Raspberry Pi Image Signal Processor (PiSP) Specification

Raspberry Pi Ltd.

Colophon

Compiled on 25/10/2023
Version 0.5(1b8074448ac1c9742fe8e33a8a0749c3fd6efae5-clean)

Revision History

Version	Date	Description
0.5	10-October-2023	Initial public draft revision.

Contents

List of Figures

List of Tables

1 Introduction

The PiSP is Raspberry Pi's Image Signal Processor (ISP). It is designed to process images from Bayer and monochrome (greyscale) camera sensors. The PiSP targets the following principal requirements:

- Similar or better image quality to previous versions of the Raspberry Pi.
- Simplified camera tuning compared to previous versions of the Raspberry Pi.
- Greater pixel throughput than previous versions of the Raspberry Pi, capable of handling up to 4Kp60, though this will depend on clock rate. We expect to target typically between 200MHz and 700MHz maximum clock rate at 2 pixels/cycle (note that 4Kp60 will require $\geq$ 320MHz).
- Raspberry Pi's own software and IP (Intellectual Property).
- The hardware and software are both flexible allowing us to implement a wider range of applications and use cases.

This document gives an overview of the PiSP, explaining how its various components operate, moving on to a complete specification of all the registers and their associated behaviours.

Software resources

A software library is available at `https://github.com/raspberrypi/libpisp` to help configure the ISP. Together with `libcamera` and Linux V4L2 drivers, it provides a complete open source software stack to drive the PiSP from user applications.

2 Overview

The PiSP is divided into two parts, one or more Front Ends and one or more (though usually only one) Back End, as shown below.

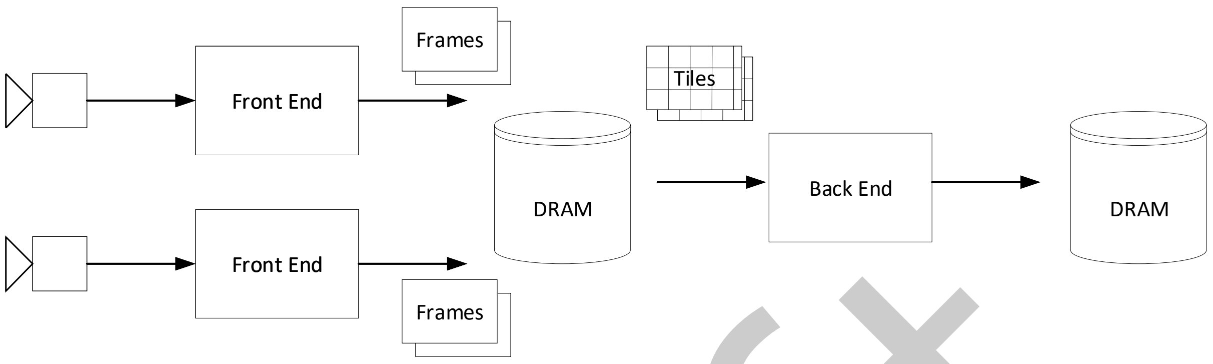

Figure 1: PiSP, showing Front and Back Ends

2.1 Front End

The Front End normally receives streaming pixel data directly from the camera (via the chip's camera peripheral), performs certain minimal processing on the pixels, and writes the results to main memory. For memory-to-memory operation, an AXI reader interface is provided which can feed data to the Front End in place of the camera.

A single Front End can be dedicated only to a single camera at a time, so any application requiring that multiple cameras operate simultaneously will require an equivalent number of Front Ends. The precise number of Front Ends is a matter for the chip as a whole, external to the PiSP, and the decision will be based on use case and other requirements. Please refer to section 3 for further information.

The Front End can perform limited image processing and rescaling, to reduce the load on the Back End in certain use cases. It can generate up to two differently cropped and scaled outputs.

The Front End gathers various image statistics which are written to memory alongside the image. The statistics are intended to be used by control algorithms running in the firmware, such as the 3A algorithms (Auto Exposure/Gain Control, Auto White Balance and Auto Focus).

The Front End has to work in "hard" real-time and requires good Quality of Service to main SDRAM. The maximum tolerable latency is governed by the size of internal FIFOs between the camera and Front End, and of the Front End's output buffer. The Back End, by contrast, can work in "soft" real-time as it operates only from memory to memory. It is free to stall, so long as its average throughput is sufficient. One Back End can multiplex between images written to DRAM by *different* Front Ends, even at a sub-frame level.

2.2 Back End

The bulk of the processing is performed in the Back End. Images are read from DRAM in *tiles*. These are limited in width (currently to 640 pixels) to reduce the requirement for internal line memories. Software must divide the image into a series of tiles so that a proper tessellation of the output image is created, with no visible discontinuities where distinct tiles abut one another. The software schedules the Back End to work on the available Front End outputs in turn, possibly giving some priority over others.

We note that the tiled approach permits:

- Smaller line memories within the Back End resulting in a smaller overall design.
- Multiplexing between different images on a sub-frame basis. For instance, we might be processing 4Kp60 video whilst simultaneously creating a JPEG still, where the latter is produced on a "best effort" basis, only when there is no unprocessed video frame outstanding.
- Video processing with sub-frame latency.

Subsequent sections of this document will present the Front and Back Ends in much greater detail.

3 Chip and Camera Peripheral Integration

This section discusses how the PiSP might be integrated into an SoC with existing camera peripherals.

3.1 Pixel Data

Most of this material consists of recommendations or illustrations; however, we do have the following strict requirements.

- There must be at least one source of camera image data, usually a camera peripheral.
- The camera peripheral(s) should be able to stream pixels directly into the Front End's streaming input, or should be able to write them directly to memory, or preferably, both. At least one of these possibilities must be available.
- Pixels streamed directly to the Front End must conform to the Front End's interface specification.
- Pixels written to memory should be either: 16-bit Bayer or monochrome pixel values, or PiSP compressed image format. The camera peripheral(s) must always undo any SMIA raw compression or packing that is used.

Beyond this we recommend the following.

- The camera peripheral(s) should support both streaming to the Front End and writing images to memory.
- Camera peripheral(s) should be able to demux multiple virtual MIPI channels.
- The system as a whole needs to consider whether it wishes to feed multiple virtual MIPI channels each to a different Front End, or whether to write (some of) them to DRAM in order to process them later using a memory-to-memory Front End.

Below we illustrate one possible configuration of camera peripherals and Front Ends. We are assuming the existence of two camera peripherals and two Front Ends (so two cameras can run simultaneously without incurring extra round-trips through DRAM).

Legend:

- *Output n* - output image stream n from the camera peripheral. Note that some are streaming outputs; the others go straight to memory.
- *Embedded n* - any embedded data associated with Output n; this is always written to memory.
- C - proprietary PiSP RAW image compression.

The illustrative configuration above allows both cameras to run simultaneously, each delivering a single image stream directly to the pair of PiSP Front Ends. Alternatively, camera peripheral 0 can deliver a pair of image streams, each from a different virtual MIPI channel, to the two Front Ends. It is of course not possible for Front End 1 to process streaming input from both camera peripherals simultaneously. If more image streams have to be processed, this could be accomplished by feeding the largest directly to one of the Front Ends and writing all the others DRAM. These would then be be processed by the second Front End operating in memory-to-memory mode. Recall that we recommend that some, if not all, of the image AXI outputs be connected through the proprietary PiSP RAW image compression block (although it should be possible to bypass compression when not appropriate, such as for a YUV camera).

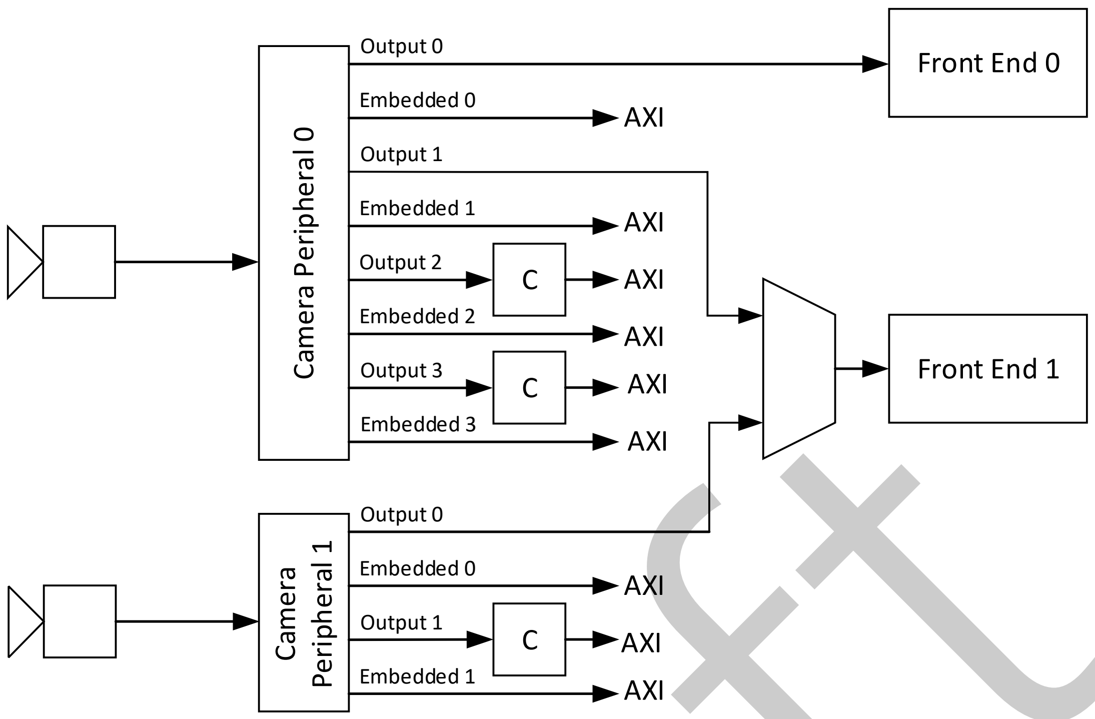

Figure 2: Two Front Ends connected to two camera peripherals

3.2 Control Signals

Besides passing pixel data to the PiSP Front End, a small number of control and other signals should also be connected. These are:

- A *Start of Frame* (*SOF*) signal. This signal indicates that a new frame of pixel data is about to start. When *SOF* is asserted, data bits [15:0] shall contain a 16-bit count (*Frame ID*) which is incremented with every new frame started by the camera.
- An *End of Frame* (*EOF*) signal. This immediately follows the end of a frame of pixel data (accompanying data bits are ignored, but error flags are significant).
- *Error Signals*. These are latched by the PiSP Front End when the *EOF* signal is asserted. Two kinds of errors may be distinguished, namely *soft* errors, meaning that if the frame just received can be ignored then the system should continue to operate normally, and *hard* errors, meaning that a permanent and fatal problem has occurred, which will probably require the camera application to terminate.
- A *Valid* signal, indicating when data (or *SOF*/*EOF*) are available to the PiSP Front End.
- A *Ready* signal, indicating that the PiSP Front End is able consume data.

This arrangement is shown in the diagrams below. The data bus for the pixel data is also shown. In this version it is 32 bits wide, for a throughput of two 16-bit pixel values per clock cycle.

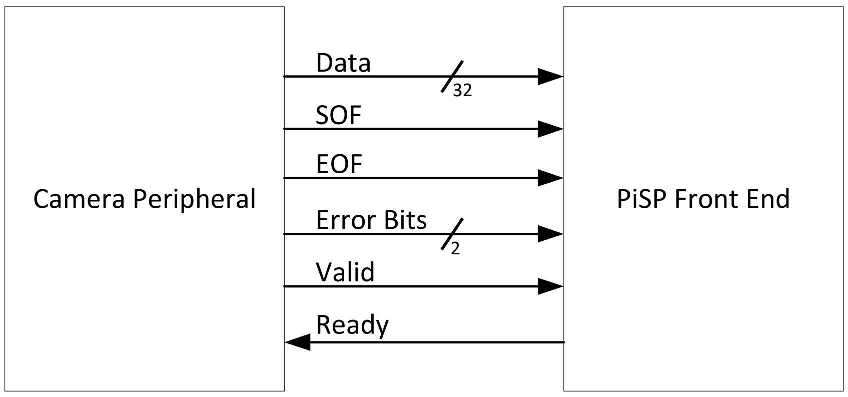

Figure 3: Connections between a Front End and a camera peripheral

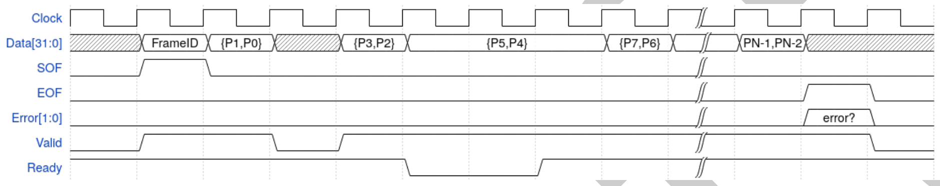

Figure 4: Example waveforms for the Camera to Front End interface

This interface always uses 16-bit pixel data (MSB-justified if smaller), and never uses compression.

Note that when multiple camera peripherals and are connected to a Front End as in figure 2, Data, SOF, EOF, Error and Valid signals should all be multiplexed under software control.

4 PiSP Compressed Raw Format

A number of the I/O blocks in the PiSP are able to read and write raw (Bayer or monochrome) image data in a compressed format. These blocks are:

- Front End input blocks
- Front End output blocks
- Back End input blocks
- TDN (Temporal Denoise) in the Back End supports writing and reading the long-term filtered frame in compressed format
- Stitching in the Back End needs to read a long/short exposure frame and write a short/long exposure frame in order to support HDR
- Additionally, camera peripherals may write directly to memory in the compressed format.

The PiSP Compressed Raw Format is a fixed-rate lossy compression format with 8 bits per pixel. The compresser accepts 16-bpp raw data in blocks of 8 horizontally adjacent pixels, encoding each block with 64 bits (in all modes). Blocks are coded independently, to reduce the cost of hardware implementation and to facilitate random access.

Source data should be justified to the MS bits of a 16-bit word and will be padded to a multiple of 8 samples. The compressor has an *offset* parameter which is subtracted from incoming pixels: this may be used to improve fidelity when there is a high black level.

The compressor has three distinct modes of operation:

- **Mode 1** is a delta-based scheme optimized for linear-light Bayer images with 10–12 significant bits, and is recommended in most such cases.
- **Mode 2** uses simple square-root-like companding, for simpler interoperation with other hardware and software.
- **Mode 3** combines companding and compression, to target HDR imagery with up to 14 significant bits, where values below FSD/16 are expected to dominate.

4.1 Outline of the delta-based scheme

When the compressor is operating in mode 1 or 3, each block of 8×1 samples is decomposed into even and odd components of 4 samples each (even if the source image was monochrome), and represented by a pair of 32-bit words, one per component, at adjacent addresses in memory.

