGA has been loaded the DAC's are updated with the settings in the default camera state storage page of flash memory.

Comparators in the Control FPGA check for changes in the DAC settings and re-load the DAC's whenever the values are changed. This can happen as a result of the host command to set camera state, or the host command to restore state from flash.

## *13.8. Calibration*

Both the MV13 and MV40 sensors have automated ADC calibration to reduce column-wise fixed-pattern noise. This is initiated at sensor reset or by using the CAL_START_N input. The Control FPGA always uses the sensor reset to initiate calibration. In addition, the MV40 sensor allows direct access to the internal calibration values on a serial interface consisting of the DATA_CLK, DATA, RE_N, and WE_N pins. This version of the Control FPGA does not use this interface.

Automatic calibration is initiated after power-up when the sensor is released from reset. Reset is released before the Data FPGA is loaded to allow the sensor to stabilize during the load process. Reset is re-asserted for two clock cycles at the end of the initialization sequence, about TBD milliseconds after the reference voltage DAC's are updated from flash. This causes the sensor to re-calibrate with the new reference voltage settings.

After power-on initialization, the sensor is only calibrated on demand by the host using the Reset and Calibrate Sensor command. It is not automatically calibrated after changing the reference voltages. Thus the camera control GUI is responsible for periodic calibration as required.

## *13.9. Power-on Initialization*

The Control FPGA has a small embedded micro that runs through an initialization sequence at power-on. This same micro also implements the host, USB, and Data FPGA communication protocols and deals with flash memory and DAC's.

1. Immediately after power-on, the voltage reference DAC's are programmed with factory default values. This allows the sensor to stabilize under conditions close to the actual operating environment.

2. Page zero is read from the flash memory. This page holds information necessary to locate actual power-on data in the flash.

3. The sensor is released from reset and allowed to self calibrate and stabilize.

4. The Data FPGA is loaded from flash. If the offset or length stored in the header are not valid, the embedded micro skips to step 10. This may take several seconds depending on the FPGA size.

5. Camera State is read from one of the 8 pages as selected by the pointer in the header. If the pointer is not valid (1 - 8) or the selected page is uninitialized, the embedded micro skips to step 10.

6. Reference DAC's are reprogrammed with the updated values from flash.

7. Camera State is forwarded to the Data FPGA.

8. If the Pointer to the Data FPGA initialization area is valid and the length is valid and non-zero, the Data FPGA initialization is read from flash and forwarded to the Data FPGA.

9. The sensor is briefly reset to start another auto-calibration cycle.

10. The embedded micro enters the command service loop.

## *14. FLASH MEMORY*

Table 2 shows the layout of flash memory. The first page is used only for main header information. Following this are eight pages for storing up to 8 camera states. Fast-Vision should reserve one of these for factory defaults. The Control FPGA will not write protect the defaults page so the Camera Control GUI should prevent the user from overwriting the factory settings, or it should have the factory default settings available in the software to restore them without using the flash. Any of these states can be chosen as the power-up default.

| **Flash Page Layout** | | |
|---|---|---|
| **Page Offset** | **Pages** | **Description** |
| 0 | 1 | Flash Memory Header Page |
| 1 | 1 | Camera State Storage 1 |
| 2 | 1 | Camera State Storage 2 |
| 3 | 1 | Camera State Storage 3 |
| 4 | 1 | Camera State Storage 4 |
| 5 | 1 | Camera State Storage 5 |
| 6 | 1 | Camera State Storage 6 |
| 7 | 1 | Camera State Storage 7 |
| 8 | 1 | Camera State Storage 8 |
| 9 | 1370 | FPGA bitstream (sized for 2V1000 + 2V250) |
| 1379 | 2716 | Available for Data FPGA Initialization and User Data |
| 4095 | 1 | Reserved by Atmel |

**Table 2 - Flash Memory Layout**

Table 3 shows the layout of the first page in flash memory. This contains enough descriptors for the Camera Control GUI to determine the camera type and its options. This data should only be programmed by Fast-Vision. The camera GUI software must enforce this, as there is no write protection built into the Control FPGA firmware. Multibyte values are little endian.

| Page Zero Flash Header | | |
|---|---|---|
| **Byte Offset** | **Bytes** | **Description** |
| 0 | 1 | Flash Memory Part Type: 0x16 or 0x32 for 16 or 32 Mb flash |
| 1 | 1 | Flash Memory Revision ID: 0x01 for this revision |
| 2 | 2 | Offset in bytes to Camera ID data from top of page 0: 0x79 0x01 |
| 4 | 4 | Flash page offset to FPGA data: 0x09 0x00 0x00 0x00 |
| 8 | 4 | FPGA bitstream length in bytes |
| 12 | 4 | Flash page offset of Data FPGA initialization data |
| 16 | 4 | Data FPGA initialization data length in bytes |
| 20 | 1 | Camera State to load at power on (1 to 8) |
| 20 | 357 | Reserved for additional header / ID info |
| 377 | 11 | Part Number, ASCII “800??-5????” |
| 388 | 3 | Part Revision, ASCII “010” |
| 391 | 9 | Serial Number, ASCII “XXX??????” |
| 400 | 128 | FPGA bitstream file header from mkbin, Null terminated string |

**Table 3 - Flash Memory Header Page**

# *15. CAMERA STATE STORAGE*

Internal to the Control FPGA all state is saved in a Block RAM. Copies of the current state can be saved to the flash or uploaded to the host. The current state can also be retrieved from flash or changed by the host. Only the host has random access to the camera state and this only when setting state. Reading back the camera state always sends the entire state to the host. Table 4 shows the layout of the camera state memory. Except for sensor reference voltages, multibyte values are little endian.

| **Byte Offset (decimal)** | **Bytes** | **Description** |
|---|---|---|
| 0 | 4 | C3, 5A, F0, 69 for detecting uninitialized buffers |
| 4 | 2 | Vln2: 14 D9 |
| 6 | 2 | Vref1: 14 D9 |
| 8 | 2 | Vtest: 20 00 |
| 10 | 2 | Vref2: 23 E0 |
| 12 | 2 | Vbias1: 30 00 |
| 14 | 2 | Vref3: 32 E8 |
| 16 | 2 | Vbias2: 40 00 |
| 18 | 2 | Vref4: 41 36 |
| 20 | 2 | Vbias3: 50 00 |
| 22 | 2 | Vln1: 54 D9 |
| 24 | 2 | Vbias4: 60 00 |
| 26 | 2 | Vlp: 64 D9 |
| 28 | 2 | Vunused1: 70 00 |
| 30 | 2 | Vclamp3: 70 00 |
| 32 | 2 | Vunused2: 80 00 |
| 34 | 2 | Vrstpix: 8D 17 |
| 36 | 2 | ROI Start Pixel |
| 38 | 2 | ROI End Pixel |

| | | |
|---|---|---|
| 40 | 2 | ROI Start Line |
| 42 | 2 | ROI End Line |
| 44 | 2 | Line Period in Pixel Clocks |
| 46 | 4 | Exposure Time in Pixel Clocks |
| 50 | 4 | Frame Period in Pixel Clocks |
| 54 | 4 | Exposure Delay in Pixel Clocks |
| 58 | 2 | Serial Link Bit Period in Pixel Clocks |
| 60 | 1 | Camera Link Readout Mode |
| 61 | 1 | Camera Link Clock Frequency |
| 62 | 1 | Binning |
| 63 | 1 | Memory Options |
| 64 | 1 | Sensor Resolution - 8/10 bit |
| 65 | 2 | Trigger Mode ($2^{nd}$ Byte is Data FPGA dependent) |
| 67 | 1 | Frame count for Multi-Trigger mode |
| 68 | 1 | CC Mode: CC2, CC3, CC4 enable and edge select |
| 69 | 59 | Reserved for Control FPGA / base system extension |
| 128 | 384 | Available for Data FPGA functionality extensions |

**Table 4 - Camera State Memory Layout**

Sensor reference voltages are presented to the DAC's exactly as they are stored in the state. The order listed above is the recommended order, but other orders may work. The format for these is MS byte first with the most significant nibble indicating the command code to the DAC. Pairs of values are sent, one to each DAC chip with the first going to U25 and the second to U26. Voltages are updated in the order sent. Default values shown are for the FastCamera 13. For more information see the LTC1660 data sheet.

Whenever the state memory is updated from host or flash, the actual internal registers that implement the camera state change, too. Some of these are located in the Control FPGA and some in the Data FPGA. The Control FPGA forwards state data to the Data FPGA whenever it is updated.

The state memory holds all of the state variables currently defined for camera operational modes as well as some additional storage that can be defined as required for more sophisticated Data FPGAs. The Data FPGAs can count on this storage to be refreshed from flash after initial FPGA load and whenever the user restores state from one of the saved sets in flash.

In addition to the camera state storage pages, some of the flash memory is available for Data FPGA storage requirements such as pixel defect maps. The amount of flash available for this depends on the size of the Data FPGA and the size of the flash device. A pointer in the flash header in page zero indicates the starting page of the Data FPGA initialization area. Its length in pages (which may be zero) is stored in the flash header as well. Data from this initialization area is read out and sent to the data FPGA after initial FPGA load. Although the data in these pages has no predefined layout, the first page must start with the sequence 3C, A5, 0F, 96. This prevents transmission of uninitialized flash pages and serves to identify the following data as Data FPGA Initialization Data rather than Camera State.

# *16. SERIAL CONTROL INTERFACE*

## *16.1. Encoding*

Commands and response use the 7-bit ASCII code zero-extended to 8 bits. This is sometimes refered to as 8 bits, no parity or 7-bits space parity. For characters that require escape codes as listed below, the 8th bit is significant in the decoding, so only the normal character encoding needs to be escaped. For example hex 0d requires the escape sequence but hex 8d does not.

The data within a command or response packet may contain 8-bit binary data with escaping as described below. The camera requires only one stop bit for framing and sends one stop bit when responding.

#### 1.1.1.2 Baud Rate

To comply with the Camera Link specification, the default baud rate is 9600. The serial baud rate can be changed with a command on the serial link or via the USB port. To allow recovery after inadvertent setting of a higher baud rate than the host can handle, the serial link will revert to 9600 baud after 5 framing errors without receiving a valid command. The factory default startup baud rate is 9600, but the actual power-on baud rate can be changed via the user start-up settings.

#### 1.1.1.3 Command Protocol

Commands all begin with a character in the range g-z or G-Z where the case is ignored.

Subsequent bytes of the command are command-specific, but most commands use hex characters 0-9 and a-f or A-F for the subsequent bytes. For commands using the hex encoding, all arguments are full bytes. Thus an odd number of hex digits are considered a protocol error. For commands that use binary encoding, the following character sequences are required:

| **Character (hex, ASCII)** | **Sequence** |
|---|---|
| 0d, <cr> | \ <cr> |
| 5c, \ | \ \ |

All commands end in a non-escaped carriage-return, hex 0d.

When commands take arguments in hex, data must consist of pairs of hex characters representing full bytes. Within each byte the most significant nibble is sent first, but for multi-byte values the least significant byte is normally sent first. Thus when a command required a 32-bit value the string “23568912” would produce the hex value 12895623. Commands using hex arguments will ignore space characters, so “23568912” is equivalent to “23 56 89 12”. Other characters will cause the command to abort and respond with negative acknowledge at the next non-escaped carriage-return.

The intent of the protocol is to allow hand typing of commands from a terminal except for long commands like flash page write that would only be done using the Fast-Vision camera control GUI.

**Response Protocol**

Each command signals completion by sending the command character G-Z (always uppercased) followed by any command-specific response and terminated with a non-escaped carriage-return, hex 0d. For commands that respond with binary data, the carriage-return (0d) and backslash (5c) are escaped as in the command protocol.

If a command cannot be completed for any reason the response will be the character? (3F) and possibly a numeric error code, followed by a carriage-return to indicate negative acknowledge.

Here again the intent is to use ASCII only response except for long data sequences like flash page read.

