# **The Karbon-CL**

# **Hardware Reference Manual**

BitFlow, Inc. 300 Wildwood Ave Woburn, MA 01801 USA Tel: 781-932-2900 Fax: 781-933-9965 Email: support@bitflow.com Web: www.bitflow.com Revision G.1

#### © 2009 BitFlow, Inc. 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 prior written consent of BitFlow, Inc.

BitFlow, Inc. makes no implicit warranty for the use of its products and assumes no responsibility for any errors that may appear in this document, nor does it make a commitment to update the information contained herein.

BitFlow, Inc. retains the right to make changes to these specifications at any time without notice.

All trademarks are properties of their respective holders.

Revision History:

Revision	Date	Comments
F.0	2006-09-01	First Printing
G.0	2008-03-21	Synchronized with SDK 5.00
G.1	2009-01-21	Synchronized with SDK 5.20

# **Table of Contents**

#### P - Preface

Purpose KBN-P-1 Support Services KBN-P-1 Technical Support KBN-P-1 Sales Support KBN-P-1 Conventions KBN-P-2

### 1 - General Description and Architecture

The Karbon-CL family KBN-1-1 Virtual vs. Hardware KBN-1-1 The Virtual Frame Grabber (VFG) KBN-1-1 Karbon Configuration Spaces KBN-1-2 Firmware, Camera Files and Downloads KBN-1-2 Firmware, Camera Files and Downloads KBN-1-5 General Description KBN-1-6 The Full CL Virtual Frame Grabber (Full CL VGF) KBN-1-7 The Base CL Virtual Frame Grabber (Base CL VGF) KBN-1-8 The Karbon-CL Models KBN-1-11

### 2 - Acquisition and Camera Control

Introduction KBN-2-1 BitFlow's Flow-Thru Architecture KBN-2-2 Camera Specific Firmware KBN-2-6 Generation of Acquisition Windows KBN-2-8 The Horizontal Active Window, HAW KBN-2-8 The Vertical Active Window, VAW KBN-2-9 The Control Tables (CTABs) KBN-2-11 Vertical Control Table KBN-2-11 The VCTAB Functions KBN-2-12 Vertical Control Table Size KBN-2-14 Horizontal Control Table KBN-2-14 The HCTAB Functions KBN-2-16 Horizontal Control Table Size KBN-2-19 Synchronizing Acquisition, Camera, CTABs and External Signals KBN-2-20 Vertical Operations and Events KBN-2-20 Horizontal Operations and Events KBN-2-24 Acquisition Command and Status KBN-2-28 The Acquisition Bitfields KBN-2-28 Trigger Processing KBN-2-33 Encoder Processing KBN-2-34 The On-Board Signal Generator KBN-2-35

#### 3 - New Timing Generator

Introduction KBN-3-1 Components and Control KBN-3-2 Periods and Frequencies KBN-3-2 Waveform polarity KBN-3-3 Triggering KBN-3-3 Output Signals KBN-3-3 Master/Slave Control KBN-3-3 Timing KBN-3-4 NTG Control Registers KBN-3-5

#### 4 - Quadrature Encoder

Introduction KBN-4-1 Simple Encoder Mode KBN-4-1 Positive or Negative Only Acquisition KBN-4-1 Interval Mode KBN-4-2 Re-Acquisition Prevention KBN-4-2 Scan Step Mode KBN-4-2 Combining Modes KBN-4-2 Control Registers KBN-4-2 Observability KBN-4-3 Electrical Connections KBN-4-3 Understanding Stage Movement vs. Quadrature Encoder Modes KBN-4-4 CON15 Register KBN-4-6 CON16 Register KBN-4-10 CON22 Register KBN-4-12 CON51 Register KBN-4-14

### 5 - Encoder Divider

Introduction KBN-5-1 Encoder Divider Details KBN-5-2 Formula KBN-5-2 Example KBN-5-2 Restrictions KBN-5-2 PLL Locking KBN-5-3 Handling Encoder Slow Down or Stopping KBN-5-3 Encoder Divider Control Registers KBN-5-4

### 6 - System Status

Introduction KBN-6-1 FACTIVE, FCOUNT KBN-6-2 PCOUNT, LCOUNT, FENCOUNT KBN-6-3 RD\_TRIG\_DIFF/TTL/OPTO, RD\_ENC\_DIFF/TTL/OPTO KBN-6-4 TRIG\_QUALIFIED KBN-6-5 VCOUNT, HCOUNT, LINES\_TOGO KBN-6-6 FIFO\_EQ KBN-6-7 DEST\_ADD KBN-6-8

#### 7 - Camera Control Registers

Introduction KBN-7-1 Bitfield definitions KBN-7-2 Example Bitfield Definition KBN-7-2 Bitfield Definition Explanation. KBN-7-2 CON0 Register KBN-7-4 CON1 Register KBN-7-8 CON2 Register KBN-7-15 CON3 Register KBN-7-21 CON4 Register KBN-7-24 CON5 Register KBN-7-31 CON6 Register KBN-7-37 CON7 Register KBN-7-39 CON8 Register KBN-7-42 CON9 Register KBN-7-48 CON10 Register KBN-7-52 CON11 Register KBN-7-56 CON12 Register KBN-7-58 CON13 Register KBN-7-60 CON14 Register KBN-7-62 CON15 Register KBN-7-66 CON16 Register KBN-7-70 CON17 Register KBN-7-73 CON18 Register KBN-7-75 CON19 Register KBN-7-77 CON20 Register KBN-7-79 CON21 Register (Bayer Version) KBN-7-82 CON22 Register KBN-7-85 CON23 Register KBN-7-87 CON24 Register KBN-7-89 CON25 Register KBN-7-93 CON26 Register KBN-7-95 CON27 Register KBN-7-97 CON36 Register KBN-7-99 CON37 Register KBN-7-101 CON38 Register KBN-7-103 CON40 Register KBN-7-106 CON41 Register KBN-7-108 CON42 Register KBN-7-110 CON43 Register KBN-7-115 CON44 Register KBN-7-117 CON51 Register KBN-7-119

#### 8 - Karbon/Neon/Alta DMA

Introduction KBN-8-1 CON28 Register KBN-8-2 CON29 Register KBN-8-4 CON30 Register KBN-8-6 CON31 Register KBN-8-8 CON32 Register KBN-8-10 CON33 Register KBN-8-12 CON34 Register KBN-8-12 CON35 Register KBN-8-14 CON35 Register KBN-8-17 Scatter Gather DMA Instructions KBN-8-19 Destination Address KBN-8-20 Size of Transfer KBN-8-21 Next Quad Address KBN-8-22

#### 9 - Register and Memory Mapping

Introduction KBN-9-1 Memory Types KBN-9-2 Registers KBN-9-2 UART KBN-9-2 DPM KBN-9-2 CTABs KBN-9-2 Memory Map KBN-9-3 Downloading Firmware KBN-9-5 PCI Configuration Space and Model/Revision Information KBN-9-6

### 10 - Electrical Interfacing

Introduction KBN-10-1 Trigger KBN-10-2 Trigger Input Types KBN-10-2 The Optocoupled Trigger KBN-10-2 Encoder KBN-10-4 Encoder Input Types KBN-10-4 The Optocoupled Encoder KBN-10-4 General Purpose Inputs (GPIN) KBN-10-6 Introduction KBN-10-6 R64 GPIN Configuration KBN-10-6 Neon-CL GPIN Configuration KBN-10-6 Karbon-CL GPIN Configuration KBN-10-6 General Purpose Outputs (GPOUT) KBN-10-7 Introduction KBN-10-7 **GPOUT Source Options KBN-10-7** R64 GPOUT Configuration KBN-10-7 Neon-CL GPOUT Configuration KBN-10-8 Karbon-CL GPOUT Configuration KBN-10-8 GPOUT Open Collector Drivers KBN-10-8 Camera Link Controls (CCs) KBN-10-11

### **11 - Specifications**

Introduction KBN-11-1 Maximum Pixels Per Line KBN-11-2 Maximum Lines Per Frame KBN-11-3

### 12 - Mechanical

Introduction KBN-12-1 The Karbon-CL Connectors KBN-12-3 The CL Connectors KBN-12-3 The I/O Connector KBN-12-4 The Auxiliary Connectors KBN-12-4 The Jumpers and Switches KBN-12-5 Jumpers KBN-12-5 Switches KBN-12-7 The Camera Link Connector Pinouts (CL1 to CL4) KBN-12-8 I/O Connector Pinout for the KBN-PCE-4-D KBN-12-9 I/O Connector Pinout for the KBN-PCE-2-D/4-F KBN-12-11 I/O Connector Pinout for the KBN-PCE-2-F KBN-12-13

# Preface

# Chapter P

## P.1 Purpose

This Hardware Reference Manual is intended for anyone using the Karbon-CL frame grabber. The purpose of this manual is two-fold. First, this manual completely describes how the board works. Second, it is a reference manual describing in detail the functionality of all of the board's registers.

### P.1.1 Support Services

BitFlow, Inc. provides both sales and technical support for the Karbon family of products.

#### P.1.2 Technical Support

Our web site is www.bitflow.com.

Technical support is available at 781-932-2900 from 9:00 AM to 6:00 PM Eastern Standard Time, Monday through Friday.

For technical support by email (support@bitflow.com) or by FAX (781-933-9965), please include the following:

Product name Camera type and mode being used Software revision number Computer CPU type, PCI chipset, bus speed Operating system Example code (if applicable)

#### P.1.3 Sales Support

Contact your local BitFlow Sales Representative, Dealer, or Distributor for information about how BitFlow can help you solve your most demanding camera interfacing problems. Refer to the BitFlow, Inc. web site (www.bitflow.com) for a list of North American representatives and worldwide distributors.

### P.1.4 Conventions

Table P-1 shows the conventions that are used for numerical notation in this manual.

Table P-1 Base Abbreviation	ns
-----------------------------	----

Base	Designator	Example
Binary	b	1010b
Decimal	None	4223
Hexidecimal	h	12fah

Table P-2 shows the numerical abbreviations that are used in this manual.

Abbreviation	Value	Example
К	1024	256K
Μ	1048576	1M

# **General Description and Architecture**

# Chapter 1

# 1.1 The Karbon-CL family

The purpose of this chapter is to explain, at a block diagram level, how the Karbon-CL works, and what different versions are available. There are four main models in the Karbon-CL family:

KBN-PCE-CL2-D, supports two base CL cameras KBN-PCE-CL2-F, supports one base, medium, full or 10-tap CL camera KBN-PCE-CL4-D, supports four base CL camera KBN-PCE-CL4-F, supports two base, medium, full or 10-tap CL cameras

The differences between the models are explained in more detail in Section 1.4.

#### 1.1.1 Virtual vs. Hardware

It's important to understand how this manual works. Some chapters of this manual discuss the Karbon-CL as a hardware platform (this chapter is a good example). While other chapters discuss the details of the virtual frame grabbers (VFG) that this hardware platform supports. The concept of the virtual frame grabber is described below, but basically the idea is that one hardware platform can support more than one device. In the case of the Karbon-CL, these devices are frame grabbers.

Note that we are not using the word virtual here in the sense of "a software virtualization of a hardware device", these VFGs are real hardware. The reason we using "virtual" is because the term "frame grabber" has more than one meaning. It can mean the piece of hardware that you put in your computer, or it can mean the device that the your software application is controlling and getting images from. For the purposes of this manual, "virtual frame grabber" means the device that your application is interfacing to. While this might sound complicated, the implementation is simple. You plug our Karbon-CL frame grabber into your PC, and your application interacts with one or more VFGs available. Everything else is taken care of by the BitFlow drivers.

#### 1.1.2 The Virtual Frame Grabber (VFG)

The Karbon family is the first board from BitFlow that supports the concept of the virtual frame grabber (VFG). The idea behind the VFG is to separate the hardware platform (connectors, laminate, FPGAs, etc.) from the frame grabbing functionality that software applications work with. The primary reason behind this separation is that the turn around time for hardware is much longer than the turn around time for modifying virtual frame grabbers. To create a brand new virtual frame grabber, or to modify an existing one, simply requires writing new firmware or updating existing firmware. The idea of modifying a frame grabber by making changes to its firmware is not new. BitFlow has been doing this since its very first product. However, what is new about the Karbon family, is the fact the entire frame grabber is written in firmware. The only fixed hardware components are the interfaces to the outside world (e.g. the CL chips on the front end). Everything else that makes up the board, camera control, data buffering, DMA engine, etc. is written in firmware. This gives the Karbon platform incredible levels of flexibility and opens the door to unlimited customization.

To understand which models support which modes see Section 1.4. Also to see examples of the different modes, see Section 1.3. However, keep in mind, the Karbon is only limited by your imagination, if you need a mode you don't see, just let us know.

#### 1.1.3 Karbon Configuration Spaces

The four basic Karbon-CL models support from one to four VFGs. Each VFG has its own configuration space (PCI interface) and will look like a separate device to the operating system. Each VFG will be connected to one or more CL interface chips. Figure 1-1shows the block diagram of the "D" models. The D models support base two or four base CL cameras. Figure 1-1 shows the diagram of the F models, which support one or two base, medium, full or 10-tap CL cameras.



Figure 1-1 KBN-PCE-CL2-D and KBN-PCE-CL4-D



Figure 1-2 KBN-PCE-CL2-F and KBN-PCE-CL4-F

## 1.2 Firmware, Camera Files and Downloads

Note that all the devices present on a Karbon will appear in SysReg as separate Bit-Flow Boards Found. The order that the VFGs appear in SysReg is determined by the operating system and is somewhat arbitrary. However, SysReg lists the connector(s) associated with each VFG, so that a connection can be made between VFG and physical connector on the board.

Recall that, even though Karbon may appear like four frame grabbers, there is only one actual hardware platform. For this reason the firmware of the of the VFGs on one board is linked. The selection of the master VFG, determines the configurations of all of slave VFGs. For example, if you have a KBN-PCE-CL2-D and you configure the master VFG with a two-tap odd/even pixel camera, then the slave VFG will also has to be configured for a two-tap odd/even pixel camera. In all other ways, however, the two configurations do not have to match. If you have a requirement where this rule must be broken, please contact BitFlow's support department. Custom combinations of firmware are available.

If there is a mismatch between the firmware required by one VFG's camera file and the firmware required by another VFG's camera file, the master VFG's firmware will get priority. In practice this means that if you change the camera file for the master VFG, and it requires different firmware than is already on the board, new firmware will be downloaded next time you start an application. However, in the case of a slave VFG, if different firmware is required than is on the board, an error message will pop up indicating the problem. Thus all the camera files for a all the VFGs on one Karbon should all require the same firmware. That said, if you have a custom need for a particular arrangement of cameras, please let us know. We can create custom firmware to solve almost any problem.

## 1.3 General Description

The Karbon-CL is a x8 PCI Express board. It can work in any PCI Express slot that it can fit it. Usually this means an x8 or x16 slot. However, some mother boards have x4 slots with x8 connectors. The Karbon-CL will work in these slots, although performance my be somewhat reduced.

Figure 1-3 shows the block diagram of the Karbon-CL hardware platform. Recall that this platform is available in a number of configurations. The block diagrams of the current supported configurations is shown in Figure 1-3. Subsequent sections discussed the details of the various models.



#### Figure 1-3 Karbon-CL Hardware Platform

As discussed previously, the Karbon platform can support a number of different virtual frame grabber arrangements. Thus the hardware resources seen in Figure 1-3 can be utilized in a number of different ways. The Karbon-CL is offered in two major configuration: the KBN-PCE-CL4, which has four CL connectors., and the KBN-PCE-CL2, which 2 CL connectors. The final D or F in the model number determines whether the board supports base CL cameras only (i.e. the D models) or base, medium and full models (i.e. the F models). The following sections discuss these different virtual frame grabbers in more detail.

#### 1.3.1 The Full CL Virtual Frame Grabber (Full CL VGF)

Figure 1-4 illustrates the Full CL virtual frame grabber (VFG) that is support on the Karbon-CL hardware. Since the Full CL interface requires two CL connectors, the KBN-PCE-CL2-F can only support one of these virtual frame grabbers, and the KBN-PCE-CL4-F can support two of these virtual frame grabbers.



#### Figure 1-4 Full CL Virtual Frame Grabber

The Full CL VFG implements the Camera Link full configuration, i.e. it can accept a single camera putting out up to 64 bits of data. In addition, it also supports the 10 tap extension to the full CL specification. This extension allows for the acquisition of up to 80 bits on two CL connectors. Not the full VFG also supports base and medium CL cameras.

The Full CL VGF can accept input data at up to 85 Mhz.

The following paragraphs are a short description of each block.

The Camera Link Interface block implements the CL full configuration This block has the Channel Link ICs, the Camera Control drivers and the serial communication tranceivers.

The MUX block packs and assembles the data from the Camera Link block before it is pushed into the FIFO. This block re-arranges on-the-fly the data from the camera's taps so that the data is written in raster scan format in the host memory.

The FIFO block decouples the camera from the DMA engine. It is implemented with dual ported memories.

The Camera Control block handles both camera synchronization as well as external I/ O. The block contains the CTABs which are uses to synchronize acquisition with the camera, determine which pixels/lines get acquired and which do not, generate control signals to the camera and to external devices. This block also handles start/stopping acquisition based on triggers and encoders.

The PCI interface block handles host reads/writes to/from the board. These reads/ writes are used to program the board, and to control its modes. This block is also responsible for DMAing image data to the host memory (or other devices). The DMA engine uses chaining scatter-gather DMA, which can DMA a virtually unlimited amount of data to memory without using any CPU cycles.

There is an on-board UART, as required by the CL specification. There is only one UART on the Karbon platform. It can be switch between the various CL connectors for programming each camera in turn.

The IO connector block has transmitters/receivers to communicate with external industrial equipment (triggers, encoders, light strobes etc.).

CL1 and CL2 are the two Camera Link connectors. CL1 is the main Camera Link connector. It carries data bits 0-23, Camera Controls and serial data port. CL2 is the auxiliary connector. It carries data bits 24-63, for the full mode and bits 24-79 in ten-tap mode.

#### 1.3.2 The Base CL Virtual Frame Grabber (Base CL VGF)

Figure 1-5 illustrates the Full CL virtual frame grabber (VFG) that is support on the Karbon-CL hardware. Since the Base CL interface requires one CL connectors, the KBN-PCE-CL2-D can support two of these virtual frame grabbers, and the KBN-PCE-CL4-D can support four of these virtual frame grabbers.



Figure 1-5 Base CL Virtual Frame Grabber

The Base CL VFG implements the Camera Link base configuration, i.e. it can accept a single camera putting out up to 24 bits of data.

The Base CL VGF can accept input data at up to 85 Mhz.

The following paragraphs are a short description of each block.

The Camera Link Interface block implements the CL base configuration This block has the Channel Link IC, the Camera Control drivers and the serial communication tranceivers.

The MUX block packs and assembles the data from the Camera Link block before it is pushed into the FIFO. This block re-arranges on-the-fly the data from the camera's taps so that the data is written in raster scan format in the host memory.

The FIFO block decouples the camera from the DMA engine. It is implemented with dual ported memories.

The Camera Control block handles both camera synchronization as well as external I/ O. The block contains the CTABs which are uses to synchronize acquisition with the camera, determine which pixels/lines get acquired and which do not, generate control signals to the camera and to external devices. This block also handles start/stopping acquisition based on triggers and encoders.

The PCI interface block handles host reads/writes to/from the board. These reads/ writes are used to program the board, and to control its modes. This block is also responsible for DMAing image data to the host memory (or other devices). The DMA engine uses chaining scatter-gather DMA, which can DMA a virtually unlimited amount of data to memory without using any CPU cycles.

There is an on-board UART, as required by the CL specification. There is only one UART on the Karbon platform. It can be switch between the various CL connectors for programming each camera in turn.

The IO connector block has transmitters/receivers to communicate with external industrial equipment (triggers, encoders, light strobes etc.).

# 1.4 The Karbon-CL Models

There are four models in the Karbon-CL family, the KBN-PCE-CL2-D, KBN-PCE-CL2-F, KBN-PCE-CL4-D and the KBN-PCE-CL4-F. Table 1-1 illustrates the capabilities of each model.

Capability	KBN-PCE- CL2-D	KBN-PCE- CL2-F	KBN-PCE- CL4-D	KBN-PCE- CL4-F
Number of base CL cameras supported	2	1	4	2
Number of medium CL cameras supported	0	1	0	2
Number of full CL cameras supported	0	1	0	2
Number of 10 tap full CL cameras supported	0	1	0	2
Number of Virtual Frame Grabbers supported	2	1	4	2
Maximum number of input bits	80	80	160	160
Number of independent trigger inputs	2	2	4	4
Number of independent encoder inputs	2	2	4	4
Number of PCI configurations (devices)	2	1	4	1
Number of GPOUTs per VFG	4	6	2	4

#### Table 1-1 Karbon-CL Models

Note that the R64 and R64e frame grabbers from BitFlow support acquisition from two cameras in what is known as dual base mode. Dual base mode requires that the two cameras be identical, and that they be synchronized. The Karbon-CL platform, supports multiple cameras in a completely independent fashion. This means there there are no restrictions on the type or resolutions of the cameras that you connect to the same Karbon. There are also no restrictions on the synchronization between the cameras. Not only do they no need to be synchronized, they don't even need to be acquiring at the same time. As far as your application is concerned each camera connected to the Karbon-CL has it's own independent frame grabber. The Karbon-CL Models

# Acquisition and Camera Control

# Chapter 2

# 2.1 Introduction

This section covers acquisition and camera control for the R64-CL, Karbon-CL, Neon-CL and the Alta-AN.

## 2.2 BitFlow's Flow-Thru Architecture

The MUX component of the block diagrams for the Alta, Karbon, Neon and R64 is composed of a chain of sub-blocks that make up the Flow-Thru Architecture (FTA). Figure 2-1 shows the structure of the FTA for the Camera Link boards. Figure 2-2 shows the structure of the Alta family. All the data paths are 64-bit. The implementation of the individual blocks depends on the camera format, i.e. it is specific to the firmware downloaded for each sensor architecture. There is a bitfield, FORMAT, which indicates the currently downloaded firmware.

Below is a description of the individual blocks. For each block are shown the signals that are defined by the user.

Data from the Camera Link or AFE is synchronized and assigned to data lanes according to the camera format. The user has no control over these operations. From this block the data goes to a Barrel Shifter.

The Barrel Shifter is composed of four 16-bit barrel shifters. All shifters receive the same command, Left/Right and the amount of shift, up to 15 bits. The main purpose of the Barrel Shifter is for cameras that have more than 8 bits per pixel. The Barrel Shifter can down-shift the data to 8-bit suitable for display. Any camera with up to four taps can be accommodated.

There is a Video Delay Line (not shown) in the data path which can delay the video by up to 8 clocks. This is useful for accurate alignment of the video on the display.

The Video Selector selects the data source: the video from the camera or the onboard generated synthetic video. The various patterns of synthetic video are useful mainly for the on-board Built In Self Test (BIST).

The Mask is a 32-bit mask replicated over the upper and lower 32 bits of the 64-bit data path. The purpose of this mask is to be able to set to zero any bit in the data path (a one will pass the data as is, a zero will set that bit to zero).

The Clip is a clipping mechanism replicated on each one of the eight 8-bit data lanes. If enabled, it will clip the 256 gray levels in each lane according to the formula:

If video > 245 then video = 245 If video <10 then video = 10

This mechanism is useful for displaying gray level data on a VGA that is set in 256color mode. In this mode the upper and lower 10 gray levels are dedicated to the Windows graphics.

The Assembler will assemble and pack the video data before it is written in the FIFO. This block does the raster scan re-arranging of the data. The packing is dependent on the pixel depth, which is defined in the PIX\_DEPTH bitfield in CON10. The DISPLAY bit will force this block to assemble the data as 8-bit pixels, suitable for display. When using this mode, the barrel shifters must be set to down-shift each pixel by the correct amount. A 10-bit camera, for example, would need a 2-bit right shift.

The SWAP bit will swap between odd-even data streams for cameras that supply oddeven pixels. The amount of data written in the FIFOs is controlled by the Acquisition Window. The vertical and horizontal size of this window is programmed in the ALPF and the ACLP registers respectively (see Section 2.4). The timing of this window is determined by the camera and the acquisition state machine.



Figure 2-1 Flow Thru Architecture - R64, Karbon and Neon



Figure 2-2 Flow Thru Architecture - Alta

# 2.3 Camera Specific Firmware

The Flow-Thru architecture is flexible and can be adapted to different cameras architectures. The main intelligence is in the firmware that gets downloaded into the onboard Field Programmable Gate Arrays (FPGAs). This firmware is different for every camera architecture. The firmware is called out in the camera file. On initialization, the driver will download into the FPGAs the firmware called-out in the camera file.

The type of the firmware is hard-coded in the FORMAT field in register CON10. The list of the formats is shown in Table 2-1

Note: Not all models support all formats. Only formats that are possible are supported. For example, the Neon, which is Base Camera Link only, will not support the MUX\_8TS format as the is a Full Camera Link format.

FORMAT	Firmware Name	Format Description
0	MUX	1 tap cameras
1	MUX_2TOEP	2 taps, odd-even pixels
2	MUX_2TOEL	2 taps, odd-even lines
3	MUX_2TS	2 taps, segmented
4	MUX_2TS1RI	2 taps, segmented, right inverted
5	MUX_4TS	4 taps, segmented
6	MUX_4T2S2RIOEP	4 taps, odd-even pixels, right taps inverted
7	MUX_4TQ2RI2BU	4 quads, right quads inverted, bottom quads upside down
8	MUX_2CAM	2 cameras: 1 tap each
9	MUX_2CAM_2TOEP	2 cameras: 2 taps, odd-even pixels
10	MUX_2CAM_2TS1RI	2 cameras: 2 taps, segmented, right- inverted
11	MUX_2CAM_2TS	2 cameras: 2 taps, segmented
12	MUX_2CAM_2TOEL	2 cameras: 2 taps, odd-even lines
13	MUX_8TS	8 taps, segmented
14	MUX_BAY	Bayer decoder, 1 tap 8 bit
15	MUX_BAY_OE	Bayer decoder, 2 taps, odd-even pixels
16	MUX_BAY_2TS	Bayer decoder, 2 taps, segmented
17	MUX_4WI	4 taps, 4-way interleaved
18	MUX_2TOEPI	2 taps, odd-even pixels, both inverted
19	MUX_1TI	1 tap, inverted

#### Table 2-1 Firmware Options

FORMAT	Firmware Name	Format Description
20	MUX_8WI	8 taps, 8-way interleaved
21	MUX_BAY_2TS_RI	Bayer decoder, 2 taps, segmented, right inverted
22	MUX_4TS2RI	Four taps, segmented, right two taps inverted
23	MUX_8TSOEP4RI	Eight taps, segments, odd/even pixel, for right taps inverted
24	MUX_10WI	Ten taps, interleaved

### Table 2-1 Firmware Options

# 2.4 Generation of Acquisition Windows

#### 2.4.1 The Horizontal Active Window, HAW

The Horizontal Active Window (HAW) is a square wave that defines the portion of the line that will be acquired horizontally. HAW can span less than the whole camera line, if we want to acquire only a portion of that line. The size of the HAW is determined by a single number, the Active Clocks Per Line (ACPL). The ACPL is defined as the number of clocks during which the HAW is active. The ACPL field is programmed in CON10. The 17 bits define the maximum HAW as minimum 128K pixels.

The total number of pixels per line that will be acquired can be different than the ACPL. For a dual tap camera that supplies odd-even pixels for example, the total number of pixels acquired will be twice the ACPL, as for every clock the camera supplies two pixels. Note that the ACPL is not a function of the bits per pixel. The relationship between the number of pixels per line and the number of clocks per line is controlled by the firmware currently downloaded to the board. The FORMAT register will indicate which firmware is currently downloaded. Each tap configuration requires a different firmware file be downloaded. The correct firmware is automatically downloaded based on the information contained in the camera configuration file.

The size of the HAW is on an arbitrary boundary. The start in time of the HAW can come from two sources, depending on the setting of the HAW\_START bit:

The HSTART bit in the HCTAB, if HAW\_START = 1. The start of LEN, if HAW\_START = 0.

In both modes, the start of the HAW can be delayed 0-7 clocks relative to the start function (HSTART or LEN). This is done by the TRIM bits in CON9.

In both modes, the start of the HAW can be advanced 0-7 clocks relative to the start function (HSTART or LEN). This is done by the DELAY bits in CON14.

The HCTAB is the Horizontal Control Table that generates the HSTART, see section on CTABs.

Figure 2-3 shows the controls that generate the HAW.



Figure 2-3 Generation of the Horizontal Active Window, HAW

#### 2.4.2 The Vertical Active Window, VAW

The Vertical Active Window (VAW) is a square wave that defines the portion of the frame that will be acquired vertically. VAW can span less than the whole camera frame, if we want to acquire only a portion of that frame. The size of the VAW is determined by a single number, the Active Lines Per Frame (ALPF). The ALPF is defined as the number of HAW periods during which the VAW is active. The ALPF field is programmed in CON17. The 17 bits define the maximum VAW as minimum 128K lines.

The total number of lines per frame that will be acquired can be different than the ALPF. For a dual tap camera that supplies odd-even lines for example, the total number of lines acquired will be twice the ALPF as in the period of one HAW the camera supplies two lines.

The size of the VAW is on an arbitrary boundary. The start in time of the VAW can come from two sources, depending on the setting of the VAW\_START bit:

The VSTART bit in the VCTAB, if VAW\_START = 1. The start of FEN, if VAW\_START = 0.

Data will be acquired in the window defined by the HAW and the VAW

Figure 2-4 shows the controls that generate the VAW.



Figure 2-4 Generation of the Vertical Active Window, VAW

# 2.5 The Control Tables (CTABs)

The CTABs are two memories that are programmed by the host computer and read by the board's acquisition circuitry. The read-out is done in a sequential fashion, i.e. the memories' addresses are scanned sequentially. There is a vertical' memory (VCTAB) and a horizontal' memory (HCTAB). The vertical and horizontal memory each has an associated counter which scans its addresses, VCOUNT and HCOUNT respectively. The concept here is similar to how a CPU runs a program. There is a PC which works it's way through memory, processing each instruction in turn. Each bit in the CTAB corresponds to a different operation. For example, one bit might control the level of a signal going to the camera. In this case the CTABs can be thought of as programmable waveform generators. Another bit might cause an interrupt to occur on the PCI bus, yet another bit might force the HCOUNT to go to zero. The CTABs are fully programmable by software. The details of the CTABs are describe in this section.

### 2.5.1 Vertical Control Table

Figure 2-5 depicts the structure of the Vertical Control Table (VCTAB). For clarity, the address and data path that allow the host to program the VCTAB are not shown.



#### Figure 2-5 Vertical Control Table

The Vertical Control Table is made up of the following blocks:

VCOUNT - a synchronous counter that can be incremented, loaded and reset. The clock that drives VCOUNT is derived from the HCTAB, see below. VCOUNT is 17-bit wide and is connected to the address input of the VCTAB.

Logic for generating INC\_V - the increment control signal to the VCOUNT.

- Logic for generating LOAD\_V the load control signal to the VCOUNT. When LOAD\_V is asserted, VCOUNT is loaded with the value of 8000h (32,768 in decimal).
- Logic for generating RESET\_V the reset control signal to the VCOUNT. When RESET\_V is asserted, VCOUNT is reset to 0.
- Logic for generating CLOCK\_V the clock to the VCOUNT.
- VCTAB a static memory (SRAM) that outputs eight VCTAB control signals. The address of this SRAM is driven by VCOUNT.

If RESET\_V and LOAD\_V are asserted simultaneous, RESET\_V overrides.

As the VCOUNT increments, it scans the addresses of the VCTAB in ascending order. The output of the VCTAB depends on the data that has been written in the VCTAB by the host. If the VCOUNT is free running, it will cyclically scan all the VCTAB's addresses. Any arbitrary cyclic waveform can be implemented by programming the VCTAB with the adequate data.

The LOAD\_V and RESET\_V will enable the synchronization between external events and the waveforms generated by the VCTAB. LOAD\_V and RESET\_V will force the VCOUNT to known values, 8000h and 0 respectively.

The INC\_V signal will allow for stopping the counter from incrementing. In that case, the output of the VCTAB will be constant. While the VCOUNT is not incrementing, it can still be loaded or reset, see the logic below.

#### 2.5.2 The VCTAB Functions

The functions assigned to the columns in the VCTAB are shown in Table 2-2.

#### Table 2-2 The VCTAB Functions

Bit	Name	Function
0	VSTART	Start of VAW
1	VRESET	Vertical Reset
2	ENVLOAD	FEN Mask, enable load
3	IRQ	CTAB Interrupt
4	GPV0	General purpose vertical function 0
5	GPV1	General purpose vertical function 1
6	GPV2	General purpose vertical function 2
7	GPV3	General purpose vertical function 3

VSTART defines the start of the Vertical Acquisition Window (VAW), if the start is provided by the VCTAB, see previous section.

VRESET defines the reset of the VCOUNT, in case this function is programmed in the VCTAB.

ENVLOAD enables the FEN to load the VCOUNT. The rationale behind the ENVLOAD column from the VCTAB is that some cameras might not give a FEN, but only two pulses: the start and end of FEN. With the ENVLOAD, we can mask out the unwanted one.

IRQ provides an interrupt to the host, allowing an interrupt to occur at any point on the vertical axis.

GPV are general purpose vertical functions, see usage below.

Note: If a VRESET pulse happens coincident with a LOAD, the LOAD is overriding.

#### The CLOCK\_V Control

CLOCK\_V is the clocking of the VCOUNT is generated by the end of the line (horizontal reset).