Two bits of each 32-bit word give the *quantization mode*. This determines how pixels are quantized and encoded. The smallest and largest quantization modes use nonlinear quantization; between them they provide step sizes ranging from 16 to 512.

When quantization mode is less than 3, pixel values are represented by four fields of {9,7,7,7} bits. The first field is usually the MIN of the two middle samples, and the other fields are horizontal deltas, but the encoding is modified for values close to 0 or to FSD (to avoid redundant codes). The compressor should find the smallest quantization mode for which the MIN and deltas can be represented within their respective fields.

Quantization mode 3 represents an "escape" case in which all four samples are independently quantized to 176 levels (roughly, 7.5 bits per pixel). This limits the worst-case error to ± 256.

4.2 Companding

In mode 2, the compressor simply maps the eight 16-bit pixel values (after offsetting for black level) to 8-bit codes, by linear interpolation between these points:

(0,0), (256,16), (512,24), (2048,48), (8192,96), (32768,192), (65024,255).

In mode 3, a related companding function maps 16-bit pixel values to a 16-bit intermediate, which is then compressed using the above delta scheme. This improves precision for tiny values at the expense of larger ones.

5 Image Format Specifications

5.1 Image Formats

The tables below explain the supported image formats. The PiSP specifices image formats using a 32-bit field which aggregates a number of flags and smaller values, and the tables indicate both the numeric value of the field as well as the names of the flags used by the accompanying software (which ORs them together).

Note: in the following

- *shifts* refer to left shifting pixel values on input, and right shifting them on output. Note that programmable shifts are supported only with 16 bit per pixel output. Pixels are implicitly shifted by 8 or 6 places for, respectively, 8 bit per pixel and 10 bit per pixel output.
- a *planar* image is one where all three colour planes are stored in disjoint memory buffers.
- 422 and 420 sampling refer to pixel sub-sampling in the manner of YUV422 and YUV420 data. Both are supported as planar formats, though only 422 is supported as an interleaved format.
- interleaved 422 output may be written with the first colour plane first (that is, YUYV...) or second ("swapped", namely UYVY...). Any other pixel reordering should be performed with the CSC block in the same output branch.
- a *semi-planar* image is one where the first colour plane is stored separately but the 2nd and 3rd colour planes are interleaved. This is only supported for 422 or 420 sampling.
- a *wallpaper* format image is where the image is stored in "wallpaper rolls". That is, images are stored in vertical stripes that are 128 bytes wide. Wallpaper format is only supported for semi-planar images (with 422 or 420 sampling).
- there is an additional 422/420 semi-planar wallpaper format that uses 10 bits per sample. In this format, three 10-bit samples are written out followed by 2 bits of padding (the two high MSBs of the 32-bit word), making up a total of 32 bits.
- We note additionally that all 422 and 420 images *must* have even widths, counted in pixels. 420 images must additionally have even height.

The following formats can be read by the Front End AXI Input block, the Back End Input block at the head of the Bayer pipe, the TDN Input block and the Stitch Input blocks. The same formats can be written by the Front End Output blocks, the TDN Output block and the Stitch Output block. These formats implicitly include (single channel) greyscale images.

Description	Value	Flags
16-bits per pixel single channel uncompressed with $16 - n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x000n0003	PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel single channel compressed with mode n, for $n = 1, 2, 3$	0x0n000000	PISP_IMAGE_FORMAT_ COMPRESSION_MODE_n

Table 2: Bayer image formats.

The Input block at the head of the RGB pipe, and the Output blocks at the very end of the Back End, support the following formats.

Description	Value	Flags
8-bits per pixel three channel interleaved	0x40000000	PISP_IMAGE_FORMAT_THREE_CHANNEL
16-bits per pixel three channel interleaved with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0003	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel interleaved 422, channel 0 first (YUYV)	0x40000100	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_SAMPLING_422
16-bits per pixel three channel interleaved 422 with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$, channel 0 first (YUYV)	0x400n0103	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel interleaved 422, channel 0 second (UYVY)	0x40001100	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_ORDER_SWAPPED
16-bits per pixel three channel interleaved 422 with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$, channel 0 second (UYVY)	0x400n1103	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_ORDER_SWAPPED PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
Three 8-bit pixel values plus 8 bits of padding (value 0x00) so that every pixel occupies a 32-bit word, and with the padding byte at the *highest* address in memory	0x40100000	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_BPP_32
Three 8-bit pixel values plus 8 bits of padding (value 0x00) so that every pixel occupies a 32-bit word, and with the padding byte at the *lowest* address in memory	0x40101000	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_BPP_32 PISP_IMAGE_FORMAT_ORDER_SWAPPED
Three 8-bit pixel values plus 8 bits of padding (value 0xFF) so that every pixel occupies a 32-bit word, and with the padding byte at the *highest* address in memory	0x40300000	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_BPP_32 PISP_IMAGE_FORMAT_X_VALUE
Three 8-bit pixel values plus 8 bits of padding (value 0xFF) so that every pixel occupies a 32-bit word, and with the padding byte at the *lowest* address in memory	0x40301000	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_BPP_32 PISP_IMAGE_FORMAT_X_VALUE PISP_IMAGE_FORMAT_ORDER_SWAPPED
8-bits per pixel three channel semi-planar 422	0x40000110	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_SAMPLING_422

Description	Value	Flags
16-bits per pixel three channel semi-planar 422 with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0113	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel semi-planar 420	0x40000210	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_SAMPLING_420
16-bits per pixel three channel semi-planar 420 with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0213	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_SAMPLING_420 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel semi-planar 422 wallpaper format	0x60000110	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_422
16-bits per pixel three channel semi-planar 422 wallpaper format with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x600n0113	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
10-bits per pixel three channel semi-planar 422 wallpaper format	0x60000111	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_BPS_10
8-bits per pixel three channel semi-planar 420 wallpaper format	0x60000210	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_420

Description	Value	Flags
16-bits per pixel three channel semi-planar 420 wallpaper format with $16 - n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x600n0213	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_420 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
10-bits per pixel three channel semi-planar 420 wallpaper format	0x60000211	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_SEMI_PLANAR PISP_IMAGE_FORMAT_WALLPAPER_ROLL PISP_IMAGE_FORMAT_SAMPLING_420 PISP_IMAGE_FORMAT_BPS_10
8-bits per pixel three channel planar	0x40000020	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR
16-bits per pixel three channel planar with $16 - n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0023	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel planar 422	0x40000120	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR PISP_IMAGE_FORMAT_SAMPLING_422
16-bits per pixel three channel planar 422 with $16 - n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0123	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR PISP_IMAGE_FORMAT_SAMPLING_422 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n
8-bits per pixel three channel planar 420	0x40000220	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR PISP_IMAGE_FORMAT_SAMPLING_420
16-bits per pixel three channel planar 420 with $16 - n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x400n0223	PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_ PLANARITY_PLANAR PISP_IMAGE_FORMAT_SAMPLING_420 PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n

Table 3: RGB image formats.

Additionally, the single channel formats listed below can be *output* by the Back End Output blocks (writing out only the first of the three channels), but it is not possible to read them in through the RGB Input block as they stand.

Description	Value	Flags
8-bits per pixel single channel	0x00000000	None
16-bits per pixel single channel with $16-n$ LSBs of dynamic range for $0 \leq n \leq 8$	0x000n0003	PISP_IMAGE_FORMAT_BPS_16 PISP_IMAGE_FORMAT_SHIFT_n

Table 4: Write-only RGB image formats.

We note however that the 16bpp single channel format could be read by the Bayer Input block with the entire Bayer pipe disabled. The 8bpp version could only be read by pretending it is a three channel planar image, reading it with the RGB Input block, and setting all three buffer addresses to the single actual image plane.

Finally, some versions of the Back End may support the following extra output formats, which cannot be read back into the PiSP at all.

Description	Value	Flags
32-bit per pixel single channel integral image	0x100n0000	PISP_IMAGE_FORMAT_INTEGRAL_IMAGE PISP_IMAGE_FORMAT_SHIFT_n
32-bit per pixel three channel integral image	0x500n0020	PISP_IMAGE_FORMAT_INTEGRAL_IMAGE PISP_IMAGE_FORMAT_THREE_CHANNEL PISP_IMAGE_FORMAT_PLANARITY_PLANAR PISP_IMAGE_FORMAT_SHIFT_n
Single channel HOG output with signed gradients	0x04000000	PISP_IMAGE_FORMAT_HOG_SIGNED
Single channel HOG output with unsigned gradients	0x08000000	PISP_IMAGE_FORMAT_HOG_UNSIGNED

Table 5: Additional image formats.

5.2 Image Format Flags

The table below lists the complete set of flags used in specifying image formats. For completeness we list the name of the features indicated by the absence of the flag (for example, if none of the "BPS" (bits per sample) flags is set, the pipeline will expect 8 bits per sample data).

Name	Value
PISP_IMAGE_FORMAT_BPS_8	-
PISP_IMAGE_FORMAT_BPS_10	0x00000001
PISP_IMAGE_FORMAT_BPS_12	0x00000002

Name	Value
PISP_IMAGE_FORMAT_BPS_16	0x00000003
PISP_IMAGE_FORMAT_PLANARITY_INTERLEAVED	-
PISP_IMAGE_FORMAT_PLANARITY_SEMI_PLANAR	0x00000010
PISP_IMAGE_FORMAT_PLANARITY_PLANAR	0x00000020
PISP_IMAGE_FORMAT_SAMPLING_444	-
PISP_IMAGE_FORMAT_SAMPLING_422	0x00000100
PISP_IMAGE_FORMAT_SAMPLING_420	0x00000200
PISP_IMAGE_FORMAT_ORDER_NORMAL	-
PISP_IMAGE_FORMAT_ORDER_SWAPPED	0x00001000
PISP_IMAGE_FORMAT_SHIFT_0	-
PISP_IMAGE_FORMAT_SHIFT_1	0x00010000
PISP_IMAGE_FORMAT_SHIFT_2	0x00020000
PISP_IMAGE_FORMAT_SHIFT_3	0x00030000
PISP_IMAGE_FORMAT_SHIFT_4	0x00040000
PISP_IMAGE_FORMAT_SHIFT_5	0x00050000
PISP_IMAGE_FORMAT_SHIFT_6	0x00060000
PISP_IMAGE_FORMAT_SHIFT_7	0x00070000
PISP_IMAGE_FORMAT_SHIFT_8	0x00080000
PISP_IMAGE_FORMAT_BPP_32	0x00100000
PISP_IMAGE_FORMAT_X_VALUE	0x00200000
PISP_IMAGE_FORMAT_UNCOMPRESSED	-
PISP_IMAGE_FORMAT_COMPRESSION_MODE_1	0x01000000
PISP_IMAGE_FORMAT_COMPRESSION_MODE_2	0x02000000
PISP_IMAGE_FORMAT_COMPRESSION_MODE_3	0x03000000
PISP_IMAGE_FORMAT_HOG_SIGNED	0x04000000
PISP_IMAGE_FORMAT_HOG_UNSIGNED	0x08000000
PISP_IMAGE_FORMAT_INTEGRAL_IMAGE	0x10000000
PISP_IMAGE_FORMAT_WALLPAPER_ROLL	0x20000000
PISP_IMAGE_FORMAT_THREE_CHANNEL	0x40000000

Table 6: Image format flags.

We repeat that not all flag combinations are meaningful; in fact, only the ones listed in previous tables are valid. In particular we point out the following (non-exhaustive list of) rules.

- The PiSP does not support 12-bit packed images so the PISP_IMAGE_FORMAT_BPS_12 value is unused.
- Compression is supported only with single channel images so PISP_IMAGE_FORMAT_COMPRESSION_MODE_n flags should not be specified for 3-channel images, and would be ignored.
- PISP_IMAGE_FORMAT_SHIFT_n flags are respected only for 16 bit per pixel images (i.e. PISP_IMAGE_FORMAT_BPS_16 set) and integral images (i.e. PISP_IMAGE_FORMAT_INTEGRAL_IMAGE set). Eight and ten bit output images are implicitly shifted by, respectively, 8 and 6 places. For integral images it will be usual to set PISP_IMAGE_FORMAT_SHIFT_8.
- PISP_IMAGE_FORMAT_ORDER_NORMAL and PISP_IMAGE_FORMAT_SWAPPED are significant only for interleaved 422 and RGB_32 formats.
- If PISP_IMAGE_FORMAT_WALLPAPER_ROLL is set then PISP_IMAGE_FORMAT_THREE_CHANNEL and PISP_IMAGE_FORMAT_SEMI_PLANAR must be set, and one of PISP_IMAGE_FORMAT_SAMPLING_422 or PISP_IMAGE_FORMAT_SAMPLING_420 must be set.
- If PISP_IMAGE_FORMAT_BPS_10 is set, then PISP_IMAGE_FORMAT_WALLPAPER_ROLL must also be set (10 bit per pixel formats are only supported as wallpaper rolls).
- If PISP_IMAGE_FORMAT_INTEGRAL_IMAGE is set, appropriate PISP_IMAGE_FORMAT_SHIFT_n bits should also be set. Where multi-channel, PISP_IMAGE_FORMAT_PLANARITY_PLANAR and PISP_IMAGE_FORMAT_THREE_CHANNEL should be set. Other flags must be clear.
- If one of PISP_IMAGE_FORMAT_HOG_SIGNED or PISP_IMAGE_FORMAT_HOG_UNSIGNED is set, no other bits should be set.
- If an image is interleaved (neither planar nor semi-planar), PISP_IMAGE_FORMAT_SAMPLING_420 may not be set.
- If PISP_IMAGE_FORMAT_BPP_32 is set, then PISP_IMAGE_FORMAT_THREE_CHANNEL must be set. PISP_IMAGE_FORMAT_ORDER_SWAPPED and PISP_IMAGE_FORMAT_X_VALUE are optional. Other flags must be clear.

Formats with the PISP_IMAGE_FORMAT_BPP_32 flag set are recognised by the Back End *only* when MINOR_VERSION ≥ 1.

5.3 Image Dimensions

Input images to the Front End and to the Bayer Input block of the Back End must have *even* image dimensions (width and height).

Input images to the RGB Input block of the Back End are also required to have even width. Further, if they have 420 sampling format they must also have even height.

Output images from the Front End must also have *even* image dimensions.

Output images from the end of the Back End are *not* required to have even dimensions, except for 422 sampling format images which must have even width, and 420 sampling format images which must have even height as well.

There is a further constraint that no single image *plane* (there being up to three planes in a Back End output image) may contain more than 2^{32} bytes.

5.4 Image Address Alignment

The base addresses of image buffers supplied to the PiSP *must* be at least 16-byte aligned. This includes all image buffers supplied as input or output to either the Front or Back Ends. Similarly, the *stride* of the image buffers, that is, the line-to-line separation in *bytes*, must always be a multiple of 16.

The Back End, however, works more efficiently when its input/output buffer addresses and strides are a multiple of 64 bytes. It is recommended, therefore, that the Front End is given 64-byte aligned addresses and strides for image buffers that are destined for the Back End.