**Synchronization**

The ASCII format of most commands combined with the escaped carriage-return in binary commands allows simple resynchronization after errors. Sending two carriage-returns will always reset the serial link to its default state, waiting for new command.

In addition, when the serial logic detects a framing error or protocol error, it will go into an idle state until the next non-escaped carriage-return. During the idle state any serial data is ignored. Any carriage-return which follows the idle state causes the negative acknowledge sequence to be sent.

**Timeout**

If the serial logic is in a state waiting for a command to be completed and no input is detected for 5 seconds, it will reset to the waiting for new command state without sending any response. This would typically happen after the serial link had noise from attaching or restarting the framegrabber.

*** TIMEOUT NOT IMPLEMENTED IN THIS VERSION ***

**Command Forwarding**

All commands are forwarded to the data FPGA. All responses from the data FPGA are forwarded to the host. Starting commands on both the Control and Data FPGA’s without waiting for completion may cause interleaving of response codes.

**Command Set Overview**

To simplify command processing, commands that are handled in the Control FPGA are assigned in ascending order starting with “G,” and commands for the Data FPGA are assigned in descending order starting with “Z.”

**G - Get Camera State**

This command takes no arguments. The camera responds with “G” followed by the current running state of the camera (512 bytes as 1024 hex digits) and a carriage-return. This can be used to synchronize the GUI with the camera at initial connection time. Camera state is sent as hex ASCII unlike the flash page read. This is because the camera state is a single page worth of data, but flash page read is likely to be used many times in a row to read large amounts of data such as the FPGA bitstream.

**H - Ping Camera for Life**

This command takes no arguments. The camera responds with “H” followed by eight hex digits and a carriage-return. The eight hex digits indicate the value of a free-running 32-bit frame counter. This counter indicates all frames read from the sensor since power on, not necessarily the number of frames sent to the framegrabber. The number of grabbed frames depends on the memory functions such as frame binning, FIFO and circular buffering. Successive pings can be used for a rough estimate of the sensor frame rate.

**I - Restore Camera State From Flash**

This command takes one argument. The argument can be 01 to 08 and indicates which of the 8 flash storage areas to retrieve the camera settings from. If the flash storage area is uninitialized the camera will send a negative acknowledge. Otherwise the camera responds with “I” followed by the flash storage area number and a carriage-return. The Restore Camera State From Flash command only restores user settable parameters other than the Data FPGA loads.

**J - Initialize FPGA From Flash**

This command takes no arguments. The Data FPGA is reloaded from the flash causing a complete re-initialization of the readout logic. If the FPGA fails to load for any reason, the camera will send a negative acknowledge and response code. Otherwise the camera responds with “J” followed by a carriage-return. Failure results when the DONE signal does not go high after download, or if the INIT# signal is reasserted after the bitstream has started to load. Either of these conditions indicates a bitstream error. The Control FPGA does not check the bitstream for correct format or CRC, this is done by the FPGA. To reduce the timeout length if the J command is issued when the flash is uninitialized, the bitstream bit counter only loads the low 24 bits of the bitstream length from the flash header. This still allows up to 128 megabits of bitstream, larger than the flash size. On error a response code is returned. The first byte is the reason which may be 00 01 or 02. 00 indicates bad header flash part type. 01 indicates illegal starting page address. 02 indicates load was attempted but did not complete. If the code is 02, six additional bytes are sent indicating the number of bytes remaining when the operation aborted. This is usually zero, unless the FPGA re-asserted the INIT line.

**K - Save Current Camera State to Flash**

This command takes one argument. The argument can be 01 to 08 and indicates which of the 8 flash storage areas to store the camera settings in. The camera responds with “K” followed by the flash storage area number and a carriage-return. This command only saves user settable parameters other than the Data FPGA loads (512 bytes total).

**L - Write Flash**

This command takes arguments that form the precise command to send to the flash memory. Data length is limited by the flash page buffer size. There is no equivalent write command to “Continuous Array Read”. The intent of the command is to allow uploading the FPGA code and to write to flash areas that need to be initialized at the factory. The data is not interpreted by the serial control logic. It is possible to use this command to directly upload camera settings to the flash pages without changing the current camera state.

This is the only Control FPGA command that uses binary encoded arguments (with escape codes for <cr> and \). The exact syntax of the Write Flash command is:

‘L’ <opcode> <addr> <data> <cr>

where opcode is one byte, addr is 3 bytes, and data can be from zero to 528 bytes depending on the command. The following opcodes are acceptable to use with this command (there is no error checking so make sure these are the only codes used).

| | |
|---|---|
| 0x84 | Buffer 1 write |
| 0x87 | Buffer 2 write |
| 0x83 | Buffer 1 to main memory program with built-in erase |
| 0x86 | Buffer 2 to main memory program with built-in erase |
| 0x88 | Buffer 1 to main memory program without built-in erase |
| 0x89 | Buffer 2 to main memory program without built-in erase |
| 0x81 | Page Erase |
| 0x50 | Block Erase |
| 0x82 | Main memory page program through buffer 1 |

0x85 Main memory page program through buffer 2

The following non-write commands are also implemented with Write Flash:

0x53 Main memory Page to Buffer 1 Transfer
0x55 Main memory Page to Buffer 2 Transfer
0x60 Main memory Page to Buffer 1 Compare
0x61 Main memory Page to Buffer 2 Compare
0x58 Auto Page Rewrite through Buffer 1
0x59 Auto Page Rewrite through Buffer 2

The camera responds with "L" followed by a carriage-return. Internally the Control FPGA waits for the flash to become non-busy then starts the flash write operation. It does not wait for the flash to complete (become not busy) after starting the command. Subsequent flash operations check the busy state of the flash memory, allowing overlap of serial communication and flash programming. Theoretically a flash write command without initial wait could give even more overlap by allowing write to a nonbusy buffer while a previous write is ongoing, but in practice the flash page write timing is shorter than the serial transmission time to send the command. At 115,200 baud it takes about 46 mS to send 528 bytes. The flash page erase/write cycle is only 20 mS max.

**M - Read Flash**

This command takes arguments that form the precise command to send to the flash memory followed by the command length and data length to be read in bytes. Command length includes any "don't care" bytes required by the flash command selected. Data length is limited by the flash page buffer size for most commands, but can be as large as the entire array if the Continuous Array Read flash command is used. All arguments are ASCII hex. The exact syntax of the Read Flash command is:

'M' <opcode> <addr> <cmd_len> <data_len> <cr>

where opcode is one byte (2 hex digits), addr is 3 bytes (6 hex digits), cmd_len is one byte (2 hex digits), and data_len is four bytes (8 hex digits). Most commands send the <opcode> and <addr> bytes to the flash and then require some number of "don't care" bytes before starting to read data from the flash. The only exception is the status read command which sends only the command byte and has no additional turn-around delay. When sending the status read command the <addr> bytes are "don't care", but must be included in the command arguments. The following opcodes are acceptable to use with this command (there is no error checking so make sure these are the only codes used).

| Opcode | Command | Command Length |
|---|---|---|
| D4 | Buffer 1 read | 5 |
| D6 | Buffer 2 read | 5 |
| E8 | Continuous array read | 8 |
| D2 | Main Memory Page Read | 8 |
| D7 | Status Register Read | 1 |

This is the only Control FPGA response that uses binary encoded arguments (with escape codes for <cr> and \).

The camera responds with “M” followed by escaped binary data from the flash and finally a non-escaped carriage-return. Internally the Control FPGA waits for the flash to become non-busy then starts the flash read operation. Thus the Status Register Read command is only useful to check the compare flag, since the busy flag will always be off.

**N - Set Camera State**

This command takes a two byte (4 hex digit) offset into the current camera state as defined in Table 4 and from one to to 512 bytes (2 to 1024 hex digits) of camera state data. The exact syntax of the Set Camera State command is:

‘N’ <addr> <data> <cr>

where <addr> is 4 hex digits of offset per Table 4 and <data> is 2 to 1024 hex digits (always a multiple of 2) of data which will be stored sequentially into the current camera state as shown in Table 4. Actual registers affected by the command are updated as the data comes in, except where further synchronization is required such as multibyte values and values that change only between frames. The Control FPGA always responds with “N” followed by a carriage-return even if the affected state is handled by the Data or DDR FPGAs. If the baud rate is changed, the command response happens at the original baud rate.

**O - Serial Trigger**

This command takes no arguments. This command has the same effect as pulsing the CC1 line high for the time period between receipt of the ‘O’ and receipt of the carriage-return. Although this command would normally be used only in edge-triggered mode, it can also be used in external exposure mode for long exposures (minimum exposure is limited by the serial baud rate). To support this mode, any characters between the “O” and the <cr> are ignored. Thus the total command length in characters can be used to determine the exposure time. The camera responds with “O” followed by a carriage-return.

**P - Reset and Calibrate Sensor**

This command takes no arguments. It causes the sensor’s LRST_N signal to be pulsed low for two clock cycles. The sensor is reset and performs an auto-calibration. This command is applied by the readout logic between frames. If enabled, the CC4 line will also initiate this command allowing operation without the GUI. The camera responds with “P” followed by a carriage-return.

**X - Send Initialization Data to Data FPGA**

This command takes any number of bytes (even number of hex digits) of FPGA initialization data. The exact syntax of the Send Initialization Data command is:

‘X’ <data> <cr>

where <data> is an even number of hex digits of data which can be used for any function required by the Data FPGA. This command is sent by the Control FPGA to the Data FPGA upon power-on initialization and after the ‘J’ command (Initialize FPGA from Flash). It is also possible for the host to issue this command via the Camera Link serial interface, but be aware that the Data FPGA does not issue a response.

**Y - Readout Image**

This command takes no arguments. This command is only effective in the memory readout modes. If enabled, the CC2 line will also initiate this command allowing operation without the GUI. The Data FPGA will respond with “Y” followed by a carriage-return.

**Z - Reset Memory**

This command takes no arguments. This command is only effective in the memory readout modes. If enabled, the CC3 line will also initiate this command allowing operation without the GUI. The Data FPGA will respond with “Z” followed by a carriage-return.

# *17. INTER-FPGA COMMUNICATION*

### Serial Commands

Commands from the host are buffered and passed to the Data FPGA on the FPGA_CTL3 wire. The Data FPGA therefore receives all commands from the host and can act on them accordingly. This allows extensions to be made to the command set for Data FPGA use. To reduce logic in the Data FPGA, each character is buffered and retransmitted at 66 MHz, synchronous to the FPGA_SYSCLK. In this way the data FPGA doesn't need to know about baud rate and can start up monitoring commands even before it receives the camera state data.

### Flash Data

In addition to forwarding commands from the host, the Control FPGA also uses the FPGA_CTL3 wire to send camera state data when the Restore Camera State From Flash command is received. It also uses the FPGA_CTL3 wire to send camera state data and Data FPGA initialization data from flash to the Data FPGA when the Initialize FPGA From Flash command is received. The Control FPGA only responds to the host after it has completed the required data transfer(s). The host must always wait until it receives the response before sending another command.

When transmitting camera state data to the Data FPGA, the Control FPGA uses the same syntax as the Set Camera State command with the starting address set to zero and a data length of 512.

When transmitting Data FPGA initialization data to the Data FPGA, the Control FPGA uses the same syntax as the Set Camera State command with the starting address set to all ones (65,535) and a data length as set in the Page Zero Flash Header. If the header indicates a zero data length, or the initialization data does not start with 3C A5 0F 96, the Control FPGA will not send this command.

### Serial Responses

Normally the data sent to the Data FPGA from the Control FPGA is limited in bandwidth at the source, either by the host serial baud rate or the flash memory read rate. Response data from the Data FPGA, which always gets forwarded to the host, must be sent at the host baud rate. To reduce the redundant logic requirement in the Data FPGA, the Control FPGA uses the FPGA_DDRCLK wire to send a baud rate pulse train. This signal is high for one cycle of FPGA_SYSCLK once per serial link bit period. The Data FPGA then uses this as a shift enable for its response transmitter.