#### The INC\_V Control

INC\_V is the logic for incrementing VCOUNT.

There are only two instances when we want to inhibit the incrementing of VCTAB. The first instance is when VCOUNT reaches 0000h, the "Stop at Zero" case. The other instance is when VCOUNT reaches 7FF0h, the "Vertical Stick" case.

#### Stop at Zero

Usually, VCOUNT will reach zero because of a RESET\_V signal. After VCOUNT is reset, there are programmable options defined by VCNT\_RLS\_ZERO. Depending on this bitfield, VCOUNT can continue to count or wait at zero till some event occurs, usually the assertion of the TRIGGER.

This operating mode is especially useful for synchronizing cameras to external events. TRIGGER is usually the output of a part-in-place signal. Until this signal is asserted, the VCOUNT waits at address 0000h. After the TRIGGER is asserted, VCOUNT starts counting, i.e., scanning the VCTAB in ascending order. At some address we will program a sync signal to be sent to the camera, usually through GPV0. In response to this sync signal, the camera will give back a frame, and it will assert FEN. The FEN will load address 8000h into VCOUNT. In the VCTAB, we will program the vertical acquisition window to start after address 8000h. At the end of the vertical acquisition window the RESET\_V will be asserted, which in turn will reset the VCOUNT. VCOUNT will wait at address 0000h until TRIGGER is asserted.

#### Vertical Stick

Using the previous example, assume that after we asserted the sync signal to the camera, we expect the camera to give us a frame, i.e., assert FEN. While we expect the camera to assert FEN, VCOUNT is still being incremented. If it takes too long for the camera to respond, VCOUNT will eventually reach and pass beyond 8000h. A vertical

acquisition window will be asserted, even though the camera did not assert FEN. To avoid such a situation, just before address 8000h, when VCOUNT reaches 7FF0h, it will stop. It will stay at 7FF0h until FEN is asserted. Then, VCOUNT will be loaded with 8000h.

The Vertical Stick will occur according to the setting of VCNT\_RLS\_STK.

#### The LOAD\_V Control

LOAD\_V is the logic of loading VCOUNT. In other words, loading VCOUNT means it jump to a new address.

VCOUNT will be loaded with the value 8000h by the rising/falling edge of FEN, if ENVLOAD is asserted. FEN usually marks the start of a valid frame. The start of the vertical acquisition window can be placed starting at address 8000h.

ENVLOAD is a column in the VCTAB. There are cameras that do not assert FEN. Some other type of cameras assert only the start and stop of a frame. In this case, ENVLOAD can mask out the unwanted signals.

Operation on the rising/falling edge of FEN is selected by FENPOL, see CON14

#### The RESET\_V Control

RESET\_V is the logic of resetting VCOUNT.

VCOUNT can be reset from different sources, under different conditions. The resetting of the VCOUNT is controlled by the VCNT\_RST bitfield.

#### 2.5.3 Vertical Control Table Size

The Vertical Control Table has 20000h (131,972) entries.

#### 2.5.4 Horizontal Control Table

The Horizontal Control Table (HCTAB) is 8 bits wide. The function of each bit is show in the following table.

Figure 2-6 depicts the structure of the HCTAB. For clarity, the address and data path that allow the host to program the HCTAB are not shown


#### Figure 2-6 Horizontal Control

The Horizontal Control Table is made up of the following blocks:

- HCOUNT a synchronous counter that can be incremented, loaded and reset. The clock that drives HCOUNT is a free running clock, PCLK/8, i.e., the pixel clock divided by eight. HCOUNT is 15 bits wide and is connected to the address input of the HCTAB.
- The 15 bit HCOUNT with the PCLK/8 can generate functions up to 256K PCLKs long, on boundaries of 8 PCLKs.

The ACPL is 17 bit, and that can generate an HAW of up to 128K PCLKs.

Logic for generating INC\_H - the increment control signal to the HCOUNT.

- Logic for generating LOAD\_H the load control signal to the HCOUNT. When LOAD\_H is asserted, HCOUNT is loaded with the value of 2000h.
- Logic for generating RESET\_H the reset control signal to the HCOUNT. When RESET\_H is asserted, HCOUNT is reset to 0.
- HCTAB a static memory (SRAM) that outputs eight HCTAB control signals. The address of this SRAM is driven by HCOUNT.
- Logic for generating CLOCK\_H the clock to the HCOUNT. This is a frequency divider. CLOCK\_H is PCLK, the pixel clock divided by eight.

Note: If RESET\_H and LOAD\_H are asserted simultaneously, RESET\_H overrides.

As the HCOUNT increments, it scans the address of the HCTAB in ascending order. The output of the HCTAB depends on the data that has been written in the HCTAB by the host. If the HCOUNT is free running, it will cyclically scan all the HCTAB's addresses. Any arbitrary cyclic waveform can be implemented by programming the HCTAB with the adequate data. The LOAD\_H and RESET\_H will enable the synchronization between external events and the waveforms generated by the HCTAB. LOAD\_H and RESET\_H will force the HCOUNT to fixed values, 2000h and 0 respectively.

The INC\_H signal will allow for stopping the counter from incrementing. In that case, the output of the HCTAB will be constant. While the HCOUNT is not incrementing, it can still be loaded or reset, see the logic below.

#### 2.5.5 The HCTAB Functions

The functions assigned to the columns in the HCTAB are shown in Table 2-3.

НСТАВ	Name	Function
D0	HSTART	Start of HAW
D1	HRESET	Reset HCOUNT and increment the VCOUNT
D2	ENHLOAD	LEN Mask, enable horizontal load
D3	reserved	
D4	GPH0	General purpose horizontal function 0
D5	GPH1	General purpose horizontal function 1
D6	GPH2	General purpose horizontal function 2
D7	GPH3	General purpose horizontal function 3

Table 2-3 The HCTAB Functions

HSTART marks the start of the Horizontal Acquisition Window, HAW. Video will be acquired only while the HAW is active.

HRESET will reset the HCOUNT.

ENHLOAD will allow the loading of the HCOUNT. A location that has 1 will allow the loading of the HCOUNT. A 0 will inhibit the loading of the HCOUNT.

GPH are general purpose horizontal functions. See usage below.

#### The INC\_H Control

INC\_H is the logic for incrementing HCOUNT.

There are only two instances when we want to inhibit the incrementing of HCTAB. The first instance is when HCOUNT reaches 0000h, "Stop at Zero" case. The other instance is when HCOUNT reaches 1FF1h, the "Horizontal Stick" case.

#### The "Stop at Zero" Case

Usually, HCOUNT will reach zero because of a RESET\_H signal. After HCOUNT is reset, there are two programmable options:

HCOUNT keeps on counting. HCOUNT stays at zero until ENCODER is asserted.

The selection between the two options is done by the HCNT\_RLS\_ZERO bitfield, see next section on camera synchronization.

This operating mode is especially useful for synchronizing line scan cameras to external events. ENCODER is usually the output of an encoder or a tachometer signal. Until this signal is asserted, the HCOUNT waits at address 0000h. After the ENCODER is asserted, HCOUNT starts counting, i.e., scanning the HCTAB in ascending order. At some address we will program a sync signal to be sent to the scan camera, usually through GPH0. In response to this sync signal, the camera will give back a line, and it will assert LEN. The LEN will load address 2000h into HCOUNT. In the HCTAB, we will program the horizontal acquisition window after address 2000h. At the end of the horizontal acquisition window the RESET\_H will be asserted, which in turn will reset the HCOUNT. HCOUNT will wait at address 0000h until ENCODER is asserted.

#### **Horizontal Stick**

Using the previous example, assume that after we asserted the sync signal to the camera, we expect the camera to give us a line, i.e., assert LEN. While we expect the camera to assert LEN, HCOUNT is still being incremented. If it takes too long for the camera to respond, HCOUNT will eventually reach and pass beyond 2000h. A horizontal acquisition window will be asserted even though the camera did not assert LEN. To avoid such a situation, just before address 2000h, when HCOUNT reaches 1FF0h, it will stop. It will stay at 1FF0h until LEN is asserted. Then, HCOUNT will be loaded with 2000h.

#### The LOAD\_H Control

LOAD\_H is the logic of loading HCOUNT.

HCOUNT will be loaded with the value 2000h by the rising/falling edge of LEN, if ENHLOAD is asserted. LEN usually marks the start of valid data in a line. The Horizontal Acquisition Window can be placed starting at address 2000h.

ENHLOAD is a column in the HCTAB that enables the LEN. There are cameras that do not assert LEN. In that case, the LEN input must be disabled, otherwise its behavior is unpredictable.

Operation on the rising/falling edge of LEN is selected by LENPOL, see CON14.

#### The RESET\_H Control

RESET\_H is the logic of reset HCOUNT.

HCOUNT can be reset from several sources, according to HCNT\_RST bitfield, see next section on camera synchronization.

#### Example

Lets look at a simple example to clarify the concept of the HCTAB. Assume we want to program a free running horizontal window of 32 pixels active area. Just before the active area we want to fire a strobe using GPH0. D0 (HSTART) defines the start of the HAW. Bit D1 defines the reset of the HCOUNT. D4, GPH0, is the strobe pulse. The size of the HAW is programmed in ACLP register.

Taking into account that the address counter is clocked by 1/8 the pixel clock, the HCTAB memory map will be as shown in Table 2-4.

HCTAB Address	HRESET	HSTART	GPH0	Comments
0	0	0	0	You got here from address 9
1	0	0	0	
2	0	0	0	
3	0	0	1	Fire the strobe
4	0	1	0	Start Horizontal Acquisition Window
5	0	0	0	Acquire
6	0	0	0	Acquire
7	0	0	0	Acquire
8	0	0	0	Acquire
9	1	0	0	Assert RESET_H
10	0	0	0	

#### Table 2-4 HCTAB Example

#### The CT Functions

The CT's are four functions derived from the HCTAB and the VCTAB. Those functions can define an arbitrary horizontal and/or vertical waveform. The definition of the CT's is given below:

CT[0] = GPV[0] AND GPH[0] CT[1] = GPV[1] AND GPH[1] CT[2] = GPV[2] AND GPH[2] CT[3] = GPV[3] AND GPH[3]

Each CT has a vertical and a horizontal component. Both components are programmed in the CTABs. The minimum horizontal pulse is 8 PCLKs. The minimum vertical pulse is one line.

The CT's can be steered to the Camera Controls (on the CL connectors) and to the GPOUTs, on the IO connector.

### 2.5.6 Horizontal Control Table Size

The Horizontal Control Table has 8000h (32,768) entries.

# 2.6 Synchronizing Acquisition, Camera, CTABs and External Signals

These boards have extremely flexible camera interfaces. They have been designed to acquire from almost any Camera Link camera and to synchronize with almost all industrial signals. There are two layers of synchronization. The first layer is to synchronize to signals coming from the industrial environment. For example triggers and encoders. The second layer is to synchronize to the camera. This requires both sending signals to the camera (e.g. exposure control) and receiving signals from the camera (e.g. Pixel Clock, Line Enable and Frame Enable). All of these synchronization problems are solved by the Control Tables (CTABs) and Vertical/Horizontal Operations. See previous section for detailed operation on the CTABs.

The Vertical and Horizontal Operations describe different state changes that the board goes through. For example one operation might be to begin acquiring pixels, another might be to reset the VCOUNT back to zero. Generally these state changes are caused by one or more events. There are a number of events, both horizontal and vertical, that the board can react to. These events are tied to operations by a set of programmable bitfields. The details of these events are outlined in this section.

#### 2.6.1 Vertical Operations and Events

The vertical operations are related to the vertical axis of an image (in memory or on the display) or frame timing (of a camera). The operations are mainly commands to VCOUNT and acquisition commands. Each operation can be initiated by some event. The selection of the event that will initiate the specific operation is done by a set of three control bits related to each operation.

Table 2-5 is a list of the vertical operations and their related control bits.

Vertical operation	Control bits
VCOUNT frozen/released from 0000h	VCNT_RLS_ZERO
VCOUNT reset to 0000h	VCNT_RST
VCOUNT load with 8000h	VCNT_LD
VCOUNT frozen/released from 7FF0h	VCNT_RLS_STK
VCOUNT increment	VCNT_INC
Acquire (SNAP, GRAB, CONTINUOUS)	ACQ_CON
FREEZE acquisition	FREEZE_CON
ABORT acquisition	ABORT_CON
START vertical acquisition window	VAW_START

#### Table 2-5 Vertical Operations

Table 2-6 is a list of the events that initiate the vertical operations:

Event description	Event Name
TRIGGER asserted	TRIG_ASRT
TRIGGER de-asserted	TRIG_DASRT
FEN asserted	FEN_ASRT
FEN de-asserted	FEN_DASRT
TRIGGER is HI	TRIG_HI
TRIGGER is LO	TRIG_LO
RESET from VCTAB	RST_VCTAB
RESET from SW	RST_SW
Host writes acquisition command	HOST_WCMD_GRAB/SNAP
Acquisition frame counter reaches pro- grammed value	AQ_COUNT

#### Table 2-6 Vertical Events

What follows is a list of each operation and the corresponding events that can be used to cause the operation. Included is the bitfield settings for each operations. It is important to understand that each operation is independent and can be programmed without regard for how the other events are programmed. However, some combinations might not make sense.

#### VCOUNT Release From Zero

This operation controls the behavior of VCOUNT when it reached zero. See Table 2-7.

Initiator	VCNT_RLS_ZERO	Comments
None	0	Normal operation mode, no stop at 0000h
TRIG_ASRT	1	Edge Mode (aka Letter Mode), always stay at 0000h, release on TRIG_ASRT
TRIG_HI	2	Level Mode (aka Luggage Mode), stay at 0000h if TRIG_LO, release on TRIG_ASRT

#### Table 2-7 VCNT\_RLS\_ZERO

#### VCOUNT Reset To Zero

This operation controls how VCOUNT is reset to zero. See Table 2-8.

Initiator	VCNT_RST	Comments
End_of_VAW	0	Default operation, reset at end of VAW
TRIG_DASRT or End_ of_VAW	1	Triggered termination
RST_VCTAB	2	Reset from VCTAB
FEN asserted or	3	Reset from start of FEN
TRIG_DASRT or RST_ VCTAB	4	Triggered termination
TRIG_DASRT	5	Triggered termination

#### Table 2-8 VCNT\_RST

Note: The VCOUNT is always reset by the RST\_SW (sofware reset ) and by the HOST\_ WCMD\_ABORT (host writes ABORT command).

#### VCOUNT Release From Stick Point (7FF0h)

This operation controls the behavior of VCOUNT when it hits the stick point. The purpose of the stick point is to allow for very long periods of time between frames. The stick point is located at 7ff0h. See Table 2-9.

#### Table 2-9 VCNT\_RLS\_STK

Initiator	VCNT_RLS_STK	Comments
None	0	Normal operation mode, no stop at 7FF0h
VLOAD or VRESET	1	Stick at 7FF0h till load (usually FEN) or reset asserted

#### VCOUNT Load To 8000h

This operation controls how and when VCOUNT loads (jumps to) 8000h. See Table 2-10.

Initiator	VCNT_LD	Comments
None	0	No load
FEN_ASRT and ENV- LOAD	1	Assertion of FEN qualified with ENV- LOAD
FEN_ASRT	2	Assertion of FEN only
TRIG_ASRT	3	Assertion of TRIGGER

#### Table 2-10 VCNT\_LD

#### Acquisition Command Control

This operations controls how the acquisition commands get initiated. There are two major acquisition commands. The SNAP command, which only acquires one frame. The GRAB command, which continuously acquires frames until a freeze or abort command is issued. In addition, the board has a continuous data mode which is not frame oriented. In continuous data mode, the board will acquire data based only on the clock and data qualifying signals. There are no acquisition commands in this mode. See Table 2-11.

#### Table 2-11 ACQ\_CON

Initiator	ACQ_CON	Comments
HOST_WCMD_GRAB/ SNAP	0	normal, host initiated GRAB/SNAP/ FREEZE
TRIG_ASRT	1	Triggered initiated GRAB/SNAP/ FREEZE
TRIG_ASRT and HOST_ WCMD_GRAB	2	Triggered SNAP
TRIG_HI	3	Continuous data, wo. CTABs

Note: See also Section 2.7 for more details on the how the acquisition commands work.

#### Freeze Command Control

This operation is used to stop acquisition when the board is in GRAB mode. Acquisition will stop immediately if the board is between frames, or at the end of the current frame, if the board is in the middle of a frame. See Table 2-12.

Initiator	FREEZE_CON	Comments
HOST_WCMD_ FREEZE	0	Normal, host initiated
AQ_COUNT or HOST_WCMD_ FREEZE	1	Acquisition counter reaches number of frames programmed in the AQ_COUNT register
TRIG_DASRT	2	Trigger de-asserted

#### Abort Command Control

This operations terminates the current acquisition immediately. This operation will terminate both a SNAP and a GRAB command. If the board is in the middle of a frame, only part of the frame will be acquired. See Table 2-13.

Table 2-13	ABORT_	CON
------------	--------	-----

Initiator	ABORT_CON	Comments
HOST_WCMD_ ABORT	0	Normal, host initiated
TRIG_DASRT or HOST_WCMD_ ABORT	1	Abort on falling edge TRIG or host com- mand.

#### 2.6.2 Horizontal Operations and Events

The horizontal operations and events are related to the horizontal axis (of an image in memory or on the display) or line timing (of a camera). The operations are mainly commands to HCOUNT. Each operation can be initiated by some event. The selection of the event that will initiate the specific operation is done by a set of three control bits related to each operation.

Table 2-14 lists the horizontal events.

#### Table 2-14 Horizontal Operations

Horizontal operation	Control bits
HCOUNT released from zero	HCNT_RLS_ZERO
HCOUNT reset to zero	HCNT_RST
HCOUNT load with 2000h	HCNT_LD
HCOUNT release from 1FF0h	HCNT_RLS7F0
HCOUNT increment	HCNT_INC
Start horizontal active window	HAW_START

The events that can initiate the horizontal operations are listed in Table 2-15.

Event description	Event Name
ENCODER asserted	ENC_ASRT
ENCODER de-asserted	ENC_DASRT
LEN asserted	LEN_ASRT
LEN de-asserted	LEN_DASRT
ENCODER is HI	ENC_HI
ENCODER is LO	ENC_LO
RESET from HCTAB	RST_HCTAB
RESET from SW	RST_SW
FEN asserted	FEN_ASRT

#### Table 2-15 Horizontal Events

The sections below enumerate all of the horizontal operations and how the various events can initiate them. The control of each operation is independent from all of the others.

#### HCOUNT Release From Zero

This operation controls the behavior of HCOUNT when it reached zero (see Table 2-16).

Initiator	HCNT_RLS_ ZERO	Comments
None	0	Normal operation mode, no stop at zero
ENC_ASRT	1	One-shot mode, wait for encoder for release

#### Table 2-16 HCNT\_RLS\_ZERO

#### **HCOUNT** Reset To Zero

This operation controls how HCOUNT is reset to zero (see Table 2-17).

Initiator	HCNT_RST	Comments
END_OF_HAW	0	Default operation, end of HAW
FEN_ASRT or RST_ HCTAB	1	Reset on FEN_ASRT, Random FEN mode
RST_HCTAB	2	Reset from HCTAB

#### Table 2-17 HCNT RST

Note: The HCOUNT can always be reset by RST\_SW or HOST\_WCMD\_ABORT

#### HCOUNT Release From Stick Point (1FF0h)

This operation controls the behavior of HCOUNT when it hits the stick point. The purpose of the stick point is to allow for very long periods of time between lines. The stick point is located at 1ff0h. See Table 2-18.

Initiator	HCNT_RLS_STK	Comments
None	0	Normal operation mode, no stop at 1FF0h
HLOAD or HRESET	1	Stay at x1FF0 till load (usually LEN) or

reset asserted

#### Table 2-18 HCNT\_RLS\_STK

#### HCOUNT Load To 2000h

This operation controls how and when HCOUNT loads (jumps to) 2000h (see Table 2-19).

Initiator	HCNT_LD	Comments
None	0	No load
LEN_ASRT	1	Load on LEN assert, qualified with ENH- LOAD column
ENC_ASRT	2	Load on ENCODER assert, qualified with ENHLOAD column

#### Table 2-19 HCNT\_LD

# 2.7 Acquisition Command and Status

This section describes how the acquisition state machine works. This state machine controls which frames from the camera are acquired and which are ignored. As the commands can be issued asynchronous to the camera's timing, the acquisition state machine will remember the command and execute it starting at the beginning of the frame. That will guarantee that whole frames will be acquired. Note that the acquisition state machine only marks the frames to be acquired. The amount of pixels/line and lines/frame to be acquired in the marked frame is determined by the HAW and VAW, see section Section 2.4.

The acquisition state machine is controlled by the following signals:

AQCMD, the acquisition command bitfield

VACTIVE, the camera's vertical active timing (usually FEN for area scan cameras).

TRIGGER, the selected trigger.

ACQ\_CON, a bitfield that defines special acquisition modes for the state machine.

The current state of the machine can be observed by the AQCMD and AQSTAT bitfields described below.

#### 2.7.1 The Acquisition Bitfields

The acquisition command bits, AQCMD describe the command to be performed in the next frame. The acquisition status bits, AQSTAT, describe the current command that is performed. The four acquisition commands are described in the Table 2-20.

AQCMD	Command	Comment
0(00b)	FREEZE	Stop acquiring at end of current frame
1 (01b)	ABORT	Stop acquiring immediately, unconditionally
2 (10b)	SNAP	Acquire one frame
3 (11b)	GRAB	Acquire continuously

#### Table 2-20 ACQCMD

The following list details the behaviors of these bitfields.

- The AQSTAT bits are set at the beginning of the VACTIVE. The last instance a command can be issued is about 4 LCLKs before the start of the VACTIVE.
- For a SNAP command, when the SNAP starts, the AQCMD bits are cleared. Note that for SNAP, the AQCMD bits are written by the host and cleared by the state machine.
- If during a SNAP/GRAB operation another SNAP/GRAB command is issued, it is ignored.

Table 2-11 below describes the acquisition modes for ACQ\_CON

ACQ_CON	Mode Description
0 (000b)	Host command performed on next frame
1 (001b)	Host command issued when TRIGGER asserted
2 (010b)	As long as GRAB command is on, a single frame will be SNAPped at every assertion of the TRIGGER.
3 (011b)	Continuous acquisition mode. Host commands are ignored. Data will be acquired continuously as long as the TRIGGER is asserted.

#### Table 2-21 ACQ\_CON

Figure 2-7 shows a timing diagram of the SNAP command, with  $ACQ\_CON = 0$ . The command is written by the host during the active frame. The acquisition will start at the beginning of the next frame.



#### Figure 2-7 Snap Command Timing

Figure 2-8 shows the timing of the GRAB operation with ACQ\_CON=0. Figure 2-9 shows the timing of the ABORT operation with ACQ\_CON=0. Note that the ABORT command will cut off part of the frame. This command is useful for resetting the board without having to wait for the end of the frame. Figure 2-10 shows a SNAP operation with ACQ\_CON=1. In this mode, after the TRIGGER has been asserted and the command executed, the host must write a new command in the AQCMD field. Figure 2-11 shows acquisition in ACQ\_CON=2 mode. Here, as long as the GRAB command is on, a frame will be acquired for every assertion of the TRIGGER. In this mode, there is no need for the host to write a new command.



AQSTAT reset, grabbing ends





Figure 2-9 Abort Command Timing



Figure 2-10 Snap Command Timing with ACQ\_CON = 2



Figure 2-11 Grab Command Timing with ACQ\_CON = 2

# 2.8 Trigger Processing

This section describes how the trigger circuit works. The trigger is used to initiate a vertical operation (for example, capturing one frame). There are three possible external hardware inputs to the trigger circuit and a software input. Assertion of the trigger can be delayed by up to 8192 lines (granularity is 8 lines). This delay works only with the external hardware trigger. Figure 2-12 illustrates the trigger circuit.

Note: The Alta only has one trigger input: TRIGGER\_TTL.



Figure 2-12 Trigger Circuit

# 2.9 Encoder Processing

This section describes how the encoder circuit works. The encoder is used to initiate a horizontal operation (for example, capturing one line). There are three possible external hardware inputs to the encoder circuit and a software input. The selected external encoder can be divided by the value in the ENC\_DIV register. Figure 2-13 illustrates the encoder circuit.

Note: The Alta does not have any encoder inputs.



Figure 2-13 Encoder Circuit

# 2.10 The On-Board Signal Generator

The on-board signal generator has been replaced the New Timing Generator (NTG). Please see Section 3.1 for more information.

# **New Timing Generator**

# Chapter 3

# 3.1 Introduction

This section covers the new timing generator (NTG) which can control cameras connected to the Karbon-CL, Neon-CL and the Alta-AN. The purpose of this timing generator is to provide a simple system of controlling a camera's exposure time and line/ frame rate from the frame grabber. The NTG is fully programmable and is easily controlled from software and/or from camera configuration files.

The NTG is based on a completely independent timing generator that is unrelated to acquisition and the CTabs. This timing generator is easy to program, is not depend on camera architecture or triggering modes, and offers the granularity and range that customers need. There is no connection between the NTG and the acquisition state machine, the CTabs, the VAW/HAW or the camera connected.

The New Timing Generator supports both triggered and free running modes. For triggered modes it supports both the trigger signal for area cameras or the encoder signal for line cameras.

The NTG requires that the camera be put in one of two modes. If the NTG is going to control just the line/frame rate, then the camera should be programmed into a "trig-gered" mode. In this case, the exposure is controlled by the camera. IF the NTG is to control both the line/frame rate as well as the exposure time, then the camera must be put into a "pulse width control" mode. In this case, neither the line/frame rate nor the exposure time are controlled by the camera. They are both completely controlled by the NTG.

Note: The NTG replaces the on-board timing generator that was prevouisly available on all boards. The NTG is much more flexible and easier to use. Please contact BitFlow if you have been using the previous on-board timing generator.

# 3.2 Components and Control

#### 3.2.1 Periods and Frequencies

The NTG consists of a programmable signal generator based on a crystal controlled clock. This clock is always running and is unrelated to the camera connected or how other parts of the board are programmed. The base frequency of the clock is designed to handle any line scan camera system. For area scan cameras the clock can be divided 128 to increase the time range if very slow frame rates and/or line exposures are needed. To use the divided clock, program the bit NTG\_TIME\_MODE to 1. The frequency of the clock is different for the different frame grabber families as shown in Table 3-1.

#### Table 3-1 NTG Base Frequencies

Family	Base frequency	Reduced frequency
Karbon, Neon	7.3728 MHz	57.6000 KHz
Alta-AN	5 MHz	39.0625 KHz

The are two main timing registers: NTG\_RATE, which controls the line/frame rate period, and NTG\_EXPOSURE, which controls the exposure period. These are both 28 bits, which should be enough to support almost all applications. Table 3-2 shows the resulting ranges

#### Table 3-2 NTG Period Ranges

Family	Mode	Granularity (1 clock period)	Max Period
Karbon, Neon	Area	~17.4 microsecond	~77 minutes
	Line	~136 nanoseconds	~36 seconds
Alta-AN	Area	25.6 microseconds	~114 minutes
	Line	200 nanoseconds	~ 53 seconds

The NTG uses a counter internally to create the programmed waveforms. Because the NTG registers can be set for very long times, reprogramming the NTG can be time consuming. In order to speed up modifications to the NTG parameters, the register NTG\_RESET can be used. Poke this bit to a 1 resets the NTG counter to zero, and starts a new cycle with the latest register values.

Note: Use the base frequency when a high resolotion timer is needed (fine granularity). Use the reduced frequency when long exposure periods and/or slow frame rates are needed (course granularity) You can use which ever mode suites your application regardless of whether you are using a line scan or an area scan camera..

#### 3.2.2 Waveform polarity

There is also a register that inverts the waveform generated, NTG\_INVERT. This is different than the old signal generator on the R64, which supports asserted-low signals by increasing the high time to be one-over-the-low time. The NTG system is much simpler, poke NTG\_INVERT to a 1 and the waveform goes from asserted high to asserted low.

#### 3.2.3 Triggering

The NTG has two modes of operation, free-running and one-shot mode. The bit that controls this mode is, NTG\_ONESHOT. In free-run mode, both NTG\_RATE and NTG\_EXPOSURE are used. In one shot mode, only NTG\_EXPOSURE is used, and the rate is controlled by the encoder/trigger.

Either the trigger input or the encoder input can be used to control the NTG in oneshot mode. This setting is controlled by the bit NTG\_TRIG\_MODE.

#### 3.2.4 Output Signals

The waveform of the NTG can be routed to almost any of the board's output signals. The waveform can be send to the CC lines on the CL connector, or the GPOUT lines on the I/O connector. The CCx\_CON bitfields can be used route the NTG signal to the CCs output. For example, program CC1\_CON to 3 to get the NTG output on CC1. Similarly, the GPOUTx\_CON bitfields can be used to route the NTG signals to the GPOUTx outputs. For example, to put the NTG output on GPOUT1, program GPOUT1\_CON to 6. The NTG waveform can be sent simultaneously to any and all of these outputs.

#### 3.2.5 Master/Slave Control

On boards that support more than one VFG (Karbon, Alta) there is the option to make the slave VFGs have the same timing as the master VFG, or to run each slave VFG's timing generator independently. The master VFGs always has its own timing. The selection is made by programming the NTG\_SLAVE bit. On a given VFG, if this bit is set to 0, the VFG generates its own independent timing. If this bit is set to 1, the VFG's timing is the same as that of the master VFG.

Note: On multi-VFG boards, there is always a master and one or more slaves for programming purpose. The master VFG must always have the bit NTG\_SLAVE set to 0. The slave VFGs can either be indpendent (NTG\_SLAVE = 0) or the same as the master VFG (NTG\_SLAVE = 1).

# 3.3 Timing

The following diagram illustrates all of the relevant parameters of the NTG.

Note: In the diagrams below NTG\_INVERT is set to 0. If it were set to 1, these diagrams would be inverted.



Figure 3-1 NTG Timing

In free running mode, the NTG counter will start at zero, one clock later it will assert the output. It will then count up to NTG\_EXPOSURE clocks, then de-assert the output. It will then continue to count to NTG\_EXPOSURE clocks then reset itself and start over.

In one-shot mode, the NTG clock will wait at zero and until the trigger is asserted, it will then start counting. On the first clock after the trigger is asserted it will assert its output. It will then count up to NTG\_EXPOSURE clocks then de-assert its output. Next it will reset itself and wait for another trigger.

# 3.4 NTG Control Registers

Name	Locations	Purpose
NTG_RATE	CON17[270]	The line/frame rate period in units of one NTG clock (see Table 3-2 for values).
NTG_ONESHOT	CON17[30]	0 = Free-run mode, 1= one-shot mode, waits for either the trigger or the encoder pules (depending on NTR_TRIG_ MODE).
NTG_TRIG_MODE	CON17[31]	1 = Encoder for NTG trigger,0 = Trigger for NTG trigger
NTG_INVERT	CON18[30]	0 = NTG asserted high, $1 = NTG$ asserted low
NTG_TIME_MODE	CON18[31]	0 = Base NTG clock, 1= Base NTG clock / 128 (see Table 3-1 for values)
NTG_EXPOSURE	CON26[270]	The exposure time in units of one NTG clock.
NTG_RESET	CON26[30]	Writing a 1 resets the NTG counter
NTG_SLAVE	CON26[31]	0 = NTG master, 1 = NTG timing slaved to master

The following table summarizes the registers:

NTG Control Registers

# Quadrature Encoder

# Chapter 4

# 4.1 Introduction

This section discusses support for quadrature encoders. A quadrature encoder is an encoder that outputs two signals A and B. Both signals are used as a line trigger. However, the signals are 90 degrees out of phase. By comparing the A and B signals, the direction of the encoder motion can be determined. There are a number of ways that quadrature encoders can be used to control acquisition. The following sections cover all of the support methods.

Most of the quadrature encoder system is based around a 24-bit counter. This normally starts at zero and then counts up or down every time the encoder moves. The counter can be observed at any time via the QENC\_COUNT register. This registers is the heart of the encoder system. For example, triggers values can be programmed to start and end acquistion of lines. Also, as the counter tracks the motion of the stage attached to the encoder exactly, the system can be programmed to only acquire forward only or backward only stage movements. The system can be be programmed to only acquire one line for each encoder count that corresponds to a physical location on the stage. The encoder counter can be used in many different ways, described in more dteails below.

#### 4.1.1 Simple Encoder Mode

The most basic method of using a quadrature encoder is to use it like a standard signal phase encoder. In this mode, the quadrature encoder provides a higher resolution signal, as both the A and B signals can be used to trigger lines. Also, by setting QENC\_DECODE = 1, both the rising and the falling edges of both the A and B signals are used to trigger lines, providing a 4x increase in resolution over a signal phase encoder.

In this mode, every encoder edge triggers a line, the direction information from the encoder is ignored.

#### 4.1.2 Positive or Negative Only Acquisition

The board can be programmed to only acquired lines when the encoder moves forward (increase the encoder count in a positive direction) or moves backwards (decrease the encoder count in a negative direction). This mode is useful in situations where a stage is moving back and forth, and lines need only be acquired if the stage is moving in one direction only. The direction of acquisition is controlled by the QENC\_AQ\_DIR register.

#### 4.1.3 Interval Mode

Often in situations when a stage is moving back and forth, acquisition is only required over a subsection of the total stage range. Interval mode has been designed for these situations. When the board is in interval mode, it only acquires lines when the encoder counter is between a lower limit and an upper limit. If the counter is outside these limits, lines are not acquired.