Finally there is one exception to these rules which is that *wallpaper* format images *must* have both the address and stride aligned to 128 bytes. (For a wallpaper image, the stride is actually the distance from the top of one wallpaper roll to the next roll; the line-to-line distance is implicitly always 128 bytes.)

Generally, firmware would be expected to calculate appropriate stride values when they are not given explicitly.

There are also constraints on the alignments of the tiles into which the Back End breaks images for processing. These constraints are discussed in subsequent sections.

6 The PiSP Front End

6.1 Front End Overview

The Front End receives pixels either from a camera, or can read them from memory. It performs some minimal processing and gathers statistics. Finally it outputs up to two images to memory which may be cropped and scaled differently. Normally it both receives and writes Bayer images though it also supports non-Bayer purely greyscale images. When writing to or reading from memory the pixel data may optionally be compressed (using the scheme from section 4) to save memory bandwidth usage.

The Front End consists of the *Input Cluster*, the *Processing Cluster*, the *Statistics Cluster* and the *Output Cluster*, as shown in figure 5.

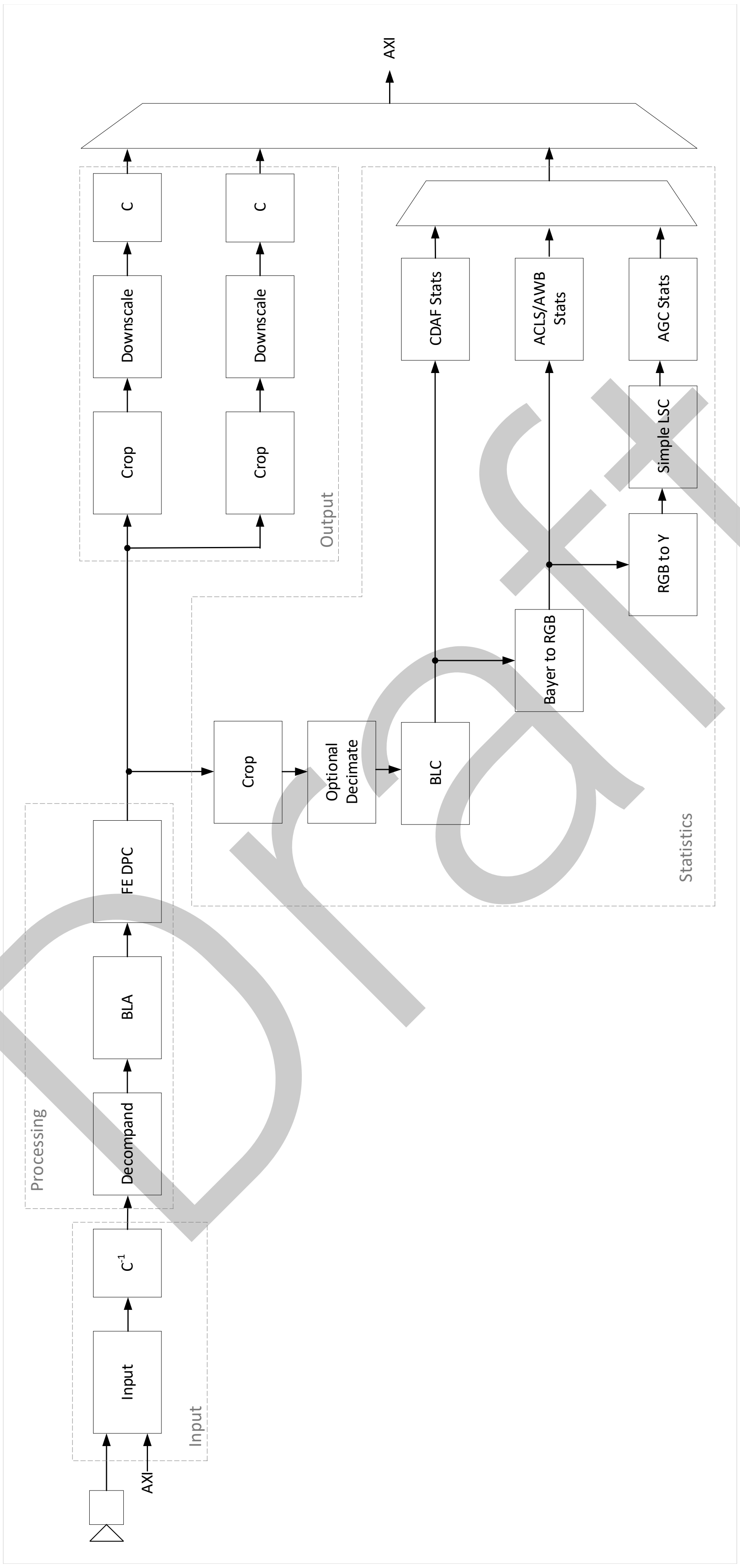

Figure 5: PiSP Front End clusters and blocks

Legend:

- C^{-1} - optional decompression.
- *Decompand* - undo any companding applied by the sensor to the pixel data.
- *BLA* - *Black Level Alignment* adjusts the black levels on all four Bayer channels to be the same, in case they are marginally different.
- *FE DPC* - a simple *Defective Pixel Correction* algorithm, tuned to require only limited context so that it can operate on full frames.
- *Crop* - crop to a window of interest in the image. Statistics and each output may be cropped independently.
- *Decimate* - optional decimation of the statistics window, by a factor of two in each axis. Only the lower-right 2×2 from each block of 4×4 pixels are used for statistics.
- *BLC* - *Black Level Correction* removes the black level from pixels in the *Statistics Cluster*. Pixels output by the *Output Cluster* do not have their black level removed.
- *CDAF Stats* - statistics for *Contrast Detect Auto Focus*.
- *Bayer to RGB* - converts each "quad" of 2×2 Bayer pixels to an RGB triple, for statistics gathering.
- *ACLS/AWB Stats* - statistics for AWB (*Auto White Balance*) and ACLS (*Auto Colour Lens Shading*) algorithms.
- *RGB to Y* - convert RGB to luminance, using configurable coefficients to take account of any digital gain and WBG (*White Balance Gains*).
- *Simple LSC* - a simple *Lens Shading Correction* algorithm, used only to correct luminance shading effects for AGC/AEC statistics gathering.
- *AGC Stats* - statistics for AGC/AEC (*Automatic Gain Control / Automatic Exposure Control*) algorithms.
- *Downscale* - downscale of Bayer or greyscale image data by a range of fixed factors.
- C - proprietary PiSP RAW image compression from section 4 (can be bypassed).

6.2 Programmer's Model

6.2.1 Registers and Interrupts

The Front End is controlled by writing to a number of 32-bit registers (where 64-bit addresses are required, they are supplied as a pair of 32-bit values). These registers are divided into three principal groups:

- *Control and Status Registers* which can be written to to start the PiSP Front End so that it will process a frame. The status registers can be read to determine what the Front End has done.
- *Input/Output Configuration Registers* (or just *I/O Registers*) which define where the PiSP Front End reads its pixels from, and where it writes its results to.
- *Processing Configuration Registers* (or just *Configuration Registers*) which control what processing the PiSP Front End performs on the pixels once it is running. These registers are further grouped according to the functional blocks shown in figure 5.

Additionally, interrupts are signalled by the Front End which the processor can respond to in order to cause the PiSP to process further frames. Four kinds of interrupts are generated:

- *Frame-Start Interrupts* occur when the PiSP Front End has started processing a new frame (or when a queued frame is aborted).
- *Frame-Done Interrupts* occur when the PiSP Front End has finished processing a frame, including writing all pixel and statistics data to memory (or when a frame is aborted).
- *Line Interrupts* are signalled when every n lines of the frame have been written, for programmable values of n. These interrupts may be configured independently for either output channel.
- *Statistics Ready Interrupts* which occur when all statistics have been written to memory.

6.2.2 Proccessing Queue

In order to reduce latency between processing frames, the PiSP Front End implements a 1-deep queue of requests to process new frames. This means that the *next* frame can be configured while the current frame is still being processed. Consequently all the configuration registers (both I/O and Processing configuration registers) have internal shadow registers. Every time the Front End prepares to process a new frame, all the values in the publicly addressable registers are latched internally and now cannot change for the duration of that frame. The *Processing Started* interrupt is signalled only after the values have been latched, *and* subsequently a new frame (sourced from a camera peripheral or from memory) begins. It is safe for firmware to overwrite the publicly addressable register values once this interrupt has been seen.

When the next frame to process has been configured and queued, if the Front End is currently idle it will become ready to process this frame immediately. If the Front End is currently busy this is delayed until the current frame is finished whereupon it will then become immediately ready for the new frame, requiring no intervention from software.

When receiving pixels from a camera, the Front End must clearly delay any activity until the frame starts to arrive. It is up to the firmware to program the Front End correctly prior to frames arriving from the camera. If the Front End is not properly configured for the new frame then that frame will be lost, and the Front End can only resume operation, if programmed to do so, with the next frame start. However, when reading pixels from memory, operations can begin immediately (subject to memory latency). Recall that the *Processing Started* interrupt is signalled only once all the publicly addressable registers have been copied *and* when the new frame start has subsequently been seen (which may involve a delay for frames from a camera, or will be immediate for frames read from memory).

Note that requests for the Front End to process new frames are also referred to as *jobs*.

6.2.3 Front End States

The Front End exists at all times in one of four states. These are:

1. IDLE - the Front End is completely idle and will not change state unless a new job is programmed into it.
2. PREPARING - when a new job is submitted the Front End immediately enters this state. It remains in this state until all the registers have been copied to their internal shadows.
3. WAITING - this state is entered spontaneously after the PREPARING state and indicates that the Front End is waiting for pixels to arrive (for example, from the camera).
4. ACTIVE - this state is entered spontaneously from either the WAITING or (if pixels are immeidately available) from the PREPARING state. The *Processing Started* interrupt is signalled upon entry to this state. It indicates that the Front End is actively processing pixels. When a frame is complete, the Front End will return to IDLE (if no further job was queued), or to PREPARING (if a job was queued), and the *Processing Done* interrupt will be signalled.

The diagram below illustrates the normal state transitions of the PiSP Front End. The dotted lines also show the immediate termination path caused by signalling the PiSP to *abort* processing.

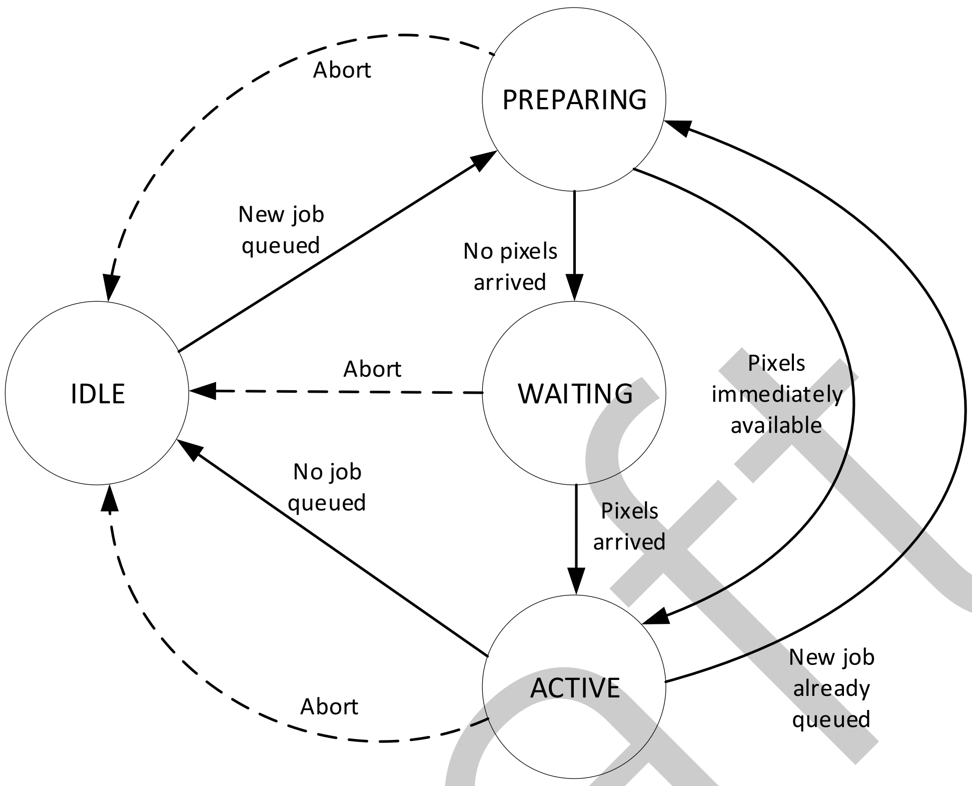

Figure 6: PiSP Front End state diagram

The *abort* paths return the PiSP to the IDLE state as soon as possible. Any pixel processing is abandonned and the active frame (if any) will not be completed. Moreover, any queued job will also be cancelled. However, if the PiSP is not IDLE an interrupt will still be raised indicating that any active frame, *plus any queued frame*, have fully completed (by the value of the *Done* count, which will also be the same as the *Started* count). During an abort, frame counters and interrupt flags are incremented atomically; firmware will not need to deal with any intermediate states.

6.2.4 The Front End Status

Besides a *status* register that returns the current state of the Front End (see section 6.2.3), the Front End also has a number of 32-bit registers which can be read (usually in response to an interrupt) to discover exactly what has been processed. These include:

- The 16-bit *Frame ID* of the frame most recently completed. This is the same frame ID as that obtained from the camera peripheral (or supplied by the caller for frames read from memory). Discontinuities in this count can be used to infer the number of dropped frames. This number is referred to as the the *DONE* count.
- The 16-bit *Frame ID* of the frame which the PiSP has most recently started processing (entered the ACTIVE state). Again this number is taken from the camera peripheral (or supplied by the caller for frames read from memory). This number is referred to as the *STARTED* count.
- There are also two sets of error indicator bits, one set for the most recently *done* frame and one for the previous frame (according to the counters above).

- Each of the Front End outputs has a line-count which records how many lines of output have been written for that branch. This line count always refers to the most recently started frame, according to the counters described above. So, for example, if the *STARTED* count has progressed onto the next frame, then we can deduce that the previous frame has been written out in its entirety, irrespective of the values in the line count registers.

For proper synchronisation it is nessary to be able to read all these status registers atomically, as if at the same instant in time. As the values will not fit into a single 32-bit register, the Front End provides a *LATCH* control bit that can be written to to force all the registers to be updated atomically from the internal hardware state. These status registers are not updated at any other time, except when a 1 is written to the *LATCH* control bit. Thereafter, software can read the register values without worrying about race conditions.

More information can be found in section 6.4.1.

6.2.5 Frames and Burstframes

We normally think of the PiSP Front End as operating on a single frame at a time. Under some circumstances, however, the Front End can operate on *several* frames at a time. This might be helpful, for example, when processing hundreds of frames per second. Such a group of frames is then referred to as a *burstframe*.