Serial response data from the Data FPGA is forwarded to the host directly by ANDing with the serial response data from the Control FPGA. For this reason it is especially important that the host does not send a new command before the previous command response has been received.

## *18. EMBEDDED SOFT PROCESSOR CORE*

The Control FPGA includes a "PicoBlaze" 8-bit controller core, programmed in assembly language that takes care of most low-speed complex operations. This soft processor handles serial protocol, flash memory, FPGA configuration, and power-on initialization. The architecture of this core is Harvard, using a 1K by 18-bit wide instruction memory composed of 5 block RAMs, and a 1K by 8-bit wide data memory composed of four block RAMs. The core was designed and optimized to run in the Virtex 2 series of Xilinx FPGA's and then adapted for the Spartan 2. It therefore is not very fast. In the MV13 it runs at 33 MHz and in the MV40 it runs at 25 MHz (1/2 the pixel clock rate). The instructions run in a fixed 2-cycle period with essentially no pipelining. This simplifies instruction sequence timing calculations.

The PicoBlaze data address space is only 256 bytes. If you look at the processor documents from Xilinx you'll find this called I/O rather than data. In our case the data space connects to 1K bytes of internal block RAM using three additional banking bits implemented as register. Access to the entire 1K of RAM as well as 8 registers is accomplished with some stunt logic. Registers normally appear at addresses 0xf8 through 0xff regardless of the state of the three banking bits. When register access is enabled, writes are shadowed in the block RAM at 0x6f8 through 0x6ff regardless of the state of the three banking bits. Register access can be disabled by reading or writing location 0xf7 and re-enabled by reading or writing locations 0 through 0xf6. In this manner a program can sequentially access the entire RAM by temporarily disabling register access. A special bit, bit 0 of register 0xff toggles each time any read is performed. This allows a simple test of the current state of the register access enable by reading location 0xff twice and comparing the two values.

The high 256 bytes, from 0x700 through 0x7ff, are dual-ported with the I2C slave unit. This provides a simple interface for introducing commands via USB.

**"Fget8" Speed-up logic**

Because the PicoBlaze runs relatively slow, there is a small sequencer to speed up FPGA configuration and flash data read. This is kicked off by a write to register 0xfe with bit 6 set. The sequencer runs the flash clock for 8 cycles, and if the fpga CCLK was high, copies the flash DOUT output to the FPGA configuration DIN input cycling the FPGA CCLK 8 cycles as well. The PicoBlaze is responsible for making sure the flash is in the appropriate state when this sequencer is started. Generally it only speeds up the inner loop of FPGA configuration and flash data read, but this is where the largest time is spent. Using the algorithm from previous PicoBlaze designs (Video Combiners) and bit wiggling, the best data rate for FPGA download would be about 1.5 MHz for CCLK. With the "Fget8" speed-up logic the rate should be about 9 MHz. This allows even large FPGA's to be loaded in under 2 seconds.

**PicoBlaze Memory Map**

| Address | Description |
|---|---|
| 000 - 1FF | Camera state storage. Sensor registers shadow locations in 000 - 0FF |
| 200 - 3FF | Header storage. Loaded from flash page 00 at power-up. |
| 400 - 6F7 | Working memory for host commands. |
| 6F8 - 6FF | Register shadow memory for readback of otherwise write-only bits. |
| 700 - 7FF | $I^2C$ shared area. |

**PicoBlaze Register Map**

All of the following registers can only be accessed when the register access enable signal is set. This is accomplished by making any access to locations 00 through 0xf6. Any access to location 0xf7 clears the register access enable signal. Accesses within the 0xf8 through 0xff register area do not affect the state of the register access enable signal.

**0xF8 Baud Rate Period, low 8 bits and Frame Count.**

**0xF9 Baud Rate Period, high 8 bits and Fget8 data.**

The Baud rate is derived by dividing the pixel clock frequency by the number entered here. In the MV13, the pixel clock runs at 66.667 MHz and the default setting for 9600 baud would be 6944. The last value written to the Baud Rate high 8-bits register can be read back from scratchpad location 0x2f9 when register access is disabled. When register access is enabled, reading 0xF9 returns the byte of flash data from the "Fget8" sequencer. Reading 0xF8 returns the frame count in four successive reads. Frame count is latched whenever address 0x00 is read.

**0xFA Sensor Control.** Bits are:

- **7** Read-only "Y" pending. Indicates that a CC2 event has occurred and the PicoBlaze should forward it to the Data FPGA as a "Y" command.
- **6** Read-only "Z" pending. Indicates that a CC3 event has occurred and the PicoBlaze should forward it to the Data FPGA as a "Z" command.
- **5** Reserved.

**4** Serial trigger. Must be toggled in software to trigger the sensor control logic in response to a serial trigger command.

**3** Reserved.

**2** Calibrate sensor. This is a pulsed signal that schedules a calibration when this bit is written to 1. Writing this bit to zero has no effect. Actual calibration may happen much later, since the sensor control logic schedules calibration only between frames.

**1** Sensor standby. Setting this bit to one places the sensor in low-power standby mode.

**0** Dark offset enable. Setting this bit allows the sensor to apply internal calibration factors to reduce fixed column noise.

**0xFB Address Extensions.** Bits 3 through 7 are reserved. Other bits are:

**2,1,0** Banking bits. These bits form the two high address bits to the 1K byte data RAM, except during register access. During register access the high bits are fixed at 6, thus register writes are always shadowed in the 7$^{th}$ of 8 banks, allowing readback of registers that don't have separate read functionality from the shadow RAM. When bank 0 is selected, sensor register write is enabled.

**0xFC Serial to FPGA and Serial Status.** When written, this register causes a byte of data to be transmitted to the Data FPGA. On reads Bits 0 through 6 are reserved. Other bits are:

**7** Transmitter to FPGA ready for data.

**6** Transmitter to FPGA overrun error. Set if write was attempted when the transmitter wasn't ready for data. Cleared when the next data is written when the transmitter is ready for data.

**0xFD UART to/from Camera Link.** Writing this location sends a byte of serial data to the framegrabber. Reading gets a byte of serial data and acknowledges its receipt. See the status bit descriptions below for more information.

**0xFE DAC and Flash Control / Flash and UART Status.** Bits are:

**7** Reserved on write, RxData Available on read. When this bit is 1, there is data available to be read from the UART via register 0xFD. The validity of the current data depends on the error bits listed below. This and other receive status bits are cleared when the UART is read.

**6** "Fget8" on write, Transmitter Ready on read. Writing this bit to one starts the sequencer for speeding up flash read and FPGA configuration write. Writing this bit to zero has no effect. When 1 this bit indicates that the UART is ready to accept a byte of data to transmit to the framegrabber.

**5** DAC chip select on write, Receiver framing error on read. DAC chip select is asserted high. This bit is inverted before driving the chip's active low pin. Receiver framing error indicates that a zero was detected in the stop bit position. This condition is cleared when the UART is read.

**4** DAC serial clock on write, Receiver overrun error on read. DAC serial clock runs directly to the DAC chips. Receiver overrun indicates that a new character came in when the previous character had not been read. In this case the older data is lost and the data in the UART is the one causing the overrun, i.e. the most recent character received. This condition is cleared when the UART is read.

**3** Flash chip select on write, Transmit overrun on read. Flash chip select is asserted high. This bit is inverted before driving the chip's active low pin. Transmit overrun is set if a write was attempted when the transmitter wasn't ready for data. This condition is cleared when data is written to the UART and the transmitter is ready for data.

**2** Flash reset on write, transmitter empty on read. Flash reset is asserted high. This bit is inverted before driving the chip's active low pin. Transmitter empty is active when no data transmission is pending. This must be checked before changing the baud rate to ensure any pending characters finish transmitting at the current baud rate.

**1** Flash serial clock on write, Flash ready on read. These correspond directly to the flash pin functions.

**0** Shared Flash and DAC serial data in on write, Flash data out on read. These correspond directly to the respective pin functions.

**0xFF FPGA Configuration.** Bit 6 is reserved. Other bits are:

**7** Read only "Fget8" status. This is high when the sequencer is still running from a prior operation.

**5** FPGA Loaded on write, FPGA Done on read. Writing this bit high indicates that the program has finished loading the data FPGA and enables the pinsaver muxes. FPGA Done is the configuration pin function.

**4** FPGA M1 on write, FPGA INIT on read. M1 is the configuration mode bit. Other mode bits are pulled up. Writing 1 sets the Data FPGA into slave serial mode for configuration load by the Control FPGA. Writing 0 sets the Data FPGA into JTAG mode for configuration by external Xilinx Parallel Cable. FPGA Init is inverted from the active low configuration /INIT pin.

**3** Read only FPGA Data Out. This allows readback functions to be implemented in the future.

**2** Write only FPGA Program. This bit is inverted before driving the chip's active low pin. Setting this bit to 1 resets the Data FPGA configuration.

**1** FPGA Configuration Clock on write, Configuration Timeout on read. Writing this bit directly affects the FPGA CCLK pin except when the "Fget8" sequencer is running. In that case this bit enables Flash to FPGA data transfer if it is 1. Running "Fget8" while this bit is 0 will not cause any changes to the FPGA DIN or CCLK signals. Configuration Timeout goes high if there have been 16,777,216 CCLK clocks since Program was last asserted. This can be used as a simple timeout mechanism for a program that only checks the Done and Init status during FPGA configuration.

**0** FPGA Data In on write, Access Test Bit on read. During FPGA configuration this is the serial data to the Data FPGA. After configuration it could be used as an additional control line to the Data FPGA. The Access Test Bit toggles on any port read. Thus reading this register twice in a row will always provide different data. This can be used to test whether the register access enable signal is active.

**Sensor Register Map**

All of the following registers can only be accessed when the sensor register write enable signal is set. This is bit 2 of register 0xfb. Writing these registers will also cause the values to be shadowed in the scratchpad RAM at whatever page is currently indicated by bits 1 and 0 of register 0xfb. Thus although the registers themselves are write-only, there is a copy in the scratchpad RAM immediately after writing them. The scratchpad copy can obviously be overwritten without affecting the sensor registers if the sensor register write enable bit is off. The layout of these registers intentionally matches the flash state storage.

**0x24 Start Pixel, low 8 bits.**

**0x25 Start Pixel, high 8 bits.** This is the leftmost pixel in the ROI. Pixels are numbered from 0 to the sensor width - 1. This is different from the old ROI settings which worked in pixel clocks. The sensor control logic handles the conversion from pixels to clocks. If the starting pixel is not a multiple of the sensor readout width (10 for MV13 or 16 for MV40) there will be some prescan pixels in the resultant image. In the future we may want to add logic in the data FPGA to mask out the prescan pixels.

**0x26 End Pixel, low 8 bits.**

**0x27 End Pixel, high 8 bits.** This is the rightmost pixel in the ROI. Pixels are numbered from 0 to the sensor width - 1. This is different from the old ROI settings which worked in pixel clocks. The sensor control logic handles the conversion from pixels to clocks. If the ending pixel is not one less than a multiple of the sensor readout width (10 for MV13 or 16 for MV40) there will be some postscan pixels in the resultant image. In the future we may want to add logic in the data FPGA to mask out the postscan pixels.

**0x28 Start Line, low 8 bits.**

**0x29 Start Line, high 8 bits.** This is the top line in the ROI. Lines are numbered from 0 to the sensor height - 1. Only lines in the ROI are scanned.

**0x2A End Line, low 8 bits.**

**0x2B End Line, high 8 bits.** This is the bottom line in the ROI. Lines are numbered from 0 to the sensor height - 1. Only lines in the ROI are scanned.

**0x2C Line Period, low 8 bits.**

**0x2D Line Period, high 8 bits.** This is the line readout time in pixel clocks - 1. i.e. the actual period is one clock more than this number.