To use interval mode, set QENC\_INTRVL\_MODE = 1, and program QENC\_INTRVL\_LL and QENC\_INTRVL\_UL to the encoder ranges that bracket the section of your stage range that you wish to acquire. Interval mode can be used in conjunction with QENC\_ AQ\_DIR to acquire lines passing over the interval in the positive direction, the negative direction or both directions.

#### 4.1.4 Re-Acquisition Prevention

Encoders are usually connected to mechanical systems which do not always move perfection smoothly. Because of these imperfections, there can be "jitter" in the quadrature encoder signal. This jitter is not an electrical imperfection, but represent the reality of the mechanical system vibrating, jumping, bouncing, etc. If these imperfections occur during the period of time where lines are being acquired, the image will be distorted. Lines on the object can be acquired more than once as the stage jitters. To prevent re-acquisition of lines, a circuit has been added to the quadrature encoder system that can prevent any line from being acquired more than once. To enable this mode, set QENC\_NO\_REAQ = 1.

#### 4.1.5 Scan Step Mode

The encoder can also be used to trigger acquisition of full frames from an area scan camera. The idea is that every N lines, a trigger is issued to the board, which causes acquisition of a frame. This can be used, for example, with a linear stage, where an image is needed in steps across the range of the stage. This mode is enable by setting SCAN\_STEP\_TRIG = 1, and programming SCAN\_STEP to the number of encoder counts per trigger.

#### 4.1.6 Combining Modes

All of the modes above can be combined to support complicated encoder requirements. For example, the board can be programmed to acquire an interval in the positive direction only, with no lines being reacquired. Many combinations are possible.

#### 4.1.7 Control Registers

Starting with Section 4.3 all of the registers needed to control the qudrature encoder system are explained.

#### 4.1.8 Observability

The status of the quadrature encoder system can be observed at any time. Shown in Table 4-1are all the registers that can be used.:

Register	Meaning
QENC_COUNT	Encoder counter
QENC_PHASEA	Phase of input A
QENC_PHASEB	Phase of input B
QENC_DIR	Direction of encoder
QENC_INTRVL_IN	Interval status
QENC_NEW_LINES	Indicates new lines are being acquired

#### Table 4-1 Observability Registers.

#### 4.1.9 Electrical Connections

Both TTL and LVDS (differential) quadrature encoders are supported. TTL connections are shown in Table 4-2 and LVDS connections are shown in Table 4-3.

Table 4-2 TTL Quadrature Encoder Connections

Encoder	Frame Grabber
А	VFGx_ENCODER_TTL
В	VFGx_ENCODER_B_TTL
Ground	GND

Table 4-3 LVDS Quadrature Encoder Connections

Encoder	Frame Grabber
A+	VFGx_ENCODER+
A-	VFGx_ENCODER-
B+	VFGx_ENCODER_B+
B-	VFGx_ENCODER_B-

Note: VFGx - refers to the VFG number that you wish to connect to. For example, if you want to connect a TLL A output to VFG 0, then you would use VFG0\_ENCODER\_TTL.

## 4.2 Understanding Stage Movement vs. Quadrature Encoder Modes

The quadrature encoder system has many modes that can be used in various combinations. These combinations are easier to understand through a few simple illustrations. Figure 4-1 shows the basic Encoder Count vs. Time graph and how it corresponds to stage movement. Keep in mind that the encoder could be attached to any mechanical system, however, a back and forth stage is a simple way to illustrate these modes.

In Figure 4-1 you can see as the stage moves back and forth, the encoder counts up and down. Further, in this example we assume  $QENC_AQ_DIR = 1$ , which tells the system to only acquire when the encoder counter is moving in the positive direction. This is illustrated by solid lines in the positive direction and dashed lines in the negative direction.



Figure 4-1 Encoder Count vs Time





Figure 4-2 Quadrature Encoder Modes vs. Acquisition

# 4.3 CON15 Register

Bit	Name
0	QENC_INTRVL_LL
1	QENC_INTRVL_LL
2	QENC_INTRVL_LL
3	QENC_INTRVL_LL
4	QENC_INTRVL_LL
5	QENC_INTRVL_LL
6	QENC_INTRVL_LL
7	QENC_INTRVL_LL
8	QENC_INTRVL_LL
9	QENC_INTRVL_LL
10	QENC_INTRVL_LL
11	QENC_INTRVL_LL
12	QENC_INTRVL_LL
13	QENC_INTRVL_LL
14	QENC_INTRVL_LL
15	QENC_INTRVL_LL
16	QENC_INTRVL_LL
17	QENC_INTRVL_LL
18	QENC_INTRVL_LL
19	QENC_INTRVL_LL
20	QENC_INTRVL_LL
21	QENC_INTRVL_LL
22	QENC_INTRVL_LL
23	QENC_INTRVL_LL
24	QENC_DECODE
25	QENC_AQ_DIR
26	QENC_AQ_DIR
27	QENC_INTRVL_MODE
28	QENC_NO_REAQ
29	QENC_DUAL_PHASE
30	SCAN_STEP_TRIG
31	QENC_RESET

LL

#### **QENC\_INTRVL** R/WR/W, CON15[23..0], Karbon, Neon

This register contains the lower limit value that is used to start acquisition when the system is in interval mode (see QENC\_INTRVL\_MODE).

**QENC\_DECODE** R/W, CON15[24], Karbon, Neon

This bit determines how often the quadrature counter is incremented.

QENC_DECODE	Meaning
0	Counter increments on the rising edge of input A and the rising edge of input B. This is also called "2x" modes.
1	Counter increments on both the rising and falling edge of A and both the rising and falling edge of B. This is also called "4x" mode.

#### QENC\_AQ\_DIR R/W, CON15[26..25], Karbon, Neon

This bit controls which quadrature encoder direction is used for acquisition.

QENC_AQ_DIR	Meaning
0 (00b)	Lines are acquired in both directions
1 (01b)	Lines are acquired only in the positive direction.
2 (10b)	Lines are acquired only in the negative direction.
3 (11b)	Reserved

#### **QENC\_INTRVL\_** R/W, CON15[27], Karbon, Neon

MODE

When this bit is 1, interval mode is turned on. When interval mode is on, lines are only capture when the encoder counter is between the lower limit (set by QENC\_INTRVL\_LL) and the upper limit (set by QENT\_INTRVL\_UL). If the counter is outside of this range, lines are not acquired. Whether lines are acquired as the counter increments through the interval, or decrements through the interval, or in both directions is controlled by QENC\_AQ\_DIR.

# QENC\_NO\_ R/W, CON15[28], Karbon, Neon REAQ This bit controls how the quadrature encoder system handles the situation where the encoder does not smoothly increase (or decrease if QENC\_AQ\_DIR = 1). If there is "jitter" in the encoder signal, often caused by problems with the mechanical systems, it is possible for the board to acquire the same line or lines more than once as the

mechanical system backs up and moves forward (jitter). This re-acquisition can cause problems as the resulting images will have distortions and will not accurately represent the object in front of the camera.

Programming this bit to a 1 turns on the no-reacquisition circuit. This circuit eliminates this problem as each line in the image will only be acquired once, regardless of how much jitter occurs in the quadrature encoder input. The circuit does this by making sure that only one line is acquired for each encoder counter value. If the quadrature encoder backs up, and then moves forward, the board will not acquire lines until a new encoder counter value is reached.

This system handles any amount of jitter, regardless of how many times the counter passes through a value, or to what extremes the counter goes. New lines will only be acquired when new values are reached.

Once the entire frame has been acquired, the system must be reset. The system can always be reset by poking QENC\_RESET to 1. There are also ways that the system can automatically be reset, see QENC\_RESET\_MODE.

QENC_NO_REAQ	Meaning
0	Lines are acquired every change in the encoder counter (as controlled by QENC_AQ_DIR)
1	Lines are only acquired when the encoder counter reaches new values (also controlled by QENC_AQ_ DIR)

QENC_DUAL_	R/W, CON15[29], Karbon, Neon
PHASE	

This bit controls which type of encoder is attached.

QENC_DUAL_PHASE	Meaning
0	A single phase encoder is attached
1	A quadrature encoder is attached

SCAN\_STEP\_ R/W, CON15[30], Karbon, Neon

TRIG

The scan step circuit uses the encoder to generate a trigger to the system. The scan step trigger generates a trigger every N lines (N is set in the SCAN\_STEP register).

SCAN_STEP_TRIG	Meaning
0	Trigger comes of the normal source
1	Trigger comes from the scan step circuit
### **QENC\_RESET** WO, CON15[31], Karbon, Neon

Poking this bit to a 1 resets the entire quadrate encoder system.

# 4.4 CON16 Register

Bit	Name
0	QENC_INTRVL_UL
1	QENC_INTRVL_UL
2	QENC_INTRVL_UL
3	QENC_INTRVL_UL
4	QENC_INTRVL_UL
5	QENC_INTRVL_UL
6	QENC_INTRVL_UL
7	QENC_INTRVL_UL
8	QENC_INTRVL_UL
9	QENC_INTRVL_UL
10	QENC_INTRVL_UL
11	QENC_INTRVL_UL
12	QENC_INTRVL_UL
13	QENC_INTRVL_UL
14	QENC_INTRVL_UL
15	QENC_INTRVL_UL
16	QENC_INTRVL_UL
17	QENC_INTRVL_UL
18	QENC_INTRVL_UL
19	QENC_INTRVL_UL
20	QENC_INTRVL_UL
21	QENC_INTRVL_UL
22	QENC_INTRVL_UL
23	QENC_INTRVL_UL
24	QENC_REAQ_MODE
25	QENC_REAQ_MODE
26	QENC_RESET_REAQ
27	N/A
28	N/A
29	N/A
30	N/A
31	N/A

# **QENC\_INTRVL** R/W, CON16[23..0], Karbon, Neon **UL**

This register contains the upper limit value that is used to start acquisition when the system is in interval mode (see QENC\_INTRVL\_MODE).

**QENC\_REAQ\_** R/W, CON16[25..24], Karbon, Neon **MODE** 

This bit controls how the circuit that prevents re-acquisition from encoder jitter is reset. Re-acquisition is prevented by keeping a list of lines that have been acquired, and making sure that only lines that are not on the list are acquired. Once the entire frame is acquired, there must be some way to reset the list, otherwise no new lines will ever be acquired See QENC\_NO\_REAQ for more information.

The reset can be either automatic or manual. Manual modes require that the host application software poke the QENC\_RESET\_REAQ bit when the reset is desired. Automatic modes do not require host interaction, the reset will occur automatically when the specified conditions are met.

QENC_REAQ_MODE	Mode	Meaning
0 (00b)	Manual	Reset the list of acquired lines when QENC_RESET_REAQ is poked to 1.
1 (01b)	Automatic	Reset the list of lines when the encoder counter is outside of the interval set by the upper limit and lower limit. Whether the reset occurs above the upper limit or below the lower limit depends on the QENC_ AQ_DIR register.
2 (10b)		Reserved
3 (11b)		Reserved

### **QENC\_RESET\_** WO, CON16[26], Karbon, Neon

#### REAQ

This register is used to reset the circuit that prevents the re-acquisition of lines when QENC\_NO\_REAQ is set to 1. Writing a 1 to this register deletes the list of acquired lines, thus next time the lines are passed over, they will be acquired again. Writing to this bit always resets the no re-acquistion circuit, regardless of the mode set by the QENC\_REAQ\_MODE. However, the register QENC\_REAQ\_MODE can be used to set the board in a mode where the no re-aquisition circuit is reset automatically every pass over the image.

# 4.5 CON22 Register

Bit	Name
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	SCAN_STEP
17	SCAN_STEP
18	SCAN_STEP
19	SCAN_STEP
20	SCAN_STEP
21	SCAN_STEP
22	SCAN_STEP
23	SCAN_STEP
24	SCAN_STEP
25	SCAN_STEP
26	SCAN_STEP
27	SCAN_STEP
28	SCAN_STEP
29	SCAN_STEP
30	SCAN_STEP
31	SCAN_STEP

### SCAN\_STEP R/WO, CON22[31..16], Karbon, Neon

This bitfield controls the number of encoder pulses that must occur before a trigger is issued to the system. See SCAN\_STEP\_TRIG for more information. The Scan Step circuit takes into account the interval and re-acquisition functions.

# 4.6 CON51 Register

Bit	Name
0	QENC_COUNT
1	QENC_COUNT
2	QENC_COUNT
3	QENC_COUNT
4	QENC_COUNT
5	QENC_COUNT
6	QENC_COUNT
7	QENC_COUNT
8	QENC_COUNT
9	QENC_COUNT
10	QENC_COUNT
11	QENC_COUNT
12	QENC_COUNT
13	QENC_COUNT
14	QENC_COUNT
15	QENC_COUNT
16	QENC_COUNT
17	QENC_COUNT
18	QENC_COUNT
19	QENC_COUNT
20	QENC_COUNT
21	QENC_COUNT
22	QENC_COUNT
23	QENC_COUNT
24	QENC_PHASEA
25	QENC_PHASEB
26	QENC_DIR
27	QENC_INTRVL_IN
28	QENC_NEW_LINES
29	Reserved
30	Reserved
31	Reserved

#### **QENC\_COUNT** RO, CON51[23..0], Karbon, Neon

This bitfield displays the current quadrature encoder count.

**QENC\_PHASEA** RO, CON51[24], Karbon, Neon

This bit displays the current logic level of the A quadrature encoder phase.

**QENC\_PHASEB** RO, CON51[25], Karbon, Neon

This bit displays the current logic level of the B quadrature encoder phase.

**QENC\_DIR** RO, CON51[26], Karbon, Neon

This bit displays the current quadrature encoder direction.

QENC_DIR	Meaning
0	Direction is negative
1	Direction is positive

### **QENC\_INTRVL\_** RO, CON51[27], Karbon, Neon

#### IN

This bit indicates the current status of the quadrature encoder if the system is in interval mode (see QENC\_INTRVL\_MODE).

QENC_INTRVL_IN	Meaning
0	System is not inside the interval. Encoder counter is not between QENC_INTRVL_LL and QENC_INTRVL_UL. Lines are not being acquired.
1	System is inside the interval. Encoder counter is between QENC_INTRVL_LL and QENC_INTRVL_UL. Lines are being acquired.

### QENC\_NEW\_ LINES

### RO, CON51[28], Karbon, Neon

This bit indicates if the system is at an encoder count that corresponds to a new line. When  $QENC_NO_REAQ = 1$ , only lines that have not yet been scanned are acquired. This bit can be used to determine of new lines are being traversed, or if the system has backed up, and is revisiting old lines.

QENC_NEW_LINES	Meaning
0	The system is traversing lines that have already been visited. If QENC_NO_REAQ = 1, lines are not being acquired.
1	The system is traversing new lines. Lines are being acquired.

# Encoder Divider

# Chapter 5

## 5.1 Introduction

This section covers the encoder divider which supported on the Karbon-C and the Neon-CL. The purpose of Encoder Divider is to provide the ability to use an encoder running at one rate to drive a line scan camera at a different rate. This circuit is only useful for line scan cameras. The Encoder Divider can scale up or down the incoming encoder frequency. The encoder divider is fully programmable and is easily controlled from software and/or from camera configuration files.

The factor used to scaled the incoming encoder frequency does not have to be a whole number. For example, the encoder could be scaled by 0.03448 or 4.2666). Of course not all ration numbers in the available scaling range can be selected (there are an infinite number of them). However, a useful selection of values is available which should support most applications.

The Encoder Divider circuit takes as its input the selected encoder input (controlled by the register SELENC). The output of the encoder divider drives the same parts of the board the normal encoder usually does. The actual circuit(s) being driven depends on how the board is programmed. However, the Encoder Divider circuit can drive any of the following:

Horizontal CTAB (HCNT\_RLS\_ZERO = 1) NTG (NTG\_ONESHOT = 1, NTG\_TRIG\_MODE = 1)

Note: The Encoder Divider circuit described in this chapter replaces the previous circuit which could only divide the incoming encoder by an integer value and could not increase the encoder frequency. Please contact BitFlow if you have been using the previous on-board encoder divider.

## 5.2 Encoder Divider Details

### 5.2.1 Formula

The following formula shows the equation used to scale the encoming encoder rate into the camera's line rate:

$$F_{out} = F_{in} \frac{2^N}{M}$$

Where:

Fout = The frequency used to driver the camera or the NTG or the CTabs Fin = The encoder (input) frequency

N = An integer between 0 and 6 (set by the register  $ENC_DIV_N$ )

M = An integer between 1 and 1023 (set by the register ENC\_DIV\_M)

The above formula provides an effective scaling factor from 0.001 (N = 0, M = 1023) to 64 (N = 6, M = 1). Not every scaling factor can be acheived between these two extremes, and the scaling factors are not evenly distributed. However, a scaling factor can be generally found that meets the requirments of most applications.

### 5.2.2 Example

Let's assume that the encoder frequency (Fin) is 10 KHz and that we need an output (Fout) of ~30 KHz. This means that we need to multiply by 3. Set N = 6 and M = 21. This will a scaling factor of 3.048. The result is an effective line rate of 30.48 KHz.

### 5.2.3 Restrictions

Because the encoder divider uses a digital PLL run by a 50 MHz clock, not all encoder input frequencies can be accurately scaled. The PLL has been designed to work in most machine visions applications. Support, therefore, is provided for the following inpute frequency range:

Minimum input encoder frequency: 1.6 KHz Maximum input encoder frequency: 300 KHz

### 5.2.4 PLL Locking

The encoder divider acheives its scaling using a PLL. By default the output waveform is lock to the input input waveform. However, this locking can result in a small amount of jitter. To reduce the jitter, the output waveform can be run open loop. This mode is accessed by setting the regsiter ENC\_DIV\_OPEN\_LOOP to 1.

### 5.2.5 Handling Encoder Slow Down or Stopping

On some machine vision systems, the encoder is attached to a mechanism that may slow and/or stop. Any PLL has a limited range that it can track (based on the PLL master clock), outside of this range, the output signal can become unpredictable. The Encoder Divider circuit's master clock is 50 MHz, which makes the minimum frequency that it can accurately track around 1.6 KHz. In order to avoid this situation and handle encoder slow down/stop gracefully, the encoder divider has has limiting circuit that can be run in one of two different mode described in the following two sections.

#### Slow Tracking Mode (ENC\_DIC\_FORCE\_DC = 0)

In this mode, when the input frequence goes below the minimum of 1.6 KHz, the Encoder Divider circuit's output will continue to track the input, but the output frequency will become simple divider on the input freqency. In this mode the output will track the input using the following formula (the variables are the same as in Section 5.2.1)

$$F_{out} = \frac{F_{in}}{4M}$$

#### DC Mode (ENC\_DIV\_FORCE\_DC = 1)

In this mode, the Fin goes below 1.6 KHz, Fout will goes to DC. This means that when the input frequency goes below the minium, the camera will be frozen, acquisition will stop. The board will stay in this state until Fin goes above 1.6 KHz. This is usefull when the encoder is being driven by a stage that is travelling back and forth. At the ends of the travel when the stage changes directions, the board will not acquire.

# 5.3 Encoder Divider Control Registers

Name	Locations	Purpose
ENC_DIV_M	CON6[2718]	This controls the M factor in the Encoder Divider equation (see Section 5.2.1
ENC_DIV_N	CON19[1817]	The controls the N factor the Encoder Divider equation
ENC_DIV_FORCE_DC	CON16[27]	Controls the behavior when Fin falls below the minimum. 0 = Output runs in simple divider mode. 1 = Output goes to DC.
ENC_DIV_OPEN_LOOP	CON16[28]	Controls whether the output signal phase of the Encoder Divider is lock to the intput or is allowed to free run. 0 = Output phased locked to input. 1 = Ouput runs open loop.
ENC_DIV_FCLK_SEL	CON16[3129]	Reserved for future support for alternate Encoder Divider PLL Master clock frequencies. Currently must be set to 0, which selects 50 MHz clock.

The following table summarizes the registers:

# System Status

# Chapter 6

## 6.1 Introduction

This chapter describes the system status report that the board supplies through its registers. The system status will help the users in setting up their system: the camera, the frame grabber, the cabling, the I/O and the software. The list of the status bits is given in the Table 6-1. If more information is available for a given specification there will be an entry in the column marked "Details". In addition, all of these registers are also described in Register Map chapter of this manual.

#### Table 6-1 Status Bits

Status Bits	Function and Relationship	Register	Details
AQSTAT	Acquisition status	CON3	Section 7.6
FACTIVE	Acquisition status, vertical active	CON3	Section 6.2
FCOUNT	Acquisition status, 3-bit frames counter	CON3	Section 6.2
LCOUNT	Camera status, LEN is toggling	CON4	Section 6.3
PCOUNT	Camera status, PCLK is toggling	CON4	Section 6.3
FENCOUNT	Camera status, FEN is toggling	CON4	Section 6.3
RD_TRIG_DIFF/TTL/OPTO	Trigger status	CON5	Section 6.4
RD_ENC_DIFF/TTL/OPTO	Encoder status	CON5	Section 6.4
TRIG_QUALIFIED	Selected trigger status	CON6	Section 6.5
VCOUNT	Acquisition status, VCTAB cycling	CON6	Section 6.6
HCOUNT	Acquisition status, HCTAB cycling	CON6	Section 6.6
LINES_TOGO	Acquisition status, current line in frame	CON19	Section 6.6
FIFO_EQ	Camera status, video value	CON20	Section 6.7
DEST_ADD	DMA running	CON22	Section 6.8

# 6.2 FACTIVE, FCOUNT

FACTIVE is 1 during the active vertical. It works for both area scan and line scan comeras. For both line scan and area scan cameras there is always a vertical size defined by ALPF.

FCOUNT is a 3-bit frame counter that is incremented by the rising edge of FACTIVE. It can be used to track acquisition, especially in triggered modes. FCOUNT works for both area scan and line scan cameras.

# 6.3 PCOUNT, LCOUNT, FENCOUNT

These three registers give an indication of the status the camera connected to the main connector:

PCOUNT is a 2-bit counter clocked by the camera's PCLK. Reading a constant value from this register indicates that the camera's clock does not reach the acquisition circuitry.

LCOUNT is a 2-bit counter clocked by the camera's LEN. Reading a constant value from this register indicates that the camera's LEN does not reach the acquisition circuitry.

FENCOUNT is a 2-bit counter clocked by the camera's FEN. Reading a constant value from this register indicates that the camera's FEN does not reach the acquisition circuitry.

# 6.4 RD\_TRIG\_DIFF/TTL/OPTO, RD\_ENC\_DIFF/TTL/OPTO

The level of all three trigger and all three encoder inputs can be read. This helps establish connection with external industrial equipment.

# 6.5 TRIG\_QUALIFIED

The bit TRIG\_QUALIFIED is the current active state of the current selected trigger. The trigger is selected by the SEL\_TRIG register. Each individual input can be monitored via the corresponding RD\_TRIG\_XXX bit, but the TRIG\_QUALIFIED always reports the state of the trigger input that is current being used by the acquisition circuitry.

# 6.6 VCOUNT, HCOUNT, LINES\_TOGO

These three registers give feedback about the operation of the horizontal and vertical CTABs. VCOUNT is the address counter of the VCTAB. This register indicates the current VCTAB address.

HCOUNT is the 2 LSB of the HCTAB address counter. This register indicates only if the HCTAB is cycling. Reading a constant value on HCOUNT indicates that the HCTAB address is stuck.

LINES\_TOGO specifies how more many lines there are till the end of the frame.

# 6.7 FIFO\_EQ

This register gives the 8-bit value of the video from the first eight bits of the main connector. It is helpful to determine if the camera is reacting to light. Covering the camera's lens will yield a low value in this register. Pointing the camera to a light source will yield a high value in this register.

# 6.8 DEST\_ADD

This register gives the DMA destination address. During acquisition, this register should change. Reading a constant value from this register suggests that the DMA operation is not progressing.

# **Camera Control Registers**

# Chapter 7

# 7.1 Introduction

This section enumerates all of the bitfields in all of the registers used to control the acquisition and external I/O. If you compare the Alta, Karbon, Neon and R64 manuals you will see that almost all of these registers are the same. There are only a few bits that are different between these two models, these will be indicated in the bitfield definitions. Registers that are related to DMA operations, which are different between the Alta, Neon, R64 and the Karbon families, have their own chapters.

All of the registers are 32 bits wide. These wide registers are named CON0, CON1, etc. Each registers is broken into one or more bitfields. Bitfields can be from one to 32 bits wide. Each bitfield controls a specific function on the board.

## 7.2 Bitfield definitions

### 7.2.1 Example Bitfield Definition

Here is what each bitfield definition looks like:

BITFIELD R/W, CON0[7..0], Alta, Karbon, Neon, R64

BitField discussion.

### 7.2.2 Bitfield Definition Explanation.

The definitions is broken into three sections (see Table 7-1).

Section	Meaning
Bitfield name	This is the name of the bitfield. This name is use to program this bitfield from software or from within and camera config- uration file. When programming bitfields from software using a Peek or Poke function, the bitfield is preceded with "REG_". For example the bitfield CFREQ is referred to in software as REG_CFREQ.
Bitfield details	This section describes how the bitfield is accessed. The first part describes the how the bits can be accessed. For exam- ple R/W means the register can be both read and writen. See the Table 7-2 for details. The second part is the wide reg- ister that the bitfield is located in. In the example above this bitfield is in CON0. Following the wide register name is a bitfield location description, in hardware engineering for- mat. For example, [70], means the bitfield has 8 bits, loca- tion in positions 0 to 7. Finally this section also indicates if the register is specific to only one product family.
Bitfield discussion	This section explains the purposed of the bitfield in detail. Usually meaning of every possible value of the bitfield is listed.

### Table 7-1 Bitfield Sections.

Table 7-2 explains the abbreviations used in the bitfield definitions.

Access	Meaning
R/W	Bitfield can be read and written.
RO	Bitfield can only be read. Writing to this bit has no effect.
WO	Bitfield can only be written. Reading from this bit will return meaning- less values.
Karbon	This bitfield is functional on the Karbon.
Neon	This bitfield is functional on the Neon
R64	This bitfield is functional on the R64 family.
Alta	This bitfield is functional on the Alta family.

# 7.3 CON0 Register

Bit	Name
0	CFGDATA
1	CFGSTATUS
2	CFGEN
3	CFGDONE
4	CFGCLOCK
5	FW_7MHZ
6	Reserved
7	POCL_EN
8	CFREQ
9	CFREQ
10	CFREQ
11	Reserved
12	L_CLKCON
13	L_CLKCON
14	SEL_UCLKC_7MHZ
15	RELOAD_FPGA
16	FW_SEL
17	FW_SEL
18	FW_SEL
19	CPLD_MODE
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

CFGDATA	R/W, CON0[0], R64
	Used for downloading firmware.
CFGSTATUS	R/W, CON0[1], R64
	Used for downloading firmware.
CFGEN	R/W, CON0[2], R64
	Used for downloading firmware.
CFGDONE	R/W, CON0[3], R64
	Used for downloading firmware.
CFGCLOCK	R/W, CON0[4], R64
	Used for downloading firmware.
FW_7MHZ	RO, CON0[5], Alta, Karbon, Neon, R64
	If this bit is set, then the board has the update Firmware which can program the fre- quency of the UART clock to 7.3 MHz. If this bit is zero, then the board has the original firmware and the UART can only be driver by an 8 MHz clock. See also the bit: SEL_ UCLK_7MHz.
POCL_EN	R/W, CON0[7], Neon

This bit turns the PoCL Safe Power system. This bit must be set to one in order to enable power to PoCL cameras. However, the system uses the Safe Power system, so a number of conditions must be met before power is actually applied to the camera.

### CFREQ R/W, CON0[10..8], Alta, Karbon, Neon, R64

These bits control the frequency of the CLOCK generated on-board.

Frequency
DC
3.75 MHz
7.5 MHz
15 MHz
24 MHz
30 MHz
48 MHz
60 MHz

### L\_CLKCON R/W, CON0[13..12], Alta, Karbon, Neon, R64

These bits control the local bus clock frequency. For normal operation, this register should always be set for 0. The other codes are for test/diagnostics.

L_CLKCON	Frequency
0 (000b)	60 MHz
1 (001b)	48 MHz
2 (010b)	24 MHz
3 (011b)	Reserved

SEL\_UCLK\_R/W, CON0[14], Alta, Karbon, Neon, R647MHZThis is the second second

This bit selects the frequency that is used to driver the UART for serial communications. This functionality is only available on boards with update firmware. The bit FW\_ 7MHZ can be used to check the version of the firmware.

SEL_UCLK_ 7MHZ	Frequency
0	8 MHz
1	7.3 MHz

#### **RELOAD\_FPGA** WO, CON0[15], Karbon, Neon

Writing to this bit causes the FGPA to be reloaded from Flash memory. This bit should only be accessed from the driver as the PCI Configuration space is overwritten by this operation. This is not a user programmable bit.

#### FW\_SEL R/W, CON0[18..16], Alta, Karbon, Neon, R64

These bits are used to select different modes for a given type of firmware. For each major type of CCD tap configuration, there is a separate firmware file that is downloaded to the board. However, in some cases different manufacturers chose slightly different ways to implement the same tap configuration. In these cases this bitfield is used to select between the different modes. As the meaning for this bitfield differ for each firmware file, and these bits are rarely used, the specific definitions of this bitfield are not enumerated here.

#### CPLD\_MODE R/W, CON0[19], Neon

On the Neon, the CPLD used to load the FPGAs has two modes. This bit is used to set the mode. This is not a user programmable bit.

# 7.4 CON1 Register

Bit	Name
0	VCNT_RLS_ZERO
1	VCNT_RLS_ZERO
2	VCNT_RLS_ZERO
3	VCNT_RST
4	VCNT_RST
5	VCNT_RST
6	VCNT_LD
7	VCNT_LD
8	VCNT_LD
9	VCNT_RLS_STK
10	VCNT_RLS_STK
11	VCNT_RLS_STK
12	ABORT_CON
13	ABORT_CON
14	ABORT_CON
15	NO_VB_WAIT
16	ACQ_CON
17	ACQ_CON
18	ACQ_CON
19	FREEZE_CON
20	FREEZE_CON
21	FREEZE_CON
22	ACQ_SAFETY
23	NO_RULE
24	INT_CTAB
25	INT_OVSTEP
26	INT_HW
27	INT_TRIG
28	INT_SER
29	INT_QUAD
30	INT_TRIGCON
31	INT_TRIGCON

### VCNT\_RLS\_ ZERO

### R/W, CON1[2..0], Alta, Karbon, Neon, R64

This register controls how the Vertical CTAB counter (VCOUNT) is released from zero.

VCNT_RLS_ZERO	Meaning
0 (000b)	Normal operation. VCOUNT does not stick at zero.
1 (001b)	Edge Mode - VCOUNT sticks at zero. VCOUNT is released from zero by the leading edge of the trigger.
2 (010b)	Level Mode - VCOUNT sticks at zero only if the trig- ger is de-asserted. If trigger is asserted, then VCOUNT does not stick at zero. VCOUNT is released from zero by the leading edge of trigger.
3 (011b)	Reserved.

### VCNT\_RST R/W, CON1[5..3], Alta, Karbon, Neon, R64

This register controls how the Vertical CTAB counter (VCOUNT) is reset to zero. In all modes the VCOUNT will also be reset by any of the following four signals/events:

The SW\_RESET The ABORT command The RST\_HVCOUNT bit in CON4 VCOUNT reaching a 1 in the VRESET CTAB

VCNT_RST	Meaning
0 (000b)	VCOUNT is reset by the End of Vertical Acquisition Window or by the VRESET column in the VCTAB.
1 (001b)	VCOUNT is reset by the de-assertion of the trigger (triggered termination) or the end of VAW or by the VRESET column in the VCTAB.
2 (010b)	VCOUNT is reset by the VRESET column in the VCTAB.
3 (011b)	VCOUNT is reset by the assertion of FEN or by the VRESET column in the VCTAB.
4 (100b)	VCOUNT is reset by the de-assertion of the trigger, or by the VRESET column in the VCTAB.

### VCNT\_LD R/W, CON1[8..6], Alta, Karbon, Neon, R64

This registers controls how the Vertical CTAB counter (VCOUNT) is loaded with the 8000h value.

VCNT_LD	Meaning
0 (000b)	No load operation performed.
1 (001b)	VCOUNT is loaded at the assertion of FEN qualified with the ENVLOAD column in the VCTAB.
2 (010b)	VCOUNT is loaded at the assertion of FEN.
3 (011b)	VCOUNT is loaded at the assertion of trigger.

### VCNT\_RLS\_STK R/W, CON1[11..9], Alta, Karbon, Neon, R64

This register controls the stick/release of the VCOUNT to/from address 7ff0h.

VCNT_RLS_STK	Meaning
0	VCOUNT does not stick at 7FF0h.
1	VCOUNT sticks at 7FF0h. It will be released from that address by a LOAD or RESET operation.

### ABORT\_CON R/W, CON1[14..12], Alta, Karbon, Neon, R64

This register controls the ABORT operation of the Acquisition State Machine.

ABORT_CON	Meaning
0	Acquisition will be aborted by a host command.
1	Acquisition will be aborted by de-assertion of trigger of by a host ABORT command.

#### NO\_VB\_WAIT R/W, CON1[15], Alta, Karbon, Neon, R64

This bit has the following properties.

NO_VB_WAIT	Meaning
0	Wait for the Vertical Active Window before executing the Head Tag Quad.
1	Do not wait for the Vertical Active Window for exe- cuting the Head Tag Quad.

### ACQ\_CON R/W, CON1[18..16], Alta, Karbon, Neon, R64

This register controls the execution of the acquisition command.