The use of burstframes significantly reduces the interrupt load on the system and correspondingly increases the time within which PiSP interrupts must be serviced. Burstframes work only when the Front End is running in *streaming* mode, and not when reading data from memory. When using burstframes:

- The Front End will ignore a specified number of frame ends and subsequent frame starts from the camera peripheral, so that several frames are received as a large single frame. The downstream parts of the Front End will treat this as a single tall frame.
- The Statistics Crop block should be configured to select the first (topmost) frame within the burstframe. Control algorithms will naturally run at the reduced burstframe rate.
- Vertical cropping should not be used on the outputs.
- If Bayer downscaling is used in the vertical axis, the denominator must be a factor of half the (real) frame height, and at least the topmost row of each output frame should be ignored. In practice, high frame rates may require cameras to run in binned mode, so downscaling might not be necessary.

The PiSP Back End can also process burstframes efficiently, by tiling the (real) frames within the burstframe individually and then submitting them all to the Back End as a single job.

6.3 Frame Size Limits

Unlike the Back End, the Front End normally operates on entire frames (or burstframes). There is a parameterized limit on the width of frames that can be processed with full functionality; in the first instantiation of PiSP the limit will be 6144.

When the source image is wider than 6144 columns:

- If FE DPC is enabled, it must be configured in horizontal-only mode.
- If statistics are used, one or both of the Statistics Crop and Statistics Decimate blocks must be enabled and configured to yield a width of 6144 columns or fewer.

- If Downscale 0 is enabled and the vertical ratio is not 1:1, its *output width* must not exceed 6144 (but the cropped image, and any unscaled outputs, may be wider).

Downscale 1 may have a different width limit, currently 4096:

- If Downscale 1 is enabled and the vertical ratio is not 1:1, its *output width* must not exceed 4096 (but the cropped image, and any unscaled outputs, may be wider).

The largest possible frame or burstframe size is 65520×65534. These functional limits are independent of performance considerations.

6.4 Register Definitions

Note that the entire register set is duplicated for each instance of the PiSP Front End. We detail first the *Control Registers*, followed by the *Input/Output Configuration Registers*, and finally all the remaining *Processing Configuration Registers* are listed block by block.

6.4.1 Control Registers

PISP_FE_VERSION **0x0000**

PiSP Front End version and capabilities. Read-only and constant.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:24	MAJOR_VERSION	Front End hardware major version number. This document describes version zero.	R	0
23:20	MINOR_VERSION	Counts minor changes other than to the parameters listed below. This document describes minor versions 0 and 1.	R	1
19:16	NUM_INPUT	Number of Streaming Input sources that may be connected to this Front End	R	1
15:12	MAXW_SCALE1	Width limit for Dowscale 1, in units of 1024 pixels. A value of zero means that there is no downscaler for output channel 1.	R	4
11:8	MAXW_SCALE0	Width limit for Downscale 0, in units of 1024 pixels.	R	6
7:4	MAXW_STATS	Width limit for statistics, in units of 1024 pixels.	R	6
3:0	MAXW_DPC	Width limit for FE DPC, in units of 1024 pixels. A value of zero means that FE DPC lacks vertical context.	R	6

Table 7: PISP_FE_VERSION register definition.

PISP_FE_CONTROL				**0x0004**
Control register for the PiSP Front End.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	-	Reserved.	-	-
3	LATCH	Writing a 1 to this bit causes the following status registers: PISP_-FE_FRAME_STATUS, PISP_FE_OUTPUT_STATUS and PISP_FE_-ERROR_STATUS to be latched atomically from the internal hardware state. Immediately after writing a 1 here, the listed registers can be read by software without the risk of them changing again (until another 1 is written to this bit).	W	0
2	RESET	Sets the PISP_FE_FRAME_STATUS, PISP_FE_OUTPUT_STATUS and PISP_FE_ERROR_STATUS registers to zero. This bit should only be written to when the PiSP is IDLE.	W	0
1	ABORT	Write a 1 to abort current PiSP Front End processing. The frame currently being processed (if any) and any queued frame are abandoned as quickly as possible. If the Front End was not IDLE, a *Processing Done* interrupt will be raised (*Statistics Ready* and *Lines* interrupt status will be undefined).	W	0
0	QUEUE	Write a 1 to queue the new job that has been programmed into the configuration registers.	W	0

Table 8: PISP_FE_CONTROL register definition.

PISP_FE_STATUS				**0x0008**
Status register for the PiSP Front End. This register returns zero when the PiSP Front End is in its IDLE state. (See also PISP_FE_DEBUG_STATUS2.)				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:3	-	Reserved.	-	-
2	ACTIVE	Reads as 1 when the Front End is actively processing pixels in a frame (is in the ACTIVE state).	R	0
1	WAITING	Reads as 1 when the Front End is ready to process a frame but is waiting for the frame to arrive (is in the WAITING state).	R	0
0	QUEUED	Reads as 1 when a job for the Front End is currently queued. If this is the only bit set, the Front End is PREPARING. It may also read as 1 during the WAITING or ACTIVE states if firmware has programmed a new job since the current one started. When it reads zero, it is safe to overwrite the configuration registers.	R	0

Table 9: PISP_FE_STATUS register definition.

PISP_FE_FRAME_STATUS				**0x000C**
Frame IDs of the frame the Front End has most recently started processing, and the frame which it has most recently completed. These frame IDs match the values supplied either by the camera peripheral or, if the frames are being read from memory, in the PISP_FE_IO_INPUT_ID register at the time the frame was begun. This value is updated only when a 1 is written to the LATCH bit of the PISP_FE_CONTROL register.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	DONE	Frame ID of the frame most recently completed.	R	0
15:0	STARTED	Frame ID of the frame most recently started.	R	0

Table 10: PISP_FE_FRAME_STATUS register definition.

PISP_FE_ERROR_STATUS				**0x0010**
Error status for the frame most recently completed (according to the DONE field of PISP_FE_FRAME_STATUS), and the preceding frame. Also, statistics output status. [P_]FULL and [P_]PANIC flags are defined only for MINOR_VERSION $\geq$ 1. This register is updated when a 1 is written to the LATCH bit of PISP_FE_CONTROL.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:17	-	Reserved.	-	-
16	STATS_READY	Reads as '1' when statistics have been written to memory for the frame most recently *started*. Not an error.	R	0
15	-	Reserved.	-	-
14	P_FULL	Value of FULL for the frame immediately before the one most recently completed.	R	0
13	P_PANIC	Value of PANIC for the frame immediately before the one most recently completed.	R	0
12	P_ABORTED	Value of ABORTED for the frame immediately before the one most recently completed.	R	0
11	P_DECERR	Value of DECERR for the frame immediately before the one most recently completed.	R	0
10	P_SLVERR	Value of SLVERR for the frame immediately before the one most recently completed.	R	0
9	P_HARD_ERR	Value of HARD_ERR for the frame immediately before the one most recently completed.	R	0
8	P_SOFT_ERR	Value of SOFT_ERR for the frame immediately before the one most recently completed.	R	0
7	-	Reserved.	-	-

Continued on next page...

PISP_FE_ERROR_STATUS (continued)				**0x0010**
Bits	***Name***	***Description***	***R/W***	***Reset***
6	FULL	Reads as '1' if streaming input FIFO was ever full during the frame most recently completed. Not an error if streaming source tolerates this. Defined only when MINOR_VERSION ≥ 1.	R	0
5	PANIC	Reads as '1' if the output FIFO reached a threshold during the frame most recently completed. Not necessarily an error. Defined only when MINOR_VERSION ≥ 1.	R	0
4	ABORTED	Reads as '1' if the last frame was aborted while preparing, waiting or active.	R	0
3	DECERR	Reads as '1' if there was an AXI Decode Error while writing out the frame most recently completed. This generally indicates an illegal output address.	R	0
2	SLVERR	Reads as '1' if there was an AXI Slave Error while writing out the frame most recently completed. This generally indicates an unsuitable output AXI burst configuration.	R	0
1	HARD_ERR	Reads as '1' if a hard (unrecoverable) error was signalled by a streaming source, or an AXI Decode Error for a memory source, for the frame most recently completed.	R	0
0	SOFT_ERR	Reads as '1' if a soft (recoverable) error was signalled by a streaming source, or an AXI Slave Error for a memory source, for the frame most recently completed.	R	0

Table 11: PISP_FE_ERROR_STATUS register definition.

PISP_FE_OUTPUT_STATUS **0x0014**

Status register for the Front End output branches indicating how many image lines have been completed. The number always refers to the most recently started frame, according to the PISP_FE_FRAME_STATUS register. This value is updated only when a 1 is written to the LATCH bit of the PISP_FE_CONTROL register.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LINES1	Number of image lines which have been written out for output branch 1.	R	0
15:0	LINES0	Number of image lines which have been written out for output branch 0.	R	0

Table 12: PISP_FE_OUTPUT_STATUS register definition.

PISP_FE_INTERRUPT_EN				**0x0018**
Interrupt enable register.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:25	-	Reserved.	-	-
24	QREADY	Set to 1 to enable *Queue Ready* interrupts, valid when MINOR_-VERSION ≥ 1 only.	RW	0
23:17	-	Reserved.	-	-
16	STATS	Set to 1 to enable *Statistics Ready* interrupts.	RW	0
15:10	-	Reserved.	-	-
9	LINES1	Set to 1 to enable *Lines* interrupts for output 1.	RW	0
8	LINES0	Set to 1 to enable *Lines* interrupts for output 0.	RW	0
7:2	-	Reserved.	-	-
1	SOF	Set to 1 to enable *Processing Started* interrupts.	RW	0
0	EOF	Set to 1 to enable *Processing Done* interrupts.	RW	0

Table 13: PISP_FE_INTERRUPT_EN register definition.

PISP_FE_INTERRUPT_STATUS				**0x001C**
Interrupt status register. Writing a 1 to a particular bit will clear the corresponding interrupt. Writing 0 to any bit, even reserved ones, has no effect. Interrupt events are recorded here even when disabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:25	-	Reserved.	-	-
24	QREADY	Read or clear *Queue Ready* interrupts, defined when MINOR_-VERSION ≥ 1 only.	RW	0
23:17	-	Reserved.	-	-
16	STATS	Read or clear *Statistics Ready* interrupt.	RW	0
15:10	-	Reserved.	-	-
9	LINES1	Read or clear output 1 *Lines* interrupt.	RW	0
8	LINES0	Read or clear output 0 *Lines* interrupt.	RW	0
7:2	-	Reserved.	-	-
1	SOF	Read or clear *Processing Started* interrupt.	RW	0
0	EOF	Read or clear *Processing Done* interrupt.	RW	0

Table 14: PISP_FE_INTERRUPT_STATUS register definition.

6.4.2 Debug Registers

These read-only registers are for debug use only. Their contents may change whenever the Front End is not idle. Do not write to them.

PISP_FE_DEBUG_CHECKSUM				**0x0020**
Fletcher checksum of 16-bit input values (after shift and before decompression) for the most recently completed frame.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	C1	Sum of sums mod 65535.	R	0
15:0	C0	Sum of pixel values mod 65535.	R	0

Table 15: PISP_FE_DEBUG_CHECKSUM register definition.

PISP_FE_DEBUG_FIFO_FULLNESS				**0x0024**
Fullness of streaming input FIFO (for all versions). When MINOR_VERSION ≥ 1, it also reports Output FIFO fullness.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	FULLNESS_OUT	Output FIFO fullness, in 16-byte beats.	R	0
15:0	FULLNESS_IN	Input FIFO fullness, in pixel-pairs.	R	0

Table 16: PISP_FE_DEBUG_FIFO_FULLNESS register definition.

PISP_FE_DEBUG_FIFO_DEPTHS				**0x0028**
Reports depths of Input and Output FIFOs. Constant. This register exists only when MINOR_VERSION ≥ 1. (In minor version zero, Input FIFO depth is always 1023.)				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	DEPTH_OUT	Output FIFO depth, in 16-byte beats.	R	511
15:0	DEPTH_IN	Input FIFO depth, in pixel-pairs.	R	1023

Table 17: PISP_FE_DEBUG_FIFO_DEPTHS register definition.

PISP_FE_DEBUG_STATUS2				**0x002C**
Additional status information. This register exists only when MINOR_VERSION $\geq$ 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	DONE	Frame ID of the frame most recently completed. Unlike in FRAME_STATUS, it is not latched.	R	0
15:8	SEQ	Count of frames completed or aborted, modulo 256.	R	0
7:3	-	Reserved.	-	-
2	ACTIVE	Reads as 1 when the Front End is actively processing pixels in a frame (is in the ACTIVE state).	R	0
1	WAITING	Reads as 1 when the Front End is ready to process a frame but is waiting for the frame to arrive (is in the WAITING state).	R	0
0	QUEUED	Reads as 1 when a job for the Front End is currently queued. If this is the only bit set, the Front End is PREPARING. It may also read as 1 during the WAITING or ACTIVE states if firmware has programmed a new job since the current one started.	R	0

Table 18: PISP_FE_DEBUG_STATUS2 register definition.

6.4.3 Input/Output Configuration Registers

These registers define the memory addresses where the PiSP Front End writes output pixels and statistics and reads pixels from (when not in streaming mode). They are typically updated by firmware on every frame.

Addresses in the PiSP have 64 bits, but depending on the target chip some high bits may be ignored. The address from which to read pixel data in memory must be aligned to a multiple of 4 bytes (8 bytes if compressed), and for writing the address must be a multiple of 16 bytes. The address for writing out statistics data must also be 16-byte aligned.

PISP_FE_IO_STATS_ADDR_LO				**0x0040**
Low 32 bits of the output address for statistics. Ignored if no statistics are enabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 19: PISP_FE_IO_STATS_ADDR_LO register definition.

PISP_FE_IO_STATS_ADDR_HI				**0x0044**
High 32 bits of the output address for statistics. Ignored if no statistics are enabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADRESSS_HI	High 32 bits of output address.	RW	-

Table 20: PISP_FE_IO_STATS_ADDR_HI register definition.

PISP_FE_IO_OUTPUT0_ADDR_LO				**0x0048**
Low 32 bits of the memory address to which Output block 0 writes pixel data. Ignored if Output 0 is not enabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 21: PISP_FE_IO_OUTPUT0_ADDR_LO register definition.

PISP_FE_IO_OUTPUT0_ADDR_HI				**0x004C**
High 32 bits of the memory address to which Output block 0 writes pixel data. Ignored if Output 0 is not enabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 22: PISP_FE_IO_OUTPUT0_ADDR_HI register definition.

PISP_FE_IO_OUTPUT1_ADDR_LO				**0x0050**
Low 32 bits of the memory address to which Output block 1 writes pixel data. Ignored if Output 1 is not enabled.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 23: PISP_FE_IO_OUTPUT1_ADDR_LO register definition.

PISP_FE_IO_OUTPUT1_ADDR_HI **0x0054**

High 32 bits of the memory address to which Output block 1 writes pixel data. Ignored if Output 1 is not enabled.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 24: PISP_FE_IO_OUTPUT1_ADDR_HI register definition.