**0x2E Exposure Period, bits <7:0>**

**0x2F Exposure Period, bits <15:8>**

**0x30 Exposure Period, bits <23:16>**

**0x31 Exposure Period, bits <31:24>** This is the desired exposure time in pixel clocks. Actual exposure time may vary, especially if readout overlaps exposure.

**0x32 Frame Period, bits <7:0>**

**0x33 Frame Period, bits <15:8>**

**0x34 Frame Period, bits <23:16>**

**0x35 Frame Period, bits <31:24>** This is the desired frame readout period in pixel clocks - 1. i.e. the actual period is one clock more than this number.

**0x36 Delay Period, bits <7:0>**

**0x37 Delay Period, bits <15:8>**

**0x38 Delay Period, bits <23:16>**

**0x39 Delay Period, bits <31:24>** This is the desired delay from trigger to exposure in edge-triggered modes. It is not used in free-running or external exposure modes. Actual exposure delay may vary, especially if readout overlaps exposure.

**0x41 Trigger Mode.** Bit 7 is reserved. Other bits are:

**6** Invert TTL Trigger. Setting this bit to 1 indicates the TTL trigger input on P2 is active low level, or falling edge.

**5** Enable TTL trigger. Set this bit to 1 to enable TTL trigger input on P2. Clear to 0 to disable TTL trigger input.

**4** Synchronous Exposure. This bit only affects free-run and multi-frame edge-triggered modes. When this bit is set, the readout timing is synchronized to the exposure.

**3** Invert Camera Link CC1 Trigger. Setting this bit to 1 indicates the CC1 trigger input on Camera Link is active low level, or falling edge.

**2** Enable Camera Link CC1 trigger. Set this bit to 1 to enable CC1 trigger input on Camera Link. Clear to 0 to disable CC1 trigger input.

**1:0** Trigger Mode.

- **0** Free-running mode.
- **1** Multi-frame edge-triggered mode.
- **2** Single edge-triggered mode.
- **3** External exposure mode.

**0x43 Trigger Count.** Total number of frames to capture in multi-frame edge-triggered mode. 0 to 255 program 0 to 255 frames.

**0x44 CC Mode.** Enable and active edge of CC2, CC3, and CC4 command inputs. Bits 7, 6 are reserved. Other bits are:

**5** Invert CC4. Setting this bit to 1 indicates the CC4 input is active on falling edge. Setting this bit to 0 indicates the CC4 input is active on rising edge.

**4** Enable Camera Link CC4. Set this bit to 1 to enable CC4 Sensor Calibration. When enabled, the active edge of CC4 causes the sensor to be reset and calibrated at the next appropriate time.

**3** Invert CC3. Setting this bit to 1 indicates the CC3 input is active on falling edge. Setting this bit to 0 indicates the CC3 input is active on rising edge.

**2** Enable Camera Link CC3. Set this bit to 1 to enable CC3 Reset Memory. When enabled, the active edge of CC3 causes the memory to be reset.

**1** Invert CC2. Setting this bit to 1 indicates the CC2 input is active on falling edge. Setting this bit to 0 indicates the CC2 input is active on rising edge.

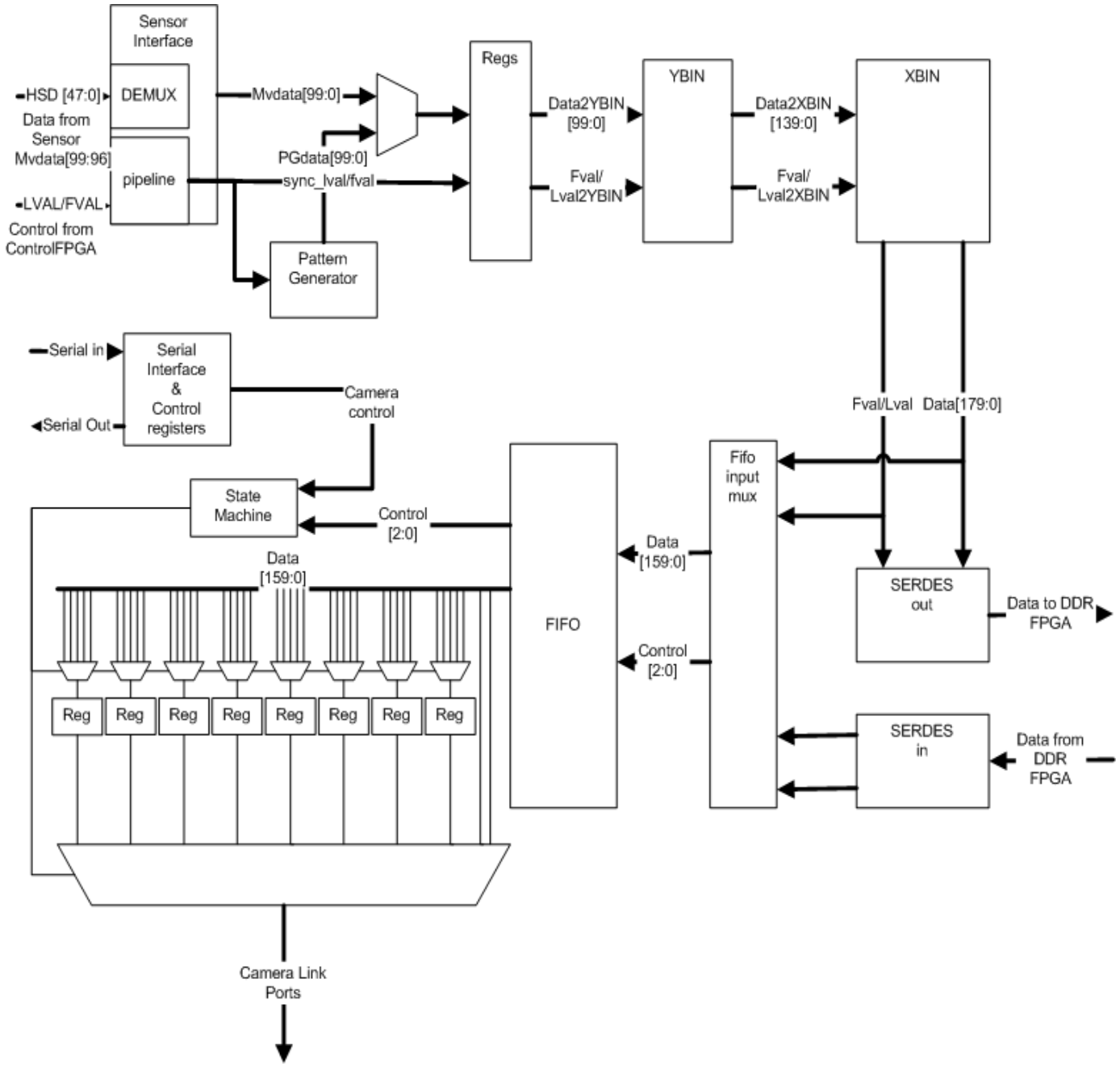
**0** Enable Camera Link CC2. Set this bit to 1 to enable CC2 Readout Image. When enabled, the active edge of CC2 causes the current memory image to be read out.

# *19. DATA FPGA*

This specification contains an overview of the functionality and design details for the Data FPGA within the FastCamera 13/40. The Data FPGA accepts the video data from the Sensor and sends it out to the host over the Camera Link and/or USB Interfaces.

## *19.1. Functional Description*

The system level block diagram for the FPGA is shown below.


Sensor Interface
DEMUX
pipeline
HSD [47:0]
Data from Sensor Mvdata[99:96]
LVAL/FVAL
Control from ControlFPGA
Mvdata[99:0]
PGdata[99:0]
sync_lval/fval
Pattern Generator
Regs
Data2YBIN [99:0]
Fval/ Lval2YBIN
YBIN
Data2XBIN [139:0]
Fval/ Lval2XBIN
XBIN
Serial in
Serial Out
Serial Interface & Control registers
Camera control
Fval/Lval
Data[179:0]
State Machine
Control [2:0]
Data [159:0]
FIFO
Fifo input mux
Data [159:0]
Control [2:0]
SERDES out
Data to DDR FPGA
SERDES in
Data from DDR FPGA
Reg
Camera Link Ports

## 19.2. Video Data path overview

The DataFpga accepts video data from the sensor and formats it for transmission over the Camera Link outputs or USB interface. The ControlFPGA controls the sensor data output and sends Lval (line valid) and Fval (frame valid) signals to the DataFPGA. The data from the sensor is valid when Lval and Fval are asserted. Data progresses through to the FIFO (or output to the DDR FPGA in memory mode) at the pixel rate of the sensor and is the width of the sensor data (100 or 160 bits wide).

The video data from the sensor is sent to the DataFPGA over the HSD[47:0] and Mvdata[99:96] signal lines. The MV13 sensor clocks out ten 10bit pixels (100 bits total) each clock (at 66Mhz) and the MV40 sensor clocks out sixteen 10bit pixels (160 bits total) each clock (60Mhz).These data lines are multiplexed to the DataFPGA over the HSD (high speed data) signal lines at 2x the pixel rate in order to reduce the I/O count needed by this FPGA. The DataFPGA demultiplexes this data and synchronizes it with the non multiplexed data from the sensor and the control signals from the ControlFPGA.

The video data is then sent into the vertical binning module. This module accumulates the programmed number of rows of video data and then forwards the binned row of data to the horizontal binning module. If vertical binning is active then the frame height is reduced by the binning factor. The vertical binner produces 140 bits of video data as ten 14bit pixels. The horizontal binning module adds together the programmed number of pixels and then forwards the reduced row of data to the FIFO input mux. The horizontal binner produces 180 bits of data as ten 18bit pixels. The FIFO input mux selects either the upper or lower 16bits from each 18 bit pixel based upon the binning multiplier mode selection (also known as Sensor Resolution). The video data is then stored in a large FIFO along with the state of Lval/Fval for each block of ten (or 16) pixels.

The section of the FPGA that reads the video data from the FIFO operates at the specified Camera link clock rate. This rate can be selected to be 33, 42.5, 66 or 85Mhz. The output state machine controls the reading of the FIFO and outputting the data to the camera links based upon the selected number of Camera Link taps and the pixel width selected. There are a multitude of Camera Link modes supported by this camera, from basic one tap mode up to ten tap 80bit mode.

The output stage over the Camera Link does not use the Dval signal as defined within the Camera Link specification. This is done to insure compatibility with a variety of framegrabbers. In order to do this, the output state machine waits until enough data has been accumulated in the FIFO in order to not under-run the FIFO while the line of video data is being output. This is only a concern at higher output modes like 8 to 10 tap modes running at 66 or 85Mhz Camera Link output clock speeds. This does not affect the maximum frame rate achievable by this camera, it just insures that once Lval is asserted out the Camera Link the whole line of data will be output without gaps.

## 19.3. Operating mode control overview

The operating state of the DataFPGA is set by the ControlFPGA or by the host via the serial interface. The ControlFPGA can load stored camera configurations from the system flash memory at powerup or when directed to load different configurations by the host.