ACQ_CON	Meaning
0 (000b)	Acquisition is initiated by host writing the command.
1 (001Ь)	The acquisition command written by host will start executing at assertion of trigger (triggered acquisi- tion). Only the SNAP and GRAB commands require a trigger to be latched. The FREEZE command will work normally.
2 (010b)	While the GRAB command is on, a frame will be acquired at the assertion of trigger.
3 (011b)	Continuous acquisition mode. Host commands are ignored. Data will be acquired continuously as long as the TRIGGER is asserted.
4 (100b)	A GRAB command will be issued every time the TRIGGER asserts. The GRAB can be terminated after a programmable number of frames by using the AQ_ COUNT register. This mode initiated by manually issuing a GRAB command, and terminated after manually issuing a FREEZE. The AQ_STAT register will return GRAB mode as long as the board is in this mode, regardless of whether the board is actually grabbing or not.

#### FREEZE\_CON R/W, CON1[21..19], Alta, Karbon, Neon, R64

This register controls the FREEZE operation.

FREEZE_CON	Meaning
0 (000b)	FREEZE initiated by host command.
1 (001b)	FREEZE initiated by the Acquisition Counter or by the host command.
2 (010b)	FREEZE initiated by the de-assertion of trigger or by the host command.

#### ACQ\_SAFETY R/W, CON1[22], Alta, Karbon, Neon, R64

Future use

### **NO\_RULE** R/W, CON1[23], R64

Test/diagnostic bit. For normal operation this bit should always be set to 0. When set to 1, the DMA engine will DMA data at maximum speed, regardless of wether the data is valid.

#### **INT\_CTAB** R/W, CON1[24], Alta, Karbon, Neon, R64

This interrupt will be set by the interrupt bit in the VCTAB or by the host writing to this bit. The interrupt will be enabled if its corresponding mask, ENINT\_CTAB, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 1.

INT_CTAB	Meaning
0	No interrupt from CTAB
1	Interrupt from CTAB asserted.

### **INT\_OVSTEP** R/W, CON1[25], Alta, Karbon, Neon, R64

This interrupt will be set if an overflow occurred in the FIFO or by the host writing to this bit. The interrupt will be enabled if its corresponding mask, ENINT\_OVSTP, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 2.

INT_OVSTP	Meaning
0	No interrupt from overflow
1	Interrupt from overflow asserted.

### INT\_HW R/W, CON1[26], Alta, Karbon, Neon, R64

This interrupt will be set by a hardware exception, a loss of sync or by the host writing to this bit. The interrupt will be enabled if its corresponding mask, ENINT\_HW, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 3.

INT_HW	Meaning
0	No interrupt from HW
1	Interrupt from HW asserted.

### INT\_TRIG R/W, CON1[27], Alta, Karbon, Neon, R64

This interrupt will be set by a trigger edge or by the host writing to this bit (see register INT\_TRIGCON below). The interrupt will be enabled if its corresponding mask, ENINT\_TRIG, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 4.

INT_TRIG	Meaning
0	No interrupt from trigger
1	Interrupt from trigger asserted.

INT\_SER RO, CON1[18], Alta, Karbon, Neon, R64

This interrupt will be set by the on board UART that implements the serial communication protocol. The interrupt will be enabled if its corresponding mask, ENINT\_SER, has been set to 1. This interrupt can be cleared by the host writing to the UART.

INT_SER	Meaning
0	No interrupt from UART
1	Interrupt from UART asserted.

**INT\_QUAD** R/W, CON1[29], Alta, Karbon, Neon, R64

This interrupt will be set by a DMA QUAD or by the host writing to this bit. The interrupt will be enabled if its corresponding mask, ENINT\_QUAD, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 5.

INT_QUAD	Meaning
0	No interrupt from QUAD
1	Interrupt from QUAD asserted.

## INT\_TRIGCON R/W, CON1[31..20], Alta, Karbon, Neon, R64

This register controls the trigger edge that will cause an interrupt:

INT_TRIGCON	Meaning
0 (00b)	reserved
1 (01b)	Assert interrupt on rising edge of trigger.
2 (10b)	Assert interrupt on falling edge of trigger.
3 (11b)	Assert interrupt on both the rising and the falling edge of the trigger.

# 7.5 CON2 Register

Bit	Name
0	HCNT_RLS_ZERO
1	HCNT_RLS_ZERO
2	HCNT_RLS_ZERO
3	HCNT_RST
4	HCNT_RST
5	HCNT_RST
6	HCNT_LD
7	HCNT_LD
8	HCNT_LD
9	HCNT_RLS_STK
10	HCNT_RLS_STK
11	HCNT_RLS_STK
12	RST_HVCOUNT
13	RST_DPM_ADDR
14	CTABHOLD
15	Reserved
16	CC1_CON
17	CC1_CON
18	CC1_CON
19	CC2_CON
20	CC2_CON
21	CC2_CON
22	CC3_CON
23	CC3_CON
24	CC3_CON
25	CC4_CON
26	CC4_CON
27	CC4_CON
28	CMDWRITE
29	CMDWRITE
30	CMDWRITE
31	QTBSRC

#### HCNT\_RLS\_ R/W, CON2[2..0], Alta, Karbon, Neon, R64 ZERO

This register controls the release of the HCOUNT from zero.

HCNT_RLS_ZERO	Meaning
0	HCOUNT does not stop at 000h.
1	HCOUNT stops at 000h. It will be released by the assertion of the encoder.

### HCNT\_RST R/W, CON2[5..3], Alta, Karbon, Neon, R64

This register controls the reset of the HCOUNT. In all cases, the HCOUNT will also be reset by any of the following functions:

SW\_RESET RST\_HVCOUNT ABORT command.

\_

HCNT_RST	Meaning
0(000b)	HCOUNT will be reset by the end of the Horizontal Active Window.
1 (001ь)	HCOUNT will be reset by the assertion of FEN or the HRESET from the HCTAB.
2 (010b)	HCOUNT will be reset by the HRESET in the HCTAB.

### HCNT\_LD R/W, CON2[8..6], Alta, Karbon, Neon, R64

This register controls the loading of the HCOUNT with 2000h.

HCNT_LD	Meaning
0 (000b)	HCOUNT will not be loaded.
1 (001b)	HCOUNT will be loaded by assertion of LEN if the ENHLOAD function in the HCTAB is set to 1.
2 (010b)	HCOUNT will be loaded by assertion of encoder if the ENHLOAD function in the HCTAB is set to 1.
#### HCNT\_RLS\_STK R/W, CON2[11..9], Alta, Karbon, Neon, R64

This register controls the HCOUNT sticking at 1FF0h.

HCNT_RLS_STK	Meaning
0	HCOUNT will not stick at 1FF0h.
1	HCOUNT will stick at 1FF0h. It will be released by a load or reset command.

**RST\_HVCOUNT** WO, CON2[12], Alta, Karbon, Neon, R64

This bit has the following properties.

RST_HVCOUNT	Meaning
0	Normal operation for HCOUNT, VCOUNT
1	Reset HCOUNT, VCOUNT

RST\_DPM\_ADDR WO, CON2[13], Alta, Karbon, Neon, R64

This bit has the following properties.

RST_DPM_ADDR	Meaning
0	Normal operation for DPM_ADDR
1	Reset DPM_ADDR

CTABHOLD R/W, CON2[14], Alta, Karbon, Neon, R64

This bit has the following properties.

CTABHOLD	Meaning
0	Normal operation for CTABs
1	Freeze outputs and operation of CTABs

# CCI\_CON R/W, CON2[18..16], Alta, Karbon, Neon, R64

This register selects the signal steered to the CC1.

CC1_CON	Signal steered to CC1
0 (000b)	CT0 from CTAB
1 (001b)	CT1 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	Free running signal generated on-board
4 (100b)	Trigger input
5 (101b)	GPIN0
6 (110b)	0
7 (111b)	1

# CC2\_CON R/W, CON2[21..19], Alta, Karbon, Neon, R64

This register selects the signal steered to the CC2.

CC2_CON	Signal steered to CC2
0 (000b)	CT0 from CTAB
1 (001b)	CT1 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	Free running signal generated on-board
4 (100b)	Trigger Input
5 (101b)	GPIN0
6 (110b)	0
7 (111b)	1

# CC3\_CON R/W, CON2[24..22], Alta, Karbon, Neon, R64

This register selects the signal steered to the CC3.

CC3_CON	Signal steered to CC3
0 (000b)	CT0 from CTAB
1 (001b)	CT1 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	Free running signal generated on-board
4 (100b)	Trigger input
5 (101b)	GPIN0
6 (110b)	0
7 (111b)	1

# CC4\_CON R/W, CON2[27..25], Alta, Karbon, Neon, R64

This register selects the signal steered to the CC4. Note that this CC control is slightly different than the previous three. CC4 can be controlled by CT3. This changes allows all four CTs to be tied to a CC.

CC4_CON	Signal steered to CC4
0 (000b)	CT0 from CTAB
1 (001b)	CT1 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	Free running signal generated on-board
4 (100b)	Trigger input
5 (101b)	CT3 from CTAB
6 (110b)	0
7 (111b)	1

## CMDWRITE R/W, CON2[30..28], Alta, Karbon, Neon, R64

This registers selects the interrupt bit to be modified by the host. While an interrupt's code is set, that interrupt can not be asserted by its source. It can be modified only by the host. This mechanism allows to perform reliable read-modify-write cycles.

CMDWRITE	Interrupt allowed for host access
0 (000b)	No interrupt can be accessed by host
1 (001b)	INT_CTAB
2 (010b)	INT_OVSTP
3 (011b)	INT_HW
4 (100b)	INT_TRIG
5 (101b)	INT_QTAB
6 (110b)	INT_EOF
7 (111b)	reserved

#### **QTBSRC** RO, CON2[31], Alta, Karbon, Neon, R64

Always read back 1.

Bit	Name
0	AQCMD
1	AQCMD
2	AQSTAT
3	AQSTAT
4	FACTIVE
5	FCOUNT
6	FCOUNT
7	FCOUNT
8	REV_DCC
9	REV_DCC
10	REV_DCC
11	REV_DCC
12	REV_DCC
13	REV_DCC
14	REV_DCC
15	REV_DCC
16	REV_DCC
17	REV_DCC
18	REV_DCC
19	REV_DCC
20	REV_DCC
21	REV_DCC
22	REV_DCC
23	REV_DCC
24	AUX_DETECT
25	GPIN0
26	GPIN1
27	GPIN2
28	GPIN3
29	GPIN4
30	SW
31	SW

## AQCMD R/W, CON3[1..0], Alta, Karbon, Neon, R64

This register is the acquisition command to be executed in the next frame.

AQCMD	Meaning
0 (00b)	FREEZE
1 (01b)	ABORT
2 (10b)	SNAP
3 (11b)	GRAB

# AQSTAT RO, CON3[3..2], Alta, Karbon, Neon, R64

The AQSTAT register describes the acquisition command currently being executed.

AQSTAT	Meaning
0 (00b)	FREEZE
1 (01b)	ABORT
2 (10b)	SNAP
3 (11b)	GRAB

FACTIVE RO, CON3[4], Alta, Karbon, Neon, R64

	FACTIVE	Meaning	
	0	Camera outside the Vertical Acquisition Window	
	1	Camera within the Vertical Acquisition Window	
FCOUNT	RO, CON3[75], Alta	RO, CON3[75], Alta, Karbon, Neon, R64	
	This is a 3-bit modulo-8 counter. The counter is incremented by the start of the Verti- cal Acquisition Window. It is used as a debug/diagnostic tool.		
REV_DCC	WO, CON3[238], Alta, Karbon, Neon, R64		
	FW revision.		

AUX_DETECT	ETECT RO, CON3[24], Karbon	
	This bit is set to a one if the Karbon auxiliary board is attached.	
<b>GPIN0</b>	RO, CON3[25], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	
GPIN1	RO, CON3[26], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	
GPIN2	RO, CON3[27], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	
GPIN3	RO, CON3[28], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	
GPIN4	RO, CON3[29], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	
SW	RO, CON3[3130], Alta, Karbon, Neon, R64	
	Controlled by inputs on the IO connector. The logical value applied to the corre- sponding pin will be reflected in this register. See also Section 10.4 for interfacing information.	

# 7.7 CON4 Register

Bit	Name
0	ENINT_CTAB
1	ENINT_OVSTEP
2	ENINT_HW
3	ENINT_TRIG
4	ENINT_SER
5	ENINT_QUAD
6	EOF_IN_AQ
7	INT_ANY
8	ENINT_ALL
9	AUX_CAM
10	GPOUT0
11	GPOUT1
12	GPOUT2
13	GPOUT3
14	GPOUT4
15	GPOUT5
16	GPOUT6
17	RST_SER
18	OVS
19	RST_OVS
20	CL_DISABLE
21	LCOUNT
22	LCOUNT
23	PCOUNT
24	PCOUNT
25	FENCOUNT
26	FENCOUNT
27	POP_TOSS
28	PUMP_OFF
29	DMA_BUSY
30	HAW_START
31	VAW_START

### ENINT\_CTAB R/W, CON4[0], Alta, Karbon, Neon, R64

This bit has the following properties.

	ENINT_CTAB	Meaning
	0	CTAB interrupt disabled
	1	CATB interrupt enabled
ENINT_OVSTEP	P R/W, CON4[1], Alta, Karbon, Neon, R64	
	This bit has the following pr	operties.
	ENINT_OVSTEP	Meaning
	0	OVERSTEP interrupt disabled
	1	OVERSTEP interrupt enabled
ENINT_HW	R/W, CON4[2], Alta, Karbon, Neon, R64	
	This bit has the following properties.	
	ENINT_HW	Meaning
	0	HW interrupt disabled
	1	HW interrupt enabled
ENINT_TRIG	R/W, CON4[3], Alta, Karb	on, Neon, R64

This bit has the following properties.

ENINT_TRIG	Meaning
0	Trigger interrupt disabled
1	Trigger interrupt enabled

#### ENINT\_SER R/W, CON4[4], Alta, Karbon, Neon, R64

This bit has the following properties.

ENINT_SER	Meaning
0	UART interrupt disabled
1	UART interrupt enabled

#### ENINT\_QUAD R/W, CON4[5], Alta, Karbon, Neon, R64

This bit has the following properties.

ENINT_QUAD	Meaning
0	QUAD interrupt disabled
1	QUAD interrupt enabled

EOF\_IN\_AQ R/W, CON4[6], Alta, Karbon, Neon, R64

This bit enables gating the INT\_EOF interrupt with acquisition. This functionality makes it easier to write software that relies on the INT\_EOF interrupt. Without this functionality, the INT\_EOF interrupt would occur continuously based on the camera or trigger's timing, even if the board is not currently acquiring. In this case, the software will have to flush these interrupts out of the queue before starting acquisition. With this functionality enable, interrupts only occur during acquisition..

EOF_IN_AQ	Meaning
0	INT_EOF interrupt will be asserted unconditionally at the end of the frame.
1	INT_EOF interrupt will be asserted at the end of the frame only if the board is acquiring, i.e. only during SNAP/GRAB states.

#### **INT\_ANY** RO, CON4[7], Alta, Karbon, Neon

On the products that use the PLDA engine, this bit indicates that an interrupt was emitted by the board. This bit can be checked first to see if some event caused the interrupt, before inquiring other bits to see the actual cause of the interrupt.

#### **ENINT\_ALL** R/W, CON4[8], Alta, Karbon, Neon

This bit enables or disables all interrupts on boards that use the PLDA engine.

#### **AUX\_CAM** R/W, CON4[8], R64

Future use.

#### **GPOUTO** R/W, CON4[10], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

GPOUT1 R/W, CON4[11], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

GPOUT2 R/W, CON4[12], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

GPOUT3 R/W, CON4[13], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

**GPOUT4** R/W, CON4[14], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

GPOUT5 R/W, CON4[15], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

GPOUT6 R/W, CON4[16], Alta, Karbon, Neon, R64

The value written in this register will be reflected on the IO connector. See also CON8 for signals steered to the GPOUTs and Section 10.5 for electrical interfacing.

# **RST\_SER** R/W, CON4[17], Alta, Karbon, Neon, R64

This bit has the following properties.

	RST_SER	Meaning	
	0	UART normal operation	
	1	UART's reset line asserted	
ovs	RO, CON4[18], Alta, Karbon, Neon, R64		
	This is a latched overstep bit.		
	OVS	Meaning	
	0	No overstep occurred since this bit was cleared	
	1	At least one overstep occurred since this bit was cleared	
RST_OVS	R/W, CON4[19], Alta, Karbon, Neon, R64		
	This bit has the following properties.		
	RST_OVS	Meaning	
	0	OVS bit in normal operation	
	1	OVS bit is reset	
CL_DISABLE	R/W, CON4[20], Alta, Karbon, Neon, R64		
	This bit enables/disables the CL chips; used for diagnostics. For normal operation this bit should always be set to 0.		
	CL_DISABLE	Meaning	
	0	All CL receivers are enabled.	
	1	All CL receivers are disabled.	
LCOUNT	RO, CON4[2221], Alta, Karbon, Neon, R64		
	This is a 2-bit counter clocked by the LEN supplied by the Camera Link main connec- tor. Reading this counter and observing changes between reads indicates an active LEN.		

#### PCOUNT RO, CON4[24..23], Alta, Karbon, Neon, R64

This is a 2-bit counter clocked by the PCLK supplied by the Camera Link main connector. Reading this counter and observing changes between reads indicates an active PCLK.

FENCOUNT RO, CON4[26..25], Alta, Karbon, Neon, R64

This is a 2-bit counter clocked by the FEN supplied by the Camera Link main connector. Reading this counter and observing changes between reads indicates an active FEN.

POP\_TOSS R/W, CON4[27], R64, Alta, Karbon, Neon, R64

For normal operation this bit should be set to 0. It is used for high-level clean-up.

POP_TOSS	Meaning
0	DMA engine's normal operation.
1	DMA engine flushes the receive FIFO without exe- cuting any QUADs.

PUMP\_OFF R/W, CON4[28], R64, Alta, Karbon, Neon, R64

For normal operation this bit should be set to 0. It is used for high-level clean-up.

PUMP_OFF	Meaning
0	DMA engine's normal operation.
1	Inhibit DMA operation.

#### DMA\_BUSY RO, CON4[29], R64, Alta, Karbon, Neon, R64

This bit indicates the state of the DMA engine.

DMA_BUYS	Meaning
0	DMA engine is idle.
1	DMA engine is currently DMAing data.

### HAW\_START R/W, CON4[30], Alta, Karbon, Neon, R64

This bit has the following properties.

HAW_START	Meaning
0	The start of the Horizontal Active Window (HAW) is controlled by the start of the LEN.
1	The start of the Horizontal Active Window is con- trolled by the HSTART column in the HCTAB.

#### VAW\_START R/W, CON4[31], Alta, Karbon, Neon, R64

This bit has the following properties.

VAW_START	Meaning
0	The start of the Vertical Active Window (VAW) is con- trolled by the start of the FEN.
1	The start of the Vertical Active Window is controlled by the VSTART column in the VCTAB.

Bit	Name
0	SEL_TRIG
1	SEL_TRIG
2	TRIGPOL
3	SW_TRIG
4	SELENC
5	SELENC
6	ENCPOL
7	SW_ENC
8	RD_TRIG_DIFF
9	RD_TRIG_TTL
10	RD_TRIG_OPTO
11	RD_ENC_DIFF
12	RD_ENC_TTL
13	RD_ENC_OPTO
14	TRIGGER_DELAY
15	TRIGGER_DELAY
16	TRIGGER_DELAY
17	TRIGGER_DELAY
18	TRIGGER_DELAY
19	TRIGGER_DELAY
20	TRIGGER_DELAY
21	TRIGGER_DELAY
22	TRIGGER_DELAY
23	TRIGGER_DELAY
24	ENINT_EOF
25	INT_EOF
26	RD_FEN
27	CCSYNC
28	CCSYNC
29	CCSYNC
30	EN_TRIGGER
31	EN_ENCODER

# SEL\_TRIG R/W, CON5[1..0], Alta, Karbon, Neon, R64

Controls the source of the internal trigger signal.

SEL_TRIG	Meaning
0 (00Ь)	The trigger used by the board is the differential trig- ger on the IO connector.
1 (01b)	The trigger used by the board is the TTL trigger on the IO connector.
2 (10b)	The trigger used by the board is the opto-coupled trigger on the IO connector.
3 (11b)	The FEN signal on the CL1 connector will be used as trigger. When this mode is used, the register FEN-POL is used to control the polarity of the trigger signal.

#### TRIGPOL R/W, CON5[2], Alta, Karbon, Neon, R64

This bit has the following properties.

TRIGPOL	Meaning
0	Trigger is asserted on the rising edge.
1	Trigger is asserted on the falling edge.

### SW\_TRIG R/W, CON5[3], Alta, Karbon, Neon, R64

The SW trigger is OR-ed with the external trigger. The polarity of the SW trigger is always active-HI. TRIGPOL has no effect on the SW trigger.

SW_TRIG	Meaning
0	SW trigger de-asserted.
1	SW trigger asserted.

#### SELENC R/W, CON5[5..4], Alta, Karbon, Neon, R64

This bitfield has the following properties.

SELENC	Meaning
0 (00Ь)	The encoder used by the board is the differential encoder on the IO connector.
1 (01Ь)	The encoder used by the board is the TTL encoder on the IO connector.
2 (10b)	The encoder used by the board is the opto-coupled encoder on the IO connector.
3 (11b)	Reserved

**ENCPOL** R/W, CON5[6], Alta, Karbon, Neon, R64

This bitfield has the following properties.

ENCPOL	Meaning
0	Encoder is asserted on rising edge.
1	Encoder is asserted on falling edge.

SW\_ENC R/W, CON5[7], Alta, Karbon, Neon, R64

The SW encoder is OR-ed with the external encoder. The polarity of the SW encoder is always active-HI. ENCPOL has no effect on the SW encoder

SW_ENC	Meaning
0	SW encoder de-asserted.
1	SW encoder asserted.

**RD\_TRIG\_DIFF** RO, CON5[8], Alta, Karbon, Neon, R64

This register reflects the status of the differential trigger input on the IO connector, pins 1,2.

**RD\_TRIG\_TTL** RO, CON5[9], Alta, Karbon, Neon, R64

This register reflects the status of the TTL trigger input on the IO connector, pin 3.

#### RD\_TRIG\_OPTO RO, CON5[10], Alta, Karbon, Neon, R64

This register reflects the status of the opto-coupled trigger input on the IO connector, pins 4,5.

**RD\_ENC\_DIFF** RO, CON5[11], Alta, Karbon, Neon, R64

This register reflects the status of the differential encoder input on the IO connector, pins 7,8.

**RD\_ENC\_TTL** RO, CON5[12], Alta, Karbon, Neon, R64

This register reflects the status of the TTL encoder input on the IO connector, pin 9

**RD\_ENC\_OPTO** RO, CON5[13], Alta, Karbon, Neon, R64

This register reflects the status of the opto-coupled encoder input on the IO connector, pins 10, 11.

TRIGGER\_DELAY R/W, CON5[23..14], Alta, Karbon, Neon, R64

The number N written in this register will delay the trigger by 8N lines.

**ENINT\_EOF** R/W, CON4[24], Alta, Karbon, Neon, R64

This bitfield has the following properties.

ENINT_EOF	Meaning
0	End of frame interrupt disabled.
1	End of frame interrupt enabled.

INT\_EOF R/W, CON4[25], Alta, Karbon, Neon, R64

This interrupt will be set by the acquisition state machine at the end of the frame (end of VAW). This ordinarily corresponds to the camera's end of frame. However, if the board is in start-stop triggered mode, this interrupt with also occur when the trigger de-asserts. The host writing a 1 to this bit will also cause and interrupt. The interrupt

will be enabled if its corresponding mask, ENINT\_EOF, has been set to 1. This interrupt can be cleared by the host writing a 0 to this location. For the host to be able to write to this location, the CMDWRITE code must be set to 6.

	INT_TRIG	Meaning	
	0	No interrupt from end of frame.	
	1	Interrupt from end of frame.	
RD_FEN	RO, CON5[26], Alta, Karbon, Neon, R64		
	This register reflects th	ne status of the FEN signal on the CL1 connector.	
CC_SYNC	R/W, CON5[2926], Alta, Karbon, Neon, R64		
	This register controls l	now the CC outputs are synchronized.	
	CCSYNC	Meaning	
	0 (000b)	CCs are not synchronized	
	1 (001b)	CCs are synchronized to the pixel clock of the pri- mary camera	
	2 (010b)	CCs are synchronized to the pixel clock of the sec- ondary camera	
	3 (011Ь)	Each set of CCs are synchronized to the pixel clock of their corresponding camera. In other words, the CCs on the primary connector are synchronized to the primary camera's pixel clock, and the CCs on the sec- ondary camera are synchronized to secondary cam- era's pixel clock.	
	4 (100b)	Reserved	
	5 (101b)	Reserved	
	6 (110b)	Reserved	
	7 (111b)	Reserved	
EN_TRIGGER	R/W, CON5[30], Alta, Karbon, Neon, R64		
	This bitfield has the following properties.		
	EN_TRIGGER	Meaning	
	0	External (HW) selected trigger is disabled.	
	1	External (HW) selected trigger is enabled.	

# **EN\_ENCODER** R/W, CON5[31], Alta, Karbon, Neon, R64

This bitfield has the following properties.

EN_ENCODER	Meaning
0	External (HW) selected encoder is disabled.
1	External (HW) selected encoder is enabled.

Bit	Name
0	VCOUNT
1	VCOUNT
2	VCOUNT
3	VCOUNT
4	VCOUNT
5	VCOUNT
6	VCOUNT
7	VCOUNT
8	VCOUNT
9	VCOUNT
10	VCOUNT
11	VCOUNT
12	VCOUNT
13	VCOUNT
14	VCOUNT
15	VCOUNT
16	VCOUNT
17	LAL
18	ENC_DIV, ENC_DIV_M
19	ENC_DIV, ENC_DIV_M
20	ENC_DIV, ENC_DIV_M
21	ENC_DIV, ENC_DIV_M
22	ENC_DIV, ENC_DIV_M
23	ENC_DIV, ENC_DIV_M
24	ENC_DIV, ENC_DIV_M
25	ENC_DIV, ENC_DIV_M
26	ENC_DIV, ENC_DIV_M
27	ENC_DIV, ENC_DIV_M
28	HCOUNT
29	HCOUNT
30	TRIG_QUALIFIED
31	Reserved

VCOUNT	RO, CON6[160], Alta, Karbon, Neon, R64		
	This is the value of th	ne VCOUNT, the VCTAB's address counter.	
LAL	R/W, CON6[17], A	R/W, CON6[17], Alta, Karbon, Neon, R64	
	LAL stands for Last A	Active Line. It controls the mode the VCOUNT is read.	
	LAL	Meaning	
	0	VCOUNT register reflects the current value of the VCOUNT.	
	1	VCOUNT register holds the last active line of the pre- vious frame.	
ENC_DIV	R/W, CON6[2718	3], R64	
	The encoder pulses example, ENC_DIV[] supply one pulse to 1.	will be divided down by the number written in this register. If, for = 5, for every five pulses on the selected encoder, the divider will the board. Programming this register to 0 or 1 will both divide by	
ENC_DIV_M	R/W, CON6[2718	], Karbon, Neon, R64	
	This register represe tion 5.1 for more info	ents the "M" parameter of the encoder divider equation. See Secormation.	
HCOUNT	R/W, CON6[2928	3], Alta, Karbon, Neon, R64	
	This register reflects register and observi	the current value of the two LSBs of the HCOUNT. Reading this ng changes in its value means that the HCOUNT is cycling.	
	RO, CON6[31], Alt	a, Karbon, Neon, R64	
ZVALIFIED	This is the current sta	ate of the selected (via SEL_TRIG) trigger input.	

# 7.10 CON7 Register

Bit	Name
0	AQ_COUNT
1	AQ_COUNT
2	AQ_COUNT
3	AQ_COUNT
4	AQ_COUNT
5	AQ_COUNT
6	AQ_COUNT
7	AQ_COUNT
8	AQ_COUNT
9	AQ_COUNT
10	AQ_COUNT
11	AQ_COUNT
12	AQ_COUNT
13	AQ_COUNT
14	AQ_COUNT
15	AQ_COUNT
16	AQ_COUNT
17	AQ_COUNT
18	AQ_COUNT
19	AQ_COUNT
20	SEL_REG_GEN
21	SEL_REG_GEN
22	GEN_ONESHOT
23	Reserved
24	TAG_BANK
25	TAG_BANK
26	TAG_BANK
27	TAG_BANK
28	TAG_BANK
29	TAG_BANK
30	NTG_TO_ENC
31	NTG_TO_TRIG

#### AQ\_COUNT R/W, CON7[19..0], Alta, Karbon, Neon, R64

The number N written in this register is used to tell the acquisition logic to FREEZE acquisition after N frames have been acquired. The register FREEZE\_CON code must be set to 1.

SEL\_REG\_GEN R/W, CON7[21..20], Alta, Karbon, Neon, R64

Future use.

#### GEN\_ONESHOT R/W, CON7[22], R64

This bit controls the mode of the special signal generator available in the TVI camera specific firmware.

GEN_ONESHOT	Meaning
0	Signal generator is free running.
1	Signal generator synchronized to the external encoder signal.

#### **TAG\_BANK** RO, CON7[29..24], R64

\_

This is the calculated bank from the address generator latched by the TAG QUAD; diagnostics/test register.

#### NTG\_TO\_ENC R/W, CON7[30], Kebon, Neon

This bit provides the ability for the NTG timing generator to rung the encoder input directly. This bit overrides the selection made by the SEL\_ENC bit.

NTG_TO_ENC	Meaning
0	The encoder circuit is driven by the selected externel encoder souce.
1	The encoder circuit is driven by the NTG.

# NTG\_TO\_TRIG R/W, CON7[31], Karbon, Neon

This bit provides the ability for the NTG timing generator to rung the trigger input directly. This bit overrides the selection made by the SEL\_TRIG bit..

NTG_TO_TRIG	Meaning
0	The trigger circuit is driven by the selected externel trigger souce.
1	The trigger circuit is driven by the NTG.

# 7.11 CON8 Register

Bit	Name
0	GPOUT0_CON
1	GPOUT0_CON
2	GPOUT0_CON
3	GPOUT1_CON
4	GPOUT1_CON
5	GPOUT1_CON
6	GPOUT2_CON
7	GPOUT2_CON
8	GPOUT2_CON
9	GPOUT3_CON
10	GPOUT3_CON
11	GPOUT3_CON
12	GPOUT4_CON
13	GPOUT4_CON
14	GPOUT4_CON
15	GPOUT5_CON
16	GPOUT5_CON
17	GPOUT5_CON
18	GPOUT6_CON
19	GPOUT6_CON
20	GPOUT6_CON
21	AFPDF
22	AFPDF
23	Reserved
24	RLE_LOAD_H
25	RLE_LOAD_H
26	RLE_LOAD_H
27	RLE_LOAD_H
28	RLE_LOAD_V
29	RLE_LOAD_V
30	RLE_LOAD_V
31	rle load v

# GPOUTO\_CON R/W, CON8[2..0], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT0 on the IO connector

GPOUT0_CON	Selected signal steered to GPOUT0
0 (000b)	GPOUT0 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	The encoder input signal is routed to the GPOUT0 output signal.

**GPOUT1\_CON** R/W, CON8[5..3], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT1 on the IO connector

GPOUT1_CON	Selected signal steered to GPOUT1
0 (000b)	GPOUT1 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	The trigger input signal is routed to the GPOUT1 out- put signal.

# GPOUT2\_CON R/W, CON8[8..6], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT2 on the IO connector

GPOUT2_CON	Selected signal steered to GPOUT2
0 (000b)	GPOUT2 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	reserved.

### GPOUT3\_CON R/W, CON8[11..9], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT3 on the IO connector

GPOUT3_CON	Selected signal steered to GPOUT3
0 (000b)	GPOUT3 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	reserved.

# GPOUT4\_CON R/W, CON8[14..12], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT4 on the IO connector

GPOUT4_CON	Selected signal steered to GPOUT4
0 (000b)	GPOUT4 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	reserved.

GPOUT5\_CON R/W, CON8[17..15], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT5 on the IO connector

GPOUT5_CON	Selected signal steered to GPOUT5
0 (000b)	GPOUT5 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	reserved.

# GPOUT6\_CON R/W, CON8[20..18], Alta, Karbon, Neon, R64

This register selects the signal steered to GPOUT6 on the IO connector

GPOUT6_CON	Selected signal steered to GPOUT6
0 (000b)	GPOUT6 bit written by host in CON4
1 (001b)	CT0 from CTAB.
2 (010b)	CT1 from CTAB.
3 (011b)	CT2 from CTAB.
4 (100b)	CT3 from CTAB.
5 (101b)	Internally generated CLOCK (frequency controlled by CFREQ in CON1).
6 (110b)	Internally generated signal (frequency and duty- cycle controlled by CON17).
7 (111b)	reserved.

#### AFPDF R/W, CON8[22..21], Karbon, Neon

This register provides the ability from multiple camera frames to be DMAed as a signle DMA frame. This helps reduce the interrupt rate for very high speed camers.sr

AFPDF	Meaning
0 (00b)	1 camera frame per DMA frame
1 (01b)	16 camera frames per DMA frame
2 (10b)	128 camera frames per DMA frame
3 (11b)	1024 camera frames per DMA frame

#### **RLE\_LOAD\_H** R/W, CON8[27..24], Karbon, Neon

For boards that use RLE CTabs, this register controls the location that the horizontal RLE counter jumps to when the LEN signal is asserted. The units of value in this bit-field is RLE entry, not the CTAB location. In other words, if the jump point is 0x8000 CTAB location, but the RLE entry for this location is 3, then this register should be programmed to 3.