PISP_FE_IO_INPUT_ADDR_LO **0x0058**

Low 32 bits of the memory address from which the Input block reads pixel data. Ignored if the Input block is receiving pixels from the camera peripheral.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_LO	Low 32 bits of the memory address, of which the two least significant bits must be zero.	RW	0

Table 25: PISP_FE_IO_INPUT_ADDR_LO register definition.

PISP_FE_IO_INPUT_ADDR_HI **0x005C**

High 32 bits of the memory address from which the Input block reads pixel data. Ignored if the Input block is receiving pixels from the camera peripheral.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 26: PISP_FE_IO_INPUT_ADDR_HI register definition.

PISP_FE_IO_INPUT_ID **0x0060**

16-bit frame ID for the frame that is to be processed. Normally this is expected to be a simple 16-bit counter. Ignored if the Input block is receiving pixels from the camera peripheral.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved.	-	-
15:0	ID	16-bit Frame ID.	RW	0

Table 27: PISP_FE_IO_INPUT_ID register definition.

6.4.4 Global Configuration Registers

These registers affect the global behaviour of the Front End pipeline.

Each Front End block can be enabled or disabled. Typically, a disabled block passes its input unmodified to its output if they are in the same format, otherwise a default conversion is performed. The configuration registers for a disabled block remain accessible, but have no effect until the block is re-enabled.

PISP_FE_GLOBAL_ENABLE **0x0064**

Register containing enable bits for blocks in the Front End pipeline.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	-	Reserved.	-	-
23	OUTPUT1	Enable Output on output branch 1.	RW	0
22	COMPRESS1	Enable Compression on output branch 1.	RW	0
21	DOWNSCALE1	Enable Downscale on output branch 1.	RW	0
20	CROP1	Enable Crop on output branch 1.	RW	0
19	OUTPUT0	Enable Output on output branch 0.	RW	0
18	COMPRESS0	Enable Compression on output branch 0.	RW	0
17	DOWNSCALE0	Enable Downscale on output branch 0.	RW	0
16	CROP0	Enable Crop on output branch 0.	RW	0
15:13	-	Reserved.	-	-
12	AGC_STATS	Enable AGC/AEC Statistics.	RW	0
11	LSC	Enable LSC (Lens Shading Correction).	RW	0
10	RGBY	Enable RGB to Y conversion.	RW	0
9	AWB_STATS	Enable AWB Statistics.	RW	0
8	CDAF_STATS	Enable CDAF Statistics.	RW	0
7	BLC	Enable BLC (Black Level Correction).	RW	0
6	DECIMATE	Enable Statistics Decimation.	RW	0
5	STATS_CROP	Enable Statistics Cropping.	RW	0
4	DPC	Enable DPC (Defective Pixel Correction).	RW	0
3	BLA	Enable BLA (Black Level Alignment).	RW	0
2	DECOMPAND	Enable Decompand.	RW	0
1	DECOMPRESS	Enable Decompression.	RW	0
0	INPUT	Must be '1' for normal operation.	RW	0

Table 28: PISP_FE_GLOBAL_ENABLE register definition.

PISP_FE_GLOBAL_BAYER_ORDER				**0x0068**
Register defining the Bayer Order used by all blocks in the Front End pipeline.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1:0	ORDER	Bayer order, either 0x0 (RGGB), 0x1 (GBRG), 0x2 (BGGR) or 0x3 (GRBG).	RW	0

Table 29: PISP_FE_GLOBAL_BAYER_ORDER register definition.

6.4.5 Input Configuration Registers

The Front End Input block accepts only three formats of pixel.

- 16-bit pixel values from the streaming interface. These pixels may optionally be left-shifted by up to 8 bits.
- 16-bit pixel values from memory. The pixels are read in little-endian format and may optionally be left shifted by up to 8 bits.
- Compressed format from memory. This is the PiSP proprietary RAW compression format at 8 bits per pixel. These values are simply passed downstream to the Decompression block.

Its line-to-line stride (pitch) must be a non-negative multiple of 16 bytes. When reading from memory the address of the image is given by the PISP_FE_IO_INPUT_ADDR_LO and PISP_FE_IO_INPUT_-ADDR_HI registers.

The registers below define the behaviour of the block.

PISP_FE_INPUT_SOURCE				**0x006C**
Register defining whether pixels are received from the streaming input (camera peripheral) or read from memory.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved.	-	-
15:8	BURST	This value indicates how many frame ends are to be ignored when receiving streaming input. If the value is zero, the block outputs a frame for every frame received. When non-zero, it will output $BURST+1$ input frames as a single burstframe.	RW	0
7:1	-	Reserved.	-	-
0	STREAMING	Set to 1 for streaming input, or 0 to read from memory.	RW	0

Table 30: PISP_FE_INPUT_SOURCE register definition.

PISP_FE_INPUT_SIZE				**0x0070**
Dimensions of the input image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being received.	RW	0
15:0	WIDTH	Width of the image being received.	RW	0

Table 31: PISP_FE_INPUT_SIZE register definition.

PISP_FE_INPUT_FORMAT				**0x0074**
Format definition of the input image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the input is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are left-shifted by this amount, for $0 \leq SHIFT \leq 8$, when COMPRESSION_MODE is zero.	RW	0
15:2	-	Ignored.	-	-
1:0	BITS_PER_SAMPLE	Number of bits in each sample (pixel). Valid settings are: 0 (8 bps) if COMPRESSION_MODE is nonzero; otherwise 3 (16 bps).	RW	0

Table 32: PISP_FE_INPUT_FORMAT register definition.

PISP_FE_INPUT_STRIDE				**0x0078**
Line to line stride of pixel data in memory. Ignored if STREAMING is set to 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 33: PISP_FE_INPUT_STRIDE register definition.

PISP_FE_INPUT_STRIDE2				0x007C
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 34: PISP_FE_INPUT_STRIDE2 register definition.

PISP_FE_INPUT_AXI				0x0080
Configuration for AXI read bursts. Ignored if STREAMING is set to 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:20	-	Reserved.	-	
19:16	QOS	Value for ARQOS field, where used.	RW	0
15	-	Reserved.	-	
14:12	PROT	Value for ARPROT field, where used.	RW	0
11:8	CACHE	Value for ARCACHE field, where used.	RW	0
7	ALIGN	When MAXLEN+1 is a power of two but address is not so aligned, truncate burst at a 16*(MAXLEN+1) byte boundary, to align following bursts.	RW	0
6:4	-	Reserved.	-	
3:0	MAXLEN	Maximum AXI burst length minus one.	RW	0

Table 35: PISP_FE_INPUT_AXI register definition.

PISP_FE_INPUT_HOLDOFF				0x0084
Configuration for input rate control. Ignored if STREAMING is set to 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:8	-	Reserved.	-	-
7:0	HOLDOFF	Extra cycles to delay before each AXI read burst.	RW	0

Table 36: PISP_FE_INPUT_HOLDOFF register definition.

The diagram below illustrates how the Input block operates and how it is controlled by the registers defined above.

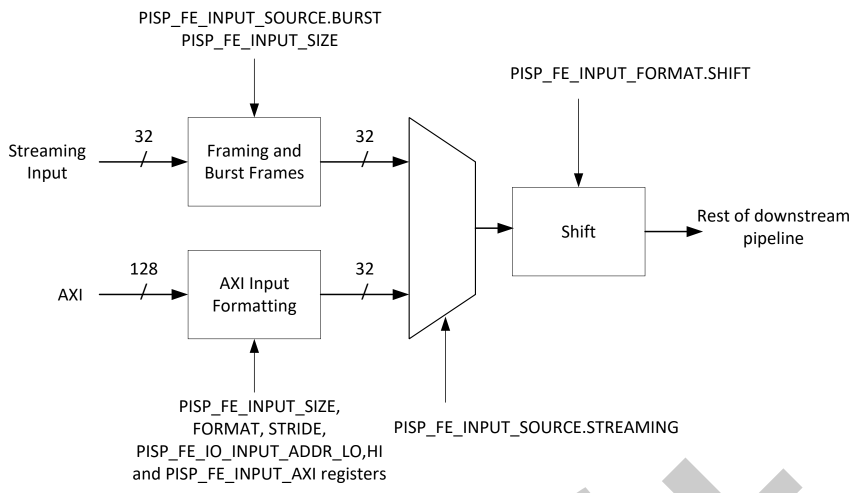
PISP_FE_INPUT_SOURCE.BURST
PISP_FE_INPUT_SIZE
PISP_FE_INPUT_FORMAT.SHIFT
Streaming Input
32
Framing and Burst Frames
32
Shift
Rest of downstream pipeline
AXI
128
AXI Input Formatting
32
PISP_FE_INPUT_SIZE, FORMAT, STRIDE, PISP_FE_IO_INPUT_ADDR_LO,HI and PISP_FE_INPUT_AXI registers
PISP_FE_INPUT_SOURCE.STREAMING

Figure 7: PiSP Front End Input block

6.4.6 Decompression Configuration Registers

This block should be enabled only if compressed image data is being read from memory by the Input block (so that block's COMPRESSED bit should be set).

PISP_FE_DECOMPRESS				**0x0088**
Parameters for decompression of PiSP compressed RAW image format data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select decompression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset value added to all pixels after decompression. Normally used to restore black level if compression was configured to subtract it.	RW	0

Table 37: PISP_FE_DECOMPRESS register definition.

6.4.7 Decompand Configuration Registers

When enabled, this block applies a piecewise linear function to pixel values; either to implement some discrete companding scheme or to approximate a smooth curve (such as a power law).

There are 64 linear segments of fixed, equal width; defined by 65 look-up table entries packed into 33 registers. Each LUT entry may differ from the previous one by 0:32767 (FSD/2). The final entry is specified modulo 65536.

PISP_FE_DECOMPAND_LUT[n]				**0x008C$+4n$**
Look-up table entries $[2n]$ and $[2n+1]$ for $n = 0, ..., 31$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LUT1	Output value for input $1024 \times (2n+1)$	RW	0
15:0	LUT0	Output value for input $1024 \times 2n$	RW	0

Table 38: PISP_FE_DECOMPAND_LUT[n] register definition.

PISP_FE_DECOMPAND_LUT[32]				**0x010C**
Final look-up table entry for decompand.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved.	-	-
15:0	LUT0	Limiting output value, modulo 65536	RW	0

Table 39: PISP_FE_DECOMPAND_LUT[32] register definition.

6.4.8 BLA (Black Level Alignment) Configuration Registers

The BLA block aligns the black levels of all the Bayer channels so that they are the same value (in case they were slightly different). For each channel it subtracts the "current black level" and adds the "output black level", before clipping the output to 0..65535.

PISP_FE_BLA_INPUT_RED				**0x0110**
Input black levels for alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	GR	Current black level of green pixels on green-red rows.	RW	0
15:0	R	Current black level of red pixels.	RW	0

Table 40: PISP_FE_BLA_INPUT_RED register definition.

PISP_FE_BLA_INPUT_BLUE				**0x0114**
Input black levels for alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	B	Current black level of blue pixels.	RW	0
15:0	GB	Current black level of green pixels on green-blue rows.	RW	0

Table 41: PISP_FE_BLA_INPUT_BLUE register definition.

PISP_FE_BLA_OUTPUT				**0x0118**
Output black level after alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved.	RW	0
15:0	LEVEL	Output black level of all pixels after alignment.	RW	0

Table 42: PISP_FE_BLA_OUTPUT register definition.

6.4.9 DPC (Defective Pixel Correction) Configuration Registers

The Front End contains a DPC block, to protect statistics and downscalers from the worst effects of defective pixels. Front End DPC is slightly less effective than Back End DPC, as it buffers fewer lines of the image. It finds the MIN..MAX range of values of 4 or 7 nearby pixels in the same Bayer channel, and extends it by a “margin” which represents the largest acceptable excursion away from neighbouring values. The margin is a function of the MIN and MAX−MIN of the neighbouring pixels (from the same Bayer channel) and optionally the MAX−MIN of 4 pixels from the *other* Bayer channel in the same row. The output is clipped to lie within this range.

PISP_FE_DPC				**0x011C**
Configuration register for Front End DPC block.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	
25	VFLAG	Normally set to one. When zero, DPC ignores pixels from other scanlines. This allows it to work (though less effectively) on images wider than 6144.	RW	0
24	FOLDBACK	Causes pixels well outside the acceptable excursion to be corrected more aggressively.	RW	0
23:21	-	Reserved.	-	-
20:16	COEFF_RANGE2	Coefficient for the range of pixels from the other Bayer channel, used to compute the margin. All coefficients are in $u1.4$ format.	RW	0
15:13	-	Reserved.	-	-
12:8	COEFF_RANGE	Coefficient for the range of pixels from this Bayer channel, used to compute the margin.	RW	0
7:5	-	Reserved.	-	-
4:0	COEFF_LEVEL	Coefficient for the darkest neighbouring pixel value, used to compute the margin.	RW	0

Table 43: PISP_FE_DPC register definition.

6.4.10 Statistics Crop Configuration Registers

A portion of the image can optionally be cropped for the statistics pipeline. This can be helpful if the image being processed is wider than the FE maximum-width parameter (currently 6144).

Only even offsets and dimensions can be specified; the bottom bit of each field will be ignored and should be written as zero. When crop is enabled, WIDTH and HEIGHT must both be at least 2.

PISP_FE_STATS_CROP_OFFSET **0x0120**

Offset of the crop window.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	OFFSET_Y	Y offset of the crop window. Must be even.	RW	0
15:0	OFFSET_X	X offset of the crop window. Must be even.	RW	0

Table 44: PISP_FE_STATS_CROP_OFFSET register definition.

PISP_FE_STATS_CROP_SIZE **0x0124**

Size of the crop window.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	HEIGHT	Height of the crop window. Must be even.	RW	0
15:0	WIDTH	Width of the crop window. Must be even.	RW	0

Table 45: PISP_FE_STATS_CROP_SIZE register definition.

Note that when the Statistics Decimate block is enabled and the source dimensions are not both multiples of 4, Statistics Crop must be enabled. When Statistics Crop and Statistics Decimate are both enabled, the cropped width and height must be multiples of 4.

6.4.11 Statistics Decimation Configuration Registers

The current version of the Statistics Decimation block is not configurable; when enabled it decimates by a factor of 2 in each axis.

PISP_FE_SPARE1 **0x0128**

This register is ignored.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	-	Reserved.	-	-

Table 46: PISP_FE_SPARE1 register definition.

6.4.12 BLC (Black Level Correction) Configuration Registers

Functionally this block is identical to the BLA block. However, it is situated at the start of the Statistics Cluster, therefore allowing the full black level to be subtracted before collecting statistics. All the input black levels would normally be programmed to the same level as the output level of the BLA block, and the output level here should be set to zero.

PISP_FE_BLC_INPUT_RED **0x012C**

Input black levels for alignment.