## 19.4. Camera State

Internal to the ControlFPGA all the camera state is saved in a Block RAM. Copies of the current state can be saved to the flash or uploaded to the host. The current state can also be retrieved from flash or changed by the host. Only the host has random access to the camera state and this only when setting state. Reading back the camera state always sends the entire state to the host. Table 4 shows the layout of the camera state memory. Except for sensor reference voltages, multibyte values are little endian. The values shown in bold apply to the DataFPGA and the values shown in grey apply to the ControlFPGA.

| Byte Offset (decimal) | Bytes | Description |
|---|---|---|
| 0 | 4 | C3, 5A, F0, 69 for detecting uninitialized buffers |
| 4 | 2 | Vln2: 14 D9 |
| 6 | 2 | Vref1: 14 D9 |
| 8 | 2 | Vtest: 20 00 |
| 10 | 2 | Vref2: 23 E0 |

| | | |
|---|---|---|
| 12 | 2 | Vbias1: 30 00 |
| 14 | 2 | Vref3: 32 E8 |
| 16 | 2 | Vbias2: 40 00 |
| 18 | 2 | Vref4: 41 36 |
| 20 | 2 | Vbias3: 50 00 |
| 22 | 2 | Vln1: 54 D9 |
| 24 | 2 | Vbias4: 60 00 |
| 26 | 2 | Vlp: 64 D9 |
| 28 | 2 | Vunused1: 70 00 |
| 30 | 2 | Vclamp3: 70 00 |
| 32 | 2 | Vunused2: 80 00 |
| 34 | 2 | Vrstpix: 8D 17 |
| 36 | 2 | ROI Start Pixel |
| 38 | 2 | ROI End Pixel |
| 40 | 2 | ROI Start Line |
| 42 | 2 | ROI End Line |
| 44 | 2 | **Line Period in Pixel Clocks** |
| 46 | 4 | Exposure Time in Pixel Clocks |
| 50 | 4 | Frame Period in Pixel Clocks |
| 54 | 4 | Exposure Delay in Pixel Clocks |
| 58 | 2 | Serial Link Bit Period in Pixel Clocks |
| 60 | 1 | **Camera Link Readout Mode** |
| 61 | 1 | **Camera Link Clock Frequency** |
| 62 | 1 | **Binning control (horizontal and vertical binning multipliers)** |
| 63 | 1 | **Memory Options** |
| 64 | 1 | **Binning Multiplier** |
| 65 | 2 | Trigger Mode (2nd Byte is Data FPGA dependent) |
| 67 | 1 | Frame count for Multi-Trigger mode |
| 68 | 1 | CC Mode: CC2, CC3, CC4 enable and edge select |
| 69 | 59 | Reserved for Control FPGA / base system extension |
| 128 | 384 | **Available for Data FPGA functionality extensions** |

**Table 4 - Camera State Memory Layout**

Whenever the state memory is updated from host or flash, the actual internal registers that implement the camera state change, too. Some of these are located in the Control FPGA and some in the Data FPGA. The Control FPGA forwards state data to the Data FPGA whenever it is updated.

## 19.5. DataFpga State controls

The following section details the modes selectable within the DataFPGA

### 19.5.1. Camera Link Readout Mode

This byte controls the operating mode of the Camera Link Interface. The table below shows the ‘ports’ assigned to each camera link connector output for the selectable modes. The Camera Link specification defines three 8bit ports A, B, C per each Camera Link channel which consists of 5 wires. In the table below CL1 is the Camera Link channel on connector J1. CL2 and CL3 are Camera Link channels on connector J2.

| Mode hex value | Camera Link Format<br>MSB ---------- LSB | Read Out Mode |
|---|---|---|

| | | |
|---|---|---|
| 0x00 | CL1_A[7:0] | Single tap 8 bits (Basic) |
| 0x01 | CL1_B[1:0],CL1_A[7:0] | Single tap 10 bits (Basic)<br>(default mode) |
| 0x02 | CL1_B[3:0],CL1_A[7:0] | Single tap 12-bits (Basic) |
| 0x03 | CL1_B[7:0],CL1_A[7:0] | Single tap 16-bits (Basic) |
| 0x04 | CL1_A[7:0] even pixels (0,2,...)<br>CL1_B[7:0] odd pixels (1,3,...) | Two Taps 8 bits (Basic) |
| 0x05 | CL1_B[1:0],CL1_A[7:0] even<br>CL1_B[5:4],CL1_C[7:0] odd | Two Taps 10 bits (Basic) |
| 0x06 | CL1_B[3:0],CL1_A[7:0] even<br>CL1_B[7:4],CL1_C[7:0] odd | Two Taps 12 bits (Basic) |
| 0x0A | CL1_A[7:0] pixels 4*n+0<br>CL1_B[7:0] pixels 4*n+1<br>CL1_C[7:0] pixels 4*n+2<br>CL2_A[7:0] pixels 4*n+3 | Four Taps 8 bits (Medium) |
| 0x0B | CL1_B[1:0],CL1_A[7:0] 4*n+0<br>CL1_B[5:4],CL1_C[7:0] 4*n+1<br>CL2_C[1:0],CL2_B[7:0] 4*n+2<br>CL2_C[4:5],CL2_A[7:0] 4*n+3 | Four Taps 10 bits (Medium) |
| 0x0C | CL1_B[3:0],CL1_A[7:0] 4*n+0<br>CL1_B[7:4],CL1_C[7:0] 4*n+1<br>CL2_C[3:0],CL2_B[7:0] 4*n+2<br>CL2_C[7:4],CL2_A[7:0] 4*n+3 | Four Taps 12 bits (Medium) |
| 0x0D | CL1_B[7:0],CL1_A[7:0] 4*n+0<br>CLA_A[7:0],CL1_C[7:0] 4*n+1<br>CL2_C[7:0],CL2_B[7:0] 4*n+2<br>CL3_B[7:0],CL3_A[7:0] 4*n+3 | Four Taps 16 bits (Medium) |
| 0x12 | CL1_A[7:0] 8*n+0<br>CL1_B[7:0] 8*n+1<br>CL1_C[7:0] 8*n+2<br>CL2_A[7:0] 8*n+3<br>CL2_B[7:0] 8*n+4<br>CL2_C[7:0] 8*n+5<br>CL3_A[7:0] 8*n+6<br>CL3_B[7:0] 8*n+7 | Eight Taps 8 bits (Full) |
| 0x13 | (pin assignment below) | Eight Taps 10 bits (Full) |
| 0x14 | (pin assignment below) | ***TEN TAPS 8 BITS*** |
| 254 | NA | Use Serial Port |
| 255 | NA | Use USB |

#### 19.5.1.1. 8 Tap 10 bitmode

This mode uses all of the RX/TX bits of the camera link interface so it needs to be described in terms of the actual RX/TX bits verses the Camera Link Specification which uses virtual Ports A-J. The port bits in the following table refer to the 8 10bit pixels A-H.

| Port Bit | Tx bit | Port Bit | Tx bit | Port Bit | Tx bit |
|---|---|---|---|---|---|
| A0 | Tx0 | C6 | Tx0 | F3 | Tx0 |
| A1 | Tx1 | C7 | Tx1 | F4 | Tx1 |

| | | | | | |
|---|---|---|---|---|---|
| A2 | Tx2 | C8 | Tx2 | F5 | Tx2 |
| A3 | Tx3 | C9 | Tx3 | F6 | Tx3 |
| A4 | Tx4 | D0 | Tx4 | F7 | Tx4 |
| A5 | Tx5 | D1 | Tx5 | F8 | Tx5 |
| A6 | Tx6 | D2 | Tx6 | F9 | Tx6 |
| A7 | Tx7 | D3 | Tx7 | G0 | Tx7 |
| A8 | Tx8 | D4 | Tx8 | G1 | Tx8 |
| A9 | Tx9 | D5 | Tx9 | G2 | Tx9 |
| B0 | Tx10 | D6 | Tx10 | G3 | Tx10 |
| B1 | Tx11 | D7 | Tx11 | G4 | Tx11 |
| B2 | Tx12 | D8 | Tx12 | G5 | Tx12 |
| B3 | Tx13 | D9 | Tx13 | G6 | Tx13 |
| B4 | Tx14 | E0 | Tx14 | G7 | Tx14 |
| B5 | Tx15 | E1 | Tx15 | G8 | Tx15 |
| B6 | Tx16 | E2 | Tx16 | G9 | Tx16 |
| B7 | Tx17 | E3 | Tx17 | H0 | Tx17 |
| B8 | Tx18 | E4 | Tx18 | H1 | Tx18 |
| B9 | Tx19 | E5 | Tx19 | H2 | Tx19 |
| C0 | Tx20 | E6 | Tx20 | H3 | Tx20 |
| C1 | Tx21 | E7 | Tx21 | H4 | Tx21 |
| C2 | Tx22 | E8 | Tx22 | H5 | Tx22 |
| C3 | Tx23 | E9 | Tx23 | H6 | Tx23 |
| LVAL | Tx24 | F0 | Tx24 | H7 | Tx24 |
| FVAL | Tx25 | F1 | Tx25 | H8 | Tx25 |
| C4 | Tx26 | F2 | Tx26 | H9 | Tx26 |
| C5 | Tx27 | LVAL | Tx27 | LVAL | Tx27 |

#### 19.5.1.2. 10 Tap 8 bit mode

This mode (Basler A501k mode) uses all of the RX/TX bits of the camera link interface so it needs to be described in terms of the actual RX/TX bits verses the Camera Link Specification which uses virtual Ports A-J. The port bits in the following table refer to the 10 8bit pixels A-K.

| Port Bit | Tx bit | Port Bit | Tx bit | Port Bit | Tx bit |
|---|---|---|---|---|---|
| A0 | Tx0 | D2 | Tx0 | G5 | Tx0 |
| A1 | Tx1 | D3 | Tx1 | G6 | Tx1 |
| A2 | Tx2 | D4 | Tx2 | G7 | Tx2 |
| A3 | Tx3 | D5 | Tx3 | H0 | Tx3 |
| A4 | Tx4 | D6 | Tx4 | H1 | Tx4 |
| A5 | Tx5 | D7 | Tx5 | H2 | Tx5 |
| A6 | Tx6 | E0 | Tx6 | H3 | Tx6 |
| A7 | Tx7 | E1 | Tx7 | H4 | Tx7 |
| B0 | Tx8 | E2 | Tx8 | H5 | Tx8 |
| B1 | Tx9 | E3 | Tx9 | H6 | Tx9 |
| B2 | Tx10 | E4 | Tx10 | H7 | Tx10 |
| B3 | Tx11 | E5 | Tx11 | J0 | Tx11 |
| B4 | Tx12 | E6 | Tx12 | J1 | Tx12 |
| B5 | Tx13 | E7 | Tx13 | J2 | Tx13 |

| | | | | | |
|---|---|---|---|---|---|
| B6 | Tx14 | F0 | Tx14 | J3 | Tx14 |
| B7 | Tx15 | F1 | Tx15 | J4 | Tx15 |
| C0 | Tx16 | F2 | Tx16 | J5 | Tx16 |
| C1 | Tx17 | F3 | Tx17 | J6 | Tx17 |
| C2 | Tx18 | F4 | Tx18 | J7 | Tx18 |
| C3 | Tx19 | F5 | Tx19 | K0 | Tx19 |
| C4 | Tx20 | F6 | Tx20 | K1 | Tx20 |
| C5 | Tx21 | F7 | Tx21 | K2 | Tx21 |
| C6 | Tx22 | G0 | Tx22 | K3 | Tx22 |
| C7 | Tx23 | G1 | Tx23 | K4 | Tx23 |
| LVAL | Tx24 | G2 | Tx24 | K5 | Tx24 |
| FVAL | Tx25 | G3 | Tx25 | K6 | Tx25 |
| D0 | Tx26 | G4 | Tx26 | K7 | Tx26 |
| D1 | Tx27 | LVAL | Tx27 | LVAL | Tx27 |

### 19.5.2. Camera Link Clock Frequency setting

This byte controls the Camera link output clock rate.

| FC13 Camera Link Clock Frequency | |
|---|---|
| 0x00 | 33MHz |
| 0x01 | 42.5MHz |
| 0x02 | 66MHz |
| 0x03 | 85MHz |

| FC40 Camera Link Clock Frequency | |
|---|---|
| 0x00 | 30MHz |
| 0x01 | 42.5MHz |
| 0x02 | 60MHz |
| 0x03 | 85MHz |

### 19.5.3. Binning Control setting

This byte controls which 16 bits are selected from the 18 bit binned pixel value.

| Binning Control byte | |
|---|---|
| Bits [3:0] | Horizontal binning setting with rates selectable in the range of 1 to 14. |
| Bits [7:0] | Vertical binning setting with rates selectable in the range of 1 to 12. |

### 19.5.4. Binning Multiplier setting

This byte controls which 16 bits are selected from the 18 bit binned pixel value.

| Binning Multiplier byte | |
|---|---|
| 0x00 | Bits [15:0] selected from the 18 bit binned pixel value. |
| 0x01 | Bits [17:2] selected from the 18 bit binned pixel value. |

# *20. FRAME RATES.*

The actual Frame rate of the camera is determined by settings within the ControlFPGA and the DateFPGA. The ControlFPGA settings determine the data rate from the sensor by adjusting the ROI (Region of Interest), exposure rate and exposure delay. The DataFPGA settings determine the binning settings(which can increase frame rate as there is less data to send over the Camera Link Interface), the Camera link mode and clock speed. To get the fastest frame rate possible for a given ROI you can calculate the minimum required Line Period (in pixels clocks) by the following formula.

