

Camera Configuration Guide

for the PCI DV Family



June 29, 2009
008-01993-03



a HEICO company

Camera Configuration Guide

EDT™ and Engineering Design Team™ are trademarks of Engineering Design Team, Inc. All other trademarks, service marks, and copyrights are the property of their respective owners†.

© 1997-2009 Engineering Design Team, Inc. All rights reserved.

Terms of Use Agreement

Definitions. This agreement, between Engineering Design Team, Inc. (“Seller”) and the user or distributor (“Buyer”), covers the use and distribution of the following items provided by Seller: a) the binary and all provided source code for any and all device drivers, software libraries, utilities, and example applications (collectively, “Software”); b) the binary and all provided source code for any and all configurable or programmable devices (collectively, “Firmware”); and c) the computer boards and all other physical components (collectively, “Hardware”). Software, Firmware, and Hardware are collectively referred to as “Products.” This agreement also covers Seller’s published Limited Warranty (“Warranty”) and all other published manuals and product information in all forms, both physical and electronic (“Documentation”).

License. Seller grants Buyer the right to use or distribute Seller’s Software and Firmware Products solely to enable Seller’s Hardware Products. Seller’s Software and Firmware must be used on the same computer as Seller’s Hardware. Seller’s Products and Documentation are furnished under, and may be used only in accordance with, the terms of this agreement. By using or distributing Seller’s Products and Documentation, Buyer agrees to the terms of this agreement, as well as any additional agreements (such as a nondisclosure agreement) between Buyer and Seller.

Export Restrictions. Buyer will not permit Seller’s Software, Firmware, or Hardware to be sent to, or used in, any other country except in compliance with applicable U.S. laws and regulations. For clarification or advice on such laws and regulations, Buyer should contact: U.S. Department of Commerce, Export Division, Washington, D.C., 20230, U.S.A.

Limited Rights. Seller grants Buyer a royalty-free right to modify, reproduce, and distribute executable files using the Seller’s Software and Firmware, provided that: a) the source code and executable files will be used only with Seller’s Hardware; b) Buyer agrees to indemnify, hold harmless, and defend Seller from and against any claims or lawsuits, including attorneys’ fees, that arise or result from the use or distribution of Buyer’s products containing Seller’s Products. Seller’s Hardware may not be copied or recreated in any form or by any means without Seller’s express written consent.

No Liability for Consequential Damages. In no event will Seller, its directors, officers, employees, or agents be liable to Buyer for any consequential, incidental, or indirect damages (including damages for business interruptions, loss of business profits or information, and the like) arising out of the use or inability to use the Products, even if Seller has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to Buyer. Seller’s liability to Buyer for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, product liability, tort including negligence, or otherwise) will be limited to fifty U.S. dollars (\$50.00).

Limited Hardware Warranty. Seller warrants that the Hardware it manufactures and sells shall be free from defects in material and workmanship for a period of 12 months from date of shipment to the initial Buyer. This warranty does not apply to any products which are misused, abused, repaired, or otherwise modified by Buyer or others. Seller’s sole obligation for breach of this warranty shall be to repair or replace (F.O.B. Sellers Plant, Beaverton, Oregon) any goods which are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. All installation and transportation expenses, and all other incidental expenses and damages, shall be borne by the Buyer. *This warranty is expressly in lieu of all other warranties, express or implied, including but not limited to any warranties of merchantability or fitness for any particular purpose.*

Limitation of Liability. *In no event shall seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise out of or are a result of breach of contract, warranty, tort (including negligence), strict liability, or otherwise.* All references to damages herein shall include, but not be limited to, loss of profit or revenues, loss of use of the goods or associated equipment, costs of substitute goods, equipment, facilities, downtime costs, or claims for damages. Seller shall not be liable for any loss, claim, expense, or damage caused by, contributed to, or arising out of the acts or omissions of Buyer, whether negligent or otherwise.

No Other Warranties. Seller makes no other warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding Seller’s Products or Documentation. Seller does not warrant, guarantee, or make any representations regarding the use or the results of the use of the Products or Documentation or their correctness, accuracy, reliability, currentness, or otherwise. All risk related to the results and performance of the Products and Documentation is assumed by Buyer. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to Buyer.

Disclaimer. Seller’s Products and Documentation, including this document, are subject to change without notice. Documentation does not represent a commitment from Seller.

Contents

Camera Configuration Guide 1	
Related Documents	2
Creating a Configuration File	3
initcam.....	3
Required Directives.....	4
width, height, and depth.....	4
Clipping and Padding.....	5
Data Format	5
Deinterleaving and Other Data Rearranging Schemes.....	5
Mode Control Lines.....	6
Decoding Bayer-filtered Images.....	6
Headers and Footers	6
Utilities	8
The checkcam Utility.....	8
The countbits Utility.....	9
Configuration File Directives.....	12
aperture_min 12	
aperture_max 12	
byteswap 12	
camera_class 12	
camera_command_file 12	
camera_download_file 12	
camera_info 13	
camera_model 13	
cameralink 13	
cameratest 13	
cameratype 13	
CL_CFG_NORM 14	
CL_CFG2_NORM 14	
CL_DATA_PATH_NORM 15	
continuous 15	
dbl_trig 15	
default_gain 15	
default_offset 15	
default_shutter_speed 16	
depth 16	
DIRECTION 16	
DIS_SHUTTER 16	
disable_mdout 16	
DOUBLE_RATE 16	
DUAL_CHANNEL 16	

ENABLE_DALSA 16
extdepth 17
fieldid_trig 17
foi_init 17
foi_rbtfiler 17
force_single 17
frame_delay 17
frame_height 17
frame_period 18
fv_once 18
fval_done 18
gain_min 18
gain_max 18
genericsim 19
hactv 19
header_dma 19
header_size 19
height 19
hskip 20
htaps 20
hwpad 20
image_offset 21
interlace 21
INV_SHUTTER 21
irris_strip 21
kbs_red_row_first 21
kbs_green_pixel_first 21
line_delay 21
markbin 22
markras 22
markrx 22
markry 22
mask 23
mc4 23
method_camera_continuous 23
method_camera_download 23
method_camera_shutter_timing 23
method_flushdma 23
method_frame_timing 23
method_header_position 24
method_interlace 24
method_lock_shutter 25
method_serial_format 25
method_serial_mode 25
method_set_gain 25
method_set_offset 25
method_shutter_speed 26
method_startdma 26
MODE_CNTL_NORM 26
offset_min 26
offset_max 26
pause_for_serial 26

pclock_speed 27
photo_trig 27
pulnix 27
rbtfile 27
rgb30 28
sel_mc4 28
serial_aperture 28
serial_baud 28
serial_binit 28
serial_binning 28
serial_exposure 29
serial_gain 29
serial_init 29
serial_init_baslerf 29
serial_init_duncanf 29
serial_init_hex 29
serial_offset 30
serial_response 30
serial_term 30
serial_timeout 30
serial_trigger 30
serial_waitc 30
shift 31
shortswap 31
shutter_speed_frontp 31
shutter_speed_min 31
shutter_speed_max 31
sim_height 31
sim_width 32
simulator_speed 32
skip 32
slop 32
timeout_multiplier 32
TRIG_PULSE 32
user_timeout 32
vactv 33
variable_size 33
vskip 33
vtaps 33
width 33
xregwrite_xx 34

Camera Configuration Guide

Before your EDT digital video board can recognize and access a camera, registers on the board and variables in the driver must be initialized to values specific to a particular camera and operating mode. The most straightforward way to do so is using a camera configuration file and the EDT `initcam` utility supplied with the PCI DV software package.

You invoke `initcam` with an argument specifying a camera configuration file; the utility then uses the values specified in that configuration file to initialize the board as required to work with a given camera model. A camera configuration file is an ASCII text file that includes directives relevant to a particular camera model. A great many directives are defined, but most cameras require only a small subset of them in order to work as required with an EDT digital video board.

EDT's PCI DV software package includes many predefined camera configuration files for use with the most common cameras in their most common operating modes.

Each supported camera has at least one configuration file, and sometimes several — different files can be used to initialize a camera in different modes. For example, one configuration may set the camera to standard full-frame mode, while another may have settings for the same camera in 2x2 binned mode. These files reside in the `camera_config` subdirectory of the distribution directory.

The `camera_config` subdirectory contains configuration files for all cameras that we've tested. This manual is intended for people who wish to create configuration files for cameras not currently supported, or to run in camera modes for which no configuration file has yet been made.

If no predefined camera configuration file suits your needs, you can modify an existing one or create a new one to do what you need. This manual describes all the camera configuration directives available to configure an EDT digital video board to work with a digital camera.

`initcam` reads the configuration file (using the EDT Digital Library subroutine `pdv_readcfg`) and then calls `pdv_initcam` to set the registers and variables based on the settings in the file. `pdv_initcam` can also send commands to the camera to put it into a known state.

EDT's digital video boards can be initialized using tools other than `initcam`; for example, when you first run `PdvShow` (or another EDT capture and display application), a dialog offers a choice of camera configurations. Based on your choice, `PdvShow` creates a script that runs `initcam`, invoking the appropriate configuration file. Your application can also initialize the board using library calls to the digital video subroutines `pdv_readcfg` and `pdv_initcam`.

Related Documents

The related publications below may prove useful.