Bits	Name	Description	R/W	Reset
31:16	GR	Current black level of green pixels on green-red rows.	RW	0
15:0	R	Current black level of red pixels.	RW	0

Table 47: PISP_FE_BLC_INPUT_RED register definition.

PISP_FE_BLC_INPUT_BLUE **0x0130**

Input black levels for alignment.

Bits	Name	Description	R/W	Reset
31:16	B	Current black level of blue pixels.	RW	0
15:0	GB	Current black level of green pixels on green-blue rows.	RW	0

Table 48: PISP_FE_BLC_INPUT_BLUE register definition.

PISP_FE_BLC_OUTPUT **0x0134**

Output black level after alignment.

Bits	Name	Description	R/W	Reset
31:16	-	Reserved.	RW	0
15:0	LEVEL	Output black level of all pixels after alignment.	RW	0

Table 49: PISP_FE_BLC_OUTPUT register definition.

6.4.13 RGBY (RGB to Luminance) Configuration Registers

This block forms either a sum or MAX of weighted R, G and B samples from each 2×2 "quad", to form a Luminance or Value image for AGC Statistics only. The lower of the two Green phases is used. Coefficients are all in $u4.10$ format; they should be configured to anticipate any white balance adjustments, digital gains or colourspace conversions applied by the Back End.

PISP_FE_RGBY_GAINS_RG **0x0138**

Red and green gains.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:30	-	Reserved.	-	-
29:16	GREEN	Coefficient for green pixels.	RW	0
15:14	-	Reserved.	-	-
13:0	RED	Coefficient for red pixels.	RW	0

Table 50: PISP_FE_RGBY_GAINS_RG register definition.

PISP_FE_RGBY_GAIN_B **0x013C**

Blue gain and MAX flag.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:17	-	Reserved.	-	-
16	MAXFLAG	Compute MAX (Value) instead of sum (Luma).	RW	0
15:14	-	Reserved.	-	-
13:0	BLUE	Coefficient for blue pixels.	RW	0

Table 51: PISP_FE_RGBY_GAIN_B register definition.

Note: When AGC Statistics are enabled but RGBY is not, these registers are ignored but conversion is still performed; R, G, B will be summed using default coefficients (0.25, 0.5, 0.25).

6.4.14 LSC (Lens Shading Correction) Configuration Registers

The Front End LSC block is used only to correct pixels prior to the AGC Statistics block. Its purpose is to apply vignetting correction so that the outer edges of the image are not overly dark, which would otherwise skew the AGC/AEC algorthm into undervaluing those regions.

For this purpose it is sufficient to apply a radial function to a single-component (luminance) image. The gain is defined by a 15-segment PWL function of r^2 where r is the distance in pixels from a central point.

PISP_FE_LSC_SCALE				**0x0140**
Parameters for converting r^2 values into the correct range for the PWL function.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:16	SCALE	Fractional multiplier used to scale shifted r^2 values into the range 0–960.	RW	0
15:5	-	Reserved.	-	-
4:0	SHIFT	Amount by which r^2 values are shifted right. Maximum $r^2/2^{SHIFT}$ should be not more than 4095.	RW	0

Table 52: PISP_FE_LSC_SCALE register definition.

PISP_FE_LSC_CENTRE				**0x0144**
Central point for the Lens Shading function, at full resolution (or half resolution if the Statistics Decimate block is enabled), relative to the start of the Statistics Crop region. It should be not more than 46340 pel from any image corner.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	CENTRE_Y	Central y coordinate. Must be even.	RW	0
15:0	CENTRE_X	Central x coordinate. Must be even.	RW	0

Table 53: PISP_FE_LSC_CENTRE register definition.

PISP_FE_LSC_GAINS[n]				**0x0148+$4n$**
Gains for the LSC radial correction function, for $n = 0, ..., 7$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	GAIN1	Gain number $2n + 1$ for the radial function LUT.	RW	0
15:0	GAIN0	Gain number $2n$ for the radial function LUT.	RW	0

Table 54: PISP_FE_LSC_GAINS[n] register definition.

6.4.15 AGC Statistics Configuration Registers

The AGC Statistics block gather statistics for the AGC/AEC algorithm. This consists of a 1024-bin histogram of luminance values, and also some row sums for the detection of flicker caused by artifical lighting. Additionally, luminance sums are collected within the floating regions.

The window in which the histogram is measured is divided into a 16x16 grid of evenly sized rectangles, and 4-bit weights can be attached to these so that they have more influence on the output histogram. Figure 8 illustrates the layout of the grid with respect to its *offset* and *size* parameters.

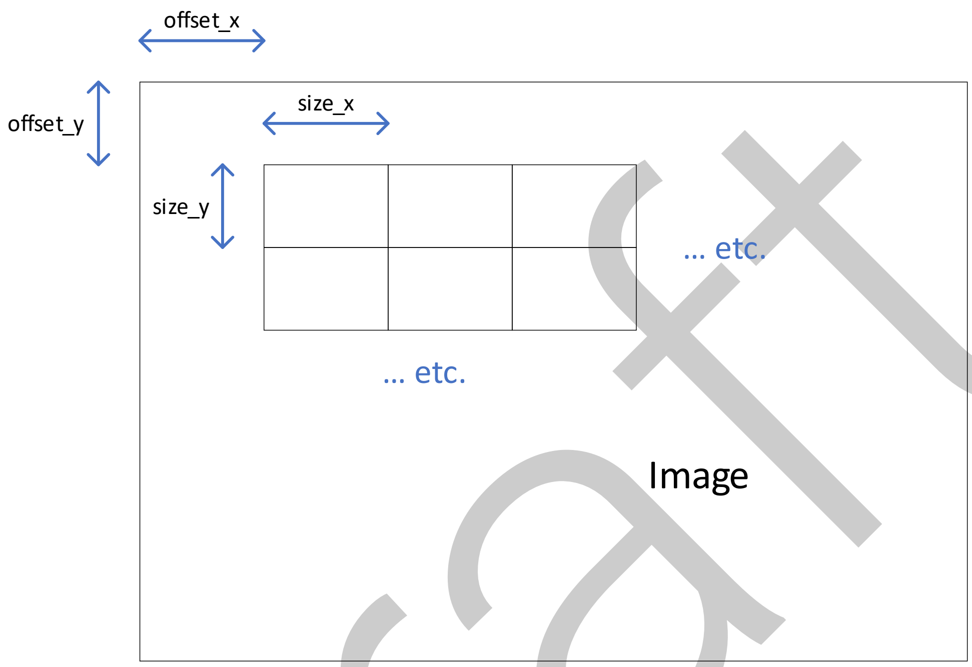

Figure 8: Grid layout for AGC statistics (not all grid rectangles shown)

Only one luminance value is formed for each 2×2 "quad" of the Bayer image, but for convenience these offsets and sizes are expressed in terms of the full-resolution image (or half resolution if the Statistics Decimate block is enabled). All dimensions must be even. The bottom bit of each offset or size field will be ignored and should be written as zero.

PISP_FE_AGC_OFFSET **0x0168**

Offset of the window within which the histogram is measured.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset of the top of the region within which the histogram is measured. Must be even.	RW	0
15:0	OFFSET_X	X offset of the left edge of the region within which the histogram is measured. Must be even.	RW	0

Table 55: PISP_FE_AGC_OFFSET register definition.

PISP_FE_AGC_SIZE				**0x016C**
Width and height of the window within which the histogram is measured.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SIZE_Y	Height of each of the 16x16 regions defining the weights. Must be even. The height of the window within which the histogram is measured is 16*SIZE_Y.	RW	0
15:0	SIZE_X	Width of each of the 16x16 regions defining the weights. Must be even. The width of the window within which the histogram is measured is 16*SIZE_X.	RW	0

Table 56: PISP_FE_AGC_SIZE register definition.

PISP_FE_AGC_WEIGHTS[n]				**0x0170$+4n$**
Weights for AGC regions, for $n = 0, ..., 31$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	WEIGHT7	Weight for region $8n + 7$.	RW	0
27:24	WEIGHT6	Weight for region $8n + 6$.	RW	0
23:20	WEIGHT5	Weight for region $8n + 5$.	RW	0
19:16	WEIGHT4	Weight for region $8n + 4$.	RW	0
15:12	WEIGHT3	Weight for region $8n + 3$.	RW	0
11:8	WEIGHT2	Weight for region $8n + 2$.	RW	0
7:4	WEIGHT1	Weight for region $8n + 1$.	RW	0
3:0	WEIGHT0	Weight for region $8n$.	RW	0

Table 57: PISP_FE_AGC_WEIGHTS[n] register definition.

PISP_FE_AGC_ROW_OFFSET				**0x01F0**
Image offset for row sum accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset before row sum accumulation begins. Must be even.	RW	0
15:0	OFFSET_X	X offset before row sum accumulation begins. Must be even.	RW	0

Table 58: PISP_FE_AGC_ROW_OFFSET register definition.

PISP_FE_AGC_ROW_SIZE				**0x01F4**
Size of rows which are accumulated.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SIZE_Y	Height of each bin for row sum accumulation. Must be even and nonzero. The total height of the region used for row sums will be 512 times this value, unless clipped at the bottom of the frame.	RW	0
15:0	SIZE_X	Width of the window within which row sums are accumulated. Must be even.	RW	0

Table 59: PISP_FE_AGC_ROW_SIZE register definition.

PISP_FE_AGC_SHIFT				**0x01F8**
Right shifts applied before row sum and floating zone accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:8	FLOAT_SHIFT	Value by which Y values are right-shifted before floating-zone sums.	RW	0
7:4	-	Reserved.	-	-
3:0	ROW_SHIFT	Value by which Y values are right-shifted before row sums.	RW	0

Table 60: PISP_FE_AGC_SHIFT register definition.

6.4.16 AWB Statistics Configuration Registers

AWB statistics are gathered for the purposes of driving AWB (Auto White Balance) and ACLS (Auto Colour Lens Shading) algorithms. They are collected for a 32x32 grid of equally sized rectangles within a window of the image.

For each rectangle within this grid we accumulate sums of the red, green and blue pixel values in each 2×2 Bayer "quad" (the lower green phase is used), along with a count of the number of "quads" summed (this may be smaller than the area of the rectangle, as quads with especially low or high R,G,B values may optionally be excluded).

The arrangement of the grid with respect to its *offset* and *size* parameters is identical to the AGC statistics, except that the grid is 32x32, and not 16x16 in size (please see section 6.4.15).

As before, the offsets and sizes of the rectangles are in full-resolution units (or half resolution if the Statistics Decimate block is enabled) and must all be even. The bottom bit of each field will be ignored and should be written as zero.

AWB statistics are collected without any lens shading correction applied as they are also intended for use by ACLS (Auto Colour Lens Shading) algorithms. AWB algorithms will need to apply the lens shading colour gains before using them. For convenience, the 32×32 grid size matches that used in the Back End LSC block.

PISP_FE_AWB_OFFSET				**0x01FC**
Offset of the window within which statistics are gathered.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset of the region within which statistics are gathered. Must be even.	RW	0
15:0	OFFSET_X	X offset of the region within which statistics are gathered. Must be even.	RW	0

Table 61: PISP_FE_AWB_OFFSET register definition.

PISP_FE_AWB_SIZE				**0x0200**
Width and height of the window within which statistics are gathered.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SIZE_Y	Height of each of the grid cells defining the weights. Must be even. The height of the window below OFFSET_Y within which statistics are collected is 32*SIZE_Y.	RW	0
15:0	SIZE_X	Width of each of the grid cells defining the weights. Must be even. The width of the window to the right of OFFSET_X within which statistics are collected is 32*SIZE_X.	RW	0

Table 62: PISP_FE_AWB_SIZE register definition.

PISP_FE_AWB_SHIFT				**0x0204**
Shift applied to pixel values before accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:3	-	Reserved.	-	-
2:0	SHIFT	Right-shift value applied to pixels before being accumulated.	RW	0

Table 63: PISP_FE_AWB_SHIFT register definition.

PISP_FE_AWB_RED				**0x0208**
Limits that red pixels must satisfy for accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Pixels must be less than or equal to this value in order to be accumulated.	RW	0

Continued on next page...

PISP_FE_AWB_RED (continued)				**0x0208**
Bits	***Name***	***Description***	***R/W***	***Reset***
15:0	LO	Pixels must be greater than or equal to this value in order to be accumulated.	RW	0

Table 64: PISP_FE_AWB_RED register definition.

PISP_FE_AWB_GREEN				**0x020C**
Limits that green pixels must satisfy for accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Pixels must be less than or equal to this value in order to be accumulated.	RW	0
15:0	LO	Pixels must be greater than or equal to this value in order to be accumulated.	RW	0

Table 65: PISP_FE_AWB_GREEN register definition.

PISP_FE_AWB_BLUE				**0x0210**
Limits that blue pixels must satisfy for accumulation.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Pixels must be less than or equal to this value in order to be accumulated.	RW	0
15:0	LO	Pixels must be greater than or equal to this value in order to be accumulated.	RW	0

Table 66: PISP_FE_AWB_BLUE register definition.

6.4.17 CDAF (Contrast Detect Autofocus) Configuration Registers

The CDAF block calculates contrast measures across the image, by summing the squares of horizontal and vertical pixel differences within Bayer components. It operates on at most two of the four components (R, G_R, G_B, B). Before calculating differences, each component is smoothed by an IIR (infinite impulse response) filter, to reduce the effects of noise.

The noise deviation that the filter needs to smooth out any pixel is estimated by $noise\_constant + noise\_slope * \sqrt{pixel\_value}$. Increasing the noise parameters will cause increased smoothing, and decreasing them will reduce the amount of smoothing applied.

Autofocus statistics are gathered for an 8×8 grid of equally sized rectangles spread across a window within the image. Each region of this grid has size $(size\_x + skip\_x) \times (size\_y + skip\_y)$, but the contrast measure is only collected within the top left $size\_x \times size\_y$ of the region. This is illustrated in the diagram below.