**Min Line Period =**

((Time_required_over_Camera_Link – Time_to_gather_data_from_sensor) / 1/66Mhz) – 132 (clocks per line)

The time required to send each row of data over the Camera Link is calculated by:

Time_required_over_Camera_Link =

(Num_Data_pixels_per_row / Num_Taps ) * 1/Selected_Clock_rate

The time required to gather each row of data from the sensor is calculated by:

Time_to_gather_data_from_sensor = 132 (clocks per row min) * 1/66Mhz = 2us

**Num_Data_pixels_per_row =**

(((# of cols in ROI * 10pixels/column) / Horiz_binning) / Vertical_binning)

So for example, when running full resolution (1280 pixels per row), no binning, in 1Tap mode at 33Mhz clock rate the calculation is:

**Time_required_over_Camera_Link – 2us / 1/66Mhz) – 132 (clocks per line)**

38.7us – 2us / 1/66Mhz = 2422 – 132 = 2290

Thus the Minimum line period needs to be set to 2290, which adds that many clocks per line read from the sensor, which results in a frame rate of 1/(1024 rows * 38.7us per row) or about 25 full resolution frames per second.

If we up the clock rate over the Camera Link to 85Mhz and switch to 8Tap mode then the limiting factor becomes the time to gather the data from the sensor:

**Time_required_over_Camera_Link = 1280/8 * 1/85Mhz = 1.88uS**

The time duration of 1.88uS to output the line is less than the time required to gather the sensor data of 2uS so we do not need to extend the line length and the Minimum Line period can be set to 0.

## 20.1. Memory Operation.

The FC13 DataFPGAa connects to the DdrFPGA via a high speed SERDES link. This link is 13 bits wide and is able to transmit 104 bits of data to the DdrFPGA at the sensor clock speed (66Mhz). This data path consists of 100 bits of data (10, 10bit pixels), 2 bits of control (Lval, Fval), and 2 spare bits.

## 20.2. DATA FPGA Technical Details.

The following diagrams show simplified state diagrams used to design the Camera Link output stage of the DataFPGA.

The data from the FIFOs are fed to 5-to-1 muxes on order to limit the fanout of the data from the FIFO. There are 8 muxs in order to support 8 Tap mode. Muxs 1,3,5,7 multiplex between pixels 1,3,5,7,9 and Muxs 2,4,6,8 multiplex between pixels 2,4,6,8,T. There are 4 mux select buses which control Muxs 1-2, 3-4, 5-6, 7-8.

The ordering of the pixels output each clock vary per the number of taps:

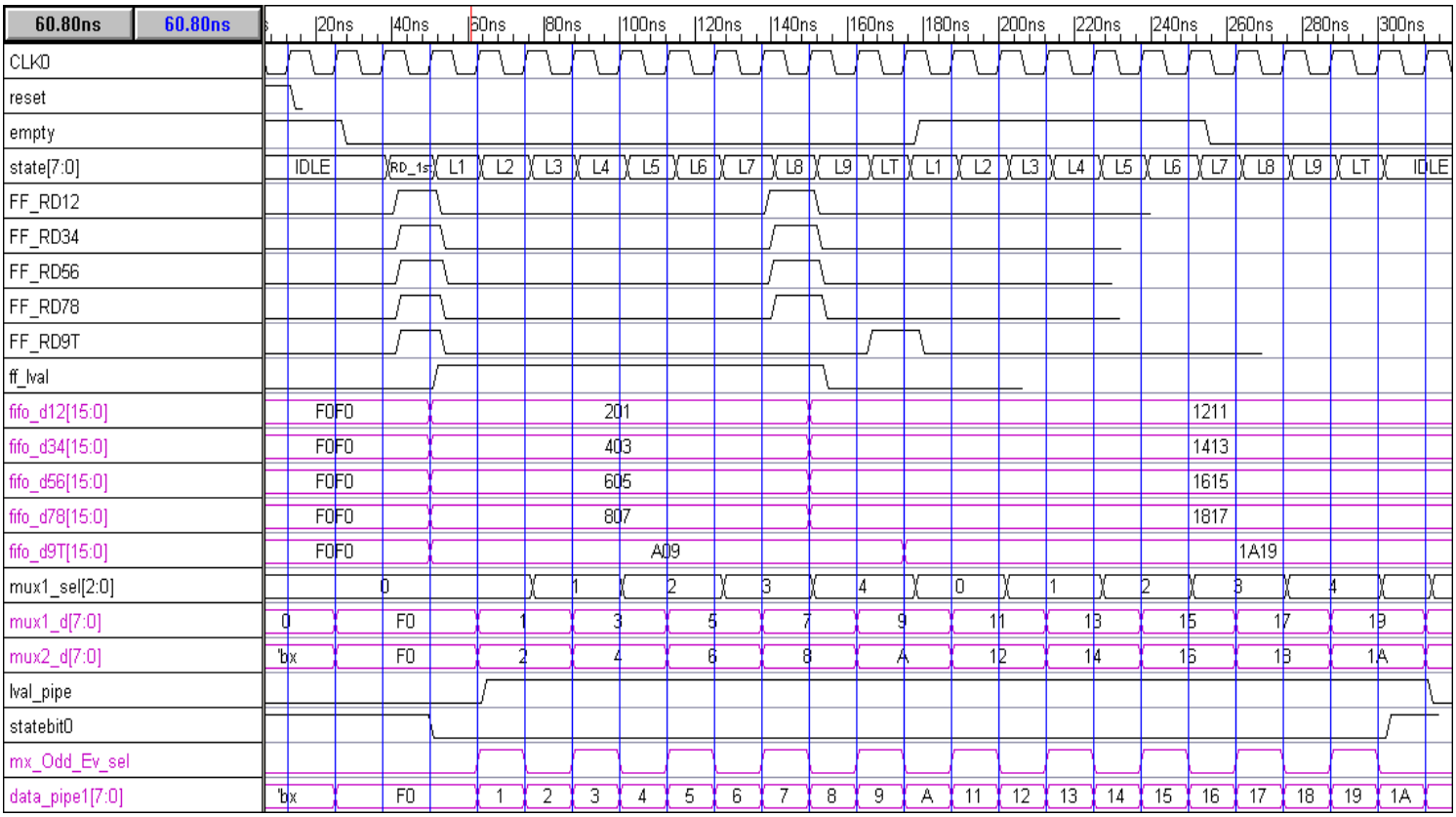
| # of Taps | Order of the pixels output at each clock cycle | Pixels read during the current clock cycle to be available for on subsequent clock cycles |
|---|---|---|
| 1 Tap | <br>1<br>2<br>3<br>4<br>5<br>6<br>7<br>8<br>9<br>T | 1 2 3 4 5 6 7 8 9 T (whole FIFO preread)<br><br><br><br><br><br><br><br>1 2 3 4 5 6 7 8<br><br>9 T |
| 2 Taps | <br>1 2<br>3 4<br>5 6<br>7 8<br>9 T | 1 2 3 4 5 6 7 8 9 T (whole FIFO preread)<br><br><br><br>1 2 3 4 5 6 7 8<br>9 T |
| 4 Taps | <br>1234<br>5678<br>9T12<br>3456<br>789T | 1 2 3 4 5 6 7 8 9 T (whole FIFO preread)<br><br>1 2 3 4 5 6 7 8<br>9 T<br>1 2 3 4 5 6<br>7 8 9 T |
| 8 Taps | <br>1 2 3 4 5 6 7 8<br>9 T 1 2 3 4 5 6<br>7 8 9 T 1 2 3 4<br>5 6 7 8 9 T 1 2<br>3 4 5 6 7 8 9 T | 1 2 3 4 5 6 7 8 9 T (whole FIFO preread)<br>1 2 3 4 5 6<br>1 2 3 4 7 8 9 T<br>1 2 5 6 7 8 9 T<br>3 4 5 6 7 8 9 T<br>1 2 3 4 5 6 7 8 9 T |
| 10 Taps | <br>1 2 3 4 5 6 7 8 9 T | 1 2 3 4 5 6 7 8 9 T (whole FIFO preread)<br>1 2 3 4 5 6 7 8 9 T |

**The following notes apply for these timing diagrams**

- CLK0 is the camera link output clock
- Empty is the signal from the FIFO that starts the state machine operation in these simplified state diagrams. The actual verilog code used a combination of Empty and other conditions to insure that the FIFO did not under-run.
- State [7:0] signal uses mnemonics to indicate the state machine states. The state machine starts in Idle, changes to RD_1$^{st}$ (first FIFO read) and then loops reading the FIFO and outputting data until the Lval signal from the FIFO deasserts.
- The FIFO was split into 5 separate FIFOs each with their own read control signals. The five FIFOs contain pixels 1-2, 3-4, 5-6, 7-8, 9-10. The data from the FIFOs was shown here as a byte per pixel, so signal FIFO_d12[15:0] represents pixel2 on bits [15:8] and pixel1 on bits [7:0]
- Signal FF_RDxx are the FIFO read signals.
- Signal ff_lval is the Lval status out of the FIFO.
- Signals FIFO_dxx[15:0] are the pixels from the FIFO.
- Mux1_sel [2:0] controls the mux selects for muxs 1, 2. Mux2_sel controls muxs 3,4,etc. Thus a mux select index of 0x0 for muxs1, 2 selects the first pixel from the two muxs which are pixels 1 and 2.
- Signals MuxX_d show the pixels selected from the mux