#### RLE\_LOAD\_V

#### R/W, CON8[31..28], Karbon, Neon

For boards that use RLE CTabs, this register controls the location that the vertical RLE counter jumps to when the FEN signal is asserted. The units of value in this bitfield is RLE entry, not the CTAB location. In other words, if the jump point is 0x20000 CTAB location, but the RLE entry for this location is 3, then this register should be programmed to 3.

# 7.12 CON9 Register

Bit	Name
0	MUX_REV
1	MUX_REV
2	MUX_REV
3	MUX_REV
4	MUX_REV
5	MUX_REV
6	MUX_REV
7	MUX_REV
8	MUX_REV
9	MUX_REV
10	MUX_REV
11	MUX_REV
12	TRIM
13	TRIM
14	TRIM
15	TRIM
16	FW_TYPE
17	FW_TYPE
18	FW_TYPE
19	FW_TYPE
20	DISPLAY
21	CLIP
22	SHORT_FRAME
23	RST_CALC_BANK
24	CALC_BANK
25	CALC_BANK
26	CALC_BANK
27	CALC_BANK
28	CALC_BANK
29	CALC_BANK
30	ACPL_MUL
31	ACPL MUL

#### MUX\_REV RO, CON9[11..0], Alta, Karbon, Neon, R64

Firmware revision.

TRIM R/W, CON9[15..12], Alta, Karbon, Neon, R64

This bit field will delay the LEN relative to the video. The delay, in units of pixels, equals the number programmed in the TRIM[] field.

The net effect for a simple, one tap camera will be a shifting to the left of the displayed image. For multi-tap cameras the visual effect is more complex and depends on the CCD architecture.

The purpose of this bit field is to align the image presented by the different taps. This bit field has the opposite effect of the DELAY field in CON14.

TRIM	Meaning
0 (000b)	LEN is not delayed
1 (001b)	LEN is delayed by 1 clocks
2 (010b)	LEN is delayed by 2 clocks
3 (011b)	LEN is delayed by 3 clocks
4 (100b)	LEN is delayed by 4 clocks
5 (101b)	LEN is delayed by 5 clocks
6 (11ob)	LEN is delayed by 6 clocks
7 (111b)	LEN is delayed by 7 clocks

FW\_TYPE RO, CON9[19..16], Alta, Karbon, Neon, R64

Firmware type.

#### DISPLAY R/W, CON9[20], Alta, Karbon, Neon, R64

This bit controls the acquisition of data that is more than 8 bits/pixel. When this bit is set, only the 8 LSB of the data will be acquired in each lane. To be able to display the 8 MSB (or any other consecutive group of 8 bits), the data must be shifted accordingly with the barrel shifter. For 9 to 16-bit cameras, setting this bit will result in an 8-bit dis-

playable pixel. For high bit depth color cameras, setting this bit will result in a 24-bit color displayable pixel. In both cases. the barrel shifter must be set correctly to get the displayable bits out of the incoming pixels.

DISPLAY	Meaning
0	Acquire full bit depth.
1	Acquire 8 LSBs of each data lane.

## CLIP R/W, CON9[21], Alta, Karbon, Neon, R64

This bit will clip the upper and lower 15 gray levels. This is useful for displaying gray level images on VGA monitors set in 256 colors mode. The upper and lower 15 gray levels are dedicated to Windows graphics.

CLIP	Meaning
0	Acquire data as is.
1	Clip upper and lower 15 gray levels to 240 and 15 respectively.

#### SHORT\_FRAME R/W, CON9[22], Alta, Karbon, Neon, R64

Future use.

#### **RST\_CALC\_** R/W, CON9[23], Alta, Karbon, Neon, R64 **BANK**

For normal operation this bit should be 0.

RST_CALC_BANK	Meaning
0	Normal operation
1	Reset the calculated starting bank.

#### CALC\_BANK RO, CON9[29..24], Alta, Karbon, Neon, R64

Value of the current calculated starting bank.

# ACPL\_MUL R/W, CON9[31..30], Alta, Karbon, Neon, R64

This register is used to increase the maximum line size the board can acquire. The settings act as multiplier for the ACPL (Active Clocks Per Line) register.

ACPL_MUL	Meaning
0 (00b)	Normal operation. ACPL is used as is
1 (01b)	ACPL is multiplied by 2
2 (10b)	Reserved
3 (11b)	Reserved

# 7.13 CON10 Register

Bit	Name
0	ACPL
1	ACPL
2	ACPL
3	ACPL
4	ACPL
5	ACPL
6	ACPL
7	ACPL
8	ACPL
9	ACPL
10	ACPL
11	ACPL
12	ACPL
13	ACPL
14	ACPL
15	ACPL
16	ACPL
17	FORMAT
18	FORMAT
19	FORMAT
20	FORMAT
21	FORMAT
22	VID_SOURCE
23	VID_SOURCE
24	VID_SOURCE
25	VID_SOURCE
26	PIX_DEPTH
27	PIX_DEPTH
28	PIX_DEPTH
29	PIX_DEPTH
30	PIX_DEPTH
31	FORCE_8BIT
#### ACPL R/W, CON10[16..0], Alta, Karbon, Neon, R64

This register defines the Active Clocks Per Line of the horizontal acquisition window. Let's assume for example a single tap camera with 2K pixels per line. If we want to acquire 400 pixels per line, the ACLP will be programmed to 400. For a 2-tap odd-even pixels camera, with 2K pixels per line, to acquire 400 pixels the ACLP will be programmed with the value 200, as for every clock the camera supplies two pixels.

#### FORMAT RO, CON10[21..17], Alta, Karbon, Neon, R64

This register defines the camera(s) format in terms of taps and scanning architecture. For every FORMAT there is an associated firmware that is downloaded in the FPGAs. The firmware is identified in the camera file by the FORMAT.

FORMAT	Firmware Name	Format Description
0 (00000b)	MUX	1 tap cameras
1 (00001b)	MUX_2TOEP	2 taps, odd-even pixels
2 (00010b)	MUX_2TOEL	2 taps, odd-even lines
3 (00011b)	MUX_2TS	2 taps, segmented
4 (00100b)	MUX_2TS1RI	2 taps, segmented, right inverted
5 (00101b)	MUX_4TS	4 taps, segmented
6 (00110b)	MUX_4T2S2RIOEP	4 taps, odd-even pixels, right taps inverted
7 (00111b)	MUX_4TQ2RI2BU	4 quads, right quads inverted, bottom quads upside down
8 (01000b)	MUX_2CAM	2 cameras: 1 tap each
9(01001b)	MUX_2CAM_2TOEP	2 cameras: 2 taps, odd-even pixels
10 (01010b)	MUX_2CAM_2TS1RI	2 cameras: 2 taps, segmented, right- inverted
11 (01011b)	MUX_2CAM_2TS	2 cameras: 2 taps, segmented
12 (01100b)	MUX_2CAM_2TOEL	2 cameras: 2 taps, odd-even lines
13 (01101b)	MUX_8TS	8 taps, segmented
14 (01110b)	MUX_BAY	Bayer decoder, 1 tap 8 bit
15 (01111b)	MUX_BAY_OE	Bayer decoder, 2 taps, odd-even pixels
16(10000b)	MUX_BAY_2TS	Bayer decoder, 2 taps, segmented
17 (10001b)	MUX_4WI	4 taps, 4-way interleaved
18 (10010b)	MUX_2TOEPI	2 taps, odd-even pixels, both inverted
19(10011b)	MUX_1TI	1 tap, inverted
20 (10100b)	MUX_8WI	8 taps, 8-way interleaved

FORMAT	Firmware Name	Format Description
21 (10101b)	MUX_BAY_2TS_RI	Bayer decoder, 2 taps, segmented, right inverted
22 (10110b)	MUX_4TS2RI	Four taps, segmented, right two taps inverted
23 (10111b)	MUX_8TSOEP4RI	Eight taps, segments, odd/even pixel, for right taps inverted
24 (11000b)	MUX_10WI	Ten taps, interleaved

#### VID\_SOURCE R/W, CON10[25..22], Alta, Karbon, Neon, R64

This register defines the video source and the pattern for the synthetic video. The synthetic patterns appear on all 8 taps, except for code 9.

VID_SOURCE	Video source
0 (0000b)	Camera
1 (0001b)	Camera, special mode for cameras that do not assert the VALID signal.
2 (0010b)	reserved
3 (0011b)	Synthetic horizontal static wedge
4 (0100b)	Synthetic dynamic wedge
5 (0101b)	Synthetic 00h
6 (0110b)	Synthetic FFh
7 (0111b)	Synthetic AAh
8 (1000b)	Synthetic 55h
9 (1001b)	Synthetic ABCDEF0123456789h
10 (1010b)	Synthetic vertical static wedge

#### PIX\_DEPTH R/W, CON10[30..26], Alta, Karbon, Neon, R64

This register defines the pixel depth as well as the color order and packing mode for RGB cameras.

PIX_DEPTH	Bit/pixel, color order and packing
0 (0000b)	8 bits
1 (0001b)	10 bits
2 (0010b)	12 bits
3 (0011b)	14 bits
4 (0100b)	16 bits
5 (0101b)	3x8 bits BGR, DMAed as 32 bits, upper MSB set to 00h
6 (0110b)	3x8 bits BGR, DMAed as 24 bits (packed)
7 (0111b)	3x10 bits RGB, DMAed as 32 bits, display mode is 24 bits
8 (1000b)	3x12 bits RGB, DMAed as 48 bits (packed), display mode is 24 bits
9 (1001b)	32 bits
10 (1010b)	64 bits
11 (1011b)	3x8 bits RGB, DMAed as 32 bits, upper MSB set to 00h
12 (1100b)	3x8 bits RGB, DMAed as 24 bits (packed)
13 (1101b)	3x10 BGR, DMAed as 32 bits, display mode is 24 bits
14 (1110b)	3x12 BGR, DMAed as 48 bits (packed), display mode is 24 bits

#### FORCE\_8BIT R/W, CON10[31], Alta, Karbon, Neon, R64

This bitfield has the following properties.

FORCE_8BIT	Meaning
0 (000b)	Normal operation
1 (001b)	Only 8 LSB of pixel will be acquired

Bit	Name
0	ALAST_ADD
1	ALAST_ADD
2	ALAST_ADD
3	ALAST_ADD
4	ALAST_ADD
5	ALAST_ADD
6	ALAST_ADD
7	ALAST_ADD
8	ALAST_ADD
9	ALAST_ADD
10	ALAST_ADD
11	ALAST_ADD
12	ALAST_ADD
13	ALAST_ADD
14	ALAST_ADD
15	DPM_WP
16	BLAST_ADD
17	BLAST_ADD
18	BLAST_ADD
19	BLAST_ADD
20	BLAST_ADD
21	BLAST_ADD
22	BLAST_ADD
23	BLAST_ADD
24	BLAST_ADD
25	BLAST_ADD
26	BLAST_ADD
27	BLAST_ADD
28	BLAST_ADD
29	BLAST_ADD
30	BLAST_ADD
31	UART MASTER

#### ALAST\_ADD RO, CON11[14..0], Alta, Karbon, Neon, R64

Last address for lane A (used for diagnostics).

DPM\_WP R/W, CON11[15], Alta, Karbon, Neon, R64

Prevents the DPM memory from be written by the acquisiton engine. Should normally be set to 0.

BLAST\_ADD RO, CON11[30..16], Alta, Karbon, Neon, R64

Last address for lane B (used for diagnostics).

UART\_MASTER R/W, CON11[31], Karbon

This bit controls which Karbon VFG is in control of the UART. Poke this bit to one in order to take control of the UART.

# 7.15 CON12 Register

Bit	Name
0	CLAST_ADD
1	CLAST_ADD
2	CLAST_ADD
3	CLAST_ADD
4	CLAST_ADD
5	CLAST_ADD
6	CLAST_ADD
7	CLAST_ADD
8	CLAST_ADD
9	CLAST_ADD
10	CLAST_ADD
11	CLAST_ADD
12	CLAST_ADD
13	CLAST_ADD
14	CLAST_ADD
15	Reserved
16	DLAST_ADD
17	DLAST_ADD
18	DLAST_ADD
19	DLAST_ADD
20	DLAST_ADD
21	DLAST_ADD
22	DLAST_ADD
23	DLAST_ADD
24	DLAST_ADD
25	DLAST_ADD
26	DLAST_ADD
27	DLAST_ADD
28	DLAST_ADD
29	DLAST_ADD
30	DLAST_ADD
31	RGBHSI

#### CLAST\_ADD RO, CON11[14..0], Alta, Karbon, Neon, R64

Last address for lane C (used for diagnostics).

DLAST\_ADD R/W, CON11[30..16], Alta, Karbon, Neon, R64

Last address for lane D (used for diagnostics).

**RGBHSI** R/W, CON11[31], Alta

Onboards that can do real time color conversion from RGB to HSI color space, the board controls what color space is put out..

RGBHSI	Meaning
0	Output RGB
1	Output HSI

# 7.16 CON13 Register

Bit	Name
0	VIDEO_MASK
1	VIDEO_MASK
2	VIDEO_MASK
3	VIDEO_MASK
4	VIDEO_MASK
5	VIDEO_MASK
6	VIDEO_MASK
7	VIDEO_MASK
8	VIDEO_MASK
9	VIDEO_MASK
10	VIDEO_MASK
11	VIDEO_MASK
12	VIDEO_MASK
13	VIDEO_MASK
14	VIDEO_MASK
15	VIDEO_MASK
16	VIDEO_MASK
17	VIDEO_MASK
18	VIDEO_MASK
19	VIDEO_MASK
20	VIDEO_MASK
21	VIDEO_MASK
22	VIDEO_MASK
23	VIDEO_MASK
24	VIDEO_MASK
25	VIDEO_MASK
26	VIDEO_MASK
27	VIDEO_MASK
28	VIDEO_MASK
29	VIDEO_MASK
30	VIDEO_MASK
31	VIDEO_MASK

#### VIDEO\_MASK R/W, CON13[31..0], Alta, Karbon, Neon, R64

With the aid of this mask, individual bits in the video data stream can be set to 0. The 32 bit mask is duplicated for the 32 MSB of a 64 bit word.

Bit N in VIDEO_MASK	Meaning
0	Set bit N to 0
1	Pass bit N as is

# 7.17 CON14 Register

Bit	Name
0	SWRESET
1	FENPOL
2	LENPOL
3	BUTTONS
4	BUTTONS
5	BUTTONS
6	BUTTONS
7	BUTTONS
8	BUTTONS
9	BUTTONS
10	BUTTONS
11	BUTTONS
12	BUTTONS
13	BUTTONS
14	BUTTONS
15	BUTTONS
16	SHIFT_RAW
17	SHIFT_RAW
18	SHIFT_RAW
19	SHIFT_RAW
20	SHIFT_RAW_LEFT
21	DELAY
22	DELAY
23	DELAY
24	SWAP
25	UART_CON
26	UART_CON
27	Reserved
28	DPM_SPLIT
29	DPM_SPLIT
30	DPM_SPLIT
31	DPM_SPLIT

#### SW\_RESET WO, CON14[0], Alta, Karbon, Neon, R64

This bit will always read back 0.

	SW_RESET	Meaning	
	0	Reset de-asserted	
	1	General reset to acquisition circuitry.	
FENPOL	R/W, CON14[1], Alta, Karbon, Neon, R64		
	FENPOL	Meaning	
	0	FEN is asserted on rising edge.	
	1	FEN is asserted on falling edge.	
LENPOL	R/W, CON14[2], Alta, Karbon, Neon, R64		
	This bitfield has the fo	his bitfield has the following properties.	
	LENPOL	Meaning	
	0	LEN is asserted on rising edge.	
	1	LEN is asserted on falling edge.	
BUTTONS	R/W, CON14[153], Alta, Karbon, Neon, R64		
	R/W register for test/diagnostics.		
SHIFT_RAW	R/W, CON14[1916], Alta, Karbon, Neon, R64		
	This register defines f acquired	or the barrel shifter the amount of shift for the data to be	

# SHIFT\_RAW\_R/W, CON14[20], Alta, Karbon, Neon, R64LEFTThis register defines for the barrel shifter the shift direction for the data to be

acquired.

SHIFT_RAW_LEFT	Meaning
0	Shift right
1	Shift left

#### DELAY R/W, CON14[23..21], Alta, Karbon, Neon, R64

This register is used for aligning the Horizontal Active Window (HAW) relative the Line ENable signal (LEN). If the first active pixel occurs simultaneously with the assertion of LEN, then no delay is needed. If the first active pixel occurs between 1 and 7 clocks later than LEN, the DELAY register must be programmed accordingly. The CTABs can be used to compensate for delays over 7 clocks. However, the granularity of the CTABs is 8 clocks, so both DELAY and the CTABS may have to be used in together to accommodate for some delays.

The net effect for a simple, one tap camera will be a shifting to the right of the displayed image. For multi-tap cameras the visual effect is more complex and depends on the taps architecture. All taps are delayed by the same amount.

The purpose of this bit field is to align the image presented by the different taps. This bit field has the opposite effect of the TRIM field in CON9.

DELAY	Meaning
0 (000b)	HAW is not delayed
1 (001b)	HAW is delayed by 1 clocks
2 (010b)	HAW is delayed by 2 clocks
3 (011b)	HAW is delayed by 3 clocks
4 (100b)	HAW is delayed by 4 clocks
5 (101b)	HAW is delayed by 5 clocks
6 (11ob)	HAW is delayed by 6 clocks
7 (111b)	HAW is delayed by 7 clocks

#### SWAP R/W, CON14[24], Alta, Karbon, Neon, R64

Future use.

#### UART\_CON R/W, CON14[26..25], Alta, Karbon, Neon, R64

This register will control the connection of the serial ports of the two camera Link connectors.

UART_CON	Connection
0 (00Ь)	On-board UART connected to CL serial port of main connector
1 (01Ь)	On-board UART connected to CL serial port of auxil- iary connector
2 (10b)	Serial port of main connector CL connected to exter- nal serial port, through the IO connector.
3 (11b)	Serial port of auxiliary connector CL connected to external serial port, through the IO connector

#### DPM\_SPLIT R/W, CON14[31..28], Alta, Karbon, Neon, R64

This register controls how incoming data is written to the DPM.

DPM_SPLIT	Mode
0 (0000b)	Normal mode
1 (0001b)	Each tap's output is split in half.
2 (0010b) to 14 (1110b)	Reserved
15 (1111b)	Each tap's output is written in 4K chunks

# 7.18 CON15 Register

Bit	Name
0	QENC_INTRVL_LL
1	QENC_INTRVL_LL
2	QENC_INTRVL_LL
3	QENC_INTRVL_LL
4	QENC_INTRVL_LL
5	QENC_INTRVL_LL
6	QENC_INTRVL_LL
7	QENC_INTRVL_LL
8	QENC_INTRVL_LL
9	QENC_INTRVL_LL
10	QENC_INTRVL_LL
11	QENC_INTRVL_LL
12	QENC_INTRVL_LL
13	QENC_INTRVL_LL
14	QENC_INTRVL_LL
15	QENC_INTRVL_LL
16	QENC_INTRVL_LL
17	QENC_INTRVL_LL
18	QENC_INTRVL_LL
19	QENC_INTRVL_LL
20	QENC_INTRVL_LL
21	QENC_INTRVL_LL
22	QENC_INTRVL_LL
23	QENC_INTRVL_LL
24	QENC_DECODE
25	QENC_AQ_DIR
26	QENC_AQ_DIR
27	QENC_INTRVL_MODE
28	QENC_NO_REAQ
29	QENC_DUAL_PHASE
30	SCAN_STEP_TRIG
31	QENC_RESET

LL

#### **QENC\_INTRVL** R/WR/W, CON15[23..0], Karbon, Neon

This register contains the lower limit value that is used to start acquisition when the system is in interval mode (see QENC\_INTRVL\_MODE).

**QENC\_DECODE** R/W, CON15[24], Karbon, Neon

This bit determines how often the quadrature counter is incremented.

QENC_DECODE	Meaning
0	Counter increments on the rising edge of input A and the rising edge of input B. This is also called "2x" modes.
1	Counter increments on both the rising and falling edge of A and both the rising and falling edge of B. This is also called "4x" mode.

#### **QENC\_AQ\_DIR** R/W, CON15[26..25], Karbon, Neon

This bit controls which quadrature encoder direction is used for acquisition.

QENC_AQ_DIR	Meaning
0 (00b)	Lines are acquired in both directions
1 (01b)	Lines are acquired only in the positive direction.
2 (10b)	Lines are acquired only in the negative direction.
3 (11b)	Reserved

#### **QENC\_INTRVL\_** R/W, CON15[27], Karbon, Neon

MODE

When this bit is 1, interval mode is turned on. When interval mode is on, lines are only capture when the encoder counter is between the lower limit (set by QENC\_INTRVL\_LL) and the upper limit (set by QENT\_INTRVL\_UL). If the counter is outside of this range, lines are not acquired. Whether lines are acquired as the counter increments through the interval, or decrements through the interval, or in both directions is controlled by QENC\_AQ\_DIR.

# QENC\_NO\_ R/W, CON15[28], Karbon, Neon REAQ This bit controls how the quadrature encoder system handles the situation where the encoder does not smoothly increase (or decrease if QENC\_AQ\_DIR = 1). If there is "jitter" in the encoder signal, often caused by problems with the mechanical systems, it is possible for the board to acquire the same line or lines more than once as the

mechanical system backs up and moves forward (jitter). This re-acquisition can cause problems as the resulting images will have distortions and will not accurately represent the object in front of the camera.

Programming this bit to a 1 turns on the no-reacquisition circuit. This circuit eliminates this problem as each line in the image will only be acquired once, regardless of how much jitter occurs in the quadrature encoder input. The circuit does this by making sure that only one line is acquired for each encoder counter value. If the quadrature encoder backs up, and then moves forward, the board will not acquire lines until a new encoder counter value is reached.

This system handles any amount of jitter, regardless of how many times the counter passes through a value, or to what extremes the counter goes. New lines will only be acquired when new values are reached.

Once the entire frame has been acquired, the system must be reset. The system can always be reset by poking QENC\_RESET to 1. There are also ways that the system can automatically be reset, see QENC\_RESET\_MODE.

QENC_NO_REAQ	Meaning
0	Lines are acquired every change in the encoder counter (as controlled by QENC_AQ_DIR)
1	Lines are only acquired when the encoder counter reaches new values (also controlled by QENC_AQ_ DIR)

QENC_DUAL_	R/W, CON15[29], Karbon, Neon
PHASE	

This bit controls which type of encoder is attached.

QENC_DUAL_PHASE	Meaning
0	A single phase encoder is attached
1	A quadrature encoder is attached

SCAN\_STEP\_ R/W, CON15[30], Karbon, Neon

TRIG

The scan step circuit uses the encoder to generate a trigger to the system. The scan step trigger generates a trigger every N lines (N is set in the SCAN\_STEP register).

SCAN_STEP_TRIG	Meaning
0	Trigger comes of the normal source
1	Trigger comes from the scan step circuit

#### **QENC\_RESET** WO, CON15[31], Karbon, Neon

Poking this bit to a 1 resets the entire quadrate encoder system.

# 7.19 CON16 Register

Bit	Name
0	QENC_INTRVL_UL
1	QENC_INTRVL_UL
2	QENC_INTRVL_UL
3	QENC_INTRVL_UL
4	QENC_INTRVL_UL
5	QENC_INTRVL_UL
6	QENC_INTRVL_UL
7	QENC_INTRVL_UL
8	QENC_INTRVL_UL
9	QENC_INTRVL_UL
10	QENC_INTRVL_UL
11	QENC_INTRVL_UL
12	QENC_INTRVL_UL
13	QENC_INTRVL_UL
14	QENC_INTRVL_UL
15	QENC_INTRVL_UL
16	QENC_INTRVL_UL
17	QENC_INTRVL_UL
18	QENC_INTRVL_UL
19	QENC_INTRVL_UL
20	QENC_INTRVL_UL
21	QENC_INTRVL_UL
22	QENC_INTRVL_UL
23	QENC_INTRVL_UL
24	QENC_REAQ_MODE
25	QENC_REAQ_MODE
26	QENC_RESET_REAQ
27	ENC_DIV_FORCE_DC
28	ENC_DIV_OPEN_LOOP
29	ENC_DIV_FCLK_SEL
30	ENC_DIV_FCLK_SEL
31	ENC_DIV_FCLK_SEL

#### **QENC\_INTRVL\_** R/W, CON16[23..0], Karbon, Neon

UL

This register contains the upper limit value that is used to start acquisition when the system is in interval mode (see QENC\_INTRVL\_MODE).

# **QENC\_REAQ\_** R/W, CON16[25..24], Karbon, Neon **MODE**

This bit controls how the circuit that prevents re-acquisition from encoder jitter is reset. Re-acquisition is prevented by keeping a list of lines that have been acquired, and making sure the only lines that are not on the list are acquired. Once the entire frame is acquired, there must be some way to reset the list, otherwise no new lines will be acquired during the next frame. See QENC\_NO\_REAQ for more information.

QENC_REAQ_MODE	Meaning
0(00b)	Reset the list of acquired lines when QENC_RESET_ REAQ is poked to 1.
1 (01Ь)	Reset the list of lines when the encoder counter is outside of the interval set by the upper limit and lower limit. Whether the reset occurs above the upper limit or below the lower limit depens on the QENC_AQ_DIR register.
2 (10b)	Reserved
3 (11b)	Reserved

#### **QENC\_RESET\_** WO, CON16[26], Karbon, Neon

REAQ

This register is used to reset the circuit that prevents the re-acquisition of lines when QENC\_NO\_REAQ is set to 1. Writing a 1 to this register deletes the list of acquired lines, thus next time the lines are passed over, they will be acquired again. Writing to this bit always resets the no re-acquisition circuit, regardless of the mode as set by the QENC\_REAQ\_MODE. However, the register QENC\_REAQ\_MODE can be used to put

the board in a mode where the no re-aquisition circuit is reset automatically every

ENC\_DIV\_ R/W, CON16[27], R64, Karbon, Neon FORCE\_DC

pass over the image.

This register is used to controls the behavior of the encoder divider when input frequency falls below the minimum.

ENC_DIV_FORCE_DC	Meaning
0	Encoder divider runs in simple divider mode.
1	Encoder divider output stops (goes to DC).

#### ENC\_DIV\_ OPEN\_LOOP

#### R/W, CON16[28], R64, Karbon, Neon

This register controls whether the output signal phase of the Encoder Divider is lock to the intput or is allowed to free run.

ENC_DIV_OPEN_LOOP	Meaning
0	Output phased locked to input
1	Ouput runs open loop

ENC\_DIV\_FCLK\_ R/W, CON16[31..29], R64, Karbon, Neon SEL

This register is reserved for future support for alternate Encoder Divider PLL Master clock frequencies. Currently must be set to 0, which selects 50 MHz clock

# 7.20 CON17 Register

Bit	Name
0	NTG_RATE
1	NTG_RATE
2	NTG_RATE
3	NTG_RATE
4	NTG_RATE
5	NTG_RATE
6	NTG_RATE
7	NTG_RATE
8	NTG_RATE
9	NTG_RATE
10	NTG_RATE
11	NTG_RATE
12	NTG_RATE
13	NTG_RATE
14	NTG_RATE
15	NTG_RATE
16	NTG_RATE
17	NTG_RATE
18	NTG_RATE
19	NTG_RATE
20	NTG_RATE
21	NTG_RATE
22	NTG_RATE
23	NTG_RATE
24	NTG_RATE
25	NTG_RATE
26	NTG_RATE
27	NTG_RATE
28	Reserved
29	Reserved
30	NTG_ONESHOT
31	NTG_TRIG_MODE

#### NTG\_RATE R/W, CON17[27..0], Alta, Karbon, Neon, R64

This register controls the line/frame rate period of the NTG. One LSB in this registers represents on clock period of the NTG clock. The NTG clock frequency depends on the model Table 7-3. See Section 3.1 for more information.

#### Table 7-3 NTG clock frequency

Model	Frequency
Karbon, Neon, R64	7.3728 MHz
Alta	5.000 MHz

#### NTG\_ONESHOT R/W, CON17[30], Alta, Karbon, Neon, R64

This bit defines whether the NTG is free running or in one shot mode.

NTG_ONESHOT	Mode
0	NTG is free running
1	NTG is in one shot mode

#### NTG\_TRIG\_ R/W, CON17[31], Alta, Karbon, Neon, R64

This bit determines what triggers the NTG when it is in one shot mode.

NTG_ONESHOT	Mode
0	NTG is triggered by the selected trigger signal
1	NTG is triggered by the selected encoder signal

MODE

CON18 Register

# 7.21 CON18 Register

Bit	Name
0	ALPF
1	ALPF
2	ALPF
3	ALPF
4	ALPF
5	ALPF
6	ALPF
7	ALPF
8	ALPF
9	ALPF
10	ALPF
11	ALPF
12	ALPF
13	ALPF
14	ALPF
15	ALPF
16	ALPF
17	TOP_REV
18	TOP_REV
19	TOP_REV
20	TOP_REV
21	TOP_REV
22	TOP_REV
23	TOP_REV
24	TOP_REV
25	TOP_REV
26	TOP_REV
27	TOP_REV
28	TOP_REV
29	TOP_REV
30	NTG_INVERT
31	NTG_TIME_MODE

#### ALPF R/W, CON18[16..0], Alta, Karbon, Neon, R64

This register defines the Active Lines Per Frame of the vertical acquisition window. Let's assume for example a single tap camera with 2K lines per frame. If we want to acquire 400 lines per frame, the ALPF will be programmed to 400. For a 2-tap oddeven lines camera, with 2K lines per frame, to acquire 400 pixels the ALPF will be programmed with the value 200, as for every LEN the camera supplies two lines.

TOP\_REV RO, CON9[11..0], Alta, Karbon, Neon, R64

Firmware revision.

NTG\_INVERT R/W, CON18[30], Alta, Karbon, Neon, R64

This bit allows for the inversion of the New Timing Generators's (NTG) output. See Section 3.1 for more information on the NTG.

NTG_INVERT	Meaning
0	NTG output is asserted high
1	NTG output is asserted low

NTG\_TIME\_ R/W, CON18[31], Alta, Karbon, Neon, R64 MODE

> This bit is used to scale down the frequency of the NTG clock. The NTG clock frequency depends on the board family. See Section 3.1 for more information. This mode is useful for area scan cameras that need very long exposure times.

NTG_TIME_MODE	Meaning
0	NTG clock is used as is.
1	NTG clock is divided by 128.

Bit	Name
0	LINES_TOGO
1	LINES_TOGO
2	LINES_TOGO
3	LINES_TOGO
4	LINES_TOGO
5	LINES_TOGO
6	LINES_TOGO
7	LINES_TOGO
8	LINES_TOGO
9	LINES_TOGO
10	LINES_TOGO
11	LINES_TOGO
12	LINES_TOGO
13	LINES_TOGO
14	LINES_TOGO
15	LINES_TOGO
16	LINES_TOGO
17	ENC_DIV_N
18	ENC_DIV_N
19	ENC_DIV_N
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

#### LINES\_TOGO R/W, CON19[16..0], Alta, Karbon, Neon, R64

This register will reflect the number of remaining lines left to be acquired till the end of the frame.

**ENC\_DIV\_N** R/W, CON19[19..17], R64, Karbon, Neon

This register represents the "N" parameter in the encoder divider equation. See Section 5.1 for more information.

Bit	Name
0	FIFO_EQ
1	FIFO_EQ
2	FIFO_EQ
3	FIFO_EQ
4	FIFO_EQ
5	FIFO_EQ
6	FIFO_EQ
7	FIFO_EQ
8	VID_BRL
9	VID_BRL
10	VID_BRL
11	VID_BRL
12	VID_BRL
13	VID_BRL
14	VID_BRL
15	VID_BRL
16	VIDEO_2DPM
17	VIDEO_2DPM
18	VIDEO_2DPM
19	VIDEO_2DPM
20	VIDEO_2DPM
21	VIDEO_2DPM
22	VIDEO_2DPM
23	VIDEO_2DPM
24	COLOR_MASK
25	COLOR_MASK
26	SHIFT_DSP_SELECT
27	SHIFT_DSP
28	SHIFT_DSP
29	SHIFT_DSP
30	SHIFT_DSP
31	SHIFT_DSP_LEFT

#### FIFO\_EQS RO, CON20[7..0], Alta, Karbon, Neon, R64

This register reflects the instantaneous value of the 8 LSB of the first tap of the main CL connector. Used for diagnostics/test.

#### VID\_BRL RO, CON20[15..8], Alta, Karbon, Neon, R64

This register reflects the instantaneous value of the 8 LSB of the barrel shifter of lane A. Used for diagnostics/test.

VIDEO\_2DPM RO, CON20[23..16], Alta, Karbon, Neon, R64

This register reflects the instantaneous value of the 8 LSB of the video written in the DPM's lane A. Used for diagnostics/test.

COLOR\_MASK R/W, CON20[25..24], Alta, Karbon, Neon, R64

This bitfield can be used to mask out color channels when acquiring color pixels formats (e.g 24-bit color, 36-bit color, etc.).

COLOR_MASK	Meaning
0 (00b)	Pass all colors
1 (01b)	Pass only the red channel, set the blue and green chan- nels to 0
2 (10b)	Pass only the green channel, set the blue and red chan- nels to 0
3 (11b)	Pass only the blue channel, set the red and green chan- nels to 0

SHIFT\_DSP\_ R/W, CON20[26], Alta, Karbon, Neon, R64 SELECT

This bitfield has the following properties.

SHIFT_DSP_SELECT	Meaning
0	Supply barrel shifter with the acquisition shift code
1	Supply barrel shifter with the display shift code.

#### SHIFT\_DISP R/W, CON20[30..27], Alta, Karbon, Neon, R64

This register holds the shift amount for data to be displayed.