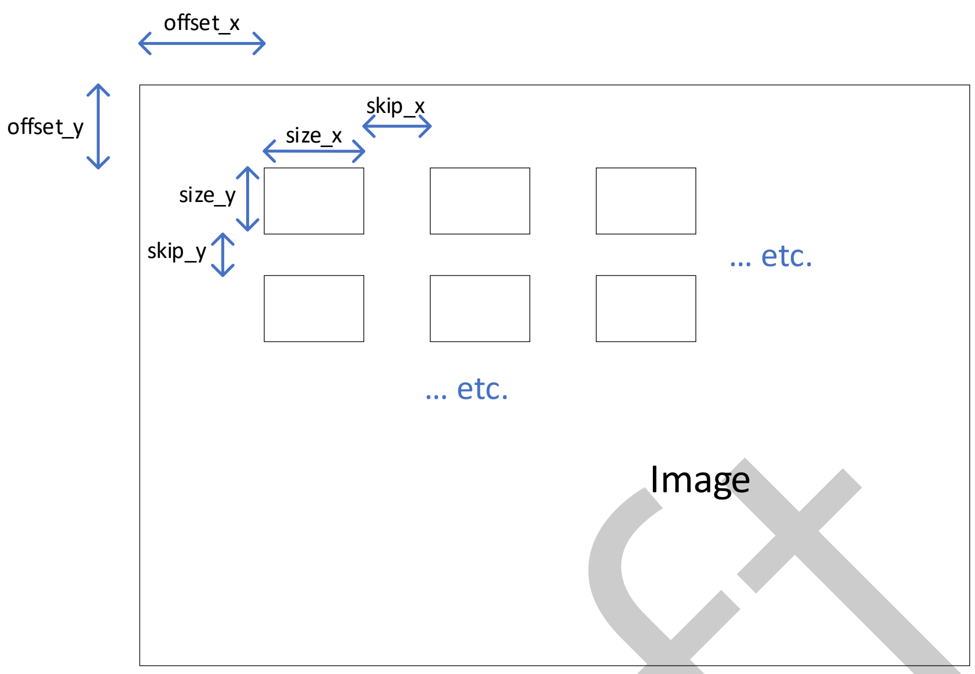

Figure 9: Grid layout for CDAF statistics (not all grid rectangles shown)

The offset and dimensions of the CDAF grid rectangles must all be even.

PISP_FE_CDAF_NOISE				**0x0214**
Noise model for the CDAF block.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SLOPE	Noise slope ($u8.8$ format).	RW	0
15:0	CONSTANT	Noise constant.	RW	0

Table 67: PISP_FE_CDAF_NOISE register definition.

PISP_FE_CDAF_OFFSET				**0x0218**
Offset for CDAF window.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset before contrast measure accumulation begins. Must be even.	RW	0
15:0	OFFSET_X	X offset before contrast measure accumulation begins. Must be even.	RW	0

Table 68: PISP_FE_CDAF_OFFSET register definition.

PISP_FE_CDAF_SIZE				0x021C
Size of CDAF regions.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SIZE_Y	Height of the CDAF regions. Must be even.	RW	0
15:0	SIZE_X	Width of the CDAF regions. Must be even.	RW	0

Table 69: PISP_FE_CDAF_SIZE register definition.

PISP_FE_CDAF_SKIP				0x0220
Skip value between CDAF regions.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SKIP_Y	Number of pixels to skip vertically. Must be even.	RW	0
15:0	SKIP_X	Number of pixels to skip horizontally. Must be even.	RW	0

Table 70: PISP_FE_CDAF_SKIP register definition.

PISP_FE_CDAF_MODE				0x0224
Bayer component selection and weighting for CDAF. Suitable values include 0x09, 0x15.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:6	-	Reserved.	-	-
5:4	CPTS	Choice of Bayer components for CDAF statistics. 0: (R, G_B); 1: (G_R, G_B); 2: (R, B); 3: (G_R, B).	RW	0
3:2	WEIGHT_G	Weight to be given to the G_B or B component, in the range 0..3. Weights should sum to ≤ 4.	RW	0
1:0	WEIGHT_R	Weight to be given to the R or G_R component, in the range 0..3. Weights should sum to ≤ 4.	RW	0

Table 71: PISP_FE_CDAF_MODE register definition.

6.4.18 Floating Statistics Configuration Registers

In addition to their normal statistics, AGC, AWB and CDAF also gather statistics for up to four "floating" regions, defined by rectangles within the Statistics Crop region. As before, the offsets and dimensions of the floating regions must all be even. If Statistics Decimation is enabled, they should be configured at half size. The floating regions may overlap one another.

The floating statistics regions cannot be disabled *per se*. If AGC statistics are enabled then AGC statistics will be gathered for the floating regions, and so forth. If an application has no need for the floating statistics regions they can simply be ignored; it is recommended to set their sizes to zero.

PISP_FE_FLOATING[n]_OFFSET				**0x0228$+8n$**
XY offset of floating rectangle n, for $n = 0, 1, 2, 3$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset of rectangle. Must be even.	RW	0
15:0	OFFSET_X	X offset of rectangle. Must be even.	RW	0

Table 72: PISP_FE_FLOATING[n]_OFFSET register definition.

PISP_FE_FLOATING[n]_SIZE				**0x022C$+8n$**
Size of floating rectangle n, for $n = 0, 1, 2, 3$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SIZE_Y	Height of rectangle. Must be even.	RW	0
15:0	SIZE_X	Width of rectangle. Must be even.	RW	0

Table 73: PISP_FE_FLOATING[n]_SIZE register definition.

6.4.19 Output AXI Configuration Registers

These registers control AXI write parameters. Unlike other PiSP AXI masters, FE output QOS can vary dynamically with Input FIFO fullness, thus four QOS levels should be configured. When MINOR_-VERSION ≥ 1, QOS may also depend on Output FIFO fullness.

PISP_FE_OUTPUT_AXI				**0x0248**
Configuration for AXI write bursts.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	QOS3	Value for AWQOS when input FIFO is $\geq 75\%$ full in non-panic mode, or $\geq 25\%$ when panicking.	RW	0
27:24	QOS2	Value for AWQOS when input FIFO is 50%–75% full in non-panic mode, or $< 25\%$ when panicking.	RW	0
23:20	QOS1	Value for AWQOS when input FIFO is 25%–50% full in non-panic mode, or $\geq 25\%$ when not panicking.	RW	0
19:16	QOS0	Value for AWQOS when input FIFO is below 25% and either not in panic mode or not panicking.	RW	0
15	-	Reserved.		
14:12	PROT	Value for AWPROT field, where used.	RW	0
11:8	CACHE	Value for AWCACHE field, where used.	RW	0

Continued on next page. . .

PISP_FE_OUTPUT_AXI (continued)				**0x0248**
Bits	***Name***	***Description***	***R/W***	***Reset***
7	ALIGN	When MAXLEN+1 is a power of two but address is not so aligned, truncate burst at a 16*(MAXLEN+1) byte boundary, to align following bursts.	RW	0
6	PAD	When '0', use WSTRB to mask valid pixels. When '1', append junk bytes up to a 16-byte boundary.	RW	0
5	PANIC	Enable "panic mode", which forces QOS2 or QOS3 whenever the output FIFO fills above a threshold. Effective only when MINOR_VERSION $\geq$ 1 and an Output FIFO is implemented.	RW	0
4	-	Reserved.		
3:0	MAXLEN	Maximum AXI burst length minus one.	RW	0

Table 74: PISP_FE_OUTPUT_AXI register definition.

PISP_FE_OUTPUT_PANIC **0x024C**

Panic thresholds, used when MINOR_VERSION $\geq$ 1, Output FIFO depth is nonzero, and the PANIC flag (above) is set.

Bits	***Name***	***Description***	***R/W***	***Reset***
31	EOF	Panic 2048 cycles after Frame End, if there are data (such as Statistics) still to be written.	RW	0
30	SOF	Panic if a new Frame Start is seem while writing out the previous frame.	RW	0
29:16	THROTTLE	Output FIFO level at which Statistics traffic is slowed to avoid spurious panic.	RW	0
15:0	THRESH	Output FIFO level at which to panic.	RW	0

Table 75: PISP_FE_OUTPUT_PANIC register definition.

6.4.20 Crop 0 Configuration Registers

This block crops the image on the first output branch of the pipeline. The offset and dimensions of the crop rectangle must all be even; the bottom bits of each field are ignored and should be written as zero. When crop is enabled, WIDTH and HEIGHT must both be at least 2.

PISP_FE_CROP0_OFFSET				**0x0250**
Offset of the crop window.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset of the crop window. Must be even.	RW	0
15:0	OFFSET_X	X offset of the crop window. Must be even.	RW	0

Table 76: PISP_FE_CROP0_OFFSET register definition.

PISP_FE_CROP0_SIZE				**0x0254**
Size of the crop window.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the crop window. Must be even.	RW	0
15:0	WIDTH	Width of the crop window. Must be even.	RW	0

Table 77: PISP_FE_CROP0_SIZE register definition.

6.4.21 Downscale 0 Configuration Registers

The Downscale block can shrink both Bayer and monochrome images. It uses a “trapezoidal” filter which is capable of downscaling by any rational factor $1 \leq factor \leq 16$, where $factor$ can be expressed as IN/OUT for integers $1 \leq OUT \leq IN \leq 31$. WIDTH and HEIGHT must be nonzero even values no greater than the proportionally scaled input dimensions, rounded down.

In Bayer mode, the block respects the Bayer pattern and maintains a regular sampling grid (unless a Binning flag is set).

When MINOR_VERSION = 0, in Monochrome mode, the XOUT field should not exceed 16.

PISP_FE_DOWNSCALE0_RATIOS				**0x0258**
Downscaling ratios, where $1 \leq OUT \leq IN \leq 31$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:24	YOUT	Denominator of the vertical downscaling factor.	RW	0
23:21	-	Reserved.	-	-
20:16	YIN	Numerator of the vertical downscaling factor.	RW	0
15:13	-	Reserved.	-	-
12:8	XOUT	Denominator of the horizontal downscaling factor.	RW	0
7:5	-	Reserved.	-	-

Continued on next page...

PISP_FE_DOWNSCALE0_RATIOS (continued)				**0x0258**
Bits	***Name***	***Description***	***R/W***	***Reset***
4:0	XIN	Numerator of the horizontal downscaling factor.	RW	0

Table 78: PISP_FE_DOWNSCALE0_RATIOS register definition.

PISP_FE_DOWNSCALE0_BAYER				**0x025C**
Bayer mode flags.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1	BIN	When BIN and BAYER are both set, scale all Bayer components identically, without offset correction.	W	0
0	BAYER	Set to 1 to select Bayer mode, or 0 to select monochrome mode.	RW	0

Table 79: PISP_FE_DOWNSCALE0_BAYER register definition.

PISP_FE_DOWNSCALE0_OUTPUT_SIZE				**0x0260**
Output image size.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Output height. Must be even.	RW	0
15:0	WIDTH	Output width. Must be even.	RW	0

Table 80: PISP_FE_DOWNSCALE0_OUTPUT_SIZE register definition.

6.4.22 Compression 0 Configuration Registers

If this block is enabled the COMPRESSED bit in the corresponding Output block should be set. This will cause the output image to be compressed to 8 bits per pixel.

PISP_FE_COMPRESS0				**0x0264**
Parameters for compression of PiSP compressed RAW image format data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select compression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0

Continued on next page. . .

PISP_FE_COMPRESS0 (continued)				**0x0264**
Bits	***Name***	***Description***	***R/W***	***Reset***
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset subtracted from pixels before compression. This may improve fidelity when there is a high black level; it is recommended to set it to a multiple of 512 below the lowest expected pixel value.	RW	0

Table 81: PISP_FE_COMPRESS0 register definition.

6.4.23 Output 0 Configuration Registers

When enabled, the Output 0 block writes final result of output branch 0 to memory. It supports two image formats:

- 16-bit pixel values. The pixels are written in little-endian format and may optionally be right shifted by up to 8 bits.
- Compressed format. This is the PiSP proprietary RAW compression format, with 8 bits per pixel.

Addresses in PiSP have 64 bits, but depending on the target chip some high bits may be ignored. The address at which to write pixel data in memory must be aligned to a multiple of 16 bytes. Its line to line stride must also be a non-negative multiple of 16 bytes.

The output should match the dimensions of the input, crop or downscale (depending on enable flags) stage. Output WIDTH and HEIGHT must both be nonzero.

PISP_FE_OUTPUT0_SIZE				**0x0268**
Dimensions of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being written.	RW	0
15:0	WIDTH	Width of the image being written. Must be even.	RW	0

Table 82: PISP_FE_OUTPUT0_SIZE register definition.

PISP_FE_OUTPUT0_FORMAT				**0x026C**
Format definition of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the output is compressed.	RW	0

Continued on next page...

PISP_FE_OUTPUT0_FORMAT (continued) **0x026C**

Bits	*Name*	*Description*	*R/W*	*Reset*
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are right-shifted by this amount if COMPRESSION_MODE is zero, for 0 <= SHIFT <= 8.	RW	0
15:2	-	Reserved.	-	-
1:0	BITS_PER_SAMPLE	Set this to 0 (8 bps) if compressed, or 3 (16 bps) if uncompressed.	RW	0

Table 83: PISP_FE_OUTPUT0_FORMAT register definition.

PISP_FE_OUTPUT0_STRIDE **0x0270**

Stride of pixel data in memory.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 84: PISP_FE_OUTPUT0_STRIDE register definition.

PISP_FE_OUTPUT0_STRIDE2 **0x0274**

This register is ignored.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	-	Reserved.	-	-

Table 85: PISP_FE_OUTPUT0_STRIDE2 register definition.

PISP_FE_OUTPUT0_ILINES **0x0278**

Line count for line-based interrupts.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	-	Reserved. Reads as zero.	-	-
15:0	LINES	When enabled, raise an interrupt for every LINES image output lines that are written.	RW	0

Table 86: PISP_FE_OUTPUT0_ILINES register definition.

PISP_FE_OUTPUT0_SPARE				**0x027C**
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 87: PISP_FE_OUTPUT0_SPARE register definition.

6.4.24 Crop 1 Configuration Registers

Functionally identical to the Crop 0 block, but for the second output branch of the pipeline.

PISP_FE_CROP1_OFFSET				**0x0280**
Offset of the crop window.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	OFFSET_Y	Y offset of the crop window. Must be even.	RW	0
15:0	OFFSET_X	X offset of the crop window. Must be even.	RW	0

Table 88: PISP_FE_CROP1_OFFSET register definition.

PISP_FE_CROP1_SIZE				**0x0284**
Size of the crop window.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the crop window. Must be even.	RW	0
15:0	WIDTH	Width of the crop window. Must be even.	RW	0

Table 89: PISP_FE_CROP1_SIZE register definition.

6.4.25 Downscale 1 Configuration Registers

Where present, Downscale 1 is similar to Downscale 0, but may have a smaller width limit.

PISP_FE_DOWNSCALE1_RATIOS				**0x0288**
Downscaling ratios, where $1 \leq OUT \leq IN \leq 31$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:24	YOUT	Denominator of the vertical downscaling factor.	RW	0
23:21	-	Reserved.	-	-

Continued on next page...

PISP_FE_DOWNSCALE1_RATIOS (continued)				**0x0288**
Bits	***Name***	***Description***	***R/W***	***Reset***
20:16	YIN	Numerator of the vertical downscaling factor.	RW	0
15:13	-	Reserved.	-	-
12:8	XOUT	Denominator of the horizontal downscaling factor.	RW	0
7:5	-	Reserved.	-	-
4:0	XIN	Numerator of the horizontal downscaling factor.	RW	0

Table 90: PISP_FE_DOWNSCALE1_RATIOS register definition.

PISP_FE_DOWNSCALE1_BAYER				**0x028C**
Bayer mode flags.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1	BIN	When BIN and BAYER are both set, scale all Bayer components identically, without offset correction.	W	0
0	BAYER	Set to 1 to select Bayer mode, or 0 to select monochrome mode.	RW	0

Table 91: PISP_FE_DOWNSCALE1_BAYER register definition.