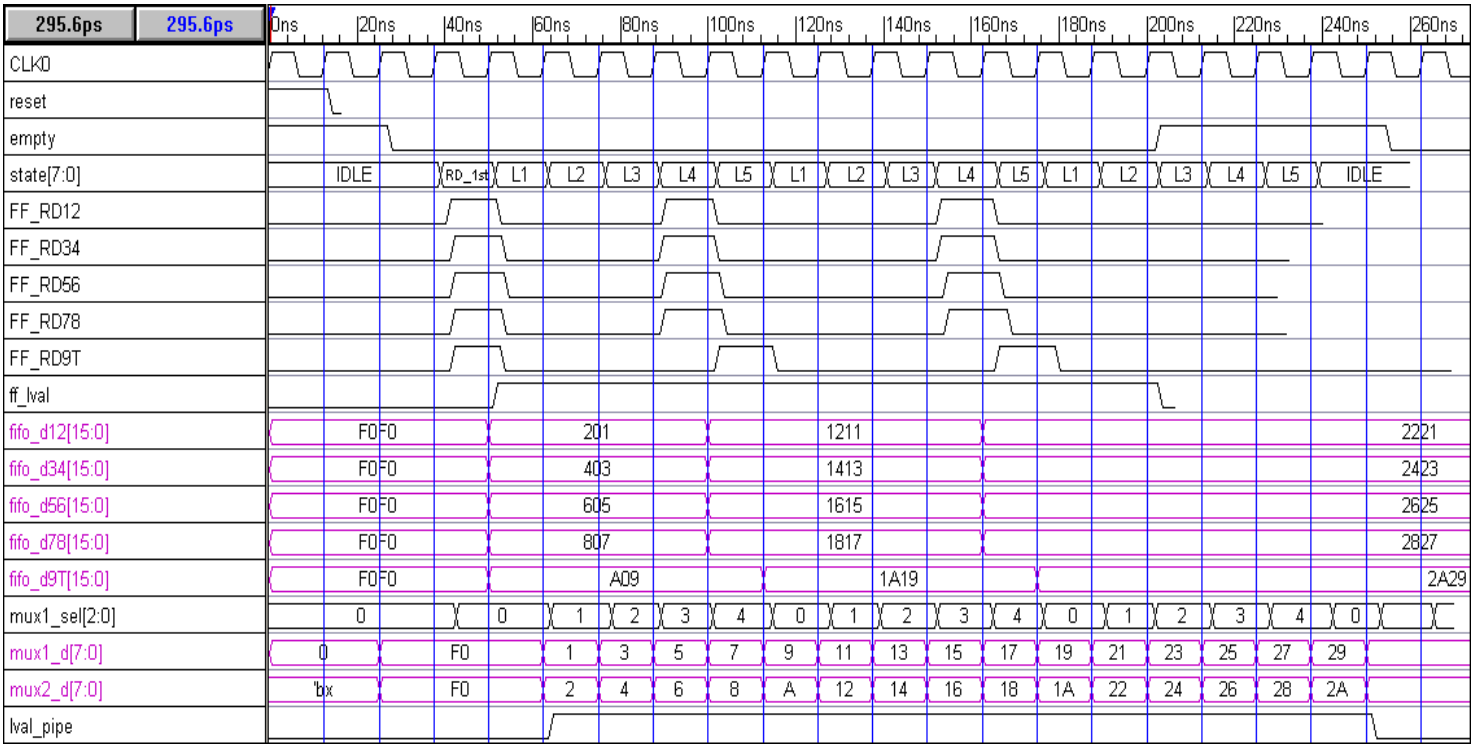
## *20.3. 1 Tap mode*

When the FIFO deasserts empty the state machine starts with a pre-read of the whole FIFO during which pixels 1-Ten are read. The state machine then continues in a 10 cycle loop until the end of the line is detected (ff_lval deasserted). The outputs from muxs 1, 2 are multiplexed again onto the single output tap (data_pipe1)



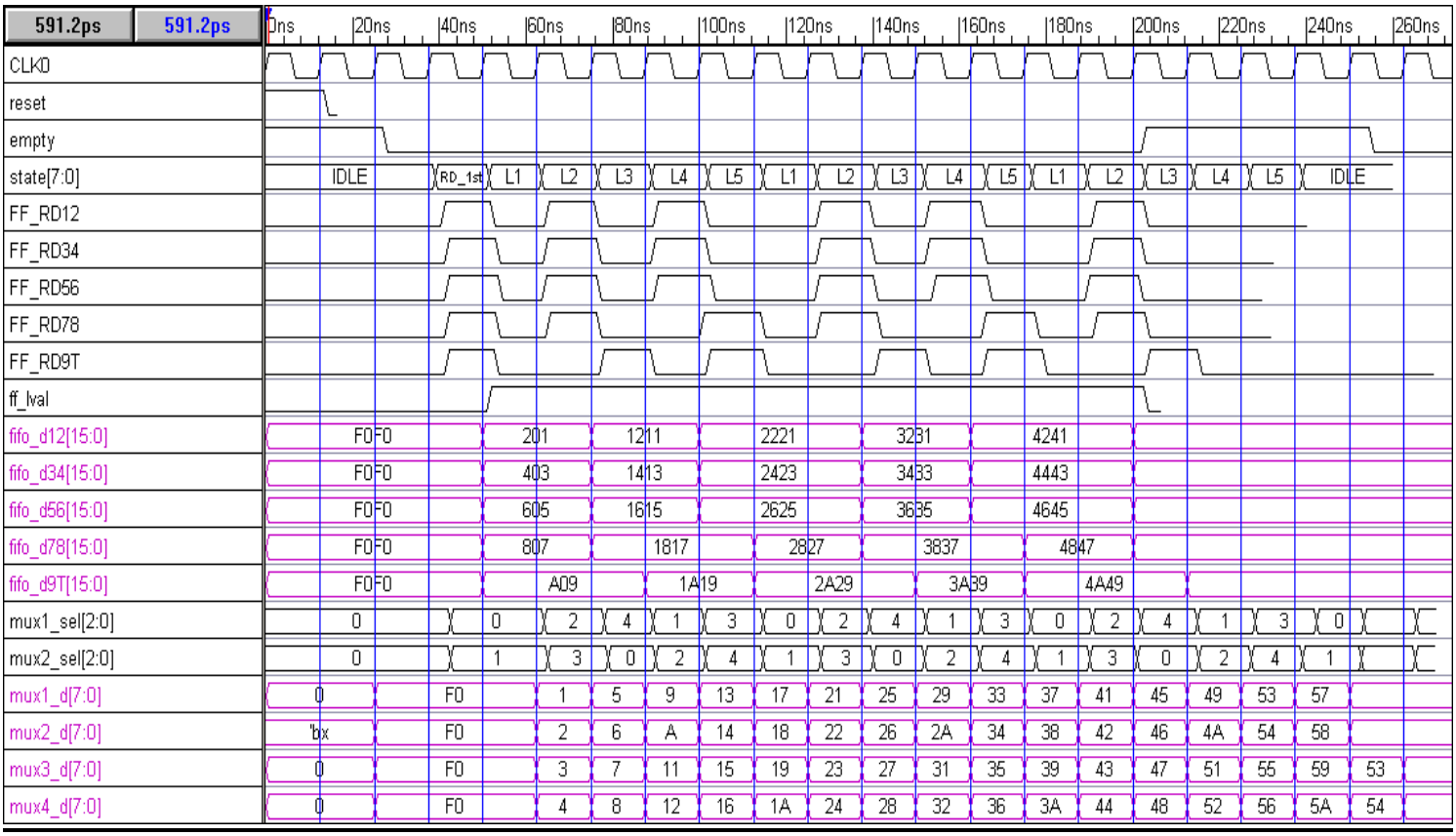

## *20.4. 2 Tap mode*

When the FIFO deasserts empty the state machine starts with a pre-read of the whole FIFO during which pixels 1-Ten are read. The state machine then continues in a 5 cycle loop until the end of the line is detected (ff_lval deasserted). The output from muxs 1, 2 drive the two taps.



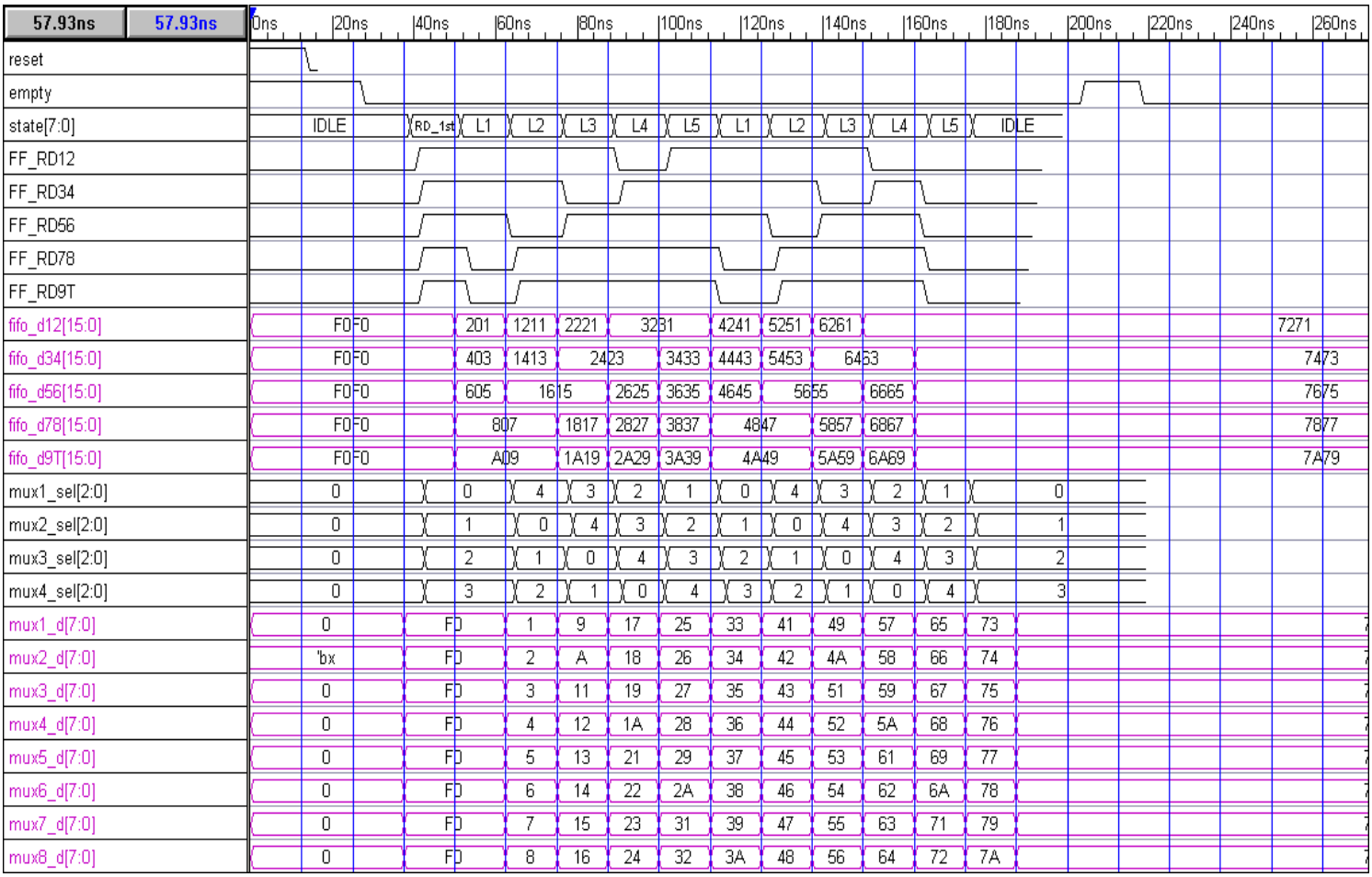

## *20.5. 4 Tap mode*

When the FIFO deasserts empty the state machine starts with a pre-read of the whole FIFO during which pixels 1-Ten are read. The state machine then continues in a 5 cycle loop until the end of the line is detected (ff_lval deasserted). The output from muxs 1,2,3,4 drives the four taps.




## *20.6. 8 Tap mode*

When the FIFO deasserts empty the state machine starts with a pre-read of the whole FIFO during which pixels 1-Ten are read. The state machine then continues in a 5 cycle loop until the end of the line is detected (ff_lval deasserted). The output from muxs 1, 2,3,4,5,6,7,8 drive the eight taps.




## *20.7. Sensor Interface*

The FPGA interface to the sensor consists of the following signals:

- mv_hsck4,2,1,0
- mv_hsck0,
- mv_hsck1,
- mv_hsck2,
- mv_hsck4,
- mv_sel0,
- mv_sel1,
- mv_sel2,
- mv_sel4,
- mv_hsd,
- mv_data

# 21. USB CAMERA OPTION

All cameras are shipped with a USB interface for control and readout of the camera. This allows the camera to be operated with or without a framegrabber, and helps diagnose system problems when a framegrabber is being used.

A camera control program comes with the camera, which uses the serial port supported by the framegrabber, it supports the standard Camera Link DLL, or may be used with the USB port in situations where the camera link serial port is not available, or too slow. In order to use this option you must be using an operating system that supports USB, and have the correct drivers installed. Currently this is supported under Windows and Linux only. The USB port may also be used to obtain image data from the camera; you may select full images or an ROI to reduce the image rate. The camera supports USB 2.0, which provides up to about 40 MBytes/sec transfer rates. You will probably get less, it depends on your computer; typically the transfer rate is more than 30 Mbytes/sec.

# 22. CAMERA CONTROL PROGRAM

The Camera Control Program provides a GUI interface that allows you to send and receive control data and images from the camera. See the manual for the camera control program.

# 23. APPLICATION ENVIRONMENTS

In this section we will discuss several operating environments to give you an idea of what you need to operate the camera with a PC.