Document	URL
EDT DMA & Digital Video Software Library	www.edt.com/api (HTML)
EDT DMA & Digital Video Software Library	www.edt.com/manuals/misc/api.pdf (PDF)
PCI DV Family User's Guide	www.edt.com/manuals/PDV/pcidv.pdf
PCI DV Camera Link Firmware Reference	www.edt.com/manuals/PDV/pcidv_cl_add.pdf
PCI DV AIA Hardware Addendum	www.edt.com/manuals/PDV/aiag_add_dv2.pdf
Camera Link Specification, online at:	www.machinevisiononline.org
PCI DV Camera Link Firmware Reference	www.edt.com/manuals/PDV/dv-clink.pdf

Creating a Configuration File

Configuration files are ASCII text files bearing the `.cfg` file extension, that can be edited with most standard text editors.

Each line is either a comment or a directive. Comments begin each line with the `#` character. Otherwise, each line in the configuration file consists of a `directive: value` pair.

NOTE Avoid editing configuration files with Windows Notepad, which cannot edit files without CR/LF line terminators.

The simplest way to create a new configuration is to start with a file for a similar camera. If you know of a camera or operating mode that is similar to the one you need, copy the file and edit it as needed, saving it under a new name. Otherwise, start with one of the example files — `generic8cl.cfg`, `generic10cl.cfg`, `generic12cl.cfg`, `generic24cl.cfg` for Camera Link boards and cameras, `generic8.cfg`, `generic10.cfg`, and `generic12.cfg` for non-Camera Link boards and cameras — and edit it as necessary to accomplish your task.

To create a configuration file:

1. Create a new file, usually by copying an existing file to one with a new name.
2. Include or modify the camera ID string directives `camera_class`, `camera_model`, and `camera_info`, so a camera chooser application (such as `PdvShow`) can present a unique and informative choice for this new camera or operating mode. These directives are typically used thus:

<code>camera_class</code>	Ordinarily, the camera manufacturer; for example, "Redlake."
<code>camera_model</code>	Ordinarily, the camera model; for example, "1310C."
<code>camera_info</code>	Ordinarily, a specific operating mode; for example, "10-bit monochrome."
3. Modify the image size and depth directives to match the exact image size, lines per frame, and bits per pixels output by the camera.
4. Modify the timing and data order directives to match the camera's output format: for Camera Link, these are `CL_DATA_PATH_NORM` and `CL_CFG_NORM`; on other boards these are some or all of `shift`, `mask`, `byteswap`, `shortswap`, `DUAL_CHANNEL` and `DOUBLE_RATE`.
5. If necessary, add the data reordering directive `method_interlace`.
6. If necessary, add Region of Interest or hardware padding directives to clip or pad the pixels per line to a four-byte boundary, and optionally clip off invalid data on the borders.
7. If you need board-controlled trigger or exposure timing, set the `MODE_CNTL_NORM` and `method_camera_shutter_timing` directives to put the board into the desired mode.
8. Optionally add a serial initialization directive (`serial_init` or a related directive) to put the camera into the desired state or mode on start-up.
9. Make other changes as needed, from the list of directives in [The checkcam Utility on page 8](#).

initcam

When you select a camera configuration (through `PdvShow`'s Camera Setup dialog, for example), the driver creates the file `camsetup0_0.bat` (on Windows) or `pdvload` (on UNIX-based systems). This file is a script that runs `initcam` with the chosen configuration file as an argument:

```
initcam -f camera_config/file.cfg
```

When creating or modifying a camera configuration file, you must run `initcam` with your new camera configuration file before it takes effect (`PdvShow` does not run `initcam` every time it is started). Then use your own application (or one of the provided GUI capture or display applications) to check your results.

Required Directives

This section describes the required directives. For descriptions of all directives, see [The checkcam Utility on page 8](#).

Most directives are optional. The following are required for all cameras. Most cameras require additional directives for proper operation; those listed here are merely those that apply in all cases.

For all EDT boards:

```
camera_class
camera_model
camera_info
width
height
depth
extdepth
```

For EDT's Camera Link-compliant frame-grabbers:

```
CL_CFG_NORM
CL_DATA_PATH_NORM
```

For the PCI DV, PCI DVK, PCI DVa, and PCI DV-FOX:

```
rbtfile
```

width, height, and depth

Initially, set both `depth` and `extdepth` to the number of bits per pixel output by the camera, unless:

- the camera is a color camera, or
- you wish to pass only eight bits from a 10- to 16-bit monochrome camera.

If you are using a color RGB or Bayer-filtered camera, set `depth` to 24 and `extdepth` to the number of bits per red, green, or blue element (usually 8, 10 or 12). If you wish to pass through only eight bits from a 10- to 16-bit monochrome camera, set `depth` to 8 and `extdepth` to the number of bits per pixel from the camera.

`width` and `height` are self-explanatory, but finding the correct settings is not always straightforward. Some cameras output a slightly different number of lines per image, and pixels per line, than the documentation states (because the active pixel area of the CCD is sometimes slightly greater than the published specification). If the camera documentation is not accurate, it can take some trial-and-error to find the right values for width and height. Diagonal skew usually indicates a problem with width. Timeouts and overruns (reported at the bottom of the pane from `PdvShow`, or as a `timeouts: or overruns:` line from the `take` application) indicates a problem with height or possibly width.

To find out what a free-running camera is outputting to PCI DV and PCI DVK boards, you can use `checkcam`, described in [The checkcam Utility on page 8](#).

Clipping and Padding

Region of interest directives can be used to clip off unwanted borders in the incoming image. Doing so is usually optional; however, if the incoming image data width is not four-byte-aligned, it causes display problems when using the Windows MFC display library (which `pdvShow` and other display programs use), because the Windows MFC display library is optimized for images aligned on four-byte boundaries. Hence, these display applications do not work correctly unless the number of pixels per line is a multiple of four (for 8-bit cameras) or two (for 10- through 16-bit cameras).

If you can't get the data to line up vertically regardless of the width value you use, the data probably needs to be clipped or padded to align on four-byte boundaries. The best solution is to use hardware Region of interest directives—in particular, `hskip` and `hactv`—to clip the data to a four-byte aligned size. Region of interest directives can also be used to clip the black inactive borders that many cameras send. The `hskip`, `hactv`, `vskip`, and `vactv` directives activate PCI DV hardware-level clipping, which eliminates the performance penalties that result from software manipulations. For example, suppose a camera with 1024 x 1024 active pixels also sends extra inactive pixel data before and after each line or frame, yielding a 1038 x 1038 image. Assuming equal borders of invalid data all around, the following directives clip the image to 1024 x 1024, both clipping off the active pixels and yielding a four-byte aligned image:

```
hskip: 7
vskip: 7
hactv: 1024
vactv: 1024
```

(This would not work with some data reordering methods, because reordering assumes sequential data.)

The `hskip`, `hactv`, `vskip` and `vactv` directives are described in detail in [Configuration File Directives](#).

Data Format

For Camera Link boards and cameras, the `CL_CFG_NORM` directive controls the expected number of taps and bits per pixel. On non-Camera Link boards and cameras, the `shift`, `mask`, `byteswap`, `shortswap`, `DOUBLE_RATE` and `DUAL_CHANNEL` directives control how the data is manipulated before being sent to system memory using DMA. If these directives are not present in the configuration file, `initcam` defaults to values that match a typical AIA-swapped camera such as the Redlake MEGAPLUS series. If `byteswap` is not set, `initcam` sets it to 0 on little-endian systems (such as Intel), or 1 on big-endian systems (such as Sun).

It is best to start by leaving these directives out of the configuration file, and add them only if the data is obviously incorrect. Grainy, noisy or banded images are usually a sign that the configuration file needs the `shift` or `mask` directives (or that `CL_DATRA_PATH_NORM` is wrong for Camera Link systems).

For such cases, the `countbits` utility can be useful. It counts the number of bit toggles for each bit in a raw image and outputs the result. For more information, see [The countbits Utility on page 9](#).

Deinterleaving and Other Data Rearranging Schemes

Some cameras need the data to be reordered before it can be displayed. The Digital Video Library provides several reordering algorithms. A `method_interlace` directive in the configuration file causes the image acquisition subroutines (for example, `pdv_wait_image`) to call the specified algorithm before handing the image to a display routine. For example, a Redlake MEGAPLUS 8-bit camera running in dual-channel mode sends the data in pairs of pixels, one from an odd line and one from an even line. The deinterleave method for this is called `BYTE_INTLV`, and the directive to enable reordering is:

```
method_interlace: BYTE_INTLV
```

See [method_interlace](#) for a description of the available reordering methods.

Mode Control Lines

Most cameras power up in free-run mode. In such cases, the board gets the next image on an acquire request. If, however, you need board-camera triggering or board-controlled exposure timing, use the Mode Control (CC) lines.

A few cameras also use mode control lines to control such things as gain, black level and binning, employing a variety of schemes. If you have a triggered camera and a cable that is wired properly, setting the `MODE_CNTL_NORM` directive to 10 is often all that is needed to enable triggering from the board. For other schemes, read your camera manufacturer's manual and see [MODE_CNTL_NORM](#).