#### SHIFT\_DSP\_LEFT R/W, CON20[31], Alta, Karbon, Neon, R64

This bitfield has the following properties.

SHIFT_DSP_LEFT	Meaning
0	Shift display data right
1	Shift display data left.

## 7.24 CON21 Register (Bayer Version)

Bit	Name
0	RED_GAIN
1	RED_GAIN
2	RED_GAIN
3	RED_GAIN
4	RED_GAIN
5	RED_GAIN
6	RED_GAIN
7	RED_GAIN
8	GREEN_GAIN
9	GREEN_GAIN
10	GREEN_GAIN
11	GREEN_GAIN
12	GREEN_GAIN
13	GREEN_GAIN
14	GREEN_GAIN
15	GREEN_GAIN
16	BLUE_GAIN
17	BLUE_GAIN
18	BLUE_GAIN
19	BLUE_GAIN
20	BLUE_GAIN
21	BLUE_GAIN
22	BLUE_GAIN
23	BLUE_GAIN
24	DECODER_OUT
25	DECODER_OUT
26	DECODER_OUT
27	Reserved
28	BAYER_BIT_DEPTH
29	BAYER_BIT_DEPTH
30	DECODER_PHASE
31	DECODER_PHASE

CON21 is a "soft" register. Soft registers change definitions depending on the version board and the firmware that is downloaded to the board.

#### **REG\_GAIN** R/W, CON21[7..0], Karbon, R64

This register controls the gain of the red channel. The video value is multiplied by the value in RED\_GAIN and after that scaled down by 64. Numbers above 255 are clipped to 255 (saturation effect).

#### **GREEN\_GAIN** R/W, CON21[15..8], Karbon, R64

This register controls the gain of the green channel. The video value is multiplied by the value in GREEN\_GAIN and after that scaled down by 64. Numbers above 255 are clipped to 255 (saturation effect).

BLUE\_GAIN R/W, CON21[23..16], Karbon, R64

This register controls the gain of the blue channel. The video value is multiplied by the value in BLUE\_GAIN and after that scaled down by 64. Numbers above 255 are clipped to 255 (saturation effect).

**DECODER\_OUT** R/W, CON21[26..24], Karbon, R64

|--|

DECODER_OUT	Meaning
0 (000b)	Decode RGB on all three channels
1 (001b)	Raw data on all three channels
2 (010b)	Decode intensity on all three channels
3 (011b)	Decode red on all three channels
4 (100b)	Decode green on all three channels
5 (101b)	Decode blue on all three channels
6 (110b)	Reserved
7 (111b)	Reserved

#### BAYER\_BIT\_ DEPTH

#### R/W, CON21[29..28], Karbon, R64

This bit is set if the pixel depth from the camera is 10 bits.

BAYER_10_BIT	Meaning
0 (00b)	8-bit pixels
1 (01b)	12-bit pixels
2 (10b)	10-bit pixels
3 (11b)	Reserved

#### DECODER\_ PHASE

R/W, CON21[31..30], Karbon, R64

These bits control the starting phase of the Bayer decoder. This register is set based on the arrangement of the color matrix in the camera's CCD.

DECODER_PHASE	Meaning
0(00b)	First two pixels: Blue, Green
1 (01b)	First two pixels: Green, Blue
2 (10b)	First two pixels: Red, Green
3 (11b)	First two pixels: Green, Red

## 7.25 CON22 Register

Bit	Name
0	FLASH_DATA
1	FLASH_DATA
2	FLASH_DATA
3	FLASH_DATA
4	FLASH_DATA
5	FLASH_DATA
6	FLASH_DATA
7	FLASH_DATA
8	FLASH_DATA
9	FLASH_DATA
10	FLASH_DATA
11	FLASH_DATA
12	FLASH_DATA
13	FLASH_DATA
14	FLASH_DATA
15	FLASH_DATA
16	SCAN_STEP
17	SCAN_STEP
18	SCAN_STEP
19	SCAN_STEP
20	SCAN_STEP
21	SCAN_STEP
22	SCAN_STEP
23	SCAN_STEP
24	SCAN_STEP
25	SCAN_STEP
26	SCAN_STEP
27	SCAN_STEP
28	SCAN_STEP
29	SCAN_STEP
30	SCAN_STEP
31	SCAN_STEP

#### FLASH\_DATA R/W, CON22[15..0], Karbon

This bitfield is used for writing to the flash memory on the board.

SCAN\_STEP R/W, CON22[31..16], Karbon, Neon

This bitfield controls the number of encoder pulses that must occur before a trigger is issued to the system. See SCAN\_STEP\_TRIG for more information. The Scan Step circuit takes into account the interval and re-acquisition functions.

Bit	Name
0	DPM_SIZE
1	DPM_SIZE
2	DPM_SIZE
3	DPM_SIZE
4	DPM_SIZE
5	DPM_SIZE
6	DPM_SIZE
7	DPM_SIZE
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	CTAB_INT_CON
16	LINES_PER_INT
17	LINES_PER_INT
18	LINES_PER_INT
19	LINES_PER_INT
20	LINES_PER_INT
21	LINES_PER_INT
22	LINES_PER_INT
23	LINES_PER_INT
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

#### **DPM\_SIZE** RO, CON23[7..0], Alta, Karbon, Neon, R64

This register specifies the size of the DPM in units of 4096 bytes.

CTAB\_INT\_CON RW, CON23[15], Karbon, Neon, Alta

This bit controls the source of the CTAB interrupt.

CTAB_INT_CON	Meaning
0	CTAB interrupt sourced from CTAB
1	CTAB interrupt sourced from lines per interrupt cir- cuit.

LINES\_PER\_INT RW, CON23[23..16], Karbon, Neon, Alta

This bit controls the lines per interrupt circuit. This circuit can be used to create a periodic interrupt on the CTAB interrupt. The interrupt rate will be every N lines, where N is the value programmed in this register. Note that CTAB\_INT\_CON must be set to one in order for the interrupts to be seen by the host.
### 7.27 CON24 Register

Bit	Name
0	LUT_HOST_DATA
1	LUT_HOST_DATA
2	LUT_HOST_DATA
3	LUT_HOST_DATA
4	LUT_HOST_DATA
5	LUT_HOST_DATA
6	LUT_HOST_DATA
7	LUT_HOST_DATA
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	LUT_ON
16	LUT_HOST_ADDR
17	LUT_HOST_ADDR
18	LUT_HOST_ADDR
19	LUT_HOST_ADDR
20	LUT_HOST_ADDR
21	LUT_HOST_ADDR
22	LUT_HOST_ADDR
23	LUT_HOST_ADDR
24	LUT_BANK
25	LUT_BANK
26	Reserved
27	LUT_DATA_WRITE_ SEL
28	LUT_HOST_LANE
29	LUT_HOST_LANE
30	LUT_WEN
31	LUT_HOST_ACCESS

CON24 is a "soft" register. Soft registers change definitions depending on the version board and the firmware that is downloaded to the board.

### LUT\_HOST\_ DATA

### R/W, CON24[7..0], Alta, Karbon, Neon, R64

This register is used to read and write data from the LUT. The LUT is programmed indirectly using this and the other registers in CON24.

The procedure to write data to the LUT is as follows:

Set LUT\_HOST\_ACCESS to 1.
Set LUT\_DATA\_WRITE\_SEL to 1.
Set LUT\_BANK to the desired bank to program.
Set LUT\_HOST\_LANE to the desired LUT lane to program.
Set LUT\_HOST\_ADDR to the desired LUT location to program (LUT input).
Set LUT\_HOST\_DATA to the value desired (LUT output).
Write LUT\_WEN to 1, this copies the value from the LUT\_HOST\_DATA register to the LUT's memory location as specified by LUT\_HOST\_ADDR.

The procedure to read data from the LUT is as follows:

Set LUT\_HOST\_ACCESS to 1.
Set LUT\_DATA\_WRITE\_SEL to 0.
Set LUT\_BANK to the desired bank to read.
Set LUT\_HOST\_LANE to the desired LUT lane to read.
Set LUT\_HOST\_ADDR to the desired LUT location to read.
Read LUT\_HOST\_DATA, the value returned in this register is the value in the LUT's memory.

LUT\_ON R/W, CON24[15], Alta, Karbon, Neon, R64

This register is used to insert or bypass the LUT from the path of incoming camera data.

LUT_ON	Meaning
0	LUT is not in the data path (bypassed)
1	LUT is in the data path

### LUT\_HOST\_ R/W CON24[23..16], Alta, Karbon, Neon, R64 ADDR

This register is used to set the address for a read operation from the LUT memory or a write operation to the LUT memory. See the description of LUT\_HOST\_DATA for more details.

#### LUT\_BANK R/W, CON24[25..24], Alta, Karbon, Neon, R64

These bits control which bank of the LUT is being programmed, and which bank is being used to pass image data.

LUT_BANK	Meaning
0 (000b)	Host access to bank 0, data passes through bank 0
1 (001b)	Host access to bank 1, data passes through bank 1
2 (010b)	Host access to bank 2, data passes through bank 2
3 (011b)	Host access to bank 3, data passes through bank 3

### LUT\_DATA\_ R/W, CON24[27], Alta, Karbon, Neon, R64

WRITE\_SEL

LANE

This bit is used to control how the LUT data is being accessed.

LUT_DATA_WRITE_SEL	Meaning
0	When reading LUT_HOST_DATA, the register returns the value currently in the LUT. In this mode LUT_ HOST_DATA is read only.
1	LUT_HOST_DATA acts like a normal register. It can be written and read normally. The value in the LUT_ HOST_DATA register is not transferred to the LUT memory until a 1 is written to LUT_WEN.

### LUT\_HOST\_ R/W, CON24[29..28], Alta, Karbon, Neon, R64

\_

These bits control which LUT lane can be access by the host.

LUT_HOST_LANE	Meaning
0 (000b)	Host access to lane 0
1 (001b)	Host access to lane 1
2 (010b)	Host access to lane 2
3 (011b)	Host access to lane 3

LUT\_WEN WO, CON24[30], Alta, Karbon, Neon, R64

When LUT\_DATA\_WRITE\_SEL is set to 1, Writing a 1 to this bit causes the value in LUT\_HOST\_DATA to be transferred to the LUT memory. When LUT\_DATA\_WRITE\_SEL is set to 0, writing to this bit has no effect. See LUT\_HOST\_DATA for more information.

### LUT\_HOST\_ ACCESS

### R/W, CON24[31], Alta, Karbon, Neon, R64

These bits turns on and off host access to the LUT..

DECODER_OUT	Meaning
0	The LUT cannot be accessed by the host
1	The LUT can be accessed by the host

### 7.28 CON25 Register

Bit	Name
0	DELAY_TAP1
1	DELAY_TAP1
2	DELAY_TAP1
3	DELAY_TAP1_SEL
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

### DELAY\_TAP1 R/W, CON25[2..0], Alta, Karbon, Neon, R64

These bits control the delay for tap 1 only when DELAY\_TAP1\_SEL = 1. In this mode, tap 0 and tap 1 can be delayed independently. These bits have no effect if DELAY\_TAP1\_SEL = 0. This register works in a similar manner to the DELAY register.

DELAY_TAP1	Meaning
0 (000b)	HAW is not delayed
1 (001b)	HAW is delayed by 1 clocks
2 (010b)	HAW is delayed by 2 clocks
3 (011b)	HAW is delayed by 3 clocks
4 (100b)	HAW is delayed by 4 clocks
5 (101b)	HAW is delayed by 5 clocks
6 (11ob)	HAW is delayed by 6 clocks
7 (111b)	HAW is delayed by 7 clocks

## DELAY\_TAP1\_ R/W, CON25[3], Alta, Karbon, Neon, R64 SEL

This bit selects the register that controls the delay for tap 1. Tap 0 is always controlled by the register DELAY.

DELAY_TAP1_SEL	Meaning
0	Tap 1 is controlled by DELAY
1	Tap 1 is controlled by DELAY_TAP1

Bit	Name
0	NTG_EXPOSURE
1	NTG_EXPOSURE
2	NTG_EXPOSURE
3	NTG_EXPOSURE
4	NTG_EXPOSURE
5	NTG_EXPOSURE
6	NTG_EXPOSURE
7	NTG_EXPOSURE
8	NTG_EXPOSURE
9	NTG_EXPOSURE
10	NTG_EXPOSURE
11	NTG_EXPOSURE
12	NTG_EXPOSURE
13	NTG_EXPOSURE
14	NTG_EXPOSURE
15	NTG_EXPOSURE
16	NTG_EXPOSURE
17	NTG_EXPOSURE
18	NTG_EXPOSURE
19	NTG_EXPOSURE
20	NTG_EXPOSURE
21	NTG_EXPOSURE
22	NTG_EXPOSURE
23	NTG_EXPOSURE
24	NTG_EXPOSURE
25	NTG_EXPOSURE
26	NTG_EXPOSURE
27	NTG_EXPOSURE
28	Reserved
29	Reserved
30	NTG_RESET
31	NTG_SLAVE

#### NTG\_EXPOSURE R/W, CON26[27..0], Alta, Karbon, Neon, R64

This register controls the exposure period of the NTG. One LSB in this registers represents on clock period of the NTG clock. The NTG clock frequency depends on the model Table 7-4. See Section 3.1 for more information.

#### Table 7-4 NTG clock frequency

Model	Frequency
Karbon, Neon, R64	7.3728 MHz
Alta	5.000 MHz

#### NTG\_RESET WO, CON26[30], Alta, Karbon, Neon, R64

This bit resets the NTG's internal counter. Writing a 1 to this bit resets the counter to 0.

#### NTG\_SLAVE R/W, CON26[31], Alta, Karbon, Neon, R64

This bit determines how whether the NTG is running on its own timing, or slave to the master VFG.

Note: This bit must be set to 0 for the master VFG.

NTG_SLAVE	Mode
0	NTG is running on its own timing
1	NTG is slaved to the master VFG

Bit	Name
0	FLASH_ADDR
1	FLASH_ADDR
2	FLASH_ADDR
3	FLASH_ADDR
4	FLASH_ADDR
5	FLASH_ADDR
6	FLASH_ADDR
7	FLASH_ADDR
8	FLASH_ADDR
9	FLASH_ADDR
10	FLASH_ADDR
11	FLASH_ADDR
12	FLASH_ADDR
13	FLASH_ADDR
14	FLASH_ADDR
15	FLASH_ADDR
16	FLASH_ADDR
17	FLASH_ADDR
18	FLASH_ADDR
19	FLASH_ADDR
20	FLASH_ADDR
21	FLASH_ADDR
22	FLASH_ADDR
23	FLASH_ADDR
24	FLASH_ADDR
25	FLASH_ADDR
26	FLASH_WP
27	FLASH_RST
28	FALSH_BE
29	FLASH_CE
30	FLASH_OE
31	FLASH_WE

FLASH_ADDR	R/W, CON27[250], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_WP	R/W, CON27[26], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_RST	R/W, CON27[27], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_BE	R/W, CON27[28], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_CE	R/W, CON27[29], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_OE	R/W, CON27[30], Karbon
	This register is used for read/writting to the on board flash memoyr.
FLASH_WE	R/W, CON27[31], Karbon
	This register is used for read/writting to the on board flash memoyr.

Bit	Name
0	MEM_ADDR_LO
1	MEM_ADDR_LO
2	MEM_ADDR_LO
3	MEM_ADDR_LO
4	MEM_ADDR_LO
5	MEM_ADDR_LO
6	MEM_ADDR_LO
7	MEM_ADDR_LO
8	MEM_ADDR_LO
9	MEM_ADDR_LO
10	MEM_ADDR_LO
11	MEM_ADDR_LO
12	MEM_ADDR_LO
13	MEM_ADDR_LO
14	MEM_ADDR_LO
15	MEM_ADDR_LO
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

### MEM\_ADDR\_LO R/W, CON25[15..0], Neon

This register is the lower 16 bits used to access the flash or ROM memory on boards that have it. This is not a user programmable register.

### 7.32 CON37 Register

Bit	Name
0	MEM_ADDR_HI
1	MEM_ADDR_HI
2	MEM_ADDR_HI
3	MEM_ADDR_HI
4	MEM_CS
5	MEM_WRITE
6	DWNLD_MODE
7	DWNLD_MODE
8	MEM_DATA
9	MEM_DATA
10	MEM_DATA
11	MEM_DATA
12	MEM_DATA
13	MEM_DATA
14	MEM_DATA
15	MEM_DATA
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

#### MEM\_ADDR\_HI R/W, CON37[3..0], Neon

This register is the upper 4 bits used to access the flash or ROM memory on boards that have it. This is not a user programmable register.

MEM\_CS R/W, CON37[4], Neon

This bit is the chip select which controls both reading and writing to either the flash or the ROM. This bit controls both host access and FPGA download source. This is not a user programmable register.

MEM_CS	Meaning
0	Host and FPGA access is to/from the ROM
1	Host and FPGA access is to/from the flash

#### MEM\_WRITE R/W, CON37[5], Alta, Neon

Used to write to SRAM. Writing a 1 to this bit force the data in MEM\_DATA to be written to the address in MEM\_ADDR.

**DWNLD\_MODE** R/W, CON37[7..6], Alta, Neon

Future use.

MEM\_DATA R/W, CON37[15..8], Neon

This bitfield provides data access used when reading or writing the flash or ROM on boards that support these features. This is not a user programmable register.

### 7.33 CON38 Register

Bit	Name
0	POCL_POWER_ON
1	POCL_EN_GND
2	POCL_CLOCK_WAIT
3	Reserved
4	POCL_SENSE
5	POCL_CLK_
	DETECTED
6	POCL_DETECTED
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

## POCL\_POWER\_ RO, CON38[0], Neon ON

г

This register indicates the state of the power on the Camera Link connector.

POCL_EN_POWER	Meaning
0	Power is not applied to the power wires on the CL cable.
1	Power is applied to the power wires on the CL cable.

### POCL\_GND\_ON RO, CON38[1], Neon

This register indicates the state of the ground on the Camera Link connector.

POCL_GND_ON	Meaning
0	The power wires on the CL cable are not grounded.
1	The power wires on the CL cable are grounded.

## POCL\_CLOCK\_ RO, CON38[2], Neon WAIT

This register indicates that the PoCL state machine is the "waiting for clock" state. In this state, the power has been applied to the camera, but the camera may be powered up yet and may not be output a pixel clock. The PoCL state machine stays in the state for a few seconds, Once it leaves this state, the PoCL state machine will immediately remove the power if it sense that the pixel clock has stopped.

POCL_CLOCK_WAIT	Meaning
0	The PoCL state machine is not in the waiting for clock state.
1	The PoCL state machine is waiting for the pixel from the camera.

### POCL\_SENSE RO, CON38[3], Neon

This register indicates that the PoCL state machine is the "sense" state. In this state, the powers has not been applied, and the PoCL state machine is watching the impedance on the CL cable. If the impedance of a PoCL camera is detected, the power will be applied. It a short is detected, indicating a legacy camera/cable has been con-

nected, the PoCL power lines will be grounded. The PoCL state machine can stay in this state indefinitely if neither of the above two conditions are detected (i.e. nothing is connected).

POCL_SENSE	Meaning
0	The PoCL state machine is not in the sense state.
1	The PoCL state machine is in the sense state.

## POCL\_CLK\_ RO, CON38[5], Neon DETECTED

\_

This register indicates that the PoCL state machine is the "PoCL camera clock has been detected" state. This state is the normal powered up steady state for the PoCL state machine.

POCL_CLK_DETECTED	Meaning
0	The PoCL state machine has not detected a camera pixel clock.
1	The PoCL state machine has detected a camera pixel clock.

POCL\_ RO, CON38[6], Neon DETECTED

This register indicates that the PoCL state machine has detected a PoCL camera..

POCL_DETECTED	Meaning
0	The PoCL state machine has not detected a PoCL- camera.
1	The PoCL state machine has detected a PoCL camera.

### 7.34 CON40 Register

Bit	Name
0	AFE_PORT_ADDR
1	AFE_PORT_ADDR
2	AFE_PORT_ADDR
3	AFE_PORT_ADDR
4	AFE_PORT_ADDR
5	AFE_PORT_ADDR
6	AFE_PORT_ADDR
7	AFE_PORT_ADDR
8	AFE_PORT_WRITE
9	AFE_PORT_ACCESS
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

#### AFE\_PORT\_ R/W, CON40[7..0], Alta ADDR

Used to access the AFE. The value written in the register will be used as the address for subsequent read/write operations.

R/W, CON40[8], Alta AFE\_PORT\_ WRITE

Determines the AFE access operation.

AFE_PORT_WRITE	Meaning
0	The next access operation will be a read.
1	The next access operation will be a write.

AFE\_PORT\_

WO, CON40[9], Alta

ACCESS

Writing a 1 to the bit causes the AFE to be accessed. The type of operation depends on the AFE\_PORT\_WRITE bit.

### 7.35 CON41 Register

Bit	Name
0	AFE_PORT_DATA
1	AFE_PORT_DATA
2	AFE_PORT_DATA
3	AFE_PORT_DATA
4	AFE_PORT_DATA
5	AFE_PORT_DATA
6	AFE_PORT_DATA
7	AFE_PORT_DATA
8	AFE_PORT_BUSY
9	AFE_PORT_ERROR
10	AFE_PORT_RESET
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

## AFE\_PORT\_ R/W, CON41[7..0], Alta DATA

Used to access the AFE. The value written in this bitfield will be use written to the AFE during the next write operation. For read operations, the value read from the AFE will be available in this bitfield after the read operation is complete.

#### AFE\_PORT\_ RO, CON41[8], Alta BUSY

Used when accessing the AFE.

AFE_PORT_BUSY	Meaning
0	The AFE access operation is completed.
1	The AFE access operation is still taking place, the data address/data ports can not be read/written.

## AFE\_PORT\_ RO, CON41[9], Alta ERROR

A 1 in this bit indicates that an error occurred during the last AFE access operation.

### AFE\_PORT\_ WO, CON41[10], Alta

Writing a 1 to the bit resets the AFE access mechanism.

RESET

### 7.36 CON42 Register

Bit	Name
0	FI
1	FIRST_FI
2	RD_WEN
3	Reserved
4	RD_HD
5	RD_VD
6	SWAP_LINES
7	FI_POL
8	SOE
9	SOE
10	ACQ_IV
11	FEN_SEL
12	FEN_SEL
13	FEN_SEL
14	HD_SEL
15	HD_SEL
16	HD_SEL
17	VD_SEL
18	VD_SEL
19	VD_SEL
20	GEN_IV
21	Reserved
22	MID
23	MID
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

FI

### RO, CON42[0], Alta

This bit indicates the current field index of the incoming video if the video is interlaced.

FI	Meaning
0	The field index is low
1	The field index is high

### FIRST\_FI R/W, CON42[1], Alta

This bit indicates the field index of the first field that was captured during a snap or grab operation.

FI	Meaning
0	The field index was low at the start of acquisition
1	The field index was high at the start of acquisition

### RD\_WEN R/W, CON42[2], Alta

\_

This bit indicates the current status of the of the WEN I/O signal.

RD_HD	Meaning
0	The WEN input is currently low
1	The WEN input is currently high

### RD\_HD R/W, CON42[4], Alta

This bit indicates the current status of the of the horizontal sync (HD) I/O signal.

RD_HD	Meaning
0	The HD input is currently low
1	The HD input is currently high

### **RD\_VD** R/W, CON42[5], Alta

This bit indicates the current status of the of the vertical sync (HD) I/O signal.

RD_VD	Meaning
0	The VD input is currently low
1	The VD input is currently high

### SWAP\_LINES R/W, CON42[6], Alta

For some interlaced camera, the odd and even fields must be swapped.

SWAP_LINES	Meaning
0	Do not swap fields
1	Swap fields.

### FI\_POL R/W, CON42[7], Alta

Used to swap the polarity of the field index signal, needed for some interlaced cameras.

FI_POL	Meaning
0	Normal polarity
1	Invert polarity

### **SOE** R/W, CON42[9..8], Alta

This bits dictates which field from an interlaced camera initiates acquisition.

SOE	Meaning
0	The odd field starts acquisition
1	The even field starts acquisition.

### **ACO\_IV** R/W, CON42[10], Alta

This bit tells the acquisition engine if the board is acquiring interlaced or non-interlaced video.

ACQIV	Meaning
0	Incoming video is non-interlaced
1	Incoming video is interlaced

### **FEN\_SEL** R/W, CON42[13..11], Alta

This bitfield controls the sorce of the FEN signal used to control the acquisition engine..

FEN_SEL	Meaning
0 (000b)	Use VD signal from AFE
1 (001b)	Use WEN input signal
2 (010b)	Reserved
3 (011b)	Reserved
4 (100b)	Reserved
5 (101b)	Reserved
6 (110b)	Reserved
7 (111b)	Reserved

### HD\_SEL R/W, CON42[16..14], Alta

This bitfield controls the sorce of the HD output signal

HD_SEL	Meaning
0 (000b)	??
1 (001b)	HD is an output, source is this VFG"s Video Genera- tor
2 (010b)	HD is an output, source is the master VFG"s Video Generator
3 (011b)	HD is an output, source the the CC3 signal

HD_SEL	Meaning
4 (100b)	HD is an output, source is the AFE's detected HD sig- nal
5 (101b)	Reserved
6 (110b)	Reserved
7 (111b)	Reserved

### **VD\_SEL** R/W, CON42[19..17], Alta

This bitfield controls the sorce of the VD output signal.

VD_SEL	Meaning
0 (000b)	??
1 (001b)	VD is an output, source is this VFG"s Video Generator
2 (010b)	VD is an output, source is the master VFG"s Video Generator
3 (011b)	VD is an output, source the the CC4 signal
4 (100b)	VD is an output, source is the AFE's detected VD sig- nal
5 (101b)	Reserved
6 (110b)	Reserved
7 (111b)	Reserved

### GEN\_IV R/W, CON42[20], Alta

This bit is used to select which video generator is used.

GEN_IV	Meaning
0	Standard video generator
1	Custom video generator

### MID R/W, CON42[23..22], Alta

This bitfield is used to comminicate between multiple VFGs on the same board. What ever value is written to this register can be read from all the other VFGs. This register does not control anything.

### 7.37 CON43 Register

Bit	Name
0	GEN_H_PERIOD
1	GEN_H_PERIOD
2	GEN_H_PERIOD
3	GEN_H_PERIOD
4	GEN_H_PERIOD
5	GEN_H_PERIOD
6	GEN_H_PERIOD
7	GEN_H_PERIOD
8	GEN_H_PERIOD
9	GEN_H_PERIOD
10	GEN_H_PERIOD
11	GEN_H_PERIOD
12	GEN_H_PERIOD
13	GEN_H_PERIOD
14	GEN_H_PERIOD
15	GEN_H_PERIOD
16	GEN_H_LOW
17	GEN_H_LOW
18	GEN_H_LOW
19	GEN_H_LOW
20	GEN_H_LOW
21	GEN_H_LOW
22	GEN_H_LOW
23	GEN_H_LOW
24	GEN_H_LOW
25	GEN_H_LOW
26	GEN_H_LOW
27	GEN_H_LOW
28	GEN_H_LOW
29	GEN_H_LOW
30	GEN_H_LOW
31	GEN H LOW

### GEN\_H\_PERIOD R/W, CON43[15..0], Alta

Horizontal period of Video Generator.

GEN\_H\_LOW R/W, CON43[15..0], Alta

Horizontal low period of Video Generator.

### 7.38 CON44 Register

Bit	Name
0	GEN_V_PERIOD
1	GEN_V_PERIOD
2	GEN_V_PERIOD
3	GEN_V_PERIOD
4	GEN_V_PERIOD
5	GEN_V_PERIOD
6	GEN_V_PERIOD
7	GEN_V_PERIOD
8	GEN_V_PERIOD
9	GEN_V_PERIOD
10	GEN_V_PERIOD
11	GEN_V_PERIOD
12	GEN_V_PERIOD
13	GEN_V_PERIOD
14	GEN_V_PERIOD
15	GEN_V_PERIOD
16	GEN_V_LOW
17	GEN_V_LOW
18	GEN_V_LOW
19	GEN_V_LOW
20	GEN_V_LOW
21	GEN_V_LOW
22	GEN_V_LOW
23	GEN_V_LOW
24	GEN_V_LOW
25	GEN_V_LOW
26	GEN_V_LOW
27	GEN_V_LOW
28	GEN_V_LOW
29	GEN_V_LOW
30	GEN_V_LOW
31	GEN V LOW

### **GEN\_V\_PERIOD** R/W, CON43[15..0], Alta

Vertical period of Video Generator.

**GEN\_V\_LOW** R/W, CON43[15..0], Alta

Vertical low period of Video Generator.

### 7.39 CON51 Register

Bit	Name
0	QENC_COUNT
1	QENC_COUNT
2	QENC_COUNT
3	QENC_COUNT
4	QENC_COUNT
5	QENC_COUNT
6	QENC_COUNT
7	QENC_COUNT
8	QENC_COUNT
9	QENC_COUNT
10	QENC_COUNT
11	QENC_COUNT
12	QENC_COUNT
13	QENC_COUNT
14	QENC_COUNT
15	QENC_COUNT
16	QENC_COUNT
17	QENC_COUNT
18	QENC_COUNT
19	QENC_COUNT
20	QENC_COUNT
21	QENC_COUNT
22	QENC_COUNT
23	QENC_COUNT
24	QENC_PHASEA
25	QENC_PHASEB
26	QENC_DIR
27	QENC_INTRVL_IN
28	QENC_NEW_LINES
29	Reserved
30	Reserved
31	Reserved

### **QENC\_COUNT** RO, CON51[23..0], Karbon, Neon

This bitfield displays the current quadrature encoder count.

**QENC\_PHASEA** RO, CON51[24], Karbon, Neon

This bit displays the current logic level of the A quadrature encoder phase.

**QENC\_PHASEB** RO, CON51[25], Karbon, Neon

This bit displays the current logic level of the B quadrature encoder phase.

**QENC\_DIR** RO, CON51[26], Karbon, Neon

This bit displays the current quadrature encoder direction.

QENC_DIR	Meaning
0	Direction is negative
1	Direction is positive

### **QENC\_INTRVL\_** RO, CON51[27], Karbon, Neon

IN

This bit indicates the current status of the quadrature encoder if the system is in interval mode (see QENC\_INTRVL\_MODE).

QENC_INTRVL_IN	Meaning
0	System is not inside the interval. Encoder counter is not between QENC_INTRVL_LL and QENC_INTRVL_UL. Lines are not being acquired.
1	System is inside the interval. Encoder counter is between QENC_INTRVL_LL and QENC_INTRVL_UL. Lines are being acquired.

### QENC\_NEW\_ LINES

### RO, CON51[28], Karbon, Neon

This bit indicates if the system is at an encoder count that corresponds to a new line. When  $QENC_NO_REAQ = 1$ , only lines that have not yet been scanned are acquired. This bit can be used to determine of new lines are being traversed, or if the system has backed up, and is revisiting old lines.

QENC_NEW_LINES	Meaning
0	The system is traversing lines that have already been visited. If QENC_NO_REAQ = 1, lines are not being acquired.
1	The system is traversing new lines. Lines are being acquired.

CON51 Register

# Karbon/Neon/Alta DMA

## Chapter 8

### 8.1 Introduction

This section enumerates all of the registers that control DMA on boards using the PLDA DMA engine. This includes the Alta, the Karbon and the Neon families. This chapter also covers the scatter gather DMA instructions (Quads or QTabs). The formatting of the register sections is explained in Section 7.2.

## 8.2 CON28 Register

Bit	Name
0	FIRST_QUAD_PTR_LO
1	FIRST_QUAD_PTR_LO
2	FIRST_QUAD_PTR_LO
3	FIRST_QUAD_PTR_LO
4	FIRST_QUAD_PTR_LO
5	FIRST_QUAD_PTR_LO
6	FIRST_QUAD_PTR_LO
7	FIRST_QUAD_PTR_LO
8	FIRST_QUAD_PTR_LO
9	FIRST_QUAD_PTR_LO
10	FIRST_QUAD_PTR_LO
11	FIRST_QUAD_PTR_LO
12	FIRST_QUAD_PTR_LO
13	FIRST_QUAD_PTR_LO
14	FIRST_QUAD_PTR_LO
15	FIRST_QUAD_PTR_LO
16	FIRST_QUAD_PTR_LO
17	FIRST_QUAD_PTR_LO
18	FIRST_QUAD_PTR_LO
19	FIRST_QUAD_PTR_LO
20	FIRST_QUAD_PTR_LO
21	FIRST_QUAD_PTR_LO
22	FIRST_QUAD_PTR_LO
23	FIRST_QUAD_PTR_LO
24	FIRST_QUAD_PTR_LO
25	FIRST_QUAD_PTR_LO
26	FIRST_QUAD_PTR_LO
27	FIRST_QUAD_PTR_LO
28	FIRST_QUAD_PTR_LO
29	FIRST_QUAD_PTR_LO
30	FIRST_QUAD_PTR_LO
31	FIRST_QUAD_PTR_LO
#### FIRST\_QUAD\_ PTR\_LO

#### R/W, CON28[31..0], Alta, Karbon, Neon

This is the low word of the 64-bit address of the first DMA scatter-gather instruction in a chain of instructions. This register can be written at any time, but the DMA engine only loads this value when byte count (as set by CHAIN\_DATA\_SIZE\_LO/CHAIN\_DATA\_SIZE\_HI) goes to zero.