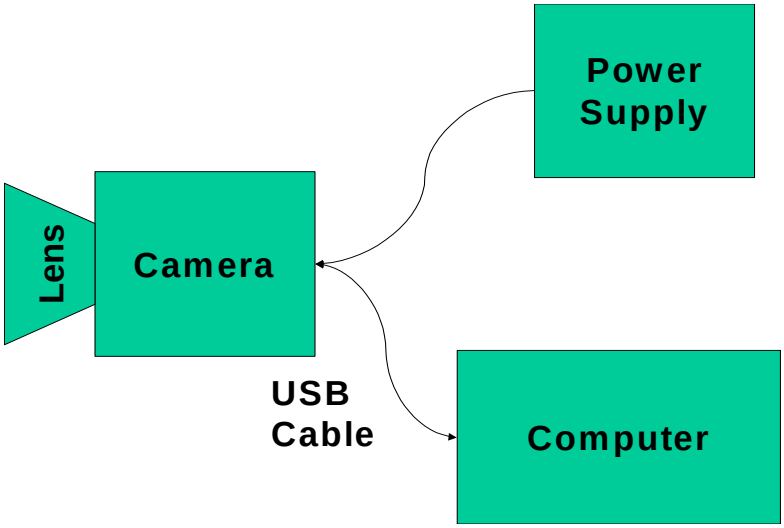
## 23.1. USB operation

In this mode of operation you will need the following items:

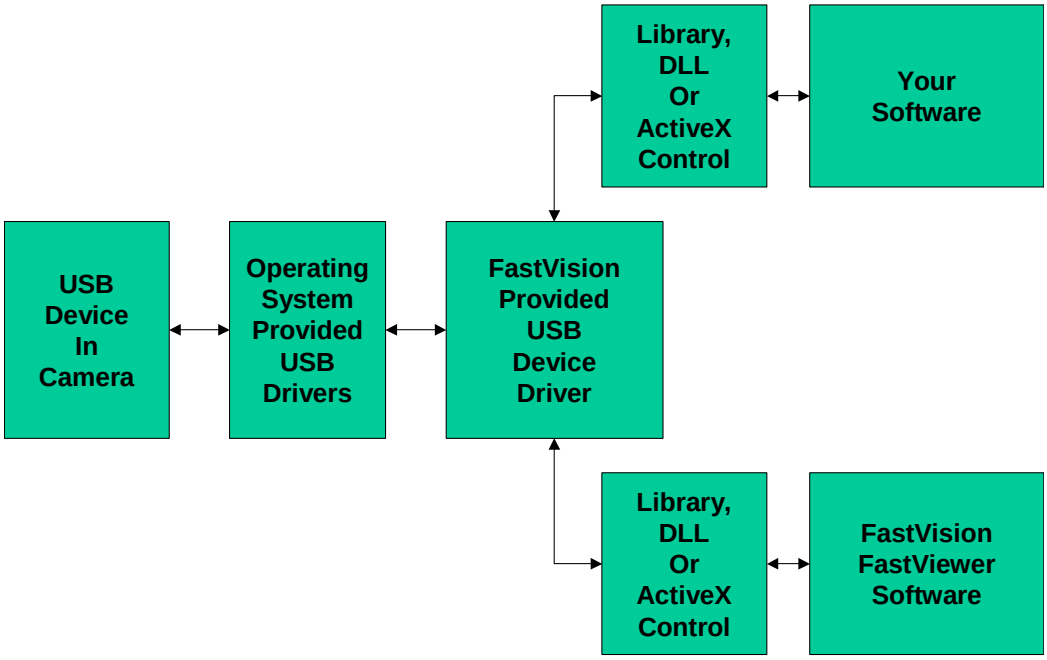
- The Camera and a lens
- A power supply for the camera.
- A USB cable
- FastVision supplied USB software for your OS.

All of the above can be supplied to you by FastVision. The software (item #4) provides two forms of access to the camera, via a library (DLL, or ActiveX control under Windows, linkable library under other operating systems), and via the supplied application called FastViewer.

## 23.2. The hardware connections are:




### *23.3. The software connections are:*

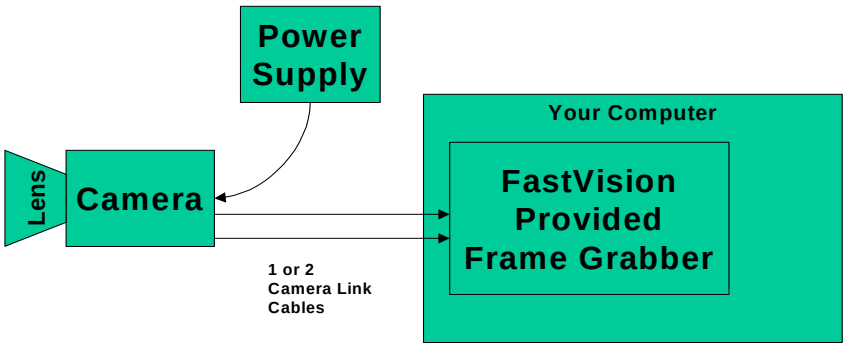



### *23.4. Using a FastVision supplied framegrabber*

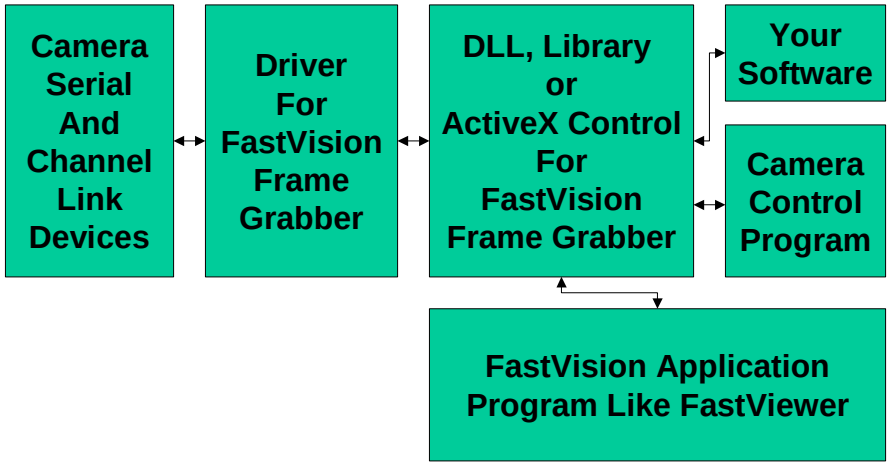
In this mode of operation you will need the following items, which can all be supplied by FastVision.

- The Camera and a lens
- A power supply for the camera
- One or more Camera Link cables
- FastVision supplied framegrabber
- FastVision supplied framegrabber software.

### 23.4.1. The hardware connections are:




### 23.4.2. The software connections are:

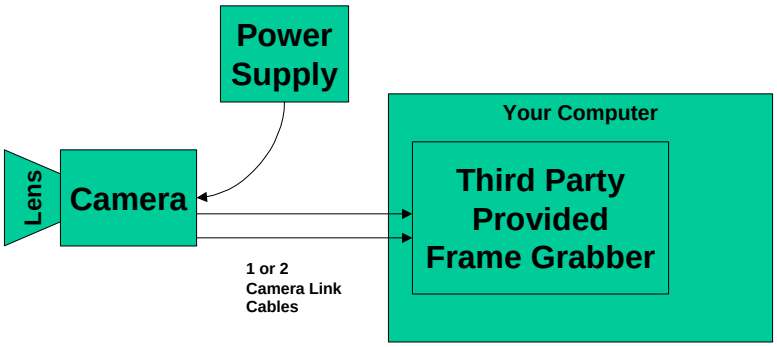


## *23.5. Using a third-party framegrabber*

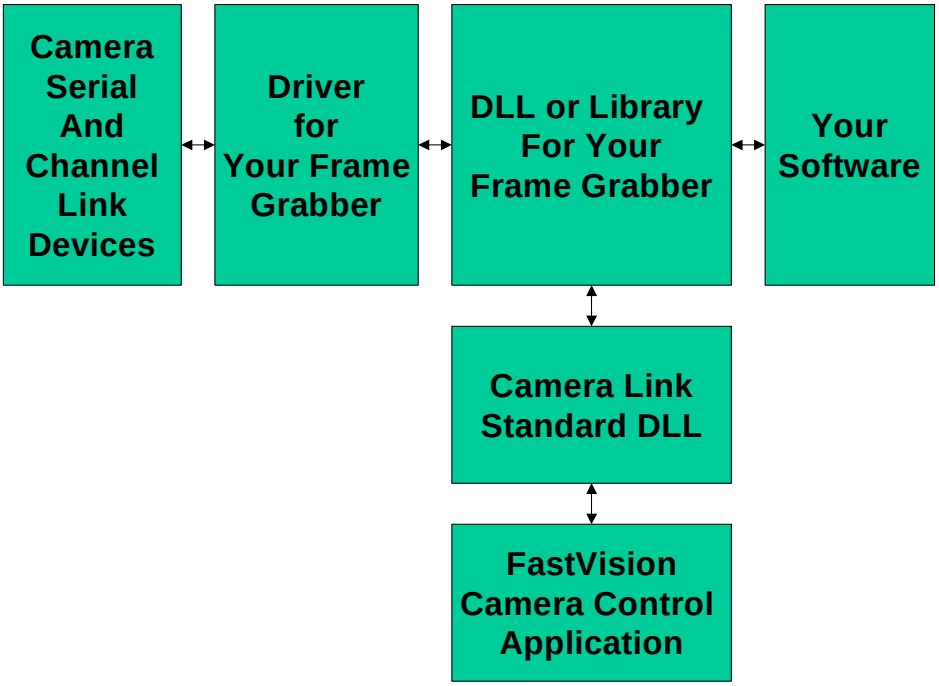
In this mode of operation you will need the following items:

- The Camera and a lens
- A power supply for the camera.
- One or more Camera Link cables.
- Your third-party framegrabber.
- AIA compliant software for your framegrabber.
- FastVision supplied camera control program.

### 23.5.1. The hardware connections are:



### 23.5.2. The software connections are:




This will only work if your framegrabber supports the Camera Link standard serial interface protocol, and your framegrabber vendor has provided the needed DLL. Otherwise you will have to send commands to the camera in the way specified by your framegrabber vendor**.**

# 24. TROUBLESHOOTING

There are several things you can try before you call FastVision 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 that the system is properly installed.

_____ 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 FastVision 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\fastvision\alinfo**

# 25. FASTVISION TECHNICAL SUPPORT

FastVision 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 camera installation and operation.

## 25.1. Contacting Technical Support

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

**Telephone: 603-891-4317**

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: 603-891-1881**

You can email a description of the problem to ***support@fast-vision.com***

## 25.2. Returning Products for Repair or Replacements

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

If, after trying everything you can do yourself, and after contacting FastVision Technical Support, you feel your hardware or software is not functioning properly, you can return the product to FastVision for service or replacement. Service or replacement may be covered by your warranty, depending upon your warranty. The first step is to call FastVision 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 a FastVision 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 FastVision. Put the camera inside its anti-static protective bags. Then pack the product(s) securely in the original shipping materials, if possible, and ship to:

**FastVision, LLC.**
**71 Spit Brook Rd, Suite 200**
**Nashua, NH 03060**
**USA**

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

Attention **RMA 90XXX**

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

## *25.3. Reporting Bugs*

We at FastVision are continually improving our products to ensure the success of your projects. In addition to ongoing improvements, every FastVision 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 FastVision.

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@fast-vision.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 FastVision address below. If the code is small enough, you can also FAX the code module to us as indicated below.

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 FastVision.

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

**FastVision, LLC.**
**131 Daniel Webster Highway #529**
**Nashua, NH 03060**
**USA**

**Telephone: 603-891-4317**
**FAX: 603-891-1881**
**Web site:**
**http://www.fast-vision.com/**
**Electronic Mail:**
**sales@fast-vision.com**
**support@fast-vision.com**