Decoding Bayer-filtered Images

To enable Bayer decoding in the library, to produce RGB image data from Bayer-filtered image data:

1. Copy a working monochrome camera configuration file and save it under a new name:
2. Change the `camera_info:` directive to include information specifying that this configuration file decodes a Bayer-filtered image.
3. Change `depth` to 24. (Do not change `extdepth`.)
4. Add the following directives:

```
method_interlace: BGGR #for 8-bit data
method_interlace: BGGR_WORD #for 10- to 16-bit data
kbs_red_row_first: 0
kbs_green_pixel_first: 0
```

5. Try configuring the board with the new configuration file and acquiring an image with the camera.

If you have not set the values for `kbs_red_row_first` and `kbs_green_pixel_first` to match the sensors' filter layout, the colors will look wrong. In this case, try different combinations of values for those directives until the colors look more or less right, or at least close. This indicates that the camera configuration matches the camera sensor. However, the colors will almost certainly not look precisely correct until you have set the white balance.

6. To set the white balance, invoke `pdvshow`.
7. Click on the color wheel toolbar icon.
8. Place something white, such as a white piece of paper, in front of the lens.
9. Click **compute white balance**.

The image should now appear correct.

For information on enabling and controlling Bayer decoding and white balance from your application, see `pdv_set_full_bayer_parameters` in the [EDT DMA & Digital Video Software Library](#).

Headers and Footers

In rare cases, the camera outputs an extra field of header or footer data before or after the image data. If this data is a multiple of the line width, increase the height to include the extra data, then clip it off if you wish. But if header data is not a multiple of the line width (an unusual situation), then the count (from `checkcam`) will not be a multiple of (lines per frame) times (pixels per line).

If the extra data comes at the end of the frame, it probably won't cause a problem; but if it is at the beginning of the frame, it will probably evidence itself as an image that is uniformly shifted to the right. In this case, add header directives so the PCI DV reads the extra data and stores it in a header before going on to the frame data itself. To do so:

1. Determine how much extra data must be stored as header, by computing the remainder of:

```
count / bytes_per_frame
```

where `count` is the output from `checkcam` and `bytes_per_frame` is defined as::

$$\frac{\text{lines}}{\text{frame}} \times \frac{\text{pixels}}{\text{line}} \times \frac{\text{bytes}}{\text{pixel}}$$

2. Use the result as the value of the `header_size` argument. For example:

```
method_header_position: HEADER_WITHIN
header_size: 398
header_dma: 1
```

See [Configuration File Directives on page 12](#) for more information on header and footer directives.

Utilities

To help you determine whether your equipment is configured and operating correctly, the distribution includes two utility applications:

`checkcam` A utility to determine the number of pixels per line and lines per frame output from a free-running camera.

`countbits` A utility to determine whether image data is properly aligned.

These are explained below.

The checkcam Utility

`checkcam` is a utility you can use to determine the number of pixels per line and lines per frame output from a free-running camera.

NOTE `checkcam` runs only on the PCI DV and PCI DVK boards.

To use `checkcam`:

1. Put the camera into continuous capture mode.

Many cameras run in continuous capture mode by default, in which case you can skip to step 2. Other cameras use a serial protocol to switch between capture modes. If you have such a camera, use the `serial_cmd` utility to put the camera into continuous mode. Before doing so:

- a. run a configuration file for a similar camera (or one of the generic files) to initialize the board and driver.
- b. Use `serial_cmd` to send the command. For example, for a Redlake MASD camera:

```
serial_cmd "MDE CS"
```

Other cameras use other special commands or mode lines — the methods for doing so are many; consult your camera documentation.

2. Enter the following at the command line:

```
initcam -f camera_config/cameratest.cfg
checkcam
```

Repeating lines such as the following appear:

```
waiting for frame valid
waiting for not frame valid
frame 1 count 614400 bad 0 lines/frame 480 pixels/line 640 dtime 0.084
frame 2 count 614400 bad 0 lines/frame 480 pixels/line 640 dtime 0.084
frame 3 count 614400 bad 0 lines/frame 480 pixels/line 640 dtime 0.084
. . . . .
. . . . .
. . . . .
```

3. Break out of this by pressing your operating system interrupt.
4. Replace the `width` and `height` values in your camera configuration file with the pixels per line and lines per frame values, respectively.

If you don't get results similar to those shown above, then the camera is not in continuous output mode or there is a signal problem. Make sure the camera is outputting good Frame Valid, Line Valid and Pixel clock signals, and make sure the cable pinout is correct. Refer to the AIA specification, and check the signal levels.

If the camera cannot be put into continuous mode (or be triggered while running the `checkcam` utility), use the trial and error method:

1. Substitute the `width` and `height` arguments in your test camera configuration file with the published pixels per line and lines per frame for the camera.
2. Run an application program such as `PdvShow`. Is the image is lined up vertically?
3. If not, repeat the procedure, adjusting the `width` argument until you see vertically aligned data.

If you can't get a value that gives you vertically aligned data, check that the number of taps is set properly for your camera.

After finding the correct `width`, adjust the `height` value using the same procedure until you have a value that is one less than that which results in a timeout from, for example, the `take` application.

The countbits Utility

`countbits` is a utility to help you determine whether image data is properly aligned. It takes a raw image file and counts the number of times each bit in a pixel changes value, compared to the value of the corresponding bit in the previous pixel. It then outputs the results of this count.

A properly aligned image ordinarily exhibits a descending number of changes in value from the least to the most significant bits of adjacent pixels. That is, the least significant bits of pixels are ordinarily the most volatile, changing values quite often to reflect small fluctuations of luminance, or color, as compared with adjacent pixels. These small fluctuations are often the result of noise in the sensor or tiny changes in the way light falls on areas of the subject. However, changes to more significant bits generally indicate larger changes in luminance or color — not a result of random noise in the sensor or the lighting, but rather changes in actual luminance or color value of the image subject. Such changes will therefore tend to occur less often. An image with properly aligned pixel data therefore shows a great many value changes in less significant bits and relatively fewer changes in more significant bits, forming a gradient of value changes, from many changes in the least significant bit to few changes in the most.

Results other than this descending gradient generally indicate a problem: the cable might be wired incorrectly or not wired for straight-through data, or the camera configuration might be in error in some way — an incorrect `CL_CFG_NORM` setting (for Camera Link cameras) or `shift`, `mask`, `byteswap` or `shortswap` settings (for non-Camera Link cameras). In this case, viewing the image in an application such as `PdvShow` displays an image that appears snowy or obviously wrong.

To use `countbits`:

1. Run `take` to acquire and save a raw image

```
take -f file.raw
```

2. To run `countbits` on a monochrome 8-bit image:

```
countbits file.raw
```

For 10- to 16-bit monochrome images, use the `-w` option:

```
countbits -w file.raw
```

For RGB images with eight bits each of red, green, and blue information per pixel, use the `-c` option:

```
countbits -c file.raw
```

For monochrome 8-bit images, a properly aligned image yields results similar to these:

```
bit 00: 1061090
bit 01: 569471
bit 02: 247656
bit 03: 156286
bit 04: 71844
bit 05: 44572
bit 06: 22244
bit 07: 15588
```

For 10- to 16-bit monochrome images, a properly aligned image yields results similar to these:

```
bit 00: 248681
bit 01: 248573
bit 02: 248453
bit 03: 244508
bit 04: 230606
bit 05: 183557
bit 06: 121598
bit 07: 71613
bit 08: 40028
bit 09: 17169
bit 10: 0
bit 11: 0
bit 12: 0
bit 13: 0
bit 14: 0
bit 15: 0
```

For 24-bit RGB images, a properly aligned image yields results similar to these, showing three different gradients, one each for red, green, and blue:

```
Blue:
bit 00: 68375
bit 01: 39681
bit 02: 23249
bit 03: 14904
bit 04: 4868
bit 05: 2299
bit 06: 1212
bit 07: 270
```

Green:

bit 00: 104357
bit 01: 104938
bit 02: 62374
bit 03: 37419
bit 04: 15010
bit 05: 5062
bit 06: 2481
bit 07: 1148

Red:

bit 00: 73512
bit 01: 44331
bit 02: 26619
bit 03: 14443
bit 04: 4966
bit 05: 2877
bit 06: 1478
bit 07: 340

Configuration File Directives

This section lists all the configuration file directives. Most apply to only one or a few cameras; most cameras need fewer than ten percent to set up properly.

aperture_min

Minimum allowable aperture setting for this camera model. Applies only to cameras with serial aperture control. For example:

```
aperture_min: 0
```

Applies only when [serial_aperture](#) is set.

aperture_max

Maximum allowable aperture setting for this camera model. Applies only to cameras with serial aperture control. For example:

```
aperture_max: 18
```

Applies only when [serial_aperture](#) is set.

byteswap

Non-Camera Link products only. Sets the BSWAP bit in the Utility2 register; tells the PCI DV whether or not to swap data bytes when transferring the image to the host's memory. A value of 1 swaps bytes; a value of 0 leaves them as is.