### 8.3 CON29 Register

Bit	Name
0	FIRST_QUAD_PTR_HI
1	FIRST_QUAD_PTR_HI
2	FIRST_QUAD_PTR_HI
3	FIRST_QUAD_PTR_HI
4	FIRST_QUAD_PTR_HI
5	FIRST_QUAD_PTR_HI
6	FIRST_QUAD_PTR_HI
7	FIRST_QUAD_PTR_HI
8	FIRST_QUAD_PTR_HI
9	FIRST_QUAD_PTR_HI
10	FIRST_QUAD_PTR_HI
11	FIRST_QUAD_PTR_HI
12	FIRST_QUAD_PTR_HI
13	FIRST_QUAD_PTR_HI
14	FIRST_QUAD_PTR_HI
15	FIRST_QUAD_PTR_HI
16	FIRST_QUAD_PTR_HI
17	FIRST_QUAD_PTR_HI
18	FIRST_QUAD_PTR_HI
19	FIRST_QUAD_PTR_HI
20	FIRST_QUAD_PTR_HI
21	FIRST_QUAD_PTR_HI
22	FIRST_QUAD_PTR_HI
23	FIRST_QUAD_PTR_HI
24	FIRST_QUAD_PTR_HI
25	FIRST_QUAD_PTR_HI
26	FIRST_QUAD_PTR_HI
27	FIRST_QUAD_PTR_HI
28	FIRST_QUAD_PTR_HI
29	FIRST_QUAD_PTR_HI
30	FIRST_QUAD_PTR_HI
31	FIRST_QUAD_PTR_HI

#### FIRST\_QUAD\_ PTR\_HI

#### R/W, CON29[31..0], Alta, Karbon, Neon

This is the high word of the 64-bit address of the first DMA scatter-gather instruction in a chain of instructions. This register can be written at any time, but the DMA engine only loads this value when byte count (as set by CHAIN\_DATA\_SIZE\_LO/CHAIN\_DATA\_SIZE\_HI) goes to zero.

### 8.4 CON30 Register

Bit	Name
0	CHAIN_DATA_SIZE_LO
1	CHAIN_DATA_SIZE_LO
2	CHAIN_DATA_SIZE_LO
3	CHAIN_DATA_SIZE_LO
4	CHAIN_DATA_SIZE_LO
5	CHAIN_DATA_SIZE_LO
6	CHAIN_DATA_SIZE_LO
7	CHAIN_DATA_SIZE_LO
8	CHAIN_DATA_SIZE_LO
9	CHAIN_DATA_SIZE_LO
10	CHAIN_DATA_SIZE_LO
11	CHAIN_DATA_SIZE_LO
12	CHAIN_DATA_SIZE_LO
13	CHAIN_DATA_SIZE_LO
14	CHAIN_DATA_SIZE_LO
15	CHAIN_DATA_SIZE_LO
16	CHAIN_DATA_SIZE_LO
17	CHAIN_DATA_SIZE_LO
18	CHAIN_DATA_SIZE_LO
19	CHAIN_DATA_SIZE_LO
20	CHAIN_DATA_SIZE_LO
21	CHAIN_DATA_SIZE_LO
22	CHAIN_DATA_SIZE_LO
23	CHAIN_DATA_SIZE_LO
24	CHAIN_DATA_SIZE_LO
25	CHAIN_DATA_SIZE_LO
26	CHAIN_DATA_SIZE_LO
27	CHAIN_DATA_SIZE_LO
28	CHAIN_DATA_SIZE_LO
29	CHAIN_DATA_SIZE_LO
30	CHAIN_DATA_SIZE_LO
31	CHAIN_DATA_SIZE_LO

#### CHAIN\_DATA\_ SIZE\_LO

#### R/W, CON30[31..0], Alta, Karbon, Neon

This is the low word if the low word of the 64-bit number of bytes in the chain. The value in this register is loaded into the DMA engine when DMA is initiated. This value is then decremented every DMA transfer. When the count reached zero, this value in this register is reloaded into the DMA engine, and the first scatter gather instruction pointed to by FIRST\_QUAD\_PTR\_HI and FIRST\_QUAD\_PTR\_LO is loaded.

### 8.5 CON31 Register

Bit	Name
0	CHAIN_DATA_SIZE_HI
1	CHAIN_DATA_SIZE_HI
2	CHAIN_DATA_SIZE_HI
3	CHAIN_DATA_SIZE_HI
4	CHAIN_DATA_SIZE_HI
5	CHAIN_DATA_SIZE_HI
6	CHAIN_DATA_SIZE_HI
7	CHAIN_DATA_SIZE_HI
8	CHAIN_DATA_SIZE_HI
9	CHAIN_DATA_SIZE_HI
10	CHAIN_DATA_SIZE_HI
11	CHAIN_DATA_SIZE_HI
12	CHAIN_DATA_SIZE_HI
13	CHAIN_DATA_SIZE_HI
14	CHAIN_DATA_SIZE_HI
15	CHAIN_DATA_SIZE_HI
16	CHAIN_DATA_SIZE_HI
17	CHAIN_DATA_SIZE_HI
18	CHAIN_DATA_SIZE_HI
19	CHAIN_DATA_SIZE_HI
20	CHAIN_DATA_SIZE_HI
21	CHAIN_DATA_SIZE_HI
22	CHAIN_DATA_SIZE_HI
23	CHAIN_DATA_SIZE_HI
24	CHAIN_DATA_SIZE_HI
25	CHAIN_DATA_SIZE_HI
26	CHAIN_DATA_SIZE_HI
27	CHAIN_DATA_SIZE_HI
28	CHAIN_DATA_SIZE_HI
29	CHAIN_DATA_SIZE_HI
30	CHAIN_DATA_SIZE_HI
31	CHAIN_DATA_SIZE_HI

#### CHAIN\_DATA\_ SIZE\_HI

#### R/W, CON31[31..0], Alta, Karbon, Neon

This is the high word if the 64-bit number bytes in the chain. The value in this register is loaded into the DMA engine when DMA is initiated. This value is then decremented every DMA transfer. When the count reached zero, this value in this register is reloaded into the DMA engine, and the first scatter gather instruction pointed to by FIRST\_QUAD\_PTR\_HI and FIRST\_QUAD\_PTR\_LO is loaded.

### 8.6 CON32 Register

Bit	Name
0	CHAIN_DATA_TOGO_LO
1	CHAIN_DATA_TOGO_LO
2	CHAIN_DATA_TOGO_LO
3	CHAIN_DATA_TOGO_LO
4	CHAIN_DATA_TOGO_LO
5	CHAIN_DATA_TOGO_LO
6	CHAIN_DATA_TOGO_LO
7	CHAIN_DATA_TOGO_LO
8	CHAIN_DATA_TOGO_LO
9	CHAIN_DATA_TOGO_LO
10	CHAIN_DATA_TOGO_LO
11	CHAIN_DATA_TOGO_LO
12	CHAIN_DATA_TOGO_LO
13	CHAIN_DATA_TOGO_LO
14	CHAIN_DATA_TOGO_LO
15	CHAIN_DATA_TOGO_LO
16	CHAIN_DATA_TOGO_LO
17	CHAIN_DATA_TOGO_LO
18	CHAIN_DATA_TOGO_LO
19	CHAIN_DATA_TOGO_LO
20	CHAIN_DATA_TOGO_LO
21	CHAIN_DATA_TOGO_LO
22	CHAIN_DATA_TOGO_LO
23	CHAIN_DATA_TOGO_LO
24	CHAIN_DATA_TOGO_LO
25	CHAIN_DATA_TOGO_LO
26	CHAIN_DATA_TOGO_LO
27	CHAIN_DATA_TOGO_LO
28	CHAIN_DATA_TOGO_LO
29	CHAIN_DATA_TOGO_LO
30	CHAIN_DATA_TOGO_LO
31	CHAIN_DATA_TOGO_LO

### CHAIN\_DATA\_ TOGO\_LO

### RO, CON32[31..0], Alta, Karbon, Neon

This register indicates the low word of the 64-bit number of bytes remaining the DMA chain.

### 8.7 CON33 Register

Bit	Name
0	CHAIN_DATA_TOGO_HI
1	CHAIN_DATA_TOGO_HI
2	CHAIN_DATA_TOGO_HI
3	CHAIN_DATA_TOGO_HI
4	CHAIN_DATA_TOGO_HI
5	CHAIN_DATA_TOGO_HI
6	CHAIN_DATA_TOGO_HI
7	CHAIN_DATA_TOGO_HI
8	CHAIN_DATA_TOGO_HI
9	CHAIN_DATA_TOGO_HI
10	CHAIN_DATA_TOGO_HI
11	CHAIN_DATA_TOGO_HI
12	CHAIN_DATA_TOGO_HI
13	CHAIN_DATA_TOGO_HI
14	CHAIN_DATA_TOGO_HI
15	CHAIN_DATA_TOGO_HI
16	CHAIN_DATA_TOGO_HI
17	CHAIN_DATA_TOGO_HI
18	CHAIN_DATA_TOGO_HI
19	CHAIN_DATA_TOGO_HI
20	CHAIN_DATA_TOGO_HI
21	CHAIN_DATA_TOGO_HI
22	CHAIN_DATA_TOGO_HI
23	CHAIN_DATA_TOGO_HI
24	CHAIN_DATA_TOGO_HI
25	CHAIN_DATA_TOGO_HI
26	CHAIN_DATA_TOGO_HI
27	CHAIN_DATA_TOGO_HI
28	CHAIN_DATA_TOGO_HI
29	CHAIN_DATA_TOGO_HI
30	CHAIN_DATA_TOGO_HI
31	CHAIN_DATA_TOGO_HI

### CHAIN\_DATA\_ TOGO\_HI

RO, CON33[31..0], Alta, Karbon, Neon

This register indicates the high word of the 64-bit number of bytes remaining the DMA chain.

### 8.8 CON34 Register

Bit	Name		
0	DMA_AUTO_START		
1	DMA_ABORT		
2	DMA_DIRECTION		
3	DMA_DONE		
4	DMA_STATUS		
5	DMA_STATUS		
6	DMA_STATUS		
7	DMA_STATUS		
8	DMA_NO_RULE		
9	Reserved		
10	Reserved		
11	Reserved		
12	Reserved		
13	Reserved		
14	Reserved		
15	Reserved		
16	DMA_INIT_FUNC		
17	DMA_PRIORITY		
18	DMA_64_BIT		
19	DMA_CHAINING		
20	DMA_COMMAND		
21	DMA_COMMAND		
22	DMA_COMMAND		
23	DMA_COMMAND		
24	DMA_BEN		
25	DMA_BEN		
26	_ DMA_BEN		
27	DMA_BEN		
28	LATCH_CONTROL		
29	LATCH_CONTROL		
30	Reserved		
31	Reserved		

DMA_AUTO_	R/W, CON34[0], Alta, Karbon, Neon			
JIANI	This bit controls how the DMA engine starts.			
	DMA_AUTO_START	Meaning		
	0	Do nothing		
	1	Reload and re-start when CHAIN_DATA_TOGO = 0		
DMA_ABORT	RO, CON34[1], Alta, Karbon, Neon This bit immediately aborts DMA. Always reads back 0.			
	R/W, CON34[2], Alta, Ka	R/W, CON34[2], Alta, Karbon, Neon		
DIRECTION	This bit indicates the direction that DMA engine will move data.			
	DMA_DIRECTION	Meaning		
	0	DMA write (to host memory)		
	1	DMA read (from host memory)		
DMA_DONE	RO, CON34[3], Alta, Kar	bon, Neon		
	Future use.			
DMA_STATUS	RO, CON34[74], Alta, Karbon, Neon			
	Future use.			
DMA_NO_RULE	R/W, CON34[8], Alta, Ka	rbon, Neon		
	Setting this bit to a 1 will cause the DMA engine to DMA data as fast as it not wait for data to be available from the acquisition engine. The actual o DMAed will be unpredictable. This bit, therefore, is only useful for diagn			
DMA_INIT_	R/W, CON34[16], Alta, K	Carbon, Neon		
FUNC	Future use.			

#### DMA\_PRIORITY R/W, CON34[17], Alta, Karbon, Neon

Future use.

DMA\_64\_BIT R/W, CON34[18], Alta, Karbon, Neon

Controls where the DMA operations are 64-bit or 32-bit.

DMA_64_BIT	Meaning
0	32-bit DMA operations
1	64-bit DMA operations

#### DMA\_CHAINING RW, CON34[19], Alta, Karbon, Neon

This bit determines whether the DMA engine will execute chaining DMA or not.

DMA_CHAINING	Meaning
0	Execute a single DMA operations
1	Execute a chain of DMA operations

DMA\_ COMMAND R/W, CON34[23..20], Alta, Karbon, Neon

Controls the DMA engine.

DMA_COMMAND	Meaning
0000b to 1110b	Reserved
1111b	Normal DMA operation

DMA\_BEN R/W, CON34[27..24], Alta, Karbon, Neon

Future use.

LATCH\_ R/W, CON34[29..28], Alta, Karbon, Neon CONTROL Future use.

### 8.9 CON35 Register

Bit	Name
0	XFR_PER_INT
1	XFR_PER_INT
2	XFR_PER_INT
3	XFR_PER_INT
4	XFR_PER_INT
5	XFR_PER_INT
6	XFR_PER_INT
7	XFR_PER_INT
8	XFR_PER_INT
9	XFR_PER_INT
10	XFR_PER_INT
11	XFR_PER_INT
12	XFR_PER_INT
13	XFR_PER_INT
14	XFR_PER_INT
15	XFR_PER_INT
16	XFR_PER_INT
17	XFR_PER_INT
18	XFR_PER_INT
19	XFR_PER_INT
20	XFR_PER_INT
21	XFR_PER_INT
22	XFR_PER_INT
23	XFR_PER_INT
24	XFR_PER_INT
25	XFR_PER_INT
26	XFR_PER_INT
27	XFR_PER_INT
28	XFR_PER_INT
29	XFR_PER_INT
30	XFR_PER_INT
31	XFR_PER_INT

#### **XFR\_PER\_INT** R/W, CON35[31..0], Alta, Karbon, Neon

This register controls how often the board issues an EOF interrupt. Every time XFR\_ PER\_INT bytes have been DMAed, the board will emit an interrupt.

### 8.10 Scatter Gather DMA Instructions

The DMA engine is run by Scatter Gather DMA instructions. These are called "quads" because they generally consist of four words, although the DMA engine only uses three words. A list of instructions are called a Quad Table or QTAB. Each quad consists of the following entries.

- 1. Destination address
- 2. Size of transfer
- 3. Next quad address.

The following sections document the structure of these quads.

### 8.11 Destination Address

Bit	Name	Bit	Name
0	Destination Address	32	Destination Address
1	Destination Address	33	Destination Address
2	Destination Address	34	Destination Address
3	Destination Address	35	Destination Address
4	Destination Address	36	Destination Address
5	Destination Address	37	Destination Address
6	Destination Address	38	Destination Address
7	Destination Address	39	Destination Address
8	Destination Address	40	Destination Address
9	Destination Address	41	Destination Address
10	Destination Address	42	Destination Address
11	Destination Address	43	Destination Address
12	Destination Address	44	Destination Address
13	Destination Address	45	Destination Address
14	Destination Address	46	Destination Address
15	Destination Address	47	Destination Address
16	Destination Address	48	Destination Address
17	Destination Address	49	Destination Address
18	Destination Address	50	Destination Address
19	Destination Address	51	Destination Address
20	Destination Address	52	Destination Address
21	Destination Address	53	Destination Address
22	Destination Address	54	Destination Address
23	Destination Address	55	Destination Address
24	Destination Address	56	Destination Address
25	Destination Address	57	Destination Address
26	Destination Address	58	Destination Address
27	Destination Address	59	Destination Address
28	Destination Address	60	Destination Address
29	Destination Address	61	Destination Address
30	Destination Address	62	Destination Address
31	Destination Address	63	Destination Address

### 8.12 Size of Transfer

Bit	Name
0	Data Size
1	Data Size
2	Data Size
3	Data Size
4	Data Size
5	Data Size
6	Data Size
7	Data Size
8	Data Size
9	Data Size
10	Data Size
11	Data Size
12	Data Size
13	Data Size
14	Data Size
15	Data Size
16	Data Size
17	Data Size
18	Data Size
19	Data Size
20	Data Size
21	Data Size
22	Data Size
23	Data Size
24	Data Size
25	Data Size
26	Data Size
27	Data Size
28	Data Size
29	Data Size
30	Data Size
31	Data Size

### 8.13 Next Quad Address

Bit	Name	Bit	Name
0	Next Quad Address	32	Next Quad Address
1	Next Quad Address	33	Next Quad Address
2	Next Quad Address	34	Next Quad Address
3	Next Quad Address	35	Next Quad Address
4	Next Quad Address	36	Next Quad Address
5	Next Quad Address	37	Next Quad Address
6	Next Quad Address	38	Next Quad Address
7	Next Quad Address	39	Next Quad Address
8	Next Quad Address	40	Next Quad Address
9	Next Quad Address	41	Next Quad Address
10	Next Quad Address	42	Next Quad Address
11	Next Quad Address	43	Next Quad Address
12	Next Quad Address	44	Next Quad Address
13	Next Quad Address	45	Next Quad Address
14	Next Quad Address	46	Next Quad Address
15	Next Quad Address	47	Next Quad Address
16	Next Quad Address	48	Next Quad Address
17	Next Quad Address	49	Next Quad Address
18	Next Quad Address	50	Next Quad Address
19	Next Quad Address	51	Next Quad Address
20	Next Quad Address	52	Next Quad Address
21	Next Quad Address	53	Next Quad Address
22	Next Quad Address	54	Next Quad Address
23	Next Quad Address	55	Next Quad Address
24	Next Quad Address	56	Next Quad Address
25	Next Quad Address	57	Next Quad Address
26	Next Quad Address	58	Next Quad Address
27	Next Quad Address	59	Next Quad Address
28	Next Quad Address	60	Next Quad Address
29	Next Quad Address	61	Next Quad Address
30	Next Quad Address	62	Next Quad Address
31	Next Quad Address	63	Next Quad Address

# **Register and Memory Mapping**

## Chapter 9

### 9.1 Introduction

This section explains how the registers and the various chunks of memory are mapped and accessed on the Alta/Karbon/Neon and their virtual frame grabbers.

### 9.2 Memory Types

### 9.2.1 Registers

All registers are 64 bits wide and on 64-bit boundary. With the exception of the DPM, only data bits 31 to 0 are used, bits 63 to 32 are always "don't care". The DPM uses all 64 bits. Out of the lower 32 LSBs, some registers use only a portion of the bits. Registers can also be accessed as 32 bit wide. Little endian addressing is used, i.e. MSB is bits 63-56.

### 9.2.2 UART

The UART is 8 bit wide and its registers are on 64-bit boundary. The UART is only on the Karbon and Neon.

### 9.2.3 DPM

The DPM is 64 bit wide. The DPM can be accessed as 64-bit (on 64-bit boundary) or as 32-bit wide (on 32-bit boundary) memory. During acquisition (GRAB/SNAP), the slave read from DPM is inhibited. In this case, data read will be always zero.

Note: The DPM is only accessible form host on the R64.

### 9.2.4 CTABs

The CTabs for the Karbon, Alta and Neon are implemented slightly differently than the R64. These CTabs are Run Length Encoded (RLE). The RLE CTabs are stored in the same address space as is used for the CTabs on the R64, however, only the first 256 locations are actually populated. This reason this works is that the RLE CTabs can compress the normal CTabs by a very large amount, considerably reduce that memory requirements for CTabs.

### 9.3 Memory Map

The following table illustrates the physical location of the various sections of memory on the board. The addresses are offset from the BAR1 PCI base address.

Memory	Address (hex)	Comment
CON0	00 80 00 00	download, clock control
CON1	00 00 00 00	Camera Control Register
CON2	00 02 00 00	Camera Control Register
CON3	00 04 00 00	Camera Control Register
CON4	00 06 00 00	Camera Control Register
CON5	00 08 00 00	Camera Control Register
CON6	00 0A 00 00	Camera Control Register
CON7	00 0C 00 00	Camera Control Register
CON8	00 0E 00 00	Camera Control Register
CON9	00 10 00 00	Camera Control Register
CON10	00 10 00 08	Camera Control Register
CON11	00 10 00 10	Camera Control Register
CON12	00 10 00 18	Camera Control Register
CON13	00 10 00 20	Camera Control Register
CON14	00 10 00 28	Camera Control Register
CON15	00 00 00 08	Camera Control Register
CON16	00 00 00 10	Camera Control Register
CON17	00 00 00 18	Camera Control Register
CON18	00 00 00 20	Camera Control Register
CON19	00 00 00 28	Camera Control Register
CON20	00 10 00 30	Camera Control Register
CON21	00 10 00 38	Camera Control Register
CON22	00 00 00 30	Camera Control Register
CON23	00 00 00 38	Camera Control Register
CON24	00 10 00 40	Camera Control Register
CON25	00 10 00 48	Camera Control Register
CON26	00 00 00 40	Camera Control Register
CON27	00 00 00 48	Camera Control Register
CON28	00 00 00 50	DMA Register
CON29	00 00 00 58	DMA Register
CON30	00 00 00 60	DMA Register
CON31	00 00 00 68	DMA Register
CON32	00 00 00 70	DMA Register
CON33	00 00 00 78	DMA Register
CON34	00 00 00 80	DMA Register
CON35	00 00 00 88	DMA Register
CON36	00 80 00 18	Alta/Neon Only
CON37	00 80 00 20	Alta/Neon Only
CON38	00 80 00 28	Neon Only
CON40	00 00 00 98	Alta Only

Memory	Address (hex)	Comment
CON41	00 00 00 A0	Alta Only
CON42	00 00 00 A8	Alta Only
CON43	00 00 00 B0	Alta Only
CON44	00 00 00 B8	Alta Only
CON45	00 00 00 C0	Alta Only
CON46	00 00 00 C8	Alta Only
CON47	00 00 00 D0	Alta Only
CON48	00 00 00 D8	Alta Only
CON49	00 00 00 E0	Alta Only
CON50	00 00 00 E8	Alta Only
CON51	00 00 00 F0	Camera Control Register
CTABS	00 20 00 00	Only first 256 address populated
DPM	00 50 00 00	Dual ported memory, R64 Only
UART	00 70 00 00	8 internal 8-bit registers on 64 bit
		boundary, Karbon/None only
RO_INFOHI	00 80 00 08	R/O info, model/rev, etc.
RO_INFOLO	00 80 00 10	R/O info, model/rev, etc.

The following pertains to the table above.

All registers are treated as 64 bits wide.

Two BARs are allocated for PCI access.

BAR0, is not currently used.

BAR1, memory mapped, 16M size, is used for access to registers, CTABs, DPM.

### 9.4 Downloading Firmware

On the Karbon family, firmware is downloaded using a special downloader module. The downloader module always resident on the board. When flashing the FPGA, either the download module can be written or the real firmware (written to the board from the host) can be written to the chip.

On the Alta and Neon families, download is facilitated by writing to board resident SRAM. This SRAM is always available on the board and is access indirectly. Once the SRAM is loaded with new firmware, the board's FPGAs can be flashed directly from the SRAM.

### 9.5 PCI Configuration Space and Model/Revision Information

Each family of boards has its own device ID as follows:

Alta - 0x5000 Karbon - 0x3000 Neon - 0x4000

Information about different models and board capabilities is stored in the INFO\_HI and INFO\_LO registers.

# **Electrical Interfacing**

## Chapter 10

### 10.1 Introduction

This chapter describes the electrical interface of the Karbon/Neon/R64. This includes detailed information on the all if the input and output signals. In addition, information is provided on recommend circuits to use when connecting to these signals.

### 10.2 Trigger

### 10.2.1 Trigger Input Types

There are four trigger inputs.

TRIGGER\_TTL - Single ended TTL level trigger TRIGGER\_DIFF - Differential (LVDS) trigger TRIGGER\_OPTO - Optocoupled trigger FEN - The FEN signal on the CL1 connector.

The hardware trigger is enabled/disabled by the bit EN\_TRIGGER. Only one input at a time is active (the software trigger bit, SW\_TRIG, is always active). The active trigger is selected by the bitfield SEL\_TRIG. The current level of these inputs can be read from software by reading the bits RD\_TRIG\_DIFF, RD\_TRIG\_TTL, RD\_TRIG\_OPTO and RD\_FEN. The unselected triggers will have no effect on the board. However, they can be used as general purpose inputs.

### 10.2.2 The Optocoupled Trigger

The opto-coupled trigger allows the acquisition circuitry to accept a trigger signal without having a galvanic connection to the trigger source. This is mandatory in some medical and industrial application. The trigger information is passed as a light pulse from an on-board LED that is coupled to a receiving phototransistor. The LED and the phototransistor are in the same package, a Sharp PC3H711. Figure 10-1 shows the electrical diagram of the optocoupled trigger circuit and a suggested circuit for the driver. The LED is driven by an open collector driver. The user must supply his +5V power to the LED. Note that there is no galvanic connection between the user's circuit and the acquisition circuitry.



Figure 10-1 Driver Circuit for Opto-Coupled Trigger

### 10.3 Encoder

### 10.3.1 Encoder Input Types

There are three encoder inputs.

ENCODER\_TTL - Single ended TTL level encoder ENCODER\_DIFF - Differential (LVDS) encoder ENCODER\_OPTO - Optocoupled encoder

The hardware encoder is enabled/disabled by the bit EN\_ENCODER. Only one input at a time is active (the software encoder bit, SW\_ENC, is always active). The active encoder is selected by the bitfield SEL\_ENC. The current level of these inputs can be read from software by reading the bits RD\_ENC\_DIFF, RD\_ENC\_TTL and RD\_ ENCOPTO. The unselected inputs will have no effect on the board. However, they can be used as general purpose inputs.

### 10.3.2 The Optocoupled Encoder

The opto-coupled encoder allows the acquisition circuitry to accept an encoder signal without having a galvanic connection to the encoder source. This is mandatory in some medical and industrial application. The encoder information is passed as a light pulse from an on-board LED that is coupled to a receiving phototransistor. The LED and the phototransistor are in the same package, a Sharp PC3H711. Figure 10-2 shows the electrical diagram of the optocoupled encoder circuit and a suggested circuit for the driver. The LED is driven by an open collector driver. The user must supply his +5V power to the LED. Note that there is no galvanic connection between the user's circuit and the acquisition circuitry.



Figure 10-2 Driver Circuit for Opto-Coupled Encoder

### 10.4 General Purpose Inputs (GPIN)

### 10.4.1 Introduction

General Purpose Inputs (GPIN) are use to relay the state of an external signal onto the board and ultimately make it available to a software program. In other words, if an external signal connected to a GPIN pin is electrically high, then an associated register will read back one, if the same signal is low, then the bit will read back zero.

There are two different type of GPIN input pins, TTL and differential (LVDS). For each GPIN pin, there is an associated GPIN register. See the pin-out in the mechanical chapter to determine the actual pins each signal resides on.

Each frame grabber family has a slightly different arrangement of GPIN signals. These are enumerated in the following sections.

#### 10.4.2 R64 GPIN Configuration

The R64 has five general purpose inputs. The signal level on each input can be read on the corresponding GPIN bit. The electrical characteristic of these inputs is shown in the following list.

GPIN0, GPIN1 - Single ended TTL level inputs GPIN2, GPIN3, GPIN4 - Differential (LVDS) inputs

### 10.4.3 Neon-CL GPIN Configuration

The Neon-CL has three general purpose inputs. They are as follows:

GPIN0, GPIN1 - Single ended TTL level inputs GPIN2 - Differential (LVDS) inputs

Note: GPIN3 and GPIN4 are not available on the Neon-CL

#### 10.4.4 Karbon-CL GPIN Configuration

The Karbon-CL has five general purpose inputs. They are only available for reading on VFG0. This means that even though all VFG's have the bit GPIN0 to GPIN4, these will only read back correctly on the first VFG (i.e. VFG0). The configuration of GPIN signals is as follows;

GPIN0, GPIN1 - Single ended TTL level inputs GPIN2, GPIN3, GPIN4 - Differential (LVDS) inputs

### 10.5 General Purpose Outputs (GPOUT)

### 10.5.1 Introduction

The General Purpose Outputs (GPOUT) are used to control external hardware (e.g. strobes, stages, etc.). Each GPOUT has a pin on the I/O connector. The level on this pin can be controlled either statically via a GPOUT register, or dynamically via one of the on-board signal generators.

There are three different electrical types of GPOUTs: TTL, Differential (LVDS) and Open Collector. Each type is described in more detail below. See the IO connector pin-out tables to determine which signals are on which pins.

Not every frame grabber family has the same number and type of GPOUTs. The specifications for each family are described later in this section.

### 10.5.2 GPOUT Source Options

The source for each GPOUT bit is controlled by the corresponding GPOUTx\_CON bitfield. These are located in CON8. The source for each GPOUT can be programmed independently of the others. Table 10-1 shows the sources for each GPOUT.

GPOUTx_CON	GPOUTx Source
0 (000b)	GPOUTx bit
1 (001b)	CT3 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	CT1 from CTAB
4 (100b)	CT0 from CTAB
5 (101b)	Internally generated clock. Frequency set by CFREQ.
6 (110b)	Free running on board signal generator. Controlled by FREE_RUN_RATE and FREE_ RUN_HIGH
7 (111b)	Reserved

#### Table 10-1 GPOUTx\_CON

### 10.5.3 R64 GPOUT Configuration

The R64 and R64e model boards have seven general purpose outputs. The electrical characteristic of these outputs is shown in the following list.

GPOUT0, GPOUT1, GPOUT2 - Differential (LVDS) outputs

GPOUT3, GPOUT4 - Single ended TTL level outputs GPOUT5, GPOUT6 - Open collector

### 10.5.4 Neon-CL GPOUT Configuration

The Neon-CL model boards have four general purpose outputs. The electrical characteristic of these outputs is shown in the following list.

GPOUT0 - Differential (LVDS) outputs GPOUT3, GPOUT4 - Single ended TTL level outputs GPOUT5 - Open collector

Note: GPOUT2, GPOUT3 and GPOUT6 are not available on the Neon-CL.

### 10.5.5 Karbon-CL GPOUT Configuration

The Karbon-CL model boards have seven general purpose outputs. However, they are divided up between up to four Virtual Frame Grabbers (VFGs). Each VFG has two GPOUTs. The electrical characteristic of these outputs on each VFG are not the same. Table 10-2 shows the options for each VFG.

#### Table 10-2 Karbon-CL GPOUTs vs. VFGs

Signal	VFG0	VFG1	VFG2	VFG3
GPOUT0	Differential	Differential	Differential	Open Collector
GPOUT1	TTL	TTL	Open Collector	TTL

Note: GPOUT2 through GPOUT6 are not available on any VFG on the Karbon-CL.

### 10.5.6 GPOUT Open Collector Drivers

The GPOUT open collector driver circuit can be used in two different ways. The circuit can be used to drive a opto-coupled circuit (see Figure 10-3, default configuration) or the circuit can be used to drive and opto-coupled circuit with galvanic isolation (see Figure 10-4). Jumpers are used to configure the driver circuits. See the Mechanical chapter on where and how the jumpers are used.



### Figure 10-3 GPOUT5 Driving Opto-Coupled Circuit (in Default Configuration)

Figure 10-3 shows how the open collector GPOUT5 in the factory configuration can drive an opto-coupling device. The user must supply the +5V to his LED and the two systems must have their grounds connected. In this configuration the board and the user's system must have a common electrical ground.



#### Figure 10-4 GPOUT5 Driving Opto-Coupled Circuit using Galvanic Isolation

Figure 10-4 shows how the board's open collector GPOUT5 can drive an user's optocoupled device configured for galvanic isolation between the board and the user. The power to the user's LED is supplied by the board's 5V through a 220 Ohm resistor. This is achieved by inserting the short in position 1-2 at JP1. The jumper at JP2 is removed. The open collector driver will sink the current from the LED. There is no galvanic connection between the board and the user's circuit. Information is passed from the board to the user as light, transmitted by the LED and received by the photo-transistor.
# 10.6 Camera Link Controls (CCs)

The Camera Link cable caries four general purpose Camera Control (CC) signals that can be use to control the camera. These are labeled CC1, CC2, CC3, CC4. The source for each CC is controlled by the corresponding CCx\_CON bitfield. Table 10-3 illustrates the source for each CCx as a function of its associated CCx\_CON bitfield.

CCx_CON	CCx Source
0 (000b)	CT0 from CTAB
1 (001b)	CT1 from CTAB
2 (010b)	CT2 from CTAB
3 (011b)	Free running on board signal generator. Controlled by FREE_RUN_RATE and FREE_ RUN_HIGH
4 (100b)	Internally generated clock. Frequency set by CFREQ.
5 (101b)	GPIN0's signal level
6 (110b)	Forced low
7 (111b)	Forced high

### Table 10-3 CCx\_CON

# **Specifications**

# Chapter 11

# 11.1 Introduction

This chapter describes the general specifications of the Neon family. The numerical values for he specifications are listed in Table 11-1. If more information is available for a given specification there will be an entry in the column marked "Details".

Specifications	Value	Units	Details
PCIe Compatibility (PCI Express Version)	x8 and x16	Slot size	
Maximum Input Pixel Clock Frequency	85	MHz	
Minimum Input Pixel Clock Frequency	20	MHz	
Maximum Pixels Per Line (1 tap)	262,144 (256K)	Pixels	Section 11.2
Maximum Lines Per Frame	131,072 (128K)	Lines	Section 11.3
Minimum clocks between lines	4	Clocks	
Minimum lines between frames	0	Lines	
Minimum pixel clocks between frames	4	Clocks	
Minimum trigger pulse	600	Nanoseconds	
Minimum encoder pulse	600	Nanoseconds	
Current, CL4 models (3.3 Volt)	4.5	Amps	
Current, CL4 models (12 Volt)	0.2	Amps	
Current, CL2 models (3.3 Volt)	4.0	Amps	
Current, CL2 models (12 Volt)	0.2	Amps	
Temperature range	0 to 50	Degrees Celsius	
Humidity	25% to 80%		
Mechanical dimensions	6.8 x 4.2	Inches	
Mechanical dimensions	17.4 x 10.6	Centimeters	
Minimum UART baud rate	110	Bits/Second	
Maximum UART baud rate	256K	Bits/Second	

### Table 11-1 Karbon Specifications

## 11.2 Maximum Pixels Per Line

The maximum number of pixels per line is given by the following formula:

Max\_pix\_per\_line = 256K x Taps

Taps is the number of taps. A tap supplies a whole pixel.