By default, `initcam` checks whether the host is little-endian (such as a Sun computer) or big-endian (such as an Intel-based computer) and sets `byteswap` accordingly. However, certain combinations of camera and platform require the use of this directive to override this default behavior. For example:

```
byteswap: 1
```

camera_class

Required. Argument is a double-quoted string that describes the camera manufacturer. Used with `camera_model` and `camera_info`, which together determine the text displayed in the camera configuration selection dialog to describe a configuration choice. When used together, these three directives replace the obsolete `cameratyp` directive. For example:

```
camera_class: "Redlake"
```

See also `camera_info`, `camera_model`.

camera_command_file

Specifies the name of a file from which to retrieve set-up commands to send to a camera. See `camera_config/si*.txt` files for command format and example use. For example:

```
camera_command_file: camera_config/si600-142_0.txt
```

See also [method_camera_download](#).

camera_download_file

Specifies the name of a file from which to retrieve binary camera download data to send to a camera. Binary download files are created by the camera manufacturer and are camera-specific. Download

files for some Spectral Instruments cameras are included in the package; however, check with the camera manufacturer to verify that you have a correct and up-to-date download file. For example:

```
method_camera_download: SPECINST_SERIAL
camera_download_file: camera_config/2350EDT.BIN
```

See also [method_camera_download](#).

camera_info

Required. Argument is a double-quoted string that describes the camera information. Specify at least the minimal information for the camera, model, and operating mode that is unique to this particular configuration file (for example, trigger mode, number of bits, or binning). With `camera_class` and `camera_model`, the information supplied determines the text displayed in the camera configuration selection dialog to describe a configuration choice. Used together, these three directives replace the obsolete `cameratyp` directive. For example:

```
camera_info: "10-bit (monochrome mode)"
```

See also [camera_class](#), [camera_model](#).

camera_model

Required. Argument is a double-quoted string that describes the camera model. With `camera_class` and `camera_info`, the information supplied determines the text displayed in the camera configuration selection dialog to describe a configuration choice. Used together, these three directives replace the obsolete `cameratyp` directive. For example:

```
camera_model: "1310C"
```

See also [camera_class](#), [camera_info](#).

cameralink

Obsolete. Has no effect.

cameratest

For diagnostic use only. Flag to tell `initcam` to abort immediately after loading the Xilinx FPGA. For example:

```
cameratest:1
```

cameratyp

Obsolete. Instead use `camera_class`, `camera_model` and `camera_info`.

`CL_CFG_NORM`

Required for Camera Link cameras and boards. Initial value for the PCI DV Camera Link Control register, specified as a two-digit hexadecimal value. For complete information on this and other registers, see the [PCI DV Camera Link Firmware Reference](#). Bits are defined as follows:

Bit	Hex Value	Description
7	80	Revision 36 or later of <code>pdrvcamlk</code> firmware: Enable region-of-interest padding. As of that revision, if the width or height of incoming data comes up short due to data loss, the region-of-interest logic no longer pads with extra bytes. Set this bit to re-enable the previous functionality. To enable region-of-interest for width only (useful for linescan cameras), set bit 3 (0x08) and bit 7 (0x80). NOTE Setting this bit on (or using older firmware) can mask timeouts because lost data is padded before the image reaches the device driver, resulting in a persistent out-of-synch condition. The application will then not be notified when it needs to perform timeout recovery.
6	40	Set to swap red (R) and blue (B) bytes in RGB triplets, so that they become BGR.
5	20	Set to invert the data-valid signal, for the few cameras that require this.
4	10	For linescan cameras; enables internal generation of Frame Valid after VACTV lines. Set bit 2 when using this feature.
3	08	Set to disable the region of interest counters, thus always acquiring the entire image. (The region of interest counters are set using the <code>hskip</code> , <code>hactv</code> , <code>vskip</code> , and <code>vactv</code> directives.)
2	04	Set to replace the frame-valid signal from the camera with a copy of the line-valid signal instead.
1	02	Set if the camera does not implement the data-valid output signal.
0	01	Set if the camera is 24-bit color: 8 bits each per red, green, and blue taps. If this bit is set, it overrides any setting of the <code>CL_DATA_PATH_NORM</code> directive.

For example:

```
CL_CFG_NORM: 02 #data valid invert bit set
```

`CL_CFG2_NORM`

Initial value for the PCI DV Camera Link Control 2 register, specified as a two-digit hexadecimal value. For complete information on this and other registers, see the [PCI DV Camera Link Firmware Reference](#). The only defined bit is bit 0 — set this bit to enable separate external triggers.

When clear (the default), an input trigger on the TRIG0 pin triggers both outgoing EXPOSE lines, one for each connector (channel). When set, an input trigger on the TRIG0 pin triggers the EXPOSE line on channel 0, and an input trigger on the TRIG1 pin triggers the EXPOSE line on channel 1.

For details on triggering, see the [PCI DV Family User's Guide](#).

For example:

```
CL_CFG2_NORM: 01 #enable separate trigger inputs and outputs
```

`CL_DATA_PATH_NORM`

Required with Camera Link boards and cameras. Initial value for the PCI DV C-Link Data Path register, specified as a two-digit hexadecimal value. Bits are defined as follows:

Bit	Definition
7-4	(number of bits per pixel) -1
3-0	(number of channels) -1

NOTE Cameras that output color-encoded pixels (typically 8 bits each of red, green, and blue per pixel), instead of Bayer-filtered output, are a special case, and the board instead uses the `CL_CFG_NORM` RGB bit to set the board into 24-bit RGB input mode. In that case, the `CL_DATA_PATH_NORM` directive is ignored; set it to 00 to indicate that it isn't being used. For more information on interface registers, see the [PCI DV Camera Link Firmware Reference](#).

For example:

```
CL_DATA_PATH_NORM: 19 # dual channel, 10 bits per pixel
```

`continuous`

By default, 0; however, certain cameras need this directive set to 1 when it has been determined that the interframe gap is too brief to wait for a frame-valid.

When 0, the DMA engine waits for a frame-valid signal to determine when to begin transferring pixels from the camera. When 1, it behaves in this way for the first frame only. For subsequent frames, the DMA engine counts pixels until it has transferred `width * height` pixels; then it begins the next frame on the next pixel without waiting for a frame-valid signal.

For example:

```
continuous:1
```

NOTE Set this directive with caution. We have set this value to 1 in configuration files for cameras that need it, in our experience — a small minority of cameras and operating modes. A spurious setting of one can cause you to lose data, resulting in misaligned frames until acquisition ends.

`dbl_trig`

An argument of 1 sets the PULNIX bit in the Utility 2 register, which enables Pulnix double-trigger mode. Necessary for board-triggering of certain Pulnix cameras. Not present in PCI DV C-Link. For more information on this bit or register, see the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
dbl_trig: 1
```

`default_gain`

Defines an initial gain value, which `pdv_open` then sets. Not generally useful; applies only to a subset of cameras for which this directive has been set in the camera's default configuration file, as supplied by EDT. For example:

```
default_gain: 6
```

`default_offset`

Defines an initial black level. If set, the first call to `pdv_open` then calls `pdv_set_blacklevel` with the specified value. Not generally useful; applies only to a subset of cameras for which this directive has been set in the camera's default configuration file, as supplied by EDT. For example:

```
default_offset: 100
```

`default_shutter_speed`

Defines an initial shutter speed, which `pdv_open` then sets. Not generally useful; applies only to a subset of cameras for which this directive has been set in the camera's default configuration file, as supplied by EDT. For example:

```
default_shutter_speed: 100
```

NOTE If this directive is not set in the configuration file, exposure time remains undefined until set by the application.

`depth`

Required. Depth of the image, in bits; must be one of 8, 10, 12, 14, 16, 24, 30, 32. With cameras deeper than eight bits, depth can be set to 8, in which case the board transfers only one byte per pixel (the most significant eight bits). For example:

```
depth: 8
```

Compare `extdepth`; see also `shift` and `mask`.

DIRECTION

Obsolete. Ignored.

DIS_SHUTTER

Obsolete.

`disable_mdout`

Non-Camera Link boards only. A value of 0 clears the ENMCOUTL bit in the Utility register (the default), allowing the four Mode Control signal pairs (MC0–3) to be used for mode control — their usual use.

A value of 1 sets the bit, enabling the PCI DV to use those four signal pairs for incoming data. This is necessary for cameras with pixels of 12 bits or greater that use some of the Mode Control lines for pixel data. For more details, see the [PCI DV Camera Link Firmware Reference](#) or the [PCI DV AIA Hardware Addendum](#). For example:

```
disable_mdout: 1
```

DOUBLE_RATE

Non-Camera Link boards only. Enable (1) or disable (0, the default) the clock doubler for high-speed cameras. For example:

```
DOUBLE_RATE: 1
```

For more information, see the description of bit 0 in the Region of Interest Control register, in the [PCI DV AIA Hardware Addendum](#).

DUAL_CHANNEL

For dual-channel cameras. A value of 0 (the default) disables dual-channel input; a value of 1 enables it. Does not apply to the PCI DV C-Link or the PCI DV FOX used with the PCI RCX C-Link; instead use `CL_DATA_PATH_NORM`. For example:

```
DUAL_CHANNEL: 1
```

ENABLE_DALSA

Non-Camera Link boards only. Enables the EN_DALSA bit in the Configuration Register, which transforms standard AIA exposure control signals to the PRIN / EXSYNC shutter-controlled timing

used by some DALSA cameras when running in board-controlled exposure mode. For more information, see the [PCI DV AIA Hardware Addendum](#). For example:

```
ENABLE_DALSA: 1
```

extdepth

Required. The actual number of bits per pixel output by the camera, regardless of how many the DMA engine actually transfers. Possible values are 8, 10, 12, 14, 16, 24, 30, 32.

For example:

```
depth: 8 # 10-bit camera, send only 8 bits.
extdepth: 10
```

Compare [depth](#). See also [shift](#) and [Compare markbin..](#)

fieldid_trig

Non-Camera Link cameras and boards only. A value of 1 sets the HWTRIG bits of the Utility 2 register, enabling the PCI DV to use the field ID signal pair to trigger the camera from an external source. A value of 0 (the default) clears the bits, which is the most useful setting for most cameras. For more information, see [PCI DV AIA Hardware Addendum](#). For example:

```
fieldid_trig: 1
```

Compare [photo_trig](#).

foi_init

Obsolete.

foi_rbtfile

Obsolete.

force_single

Tells the application to use only one buffer, for cases where the camera uses a serial command or other method to trigger the camera (for example, a Spectral Instruments camera) that violates the ring buffers' pipelining. This flag is used by the EDT Digital Video library, but if there is a chance that you're using such a camera, then your application must check for this flag as well. To do so, check the return value from `pdv_force_single`. See the [EDT DMA & Digital Video Software Library](#).

NOTE To avoid application failure, if this flag is set, do not start multiple buffers at once, as is the normal, pipelined case (as `take.c` and `simple_take.c` do). For more information, see the source code for `take.c`, which checks this flag and deals with the result. For example:

```
force_single: 1
```

frame_delay

Applies only on a PCI DV or PCI DVK, and only when the `genericsim` directive is also used. The number of lines to delay between frame-valid signals when using board-generated simulator data. For example:

```
frame_delay: 255
```

frame_height

Used to give an application an idea of the frame height. Not used by the driver or Digital Video library, this directive has no effect; it is provided to pass through to applications with graphical user interfaces,

such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. For example:

```
frame_height: 80
```

`frame_period`

Useful for cameras that are not ready for a trigger immediately after acquiring a frame, or when you want the board to output a continuous trigger at a specified interval. Units are microseconds. Applies only when the application specifies `method_frame_timing` with a valid argument — either `FMRATE_ENABLE` OR `FVAL_ADJUST`.

Depending on the state of `method_frame_timing`, an integer that sets one of two things:

- if continuous frame triggers are used (`FMRATE_ENABLE`), the interval between triggers;
- otherwise (`FVAL_ADJUST`), the number of microseconds after the end of a frame before starting the next.

See [method_frame_timing](#).

`fv_once`

Enables (a value of 1) or disables (a value of 0 — the default) continuous acquisition after the first frame valid. Causes the board to acquire successive frames without waiting for subsequent `FVAL` signals to go `TRUE`, after the first. Not normally needed, use this flag when latency between frames is very short, or to overcome the 64 MB per frame limit by capturing single images over multiple buffers. Use with [frame_height](#). For example:

```
fv_once: 1
```

`fval_done`

Camera Link boards only. Enables (a value of 1) or disables (a value of 0 — the default) image acquisition termination when the frame-valid signal goes `FALSE`. By default, the board terminates acquisition only when the expected image data (`width * height`) has been transferred, or after the timeout period has expired (see `pdv_get_timeout`). When `fval_done` is enabled, the board aborts acquisitions when the driver detects the frame valid going `FALSE`, whether or not all of the expected data has come in.

Line scan application typically set `height` to the maximum possible number of lines, and the frame valid signal is generated from an external sensor (for example, on a conveyor belt). Images are then read into a fixed-sized buffer that may be only partially filled. To determine how many lines were transferred before the frame valid signal terminated the acquisition, use the `pdv_get_lines_xferred` subroutine. For example:

```
fval_done: 1
```

`gain_min`

Minimum allowable gain setting for the camera model. Applies only to cameras that have computer-controlled gain, such as Redlake MEGAPLUS serial cameras. Not used by the driver or Digital Video library, this directive has no effect; it is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. For example:

```
gain_min: 0
```

`gain_max`

Maximum allowable gain setting for the camera model. Applies only to cameras that have computer-controlled gain, such as Redlake MEGAPLUS serial cameras. Not used by the driver or Digital Video

library, this directive is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. For example:

```
gain_max: 255
```

genericsim

PCI DV and PCI DVK boards only (for internal simulator functionality on Camera Link, see the Camera Link Internal Simulator section of the [PCI DV Family User's Guide](#)). If this value is not zero, `initcam` configures the device to simulate a camera; it then generates its own data. Three nonzero values are possible:

- 1 = *single* `initcam` prompts the user to press **Return** to generate one frame.
- 2 = *continuous* The simulator generates frames continuously.
- 3 = *triggered* The board generates a frame each time an exposure is requested.

For example:

```
genericsim: 3
```

Also see [sim_height](#), [sim_width](#), and [simulator_speed](#).

hactv

The width, in pixels, of a rectangular region of interest; use with `hskip`, `vskip`, and `vactv` to set the other coordinates. When set, this value overrides the image width throughout the software and firmware. For more information, see `pdv_set_roi` and `pdv_enable_roi` in the [EDT DMA & Digital Video Software Library](#), and Region of Interest section of the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example

```
hactv: 1024
```

See [hskip](#), [vactv](#), and [vskip](#).

header_dma

Initial setting for header DMA flag. Causes `initcam` to call `pdv_set_header_dma` with the specified value. Valid arguments are 0 (FALSE—the default) or 1 (TRUE). For complete information on the header functionality, see the `pdv_get/set_header_*` library subroutines in the [EDT DMA & Digital Video Software Library](#). For example:

```
header_dma: 1
```

header_size

An integer value representing the initial setting for header size. Causes `initcam` to call `pdv_set_header_size` with the indicated value. , valid range 0-65535. For complete information on the header functionality, see the `pdv_get/set_header_*` library subroutines in the [EDT DMA & Digital Video Software Library](#).

For example:

```
header_size: 1024
```

height

Required. Specifies the height, in pixels, of the image data output by the camera. Because some cameras output more lines per image than are in the CCD's active image area, this value does not always match the height described in the camera documentation.

For example:

```
height: 1024
```

If `vactv` is specified, its value overrides that specified in `height`.

See also `vactv` and `vskip`. Compare `width`.

`hskip`

The upper left X coordinate of a rectangular region of interest; use with `hactv`, `vskip`, and `vactv` to set the other coordinates. For more information, see `pdv_set_roi` and `pdv_enable_roi` in the [EDT DMA & Digital Video Software Library](#), and Region of Interest section of the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
hskip: 10
```

See `hactv`, `vactv`, and `vskip`.

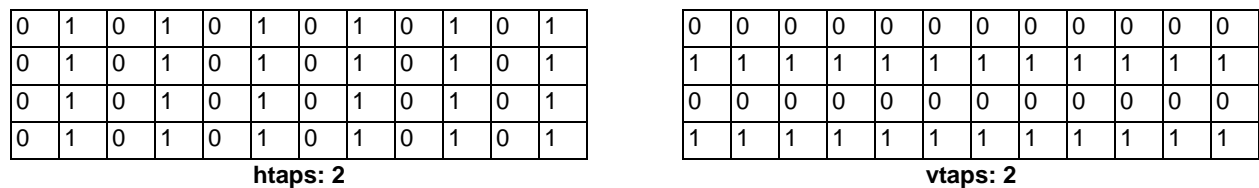
`htaps`

Number of horizontal taps per pixel clock cycle. Sets up the board's DMA logic to sequence the DMA data properly for a correctly displayed image. To display an image correctly, the board must be set correctly for:

- the number of DMA channels, corresponding to the number of taps in the camera (set with `CL_DATA_PATH_NORM`), and
- for two-tap cameras, whether the pixels coming in from alternate taps are supposed to be next to each other on the same line (`htaps: 2`), or in the same relative position on adjacent lines (`vtaps: 2`).

[Figure 1](#) shows the difference between the two types of pixel ordering (for an imaginary camera with only twelve pixels per line). In this figure, pixels are labeled according to the DMA channel, or camera tap, from which they originate:

Figure 1. Horizontal vs. Vertical Pixel Ordering in Two-tap cameras



For a two-tap camera, set either `htaps` or `vtaps` to 2, but never both, and only if `CL_DATA_PATH_NORM` is set to specify a two-tap camera.

See also `vtaps` and `CL_CFG2_NORM`. For example:

```
htaps: 2
```

`hwpad`

Included for backwards compatibility with older PCI DV or PCI DVK boards. For EDT digital video products developed more recently than 1999, which include the Region of Interest functionality, instead clip image width to a four-byte boundary using the `hactv` directive.

Number of pixels to append to each line of data — useful for cameras that output a number of bytes per line that is not an even multiple of four, setting this value is often necessary for display operability (for example, `PdvShow` and any other applications that use the MFC library). Valid values are 1, 2, or 3.

For example:

```
hwpad: 3
```

image_offset

Obsolete. Use [header_size](#).

interlace

Required for the Photonic Systems CCIR camera only. The offset from the beginning of the frame to the second bank —necessary to properly order the data in memory for correct image display. For example:

```
interlace: 185101
```

INV_SHUTTER

Invert the polarity on the shutter (EXPOSE) line, so that negative is true. By default, positive is true. Valid values are 0 (positive is true) and 1 (negative is true). For example:

```
INV_SHUTTER: 1
```

irris_strip

Alias for [hwpad](#).

kbs_red_row_first

Only for color cameras that use Bayer filters. A value of 1 indicates the red row is first; a value of 0 indicates the blue row is first.



kbs_red_row_first: 0



kbs_red_row_first: 1

For example:

```
kbs_red_row_first: 0
```

kbs_green_pixel_first

Only for color cameras that use Bayer filters. A value of 1 indicates the green pixel is first in the first row; a value of 0 indicates the green pixel is second in the first row.



kbs_green_pixel_first: 0



kbs_green_pixel_first: 1

For example:

```
kbs_green_pixel_first: 1
```

line_delay

Applies only when the PCI DV or DVK internal simulator is enabled (see [genericsim](#)). Specifies the number of pixel clock cycles to delay between line valid signals. For example:

```
line_delay: 255
```

markbin

When enabled (any nonzero value), produces a 32-bit frame counter and replaces four pixels on the output image (for 8-bit per pixel images) or two pixels in the output image (for 10- to 16-bits per pixel images, which display using two bytes per pixel) with the four bytes representing that value. The position of the first pixel to be replaced is specified by the value given as an argument; this is an offset into the image in host memory. Subsequent pixels are contiguous.

The counter specifies the number of frames that have been acquired since `pdv_open` was last called. Your application can then access this integer by specifying the offset into the image. This can be useful, for example, to ensure that images are being continually acquired when successive images do not change (for example, images from a simulator). Pixels are counted from the top left of the image. Visible effects are minimal; displaying the image shows one pixel flickering.

Because a value of zero (the default) disables the counter, pixel 0, the topmost leftmost pixel, is not available for this purpose.

For example:

```
markbin:4
```

Compare [markras](#), [markrx](#), and [markry](#).

markras

When enabled (a value of one), produces a 7-digit frame counter and superimposes it onto an image in host memory in a 56- by 8-pixel rectangle, in the position specified by `markrx`, `markry`. The counter displays the number of frames that have been acquired since `pdv_open` was last called. This can be useful, for example, to ensure that images are being continually acquired when successive images do not change (for example, images from a simulator), or to ensure that no images in a sequence have been deleted. The default value, zero, disables this feature. Use with the directives [markrx](#) and [markry](#) to specify where in the frame to display the counter. For example:

```
markras:1
markry:100
markrx:200
```

Compare [markbin](#).

markrx

Assuming that `markras` has been enabled, specifies the X coordinate within the image at which to place the top left corner of the image counter rectangle. Use with the directive [markras](#) to enable the counter feature, and [markry](#) to specify the Y coordinate. For example:

```
markras:1
markry:100
markrx:200
```

Compare [markbin](#).

markry

Assuming that `markras` has been enabled, specifies the Y coordinate within the image at which to place the top left corner of the image counter rectangle. Use with the directive [markras](#) to enable the counter feature, and [markrx](#) to specify the X coordinate. For example:

```
markras:1
markry:100
markrx:200
```

Compare [markbin](#).

mask

Only for non-Camera Link cameras. The value to which to set the Mask Lo and Mask Hi registers. Bits set to 0 force the corresponding camera data bit to 0. (For Camera Link cameras, see `CL_DATA_PATH_NORM`). For more details, see the PCI DV Addendum A:AIA Generic. For example:

```
mask: 3FF (for a 10-bit camera)
```

mc4

Obsolete.

method_camera_continuous

Obsolete. Ignored.

method_camera_download

Specifies the download method to use for the file specified by `camera_download_file`. Valid values are `IRC_160` (some Cincinnati Electronics cameras) and `SPECINST_SERIAL` (Spectral Instruments cameras). For example:

```
method_camera_download: SPECINST_SERIAL
```

method_camera_shutter_timing

Specifies the use of board-controlled expose timing or one of the camera-specific expose timing methods defined in the software library.

NOTE This directive sets only the board's exposure mode, not the camera's. To avoid unpredictable results, ensure that the camera is in a compatible exposure mode (ordinarily, by means of a serial command).

By default (when no `camera_shutter_timing` directive is present), the board does not control the timing. Instead, timing is controlled by serial commands (sent by the serial command utility `serial_cmd`, or one of the `pdv_serial_command` library subroutines), or by means of a camera manufacturer-supplied Camera Link serial control panel, or by some other camera-specific method.

Valid values are:

`AIA_SER` Default. Generic serial timing mode. No timing from the board; camera's exposure time controlled by means of a serial command or other camera-specific method, as is typically the case for for freerun mode.

`AIA_MCL` Mode control timing. The board holds the CC1 (EXPOSE) line high for the duration of the exposure, which is set using `pdv_set_exposure`. Units are milliseconds.

`AIA_MCL_100US` Mode control timing. Same as `AIA_MCL` except that the units are microseconds.

For example:

```
method_camera_shutter_timing: AIA_MCL
```

method_flushdma

Obsolete. This functionality has been incorporated into the driver, so this directive is no longer necessary.

method_frame_timing

Enables the frame timer and determines its function. Valid values are:

`FMRATE_ENABLE` The board sends continuous triggers to the camera at an interval set by the `frame_period` directive. (Applies to firmware of version 35 or later.)

FVAL_ADJUST The board waits until the frame timer value (set by the `frame_period` directive) has counted down to zero, instead of sending the next trigger immediately upon seeing a frame valid TRUE (as is the normal case). This is necessary for cameras whose minimum interval between frame triggers is longer than the time it takes to acquire and transfer the image.

Both assumes `MODE_CNTL_NORM` has been set as appropriate for board triggering. For example:

```
# adjust trigger timing such that a trigger always comes 100 ms after
# beginning of previous frame readout
MODE_CNTL_NORM: 10
method_frame_timing: FVAL_ADJUST
frame_period: 100000

# send a trigger pulse out on EXPOSE line every 300 ms
MODE_CNTL_NORM: 10
method_frame_timing: FMRATE_ENABLE
frame_period: 300000
```

method_header_position

Initial setting for header position, if header is present. Causes `initcam` to call `pdv_set_header_position` with the indicated value. Valid arguments are `HEADER_BEFORE`, `HEADER_AFTER`, `HEADER_WITHIN`. Typically used along with `header_dma` and `header_size`. For complete information about the header functionality, see the `pdv_get/set_header_*` library subroutines in the [EDT DMA & Digital Video Software Library](#).

For example:

```
method_header_position: HEADER_WITHIN
```

method_interlace

Tells the library which method to use to reorder the pixels with a frame, for interleaved or interlaced cameras. Valid values are:

BGGR Decodes data in Bayer-filtered format for cameras with 8 bits per pixel. Use with `kbs_green_pixel_first` and `kbs_red_row_first` to set the decoding to match the specific order of the Bayer filter.

BGGR_WORD Decodes data in Bayer-filtered format for cameras with 10–16 bits per pixel. Use with `kbs_green_pixel_first` and `kbs_red_row_first` to set the decoding to match the specific order of the Bayer filter.

BYTE_INTLV Bytes are interleaved. Two-byte pairs, from adjacent lines, are contiguous in the data stream. Most commonly used with dual-tap, non-Camera Link cameras.

DALSA_2CH_INTLV See `INVERT_RIGHT_INTLV`.

DALSA_4CH_INTLV See `QUADRANT_INTLV`.

ES10_BGGR_INTLV See `BGGR`.

EVEN_RIGHT_INTLV Data is organized in pairs of pixels — odd pixels from the left progressing right, and even pixels from the horizontal center, also progressing right.

FIELD_INTLC Data is organized in pairs of pixels — odd pixels starting at the top left corner of the frame, and even pixels starting at the vertical center, lefthand edge.

INVERT_RIGHT_INTLV

For cameras with sensors like that of the Dalsa A series, which send the data in pixel pairs, odd bytes starting from the left and progressing right, even bytes starting from the right and progressing left, with pixel pairs meeting in the center.

INVERT_RIGHT_BGGR_INTLV

Combined **INVERT_RIGHT_INTLV** and **BGGR**. Decodes Bayer-filtered data using the Dalsa A model type sensor.

PIRANHA_4CH_INTLV

See **QUADRANT_INTLV**.

QUADRANT_INTLV

For cameras with Dalsa four-tap sensors having data coming in in four-pixel quads, progressing inward (first horizontally, then vertically) from the four corners.

SPECINST_4PORT_INTLV

See **QUADRANT_INTLV**.

WORD_INTLV

Same as **BYTE_INTLV**, for cameras with more than eight bits per pixel.

WORD_INTLV_HILO

Same as **BYTE_INTLV** except pixels start at the left sides of both the top and bottom of the frame, converging toward the center.

WORD_INTLV_ODD

Same as **WORD_INTLV** but with the first pixel coming from the second line.

For example:

```
method_interlace: BYTE_INTLV
```

method_lock_shutter

Obsolete.

method_serial_format

Obsolete. To specify special handling of the **serial_init** string, use one of the special directives **serial_init_hex**, **serial_init_baslerf**, or **serial_init_duncanf**.

method_serial_mode

Required for RS-232 camera serial control through the EDT board. If this directive is missing, the default is differential (RS-422 or LVDS).

Applies only to the PCI DVa, and to PCI DV RCI boards of Revision 02 or greater.

NOTE To enable RS-232 serial control through the EDT board, the RS-232 jumper on the board must also be set.