Examples:

A two tap camera that supplies odd/even pixels, Max\_pix\_per\_line = 512K.

An RGB camera that supplies RGB over 24 bits, Max\_pix\_per\_line = 256K, as every clock the camera supplies one single pixel.

A four tap, two segments, each left right, Max\_pix\_per\_line = 1M

### 11.3 Maximum Lines Per Frame

For area scan cameras, the maximum number of lines per frame is given by the following formula:

Max\_lines\_per\_frame = 128K x Line\_taps

Line\_taps is the number of taps that supply a whole line.

Examples:

A one tap camera, Max\_lines\_per\_frame = 128K.

A two tap camera that supplies odd/even lines, Max\_lines\_per\_frame = 256K.

A two tap camera that supplies odd/even pixels, Max\_lines\_per\_frame = 128K.

Maximum Lines Per Frame

# Mechanical

# Chapter 12

### 12.1 Introduction

This chapter describes the mechanical characteristics of the Karbon-CL. This includes description of all of the connectors on the board and pin-outs for these connectors.

The Karbon-CL is available in two version. The KBN-PCE-CL2, which consists just of the main board, and the KBN-PCE-CL4 which consists of the main board plus the auxiliary board. The primary purpose of the auxiliary board is to hold the third and fourth Camera Link connectors. This board does not need to be in a PCI Express slot. In fact, it does not require any power from the PC and can be located in any slot that is not used.

The mechanical layout of the Karbon-CL main board is shown in Figure 12-1. The mechanical layout of the Karbon-CL auxiliary board is shown in Figure 12-2.



Figure 12-1 Karbon-CL Main Board Layout



Figure 12-2 Karbon Auxiliary Board Layout

## 12.2 The Karbon-CL Connectors

There are six connectors on the Karbon-CL main board:

CL1 - Camera Link Connector 1

- CL2 Camera Link Connector 2
- P1 Auxiliary Connector 1
- P2 Auxiliary Connector 2
- P3 Auxiliary Connector 3
- P11 The I/O connector

Figure 12-1 shows the locations of these connectors.

There are five connectors on the Karbon-CL auxiliary board:

CL3 - Camera Link Connector 3

- CL4 Camera Link Connector 4
- P12 Auxiliary Connector 1
- P13 Auxiliary Connector 2
- P14 Auxiliary Connector 3

Figure 12-2 shows the locations of these connectors.

### 12.2.1 The CL Connectors

The CL connectors are for connecting Camera Link cameras. Table 12-1 illustrates how to connect the Karbon to various types and numbers of Camera Link Cameras. Also for each camera, the Virtual Frame Grabber (VFG) that the camera will be connected with is shown.

Camera(s)	CL1	CL2	CL3	CL4
One Base Camera	Camera 1 VFG0			
Two Base Cameras	Camera 1 VFG0	Camera 2 VFG1		
Three Base Cameras	Camera 1 VFG0	Camera 2 VFG1	Camera 3 VFG2	
Four Base Cameras	Camera 1 VFG0	Camera 2 VFG1	Camera 3 VFG2	Camera 4 VFG3

#### Table 12-1 CL Connector Configuration

Camera(s)	CL1	CL2	CL3	CL4
One Medium Camera	Camera 1, Connector 1 VFG0	Camera 1, Connector 2 VFG0		
Two Medium Cameras	Camera 1, Connector 1 VFG0	Camera 1, Connector 2 VFG0	Camera 2, Connector 1 VFG1	Camera 2, Connector 2 VFG1
One Full Camera	Camera 1, Connector 1 VFG0	Camera 1, Connector 2 VFG0		
Two Full Cameras	Camera 1, Connector 1 VFG0	Camera 1, Connector 2 VFG0	Camera 2, Connector 1 VFG1	Camera 2, Connector 2 VFG1

#### Table 12-1 CL Connector Configuration

Another way to look at the connectors is shown in Table 12-2. This describes the relation ship between the physical connectors on the board and the VFG's seen in Sys-Reg.

Connector	CL2-D models	CL2-F models	CL4-D models	CL4-F models
CL1	VFG0	VFG0	VFG0	VFG0
CL2	VFG1	VFG1	VFG1	VFG0
CL3			VFG2	VFG1
CL4			VFG3	VFG1

### 12.2.2 The I/O Connector

The I/O connector, P11, contains a number of general purpose inputs and outputs. The outputs can be used, for example, to control a strobe light or other devices. Some of the inputs have specific functions, for example the Trigger and Encoder, and some are general purpose, for example GPINO, whose state can be read by software. These signals are described in detail in the following sections.

### 12.2.3 The Auxiliary Connectors

The auxiliary connectors, P1, P2 and P3, are used to connect the main board to the auxiliary board connectors P12, P13 and P14 respectively. The connectors are different sizes to avoid incorrect pairing.

### 12.3 The Jumpers and Switches

On the Karbon-CL main board there are four jumper fields that are user configurable, JP2, JP3, JP4 and JP5 and two that are not, JP10 and JP11. Figure 12-3 shows the locations of these jumpers and switches. On the Karbon-CL auxiliary board, there are two jumper fields, JP1 and JP2, which are also not user configurable and that have the same functionality JP10 and JP11 on the main board. See Figure 12-2 for the location of these jumpers.

### 12.3.1 Jumpers

Figure 12-3 shows the detailed structure of the jumpers.



Figure 12-3 Karbon-CL Jumpers

Jumper fields JP2, JP3, JP4 and JP5 control the configuration of the open collector GPOUT5 and GPOUT6. See Section 10.5.6 on GPOUT5 and GPOUT6 configuration.

Jumper fields JP10 and JP11 on the main board and JP1 and JP2 on the auxiliary board are not user configurable. They are set at the factory in conjunction with FLASH programmable logic, when the board is configured as a FULL or DUAL model. The user should not change the factory configuration of these jumpers as it will result in malfunction of the board. The jumpers setting will disagree with the on-board FLASH logic. Table 12-3 illustrates the configurations of JP2, JP3, JP4 and JP5. See also Section 10.5.6 on configuration of GPOUT5 and GPOUT6.

Jumper	Position 1-2	Position 2-3	No Jumper
JP2	5 Volt connected to GPOUT5_VCC through a 220 Ohm resistor	12 Volt connected to GPOUT5_VCC through a 680 Ohm resistor	Power disconnected from GPOUT5_VCC (factory default)
JP3	1K pull-up resistor installed between GPOUT5_OC and GPOUT5_VCC (fac- tory default)	N. A.	No pull-up on GPOUT5
JP4	5 Volt connected to GPOUT6_VCC through a 220 Ohm resistor	12 Volt connected to GPOUT6_VCC through a 680 Ohm resistor	Power disconnected from GPOUT6_VCC (factory default)
JP5	1K pull-up resistor installed between GPOUT6_OC and GPOUT6_VCC (fac- tory default)	N.A.	No pull-up on GPOUT6

#### Table 12-3 Jumper Set 1

Table 12-4 illustrates the factory setting for JP10 and JP11 on the main board and JP1 and JP2 on the auxiliary board.

#### Table 12-4 Jumper Set 2 and 3

Jumper Field	Position 1-2	Position 2-3
JP10 main, JP1 aux	Full Configuration Cameras	Dual Base and Medium Configu- ration Cameras
JP11 main, JP2 aux	Full Configuration Cameras	Dual Base and Medium Configu- ration Cameras

Note: Users should not change the factory setting of JP1,. JP2, JP10 and JP11.

### 12.4 Switches

There is one piano-type switch block on the Karbon-CL with two switches. These are used to identify individual boards when there is more than one board in a system. The idea is to set the switches differently on each board in the system. The switch settings can be read for each board from software (by reading the SW bitfield). SysReg also shows the switch setting for each board. See Table 12-5 below shows the switch settings and the corresponding value in the SW bitfield.

S1	S2	SW register
down	down	0
down	up	1
up	down	2
up	up	3

# 12.5 The Camera Link Connector Pinouts (CL1 to CL4)

The pinouts for connectors CL1, CL2, CL3 and CL4 conform to the Camera Link specification for frame grabbers. This specification is maintained by the Automated Imaging Association. Please contact this organization for a copy of the specification. At the time of this printing, it is available on the web at www.machinevisiononline.org. It is important to understand that some of these signals are the output of a high speed serial converter chip, and require special instrumentation to be observed.

### 12.6 I/O Connector Pinout for the KBN-PCE-4-D

The pin-out for the I/O Connector (P11) for the Karbon model KBN-PCE-CL4-D is illustrated in the Table 12-6.

Note: Signal names start with the Virtual Frame Grabber (VFG) that they are routed to. For example, the signal VFG0\_TRIGGER\_TTL is wired to VFG0., while VFG2\_TRIGGER\_TTL is wired to VGF2.

Pin	I/O	Signal	Comment
1	In	VFG0_TRIGGER+	LVDS
2	In	VFG0_TRIGGER-	LVDS
3	In	VFG0_TRIGGER_TTL	TTL
4	In	VFG0_TRIGGER_OPTO_K	Cathode of optocoupling sensor
5	In	VFG0_TRIGGER_OPTO_A	Anode of optocoupling sensor
6		GND	
7	In	VFG0_ENCODER+	LVDS
8	In	VFG0_ENCODER-	LVDS
9	In	VFG0_ENCODER_TTL	TTL
10	In	VFG0_ENCODER_OPTO_K	Cathode of optocoupling sensor
11	In	VFG0_ENCODER_OPTO_A	Anode of optocoupling sensor
12		GND	
13	In	VFG0_GPIN0_TTL	TTL
14	In	VFG0_GPIN1_TTL	TTL
15	In	VFG0_GPIN2+	LVDS
16	In	VFG0_GPIN2-	LVDS
17	In	VFG0_GPIN3+	LVDS
18	In	VFG0_GPIN3-	LVDS
19	In	VFG0_GPIN4+	LVDS
20	In	VFG0_GPIN4-	LVDS
21		GND	
22	Out	VFG0_GPOUT0+	LVDS
23	Out	VFG0_GPOUT0-	LVDS
24	Out	VFG1_GPOUT0+	LVDS
25	Out	VFG1_GPOUT0-	LVDS
26	Out	VFG2_GPOUT0+	LVDS
27	Out	VFG2_GPOUT0-	LVDS
28	Out	VFG0_GPOUT1_TTL	TTL
29		GND	
30	Out	VFG1_GPOUT1_TTL	TTL
31	Out	VFG3_GPOUT0_OC	Open collector driver

Table 12-6 I/O Connector for the KBN-PCE-4-D

Pin	I/O	Signal	Comment
32	Out	VFG3_GPOUT0_VCC	Pull-up or power for the open col- lector driver
33	Out	VFG2_GPOUT1_OC	Open collector driver
34	Out	VFG2_GPOUT1_VCC	Pull-up or power for the open col- lector driver
35		GND	
36		GND	
37		Reserved	
38		GND	
39		Reserved	
40		GND	
41	In	VFG1_TRIGGER+	LVDS
42	In	VFG1_TRIGGER-	LVDS
43	In	VFG1_TRIGGER_TTL	TTL
44	In	VFG1_ENCODER+	LVDS
45	In	VFG1_ENCODER-	LVDS
46	In	VFG1_ENCODER_TTL	TTL
47	ln	VFG2_TRIGGER+	LVDS
48	In	VFG2_TRIGGER-	LVDS
49	In	VFG2_TRIGGER_TTL	TTL
50	In	VFG2_ENCODER+	LVDS
51	In	VFG2_ENCODER-	LVDS
52	In	VFG2_ENCODER_TTL	TTL
53	In	VFG3_TRIGGER+	LVDS
54	In	VFG3_TRIGGER-	LVDS
55	In	VFG3_TRIGGER_TTL	TTL
56	In	VFG3_ENCODER+	LVDS
57	In	VFG3_ENCODER-	LVDS
58	In	VFG3_ENCODER_TTL	TTL
59		Reserved	
60	Out	VFG3_GPOUT1_TTL	TTL

### Table 12-6 I/O Connector for the KBN-PCE-4-D

## 12.7 I/O Connector Pinout for the KBN-PCE-2-D/4-F

The pin-out for the I/O Connector (P11) for both the Karbon models KBN-PCE-2-D and KBN-PCI-4-F are illustrated in the Table 12-7.

Note: Signal names start with the Virtual Frame Grabber (VFG) that they are routed to. For example, the signal VFG0\_TRIGGER\_TTL is wired to VFG0., while VFG2\_TRIGGER\_TTL is wired to VGF2.

Pin	I/O	Signal	Comment
1	In	VFG0_TRIGGER+	LVDS
2	In	VFG0_TRIGGER-	LVDS
3	In	VFG0_TRIGGER_TTL	TTL
4	In	VFG0_TRIGGER_OPTO_K	Cathode of optocoupling sensor
5	In	VFG0_TRIGGER_OPTO_A	Anode of optocoupling sensor
6		GND	
7	In	VFG0_ENCODER+	LVDS
8	In	VFG0_ENCODER-	LVDS
9	In	VFG0_ENCODER_TTL	TTL
10	In	VFG0_ENCODER_OPTO_K	Cathode of optocoupling sensor
11	In	VFG0_ENCODER_OPTO_A	Anode of optocoupling sensor
12		GND	
13	In	VFG0_GPIN0_TTL	TTL
14	In	VFG0_GPIN1_TTL	TTL
15	In	VFG0_GPIN2+	LVDS
16	In	VFG0_GPIN2-	LVDS
17	In	VFG0_GPIN3+	LVDS
18	In	VFG0_GPIN3-	LVDS
19	In	VFG0_GPIN4+	LVDS
20	In	VFG0_GPIN4-	LVDS
21		GND	
22	Out	VFG0_GPOUT0+	LVDS
23	Out	VFG0_GPOUT0-	LVDS
24	Out	VFG1_GPOUT0+	LVDS
25	Out	VFG1_GPOUT0-	LVDS
26	Out	VFG0_GPOUT3+	LVDS
27	Out	VFG0_GPOUT3-	LVDS
28	Out	VFG0_GPOUT1_TTL	TTL
29		GND	
30	Out	VFG1_GPOUT1_TTL	TTL
31	Out	VFG0_GPOUT2_OC	Open collector driver

Table 12-7 I/O Connector for the KBN-PCE-2-D/4-F

Pin	I/O	Signal	Comment
32	Out	VFG0_GPOUT2_VCC	Pull-up or power for the open col- lector driver
33	Out	VFG1_GPOUT2_OC	Open collector driver
34	Out	VFG1_GPOUT2_VCC	Pull-up or power for the open col- lector driver
35		GND	
36		GND	
37		Reserved	
38		GND	
39		Reserved	
40		GND	
41	In	VFG1_TRIGGER+	LVDS
42	In	VFG1_TRIGGER-	LVDS
43	In	VFG1_TRIGGER_TTL	TTL
44	In	VFG1_ENCODER+	LVDS
45	In	VFG1_ENCODER-	LVDS
46	In	VFG1_ENCODER_TTL	TTL
47	In	Reserved	
48	In	Reserved	
49	In	Reserved	
50	In	VFG0_ENCODER_B+	LVDS
51	In	VFG0_ENCODER_B-	LVDS
52	In	VFG0_ENCODER_B_TTL	TTL
53	In	Reserved	
54	In	Reserved	
55	In	Reserved	
56	In	VFG1_ENCODER_B+	LVDS
57	In	VFG1_ENCODER_B-	LVDS
58	In	VFG1_ENCODER_B_TTL	TTL
59		Reserved	
60	Out	VFG1_GPOUT3_TTL	TTL

### Table 12-7 I/O Connector for the KBN-PCE-2-D/4-F

## 12.8 I/O Connector Pinout for the KBN-PCE-2-F

The pin-out for the I/O Connector (P11) for the Karbon model KBN-PCE-2-F is illustrated in the Table 12-8.

Note: Signal names start with the Virtual Frame Grabber (VFG) that they are routed to. For example, the signal VFG0\_TRIGGER\_TTL is wired to VFG0., while VFG2\_TRIGGER\_TTL is wired to VGF2.

Pin	I/O	Signal	Comment
1	In	VFG0_TRIGGER+	LVDS
2	In	VFG0_TRIGGER-	LVDS
3	In	VFG0_TRIGGER_TTL	TTL
4	In	VFG0_TRIGGER_OPTO_K	Cathode of optocoupling sensor
5	In	VFG0_TRIGGER_OPTO_A	Anode of optocoupling sensor
6		GND	
7	In	VFG0_ENCODER+	LVDS
8	In	VFG0_ENCODER-	LVDS
9	In	VFG0_ENCODER_TTL	TTL
10	In	VFG0_ENCODER_OPTO_K	Cathode of optocoupling sensor
11	In	VFG0_ENCODER_OPTO_A	Anode of optocoupling sensor
12		GND	
13	In	VFG0_GPIN0_TTL	TTL
14	In	VFG0_GPIN1_TTL	TTL
15	In	VFG0_GPIN2+	LVDS
16	In	VFG0_GPIN2-	LVDS
17	In	VFG0_GPIN3+	LVDS
18	In	VFG0_GPIN3-	LVDS
19	In	VFG0_GPIN4+	LVDS
20	In	VFG0_GPIN4-	LVDS
21		GND	
22	Out	VFG0_GPOUT0+	LVDS
23	Out	VFG0_GPOUT0-	LVDS
24	Out	VFG0_GPOUT1+	LVDS
25	Out	VFG0_GPOUT1-	LVDS
26	Out	VFG0_GPOUT2+	LVDS
27	Out	VFG0_GPOUT2-	LVDS
28	Out	VFG0_GPOUT3_TTL	TTL
29		GND	
30	Out	VFG0_GPOUT4_TTL	TTL
31	Out	VFG0_GPOUT5_OC	Open collector driver

Table 12-8 I/O Connector for the KBN-PCE-2-F

Pin	I/O	Signal	Comment
32	Out	VFG0_GPOUT5_VCC	Pull-up or power for the open col- lector driver
33	Out	VFG0_GPOUT6_OC	Open collector driver
34	Out	VFG0_GPOUT6_VCC	Pull-up or power for the open col- lector driver
35		GND	
36		GND	
37		Reserved	
38		GND	
39		Reserved	
40		GND	
41	In	Reserved	
42	In	Reserved	
43	In	Reserved	
44	In	VFG0_ENCODER_B+	LVDS
45	In	VFG0_ENCODER_B-	LVDS
46	In	VFG0_ENCODER_B_TTL	TTL
47	In	Reserved	
48	In	Reserved	
49	In	Reserved	
50	In	Reserved	
51	In	Reserved	
52	In	Reserved	
53	In	Reserved	
54	In	Reserved	
55	In	Reserved	
56	In	Reserved	
57	In	Reserved	
58	In	Reserved	
59		Reserved	
60	Out	Reserved	

### Table 12-8 I/O Connector for the KBN-PCE-2-F

# Index

### A

ABORT CON KBN-7-10 ACPL KBN-7-53 ACPL MUL KBN-7-51 ACQ\_CON KBN-7-11 ACQ IV KBN-7-113 ACQ SAFETY KBN-7-11 AFE\_PORT\_ACCESS KBN-7-107 AFE PORT ADDR KBN-7-107 AFE\_PORT\_BUSY KBN-7-109 AFE\_PORT\_DATA KBN-7-109 AFE PORT ERROR KBN-7-109 AFE\_PORT\_RESET KBN-7-109 AFE PORT WRITE KBN-7-107 AFPDF KBN-7-46 ALAST ADD KBN-7-57 ALPF KBN-7-76 AQ COUNT KBN-7-40 AQCMD KBN-7-22 AQSTAT KBN-7-22, KBN-8-15

### B

BAYER\_BIT\_DEPTH KBN-7-84 BITFIELDNAME KBN-7-2 BLAST\_ADD KBN-7-57 BLUE\_GAIN KBN-7-83 BUTTONS KBN-7-63

### C

CALC\_BANK KBN-7-50 Camera Link Controls (CCs) KBN-10-11 CC\_SYNC KBN-7-35 CC1\_CON KBN-7-18 CC2\_CON KBN-7-18 CC3\_CON KBN-7-19 CC4\_CON KBN-7-19 CFGCLOCK KBN-7-5 CFGDATA KBN-7-5 CFGDATA KBN-7-5 CFGEN KBN-7-5 CFGEN KBN-7-5 CFGSTATUS KBN-7-5 CFREQ KBN-7-6 CHAIN\_DATA\_SIZE\_HI KBN-8-9 CHAIN\_DATA\_SIZE\_LO KBN-8-7

CHAIN\_DATA\_TOGO\_HI KBN-8-13 CHAIN DATA TOGO LO KBN-8-11 CL DISABLE KBN-7-28 CLAST\_ADD KBN-7-59 CLIP KBN-7-50 CMDWRITE KBN-7-20 CON0 KBN-7-4 CON1 KBN-7-8 CON10 KBN-7-52 CON11 KBN-7-56 CON12 KBN-7-58 CON13 KBN-7-60 CON14 KBN-7-62 CON15 KBN-4-6, KBN-7-66 CON16 KBN-4-10, KBN-7-70 CON17 KBN-7-73 CON18 KBN-7-75 CON19 KBN-7-77 CON2 KBN-7-15 CON20 KBN-7-79 CON21 KBN-7-82 CON22 KBN-4-12, KBN-7-85 CON23 KBN-7-87 CON24 KBN-7-89 CON25 KBN-7-93 CON26 KBN-7-95 CON27 KBN-7-97 CON28 KBN-8-2 CON29 KBN-8-4 CON3 KBN-7-21 CON30 KBN-8-6 CON31 KBN-8-8 CON32 KBN-8-10 CON33 KBN-8-12 CON34 KBN-8-14 CON35 KBN-8-17 CON36 KBN-7-99 CON37 KBN-7-101 CON38 KBN-7-103 CON4 KBN-7-24 CON40 KBN-7-106 CON41 KBN-7-108 CON42 KBN-7-110 CON43 KBN-7-115 CON44 KBN-7-117 CON5 KBN-7-31

CON51 KBN-4-14, KBN-7-119 CON6 KBN-7-37 CON7 KBN-7-39 CON8 KBN-7-42 CON9 KBN-7-48 CPLD\_MODE KBN-7-7 CTAB\_INT\_CON KBN-7-88 CTABHOLD KBN-7-17

### D

DECODER\_OUTPUT KBN-7-83 **DECODER PHASE KBN-7-84** DELAY KBN-7-64 DELAY TAP1 KBN-7-94 DELAY TAP1 SEL KBN-7-94 **DISPLAY KBN-7-49** DLAST ADD KBN-7-59 DMA 64 BIT KBN-8-16 DMA\_AUTO\_START KBN-8-15 DMA BEN KBN-8-16 DMA\_BUSY KBN-7-29 DMA\_CHAINING KBN-8-16 DMA COMMAND KBN-8-16 DMA\_DIRECTION KBN-8-15 DMA\_DONE KBN-8-15 DMA INIT FUNC KBN-8-15 DMA\_NO\_RULE KBN-8-15 DMA PRIORITY KBN-8-16 DMA STATUS KBN-8-15 DPM SIZE KBN-7-88 DPM SPLIT KBN-7-65 DPM WP KBN-7-57 DWNLD\_MODE KBN-7-102

### E

EN\_ENCODER KBN-7-36 EN\_TRIGGER KBN-7-35 ENC\_DIV KBN-7-38 ENC\_DIV\_FCLK\_SEL KBN-7-72 ENC\_DIV\_FORCE\_DC KBN-7-71 ENC\_DIV\_M KBN-7-38 ENC\_DIV\_N KBN-7-78 ENC\_DIV\_OPEN\_LOOP KBN-7-72 Encoder KBN-10-4 Encoder Divider KBN-5-1 ENCPOL KBN-7-33 ENINT\_CTAB KBN-7-25 ENINT\_EOF KBN-7-34 ENINT\_HW KBN-7-25 ENINT\_OVSTEP KBN-7-25 ENINT\_QUAD KBN-7-26 ENINT\_SER KBN-7-26 ENINT\_TRIG KBN-7-25 EOF\_IN\_AQ KBN-7-26

### F

FACTIVE KBN-7-22 FCOUNT KBN-7-22 FEN\_SEL KBN-7-113, KBN-7-114 FENCOUNT KBN-7-29 FENPOL KBN-7-63 FI KBN-7-111 FI POL KBN-7-112, KBN-7-114 FIFO\_EQS KBN-7-80 FIRST FI KBN-7-111 FIRST QUAD PTR HI KBN-8-5 FIRST\_QUAD\_PTR\_LO KBN-8-3 FLASH ADDR KBN-7-98 FLASH BE KBN-7-98 FLASH\_CE KBN-7-98 FLASH DATA KBN-7-86 FLASH\_OE KBN-7-98 FLASH\_RST KBN-7-98 FLASH WE KBN-7-98 FLASH\_WP KBN-7-98 FORCE 8BIT KBN-7-55 FORMAT KBN-7-53 FREEZE CON KBN-7-11 FW 7MHZ KBN-7-5 FW SEL KBN-7-7 FW\_TYPE KBN-7-49

### G

GEN\_H\_LOW KBN-7-116, KBN-7-118 GEN\_H\_PERIOD KBN-7-116 GEN\_ONESHOT KBN-7-40 GEN\_V\_PERIOD KBN-7-118 General Purpose Inputs (GPIN) KBN-10-6 General Purpose Outputs (GPOUT) KBN-10-7 GPIN0 KBN-7-23 GPIN1 KBN-7-23 GPIN2 KBN-7-23 GPIN3 KBN-7-23 GPIN4 KBN-7-23 GPOUT0 KBN-7-27 GPOUT0\_CON KBN-7-43 GPOUT1 KBN-7-27 GPOUT1\_CON KBN-7-43 GPOUT2 KBN-7-27 GPOUT2\_CON KBN-7-44 GPOUT3 KBN-7-27 GPOUT3\_CON KBN-7-44 GPOUT4 KBN-7-27 GPOUT4\_CON KBN-7-45 GPOUT5 KBN-7-27 GPOUT5\_CON KBN-7-45 GPOUT6 KBN-7-27 GPOUT6\_CON KBN-7-46 GREEN\_GAIN KBN-7-83

### H

HAW\_START KBN-7-30 HCNT\_LD KBN-7-16 HCNT\_RLS\_STK KBN-7-17 HCNT\_RLS\_ZERO KBN-7-16 HCNT\_RST KBN-7-16 HCOUNT KBN-7-38 HD\_SEL KBN-7-113 Horizontal Control Table Size KBN-2-19

# I

I/O Connector Pinout for the KBN-PCE-CL2-D/ 4-F KBN-12-11 I/O Connector Pinout for the KBN-PCE-CL2F KBN-12-13 I/O Connector Pinout for the KBN-PCE-CL4-D KBN-12-9 INT\_ANY KBN-7-26, KBN-7-27 INT\_CTAB KBN-7-26, KBN-7-27 INT\_CTAB KBN-7-12 INT\_EOF KBN-7-12 INT\_EOF KBN-7-12 INT\_OVSTEP KBN-7-12 INT\_OVSTEP KBN-7-13 INT\_SER KBN-7-13 INT\_SER KBN-7-13 INT\_TRIG KBN-7-14

# J

Jumpers KBN-12-5

### L

L\_CLKCON KBN-7-6 LAL KBN-7-38 LATCH\_CONTROL KBN-8-16 LCOUNT KBN-7-28 LENPOL KBN-7-63 LINES\_PER\_INT KBN-7-88 LINES\_TOGO KBN-7-78 LUT\_BANK KBN-7-91 LUT\_DATA\_WRITE\_SEL KBN-7-91 LUT\_HOST\_ACCESS KBN-7-92 LUT\_HOST\_ADDR KBN-7-90 LUT\_HOST\_DATA KBN-7-90 LUT\_HOST\_LANE KBN-7-91 LUT\_ON KBN-7-90 LUT\_WEN KBN-7-91

### Μ

MEM\_ADDR\_LO KBN-7-100, KBN-7-102 MEM\_CS KBN-7-102 MEM\_DATA KBN-7-102 MEM\_WRITE KBN-7-102 MUX\_REV KBN-7-49

### N

New Timing Generator KBN-3-1 NO\_RULE KBN-7-12 NO\_VB\_WAIT KBN-7-10 NTG KBN-3-1 NTG\_EXPOSURE KBN-7-96 NTG\_INVERT KBN-7-76 NTG\_ONESHOT KBN-7-74 NTG\_RATE KBN-7-74 NTG\_RESET KBN-7-96 NTG\_SLAVE KBN-7-96 NTG\_TIME\_MODE KBN-7-76 NTG\_TO\_ENC KBN-7-40 NTG\_TO\_TRIG KBN-7-41 NTG\_TRIG\_MODE KBN-7-74

### 0

OVS KBN-7-28

### Ρ

PCOUNT KBN-7-29 Pinouts (CL1 & CL2) KBN-12-8 PIX\_DEPTH KBN-7-55 POCL\_CLK\_DETECTED KBN-7-105 POCL\_CLOCK\_WAIT KBN-7-104 POCL\_DETECTED KBN-7-105 POCL\_EN KBN-7-5 POCL\_EN\_POWER KBN-7-104 POCL\_GND\_ON KBN-7-104 POCL\_SENSE KBN-7-104 POP\_TOSS KBN-7-29 PUMP\_OFF KBN-7-29

### Q

QENC AQ DIR KBN-4-7, KBN-7-67 QENC\_COUNT KBN-4-15, KBN-7-120 QENC\_DECODE KBN-4-7, KBN-7-67 QENC\_DIR KBN-4-15, KBN-7-120 QENC\_DUAL\_PHASE KBN-4-8, KBN-7-68 QENC INTRVL IN KBN-4-15, KBN-7-120 QENC INTRVL LL KBN-4-7, KBN-7-67 QENC\_INTRVL\_MODE KBN-4-7, KBN-7-67 QENC INTRVL UL KBN-4-11, KBN-7-71 QENC NEW LINES KBN-4-16, KBN-7-121 QENC\_NO\_REAQ KBN-4-7, KBN-7-67 QENC PHASEA KBN-4-15, KBN-7-120 QENC\_PHASEB KBN-4-15, KBN-7-120 QENC\_REAQ\_MODE KBN-4-11, KBN-7-71 QENC RESET KBN-4-9, KBN-7-69 QENC\_RESET\_REAQ\_KBN-4-11, KBN-7-71 QTAB KBN-8-19 QTBSRC KBN-7-20 Quad Table KBN-8-19

### R

**R/W KBN-7-3** RD ENC\_DIFF KBN-7-34, KBN-7-35 RD ENC OPTO KBN-7-34 RD ENC TTL KBN-7-34 RD\_HD KBN-7-111 RD\_TRIG\_DIFF KBN-7-33 RD\_TRIG\_OPTO KBN-7-34 RD\_TRIG\_TTL KBN-7-33 RD\_VD KBN-7-112 RD\_WEN KBN-7-111 **REG GAIN KBN-7-83 RELOAD FPGA KBN-7-7** REV\_DCC KBN-7-22 RLE\_LOAD\_H KBN-7-46 RLE\_LOAD\_V KBN-7-47 RO KBN-7-3 RST CALC BANK KBN-7-50 RST DPM ADDR KBN-7-17 RST\_HVCOUNT KBN-7-17

RST\_OVS KBN-7-28 RST\_SER KBN-7-28

## S

SCAN STEP KBN-4-13, KBN-7-86 SCAN STEP TRIG KBN-4-8, KBN-7-68 SEL\_REG\_GEN KBN-7-40 SEL TRIG KBN-7-32 SEL\_UCLK\_7MHZ KBN-7-6 SELENC KBN-7-33 SHIFT\_DISP KBN-7-80 SHIFT\_DSP\_LEFT KBN-7-81 SHIFT\_DSP\_SELECT KBN-7-80 SHIFT RAW KBN-7-63 SHIFT RAW LEFT KBN-7-64 SHORT\_FRAME KBN-7-50 SOE KBN-7-112 Specifications KBN-11-1 SW KBN-7-23 SW ENC KBN-7-33 SW RESET KBN-7-63 SW\_TRIG KBN-7-32 SWAP KBN-7-64 SWAP\_LINES KBN-7-112 Switches KBN-12-7

# T

TAG\_BANK KBN-7-40 TOP\_REV KBN-7-76 TRIG\_QUALIFIED KBN-7-38 Trigger KBN-10-2 TRIGGER\_DELAY KBN-7-34 TRIGPOL KBN-7-32 TRIM KBN-7-49

### U

UART\_CON KBN-7-65 UART\_MASTER KBN-7-57

# V

VAW\_START KBN-7-30 VCNT\_LD KBN-7-10 VCNT\_RLS\_STK KBN-7-10 VCNT\_RLS\_ZERO KBN-7-9 VCNT\_RST KBN-7-9 VCOUNT KBN-7-38 Vertical Control Table Size KBN-2-14 VFG KBN-1-1 VID\_BRL KBN-7-80 VID\_SOURCE KBN-7-54 VIDEO\_2DPM KBN-7-80 VIDEO\_MASK KBN-7-61

### W

WO KBN-7-3

### Х

XFR\_PER\_INT KBN-8-18

Index