Currently the only valid argument for this directive is **RS232**; if the camera uses differential signal levels (RS-422 or LVDS) for the serial channel, omit the directive from the configuration file, and set the jumper to **DIFF**.

For example:

```
method_serial_mode: RS232
```

method_set_gain

Obsolete.

method_set_offset

Obsolete.

`method_shutter_speed`

An alias for `method_camera_shutter_timing`, described on [page 23](#).

`method_startdma`

Obsolete; this functionality has been incorporated into the driver so that this directive is no longer necessary. The driver now automatically flushes the DMA before acquiring a frame.

`MODE_CNTL_NORM`

Sets the state of the camera control (CC) lines to the camera. Value is an 8-bit hexadecimal number. The right nibble sets the steady state of the four CC lines, and the left nibble selects which of the lines, if any, to use for sending trigger or expose pulses. For cameras in free-run mode, set the left nibble to zero. For cameras that expect a trigger or expose pulse, the left nibble is ordinarily one, because the EXPOSE line for almost all cameras is CC1.

If the leftmost nibble is not zero, the board sends out a one-millisecond pulse on the indicated CC line for every acquire, unless `method_camera_shutter_timing` is set to `AIA_MCL`, in which case the pulse instead lasts for the duration set by the `pdv_set_exposure` library subroutine.

NOTE These modes control the operation of the frame-grabber only. To achieve the expected results, the camera must also be configured to run in a compatible or matching mode, either by means of serial commands (see the `serial_init` directive) or the steady state of another CC line or lines, or some other external means such as a camera manufacturer-supplied application.

For more information, see your camera documentation and the description of the the Mode Control register in the [PCI DV Camera Link Firmware Reference](#).

For example:

```
# send a trigger or pulse on CC2, set CC2 high (on)
# CC3 and CC4 are low (off)
MODE_CNTL_NORM: 12
```

`offset_min`

Minimum allowable offset (black level) setting for the camera model. Applies only to cameras that have computer-controlled offset, such as the Redlake MEGAPLUS serial cameras. Not used by the driver or library, this directive is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. For example:

```
offset_min: 0
```

`offset_max`

Maximum allowable offset (black level) setting for this camera model. Applies only to cameras that have computer-controlled offset, such as Redlake MEGAPLUS serial cameras. Not used by the driver or library, this directive is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. For example:

```
offset_max: 255
```

`pause_for_serial`

Can be used to add a pause between serial characters if required for reliable serial communication. Not needed for most cameras. The value is an integer; units are milliseconds. For example:

```
pause_for_serial: 50
```

`pclock_speed`

This directive specifies a pixel clock speed, in MHz. It has two uses:

- As part of its initial configuration, `initcam` sets an automatic timeout value. To do so, it uses a conservative estimate of 5 MHz for the pixel clock speed, which is slow for many cameras. Most cameras can therefore use this directive to bias `initcam`'s automatic timeout to a more appropriate value, resulting in more reasonable wait times for image timeouts, should data loss occur.
- A few non-Camera Link cameras do not generate a pixel clock when the frame-valid signal is false. In this case, use this directive in concert with the `rbtfile` directive, specifying `aia_async.bit` or some other firmware file that uses an asynchronous pixel clock. In such cases, valid values for this directive are 5, 10, and 20. For example:

```
pclock_speed: 20
```

`photo_trig`

A value of 1 sets bit 0 of the Utility 2 register, enabling the PCI DV to use the external optical trigger input pins to trigger the camera. (For the location of these pins, see the [PCI DV Family User's Guide](#).) A value of 0 (the default) clears the bit.

For more details, see the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
photo_trig: 1
```

Compare `fieldid_trig`.

`pulnix`

Non Camera-Link boards only. Set this to 1 to set the PULNIX bit in the Utility 2 register — required for certain Pulnix cameras, notably the TM-9701 in mode 9. For more information, see the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
pulnix: 1
```

`rbtfile`

Required for non-Camera Link boards and the PCI DV FOX. Ignored on Camera Link boards. Specifies which firmware file to download into the onboard FPGA for this device. If no absolute path is specified, searches the `camera_config/bitfiles` subdirectory of the EDT distribution directory for the appropriate file for the board, as shown below.

<code>aiag.bit</code>	PCI DV, PCI DVk, PCI DVa
<code>aiag_2ch.bit</code>	PCI DVa boards with 2-channel cameras of 10 or more bits per pixel
<code>aiagcl.bit</code>	Camera Link and FOX boards (for example, PCI DV C-Link and PCI DV FOX)

For specific characteristics of other special firmware files, contact EDT.

For Camera Link FPGA configuration file functionality, see [PCI DV Camera Link Firmware Reference](#).

For non-Camera Link FPGA configuration file functionality, see [PCI DV Firmware Reference \(other than Camera Link\)](#).

For example:

```
rbtfile: aiagcl.bit
```

rgb30

For 30- to 32-bit cameras, sets the 30-bit RGB multiplex method. Applies only when the board is loaded with 32-bit firmware, such as `pdvcamlk_pir.bit`. Valid values are:

- 1 Redlake MS and DT series (formerly DuncanTech) 30-bit per pixel cameras that order the data in a manner inconsistent with the Camera Link specification.
- 3 Camera Link standard cameras.

For example:

```
rgb30: 3
```

See the [Redlake/DuncanTech Application Note](#) for more details.

sel_mc4

A value of 1 sets the `SELECT_MC4` bit in the Utility 2 register, enabling the PCI DV to use the signal pair ordinarily assigned to Serial Control Out as a fifth mode control bit. A value of 0 (the default) clears this bit. For more details, see the [PCI DV AIA Hardware Addendum](#). For example:

```
sel_mc4: 1
```

serial_aperture

For cameras that accept an ASCII serial command with one argument to set the aperture. An ASCII string enclosed in double quotes and passed to the PDV Library subroutine `pdv_set_aperture`, to define the serial command to send when the application calls `pdv_set_aperture`. If the camera uses a different serial format to set the aperture, you can instead set the aperture using calls to lower-level library routines for serial control, such as `pdv_serial_command`; or set up an initial aperture value using [serial_init](#). For example:

```
serial_aperture: "APT"
```

serial_baud

Sets the baud rate (by default, 9600) for cameras that use serial commands for camera control. If this directive is omitted, the default baud rate is used. Valid values are 9600, 19200, 38400, 57600, and 115200. For example:

```
serial_baud: 115200
```

serial_binit

For cameras that use binary numbers instead of ASCII characters for serial control. A string enclosed in double quotes, representing serial bytes to send out the serial transmit line during camera initialization. For example:

```
serial_binit: "00 11 22 33"
```

serial_binning

For cameras that accept an ASCII serial command with one argument to set the binning mode. An ASCII string enclosed in double quotes and passed to the PDV Library subroutine `pdv_set_binning`, to define the serial command to send when the application calls `pdv_set_binning`. If the camera uses a different serial format to set the binning, you can instead set the binning mode using calls to lower-level library routines for serial control, such as `pdv_serial_command`; or set up an initial binning mode using [serial_init](#). For example:

```
serial_binning: "BIN"
```

For more information on PDV Library subroutines, see the [EDT DMA & Digital Video Software Library](#).

serial_exposure

For cameras that accept an ASCII serial command with one argument to set the exposure time. An ASCII string enclosed in double quotes and passed to the PDV Library subroutine `pdv_set_exposure`, to define the serial command to send when the application calls `pdv_set_exposure`. If the camera uses a different serial format to set the exposure time, you can instead set the exposure time using calls to lower-level library routines for serial control, such as `pdv_serial_command`; or set up an initial exposure time using `serial_init`. For example:

```
serial_exposure: "EXE"
```

serial_gain

For cameras that accept an ASCII serial command with one argument to set the gain. An ASCII string enclosed in double quotes and passed to the PDV Library subroutine `pdv_set_gain`, to define the serial command to send when the application calls `pdv_set_gain`. If the camera uses a different serial format to set the gain, you can instead set the gain using calls to lower-level library routines for serial control, such as `pdv_serial_command`; or set up an initial gain value using `serial_init`. For example:

```
serial_gain: "GAE"
```

serial_init

For cameras that accept ASCII serial commands, defines a double-quoted, colon-delimited set of serial commands to send the camera at initialization. Use this, for example, to set the camera to a mode compatible with the frame-grabber mode. For example:

```
serial_init: "RDM 2:TRM N:MDE TR"
```

serial_init_baslerf

Similar to `serial_init`, except `serial_init_baslerf` causes `initcam` to treat the command string as a series of colon-separated hexadecimal commands and to add Roper/Basler-format framing to each command in the string before sending it to the camera. Specifically, an STX character (0x02) is prepended, and the BCC is calculated and appended along with an ETX (0x03) character. Refer to the *Basler A202K Camera Manual*, and the PDV Library subroutine `pdv_send_basler_frame`. For example:

```
serial_init_baslerf: "c00103:a00100:a603530000:a7030b5100"
```

serial_init_duncanf

Similar to `serial_init`, except `serial_init_duncanf` causes `initcam` to treat the command string as a series of colon-separated hexadecimal commands and to add Roper/DuncanTech-format framing to each command in the string before sending it to the camera. Specifically, an STX character (0x02) is prepended, and BCC is calculated and appended. Refer to the DuncanTech User Manual, and the PDV Library subroutine `pdv_send_duncan_frame`. For example:

```
serial_init_duncanf: "0300160000:04001a01ba00"
```

serial_init_hex

Similar to `serial_init`, except `serial_init_hex` causes the initialization string to be treated as a series of hexadecimal bytes, separated by spaces, which will be converted from ASCII to binary before being sent to the camera. Use this, for example, to set the camera to a mode compatible with the frame-grabber mode. See also the PDV Library subroutine `pdv_serial_binary_command`. For example:

```
serial_init_hex: "80 82 40 82 80 85 11"
```

serial_offset

For cameras that accept an ASCII serial command with one argument to set the offset (black level). An ASCII string enclosed in double quotes and passed to the PDV Library subroutine `pdv_set_blacklevel`, to define the serial command to send when the application calls `pdv_set_blacklevel`. If the camera uses a different serial format to set the black level, you can instead set the black level using calls to lower-level library routines for serial control, such as `pdv_serial_command`; or set up an initial black level using `serial_init`.

Valid values are as defined by the camera manufacturer — consult your camera documentation. For example:

```
serial_offset: "BKE"
```

serial_response

For ASCII serial commands, sets the expected response from the camera. Used only for a few cameras that have a specific known response to all commands. The application can use this directive to set the expected response to which it can compare the actual response, in order to perform serial command-response handshaking. For example:

```
serial_response: "Y"
```

serial_term

A string enclosed in double quotes that sets the terminator character(s) to append to ASCII serial commands sent using the `serial_init` directive and other ASCII serial directives, as well as the `pdv_serial_command` library subroutine. The default is `"\r"` (carriage return). To specify no appended characters when sending ASCII serial commands, set this to the empty string (`" "`). For example:

```
serial_term: ""
```

serial_timeout

The number of milliseconds to wait for a response from a serial command sent by `pdv_serial_wait`. If the camera doesn't respond overtly to serial commands, set this to a value high enough to ensure that the command has time to complete; check the camera manufacturer's specifications for details. Valid values are 0– 65535; the default is 1000.

Note that the value set by this directive only effects higher level serial communications such as the `serial_init` phase of the initialization, and convenience routines such as `pdv_set_gain`. When sending serial via the direct serial subroutines such as `pdv_serial_command`, the value set by `serial_timeout` will be ignored since the timeout value gets sent explicitly as part of the subroutine call. For example:

```
serial_timeout: 500
```

serial_trigger

A string enclosed in double quotes representing the ASCII serial string the driver sends to trigger the camera. Few cameras provide this functionality.

NOTE To avoid timing problems, do not use this directive to trigger the camera when another triggering method is possible.

For example:

```
serial_trigger: "D"
```

serial_waitc

An eight-bit hexadecimal value that sets the expected terminator for serial responses. For ASCII serial cameras, the `pdv_serial_read` subroutine normally waits for the expected number of characters (see

library subroutines `pdv_serial_wait` and `pdv_serial_read`) or a serial timeout (see the `serial_timeout` directive) before returning. This directive can help significantly shorten the time it takes for a serial command and response sequence to complete: without it, an application must either know the exact number of characters it expects in response to every serial command, or else wait for the largest possible number of characters, and then wait for the `serial_timeout` value to expire every time `pdv_serial_wait` is called. For example:

```
serial_waitc: 0D
```

shift

Non-Camera Link boards only. The hexadecimal value to which to set the eight-bit Shift register. Set bit 4 to cause incoming pixels to be sent in the opposite order (most significant bit first); bits 0–3 barrel-shift incoming data by the specified amount. (Bits 5–7 are not used.) For more details, see the [PCI DV AIA Hardware Addendum](#). For example:

```
shift: 10
```

shortswap

Enables (a value of 1) or disables (a value of zero, the default) swapping of shorts (two-byte words). Analogous to [byteswap](#). This is a hardware operation and does not degrade performance. For example:

```
shortswap: 1
```

shutter_speed_frontp

Obsolete.

shutter_speed_min

Minimum allowable shutter speed (exposure time) setting for the camera model. Not used by the driver nor the software library, this setting is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. When using EDT's camera shutter timer (`MODE_CNTL_NORM 10` and `method_camera_shutter_timing: AIA_MCL`), the minimum is 0; otherwise the setting is camera-dependent (assuming a particular camera's method of shutter timing is implemented in the EDT software library). For example:

```
shutter_speed_min: 0
```

shutter_speed_max

Maximum allowable shutter speed (exposure time) setting for the camera model. Not used by the driver or the software library, this setting is provided to pass through to applications with graphical user interfaces, such as `PdvShow`, which can use it, for example, to determine what values should mark the ends of user-controlled sliders. When using EDT's camera shutter timer (`MODE_CNTL_NORM 10` and `method_camera_shutter_timing: AIA_MCL`), the maximum is 25500, otherwise the setting is camera-dependent (assuming a particular camera's method of shutter timing is implemented in the EDT software library). For example:

```
shutter_speed_max: 20000
```

sim_height

PCI DV and PCI DVK boards only. Height of the image, in pixels. Used by `initcam` to configure the height of the simulated image when the device is in simulator mode. For example:

```
sim_height: 1024
```

See [genericsim](#). For internal simulator functionality on Camera Link, see the Camera Link Internal Simulator section of the [PCI DV Family User's Guide](#).

sim_width

PCI DV and PCI DVK boards only. The width of the image, in pixels. Used by `initcam` to configure the width of the simulated image, when the device is in simulator mode. For example:

```
sim_width: 1024
```

See [genericsim](#). For internal simulator functionality on Camera Link, see the Camera Link Internal Simulator section of the [PCI DV Family User's Guide](#).

simulator_speed

PCI DV and PCI DVK boards only. Sets the speed at which `genericsim` transfers internally generated data. Applies only when the firmware file is `xtest.bit`. Valid values are:

0	5 MHz
1	10 MHz
2	20 MHz

For example:

```
sim_speed: 2
```

See [genericsim](#). For internal simulator functionality on Camera Link, see the Camera Link Internal Simulator section of the [PCI DV Family User's Guide](#).

skip

Obsolete.

slop

Obsolete.

timeout_multiplier

Used as a multiplier for the default timeout value. Valid values are 1–65535. Useful with slow cameras when the timeout value that `initcam` (or `pdv_set_exposure`) calculates isn't long enough. See `edt_timeout` in the EDT software library for an explanation. Compare `user_timeout`. For example:

```
timeout_multiplier: 2
```

Compare [pclock_speed](#).

TRIG_PULSE

Obsolete.

user_timeout

Sets a specific timeout value, in milliseconds, for acquisition routines to wait for all the image data before returning. Overrides the default, which is to adjust the timeout value automatically depending on exposure time and image size. Normally the automatic value is adequate, and the preferred method for making it longer is to use the `pclock_speed` or `timeout_multiplier` directive. Therefore, use this directive only when a fixed or infinite timeout is required — for example, with some externally triggered camera applications. Valid range is 0–65535, with zero indicating forever. For example:

```
user_timeout: 1000
#wait for one second before timing out
```

vactv

The height, in pixels, of a rectangular region of interest; use with `hskip`, `hactv`, and `vskip` to set the other coordinates. When set, this value overrides the image height throughout the software and firmware. For more information, see `pdv_set_roi` and `pdv_enable_roi` in the [EDT DMA & Digital Video Software Library](#), and Region of Interest section of the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
vactv: 1033
```

See [hactv](#), [hskip](#), and [vskip](#).

variable_size

Obsolete. Ignored.

vskip

The upper left Y coordinate of a rectangular region of interest; use with `hactv`, `hskip`, and `vactv` to set the other coordinates. For more information, see `pdv_set_roi` and `pdv_enable_roi` in the [EDT DMA & Digital Video Software Library](#), and Region of Interest section of the [PCI DV AIA Hardware Addendum](#) or the [PCI DV Camera Link Firmware Reference](#). For example:

```
vskip: 10
```

See [hactv](#), [hskip](#), and [vactv](#).

vtaps

Number of vertical taps per pixel clock cycle. Sets up the board's DMA logic to sequence the DMA data properly for a correctly displayed image. To display an image correctly, the board must be set correctly for:

- the number of DMA channels, corresponding to the number of taps in the camera (set with `CL_DATA_PATH_NORM`), and
- for two-tap cameras, whether the pixels coming in from alternate taps are supposed to be next to each other on the same line (`htaps: 2`), or in the same relative position on adjacent lines (`vtaps: 2`).

Figure 1 shows the difference between the two types of pixel ordering. For a two-tap camera, set either `htaps` or `vtaps` to 2, but never both, and only if `CL_DATA_PATH_NORM` is set to specify a two-tap camera.

See also [htaps](#) and [CL_CFG2_NORM](#). For example:

```
vtaps: 2
```

width

Required. Specifies the width, in pixels, of the camera image. Because some cameras output more pixels per line than are in the CCD's active image area, this value does not always match the width described in the camera documentation.

If `hactv` is specified, its value overrides that specified in `width`.

For example:

```
width: 1024
```

See also [hactv](#) and [hskip](#). Compare [height](#).

xregwrite_xx

Sets specific register values on the board, overriding any values for the register set by other directives.

NOTE Do not add or modify any instances of this directive in existing camera configuration files before consulting EDT.