PISP_FE_DOWNSCALE1_OUTPUT_SIZE				**0x0290**
Output image size.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Output height. Must be even.	RW	0
15:0	WIDTH	Output width. Must be even.	RW	0

Table 92: PISP_FE_DOWNSCALE1_OUTPUT_SIZE register definition.

6.4.26 Compression 1 Configuration Registers

Functionally identical to the Compression 0 block, but for the second output branch of the pipeline.

PISP_FE_COMPRESS1				**0x0294**
Parameters for compression of PiSP compressed RAW image format data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select compression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset subtracted from pixels before compression. This may improve fidelity when there is a high black level.	RW	0

Table 93: PISP_FE_COMPRESS1 register definition.

6.4.27 Output 1 Configuration Registers

Functionally identical to the Output 0 block, but for the second output branch of the pipeline.

PISP_FE_OUTPUT1_SIZE				**0x0298**
Dimensions of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being written.	RW	0
15:0	WIDTH	Width of the image being written.	RW	0

Table 94: PISP_FE_OUTPUT1_SIZE register definition.

PISP_FE_OUTPUT1_FORMAT				**0x029C**
Format definition of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the output is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are right-shifted by this amount if COMPRESSION_MODE is zero, for 0 <= SHIFT <= 8.	RW	0
15:2	-	Reserved.	-	-
1:0	BITS_PER_SAMPLE	Set this to 0 (8 bps) if compressed, or 3 (16 bps) if uncompressed.	RW	0

Table 95: PISP_FE_OUTPUT1_FORMAT register definition.

PISP_FE_OUTPUT1_STRIDE				0x02A0
Stride of pixel data in memory.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 96: PISP_FE_OUTPUT1_STRIDE register definition.

PISP_FE_OUTPUT1_STRIDE2				0x02A4
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 97: PISP_FE_OUTPUT1_STRIDE2 register definition.

PISP_FE_OUTPUT1_ILINES				0x02A8
Line count for line-based interrupts.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved. Reads as zero.	-	-
15:0	LINES	When enabled, raise an interrupt for every LINES image output lines that are written.	RW	0

Table 98: PISP_FE_OUTPUT1_ILINES register definition.

PISP_FE_OUTPUT1_SPARE				0x02AC
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 99: PISP_FE_OUTPUT1_SPARE register definition.

6.5 Statistics Output Definitions

The various statistics blocks write their results to memory via DMA. Here we define the format in memory that these outputs take. The output statistics are written contiguously to the address defined by PISP_FE_IO_STATS_ADDR_LO and PISP_FE_IO_STATS_ADDR_HI, in the order given below.

- AWB statistics, starting at the offset 0 bytes from the address that was specified.
- AGC statistics, starting at the offset 16448 bytes from the address that was specified.
- CDAF statistics, starting at the offset 22656 bytes from the address that was specified.

Note that there is a total of 23200 bytes of statistics data in total, and that the offsets given are respected even when any of the statistics in question are not enabled.

6.5.1 AWB Statistics

For each of its 32x32 zones, AWB writes out red, green and blue sums, plus the number of pixels it counted (that is, passed the inclusion criteria) in each zone. This latter value is the same for all the colour sums. All values are 32-bit little-endian format; offsets and sizes are given in bytes.

Field	Offset	Size	Description
Red Sum 0	0	4	Red pixel sum for zone 0.
Green Sum 0	4	4	Green pixel sum for zone 0.
Blue Sum 0	8	4	Blue pixel sum for zone 0.
Counted 0	12	4	Number of pixels counted in zone 0.
Red Sum 1	16	4	Red pixel sum for zone 1.
Green Sum 1	20	4	Green pixel sum for zone 1.
Blue Sum 1	24	4	Blue pixel sum for zone 1.
Counted 1	28	4	Number of pixels counted in zone 1.
...			
Red Sum 1023	16368	4	Red pixel sum for zone 1023.
Green Sum 1023	16372	4	Green pixel sum for zone 1023.
Blue Sum 1023	16376	4	Blue pixel sum for zone 1023.
Counted 1023	16380	4	Number of pixels counted in zone 1023.

Table 100: Output format of AWB zones.

Field	Offset	Size	Description

Field	Offset	Size	Description
Red Sum 0	0	4	Red pixel sum for floating zone 0.
Green Sum 0	4	4	Green pixel sum for floating zone 0.
Blue Sum 0	8	4	Blue pixel sum for floating zone 0.
Counted 0	12	4	Number of pixels counted in floating zone 0.
...			
Red Sum 3	48	4	Red pixel sum for floating zone 3.
Green Sum 3	52	4	Green pixel sum for floating zone 3.
Blue Sum 3	56	4	Blue pixel sum for floating zone 3.
Counted 3	60	4	Number of pixels counted in floating zone 3.

Table 101: Output format of AWB floating zones.

Note that this makes a total of $16384 + 64 = 16448$ bytes of AWB statistics data.

6.5.2 AGC Statistics

First of all, AGC writes 512 32-bit row sums to memory, as shown in the table below. The offset and size values are in bytes, and all are output in little-endian format.

Field	Offset	Size	Description
Row 0 Sum	0	4	Row 0 total.
Row 1 Sum	4	4	Row 1 total.
...			
Row 511 Sum	2044	4	Row 511 total.

Table 102: Output format of AGC row sums.

Next, AGC writes its 1024-bin histogram of 32-bit values to memory in the order shown below, immediately following the row sums.

Field	Offset	Size	Description
Bin 0	0	4	Number of pixels in bin 0.
Bin 1	4	4	Number of pixels in bin 1.
...			
Bin 1023	4092	4	Number of pixels in bin 1023.

Table 103: Output format of AGC histogram.

Finally, AGC outputs Y sums and counts for the four floating zones, immediately following the histogram.

Field	Offset	Size	Description
Zone 0 Sum	0	8	Sum of pixel Y values in floating zone 0.
Zone 0 Count	8	8	Count of pixels in floating zone 0.
...			
Zone 3 Sum	48	8	Sum of pixel Y values in floating zone 3.
Zone 3 Count	56	8	Count of pixels in floating zone 3.

Table 104: Output format of AGC floating zones.

Note that this makes a total of $2048 + 4096 + 64 = 6208$ bytes of AGC statistics data.

6.5.3 CDAF Statistics

The CDAF block stores a 64-bit FOM (figure of merit) value for each of the 64 zones.

Field	Offset	Size	Description
FOM 0	0	8	Contrast measure for zone 0.
FOM 1	8	8	Contrast measure for zone 1.
...			
FOM 63	504	8	Contrast measure for zone 63.

Table 105: Output format of CDAF FOMs.

The are four further FOMs for the floating zones immediately following those above.

Field	Offset	Size	Description
FOM 0	0	8	Contrast measure for floatingzone 0.
...			
FOM 3	24	8	Contrast measure for floating zone 3.

Table 106: Output format of CDAF floating zone FOMs.

Note that this makes a total of $512 + 32 = 544$ bytes of CDAF statistics data.

7 The PiSP Back End

7.1 Back End Overview

The Back End reads pixels from DRAM in a variety of formats. It performs the bulk of the required image processing on the pixels before writing up to two output images back to memory. These two images are normally cropped to the same field of view but may be scaled differently. Just like the Front End it supports both Bayer and greyscale images. It also supports the same compressed image format as the Front End, meaning the two can cooperate to reduce memory bandwidth usage.

For convenience the Back End is sometimes thought of as consisting of a *Bayer Pipe* (everything up to and including *Demosaic*) and an *RGB Pipe* (everything after *Demosaic*). Images are normally read from memory into the *Bayer Pipe* though the *RGB Pipe* is capable of reading 3-channel images (such as YUV or RGB) too. The *RGB Pipe* also includes stages where the pixels may not actually be represented in the RGB colour space (such as YCbCr), and terminates in a pair of *Output Pipes*, responsible for writing images out to memory. The two *Output Pipes* are very similar in their capabilities. This arrangement is illustrated below.

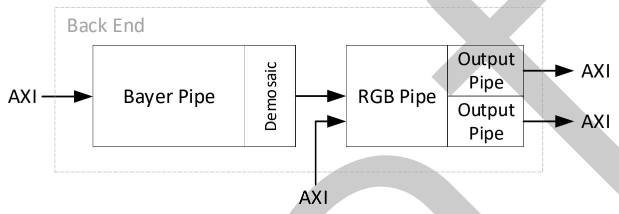

Figure 10: Back End divided into Bayer and RGB pipes

One of the chief differences between the Front and Back Ends, is that the Back End processes images in *tiles*. Hardware imposes a maximum tile width (typically 640); otherwise the widths and heights of tiles are determined by software. Because many of the filtering operations in the Back End require pixel context, the source rectangles of neighbouring tiles will generally overlap; output tiles may be cropped to remove this overlap.

Note also that it is possible for a tile to produce outputs on only some of the output branches even though all the branches are enabled – the pipeline supports the notion that some branches may crop the entire tile to nothing. This is useful when outputs are differently cropped, or when size and alignment constraints would make a common tiling difficult.

7.2 Back End Pipe in Detail

7.2.1 Bayer Pipe

The Bayer Pipe is shown below, reading image data from memory and passing it, after the Demosaic block, to the RGB Pipe.

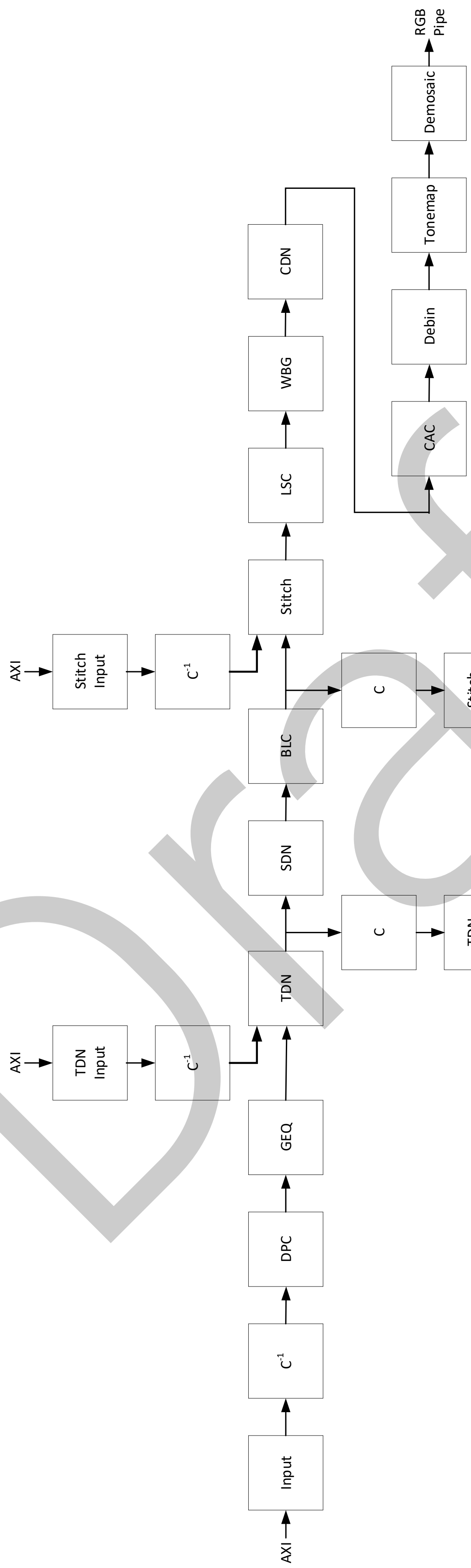

Figure 11: PiSP Back End Bayer Pipe

Legend:

- *Input* - Input block can read compressed and uncompressed Bayer pixel data.
- C^{-1} - Decompression, only to be enabled when compressed pixel data is read.
- *DPC* - Defective Pixel Correction block. This version of the block, unlike the one in the Front End, has no line length limit.
- *GEQ* - Green Equalisation. Corrects “imbalance” between the two types of green (on red rows and on blue rows) in the Bayer pattern.
- *TDN Input* - Reads the previous LTA (Long Term Average) frame output by the TDN block when the previous frame was processed.
- *TDN* - Temporal Denoise. Requires the current frame being processed, and the previous LTA output of this block, to be supplied as inputs.
- C - optional compression, to reduce memory traffic when using TDN.
- *TDN Output* - writes the output of TDN (the LTA frame) to memory, so that it can be read by the TDN Input block when the next frame is processed.
- *SDN* - Spatial Denoise.
- *BLC* - Black Level Correction. Black levels are finally subtracted from the pixels here.
- *Stich Output* - outputs the current frame to memory so that it can be re-read next time round by the Stitch Input block as the second input to the Stitch block.
- *Stitch Input* - reads the previous frame that is to be stitched with the frame currently being processed by the pipeline. Normally the other frame is of a different exposure and must be read to create HDR images.
- *Stitch* - stitches its two input frames together. Normally one is a short exposure image and the other is a long exposure image, and the two are stitched to create an HDR (High Dynamic Range) image.
- *LSC* - Lens Shading Correction.
- *WBG* - apply white balance gains to pixels.
- *CDN* - Colour Denoise.
- *CAC* - Chromatic Aberration Correction.
- *Debin* - applies filtering to pixels to reduce the “staircase” effects caused by standard camera 2x2 binning of pixels.
- *Tonemap* - apply tonemapping to the image data. Usually used in conjunction with the Sitch blocks to create HDR images.
- *Demosaic* - Demosaic block.

Mostly the Bayer Pipe reads pixels, processes them straight through, and passes three-channel RGB pixels to the RGB pipe. However, the reader’s attention is drawn to the following use cases.

1. **Temporal Denoise**. When using temporal denoise, the TDN Output block should be enabled so that the output of TDN itself (which is the LTA, or Long Term Average, frame) can be written out to memory. Similarly, the TDN Input block should be enabled and configured to read the previous LTA frame that was produced by TDN. The TDN block combines the current frame, and the previous LTA frame, to produce the new LTA frame.

2. **HDR**. This will use both the Stitch and Tonemap blocks. The Stitch Output block should be enabled to write the current frame to memory. Similarly, the Stitch Input block will be enabled to read the frame previously written by Stitch Output. One of these frames should be a long exposure image, and the other a short exposure image. The Stitch block itself combines them to create an HDR image. Finally, the Tonemap block will normally be enabled so as to render the different parts of the scene at a level of brightness more visible in (for example) an 8-bit final output image.

3. **Outputting Intermediate Bayer Images**. Sometimes it may be desirable to obtain Bayer data from the Back End pipeline. In this case, the Demosaic block should be *disabled*, which causes it to duplicate the incoming Bayer pixel onto all three of the output channels. All the downstream pipeline stages, except for the final output stage, should be disabled, resulting in the writing to memory of the intermediate Bayer format image.

7.2.2 RGB Pipe

The RGB Pipe is shown below, receiving pixel data from the Bayer Pipe, and optionally writing multiple output images to memory.

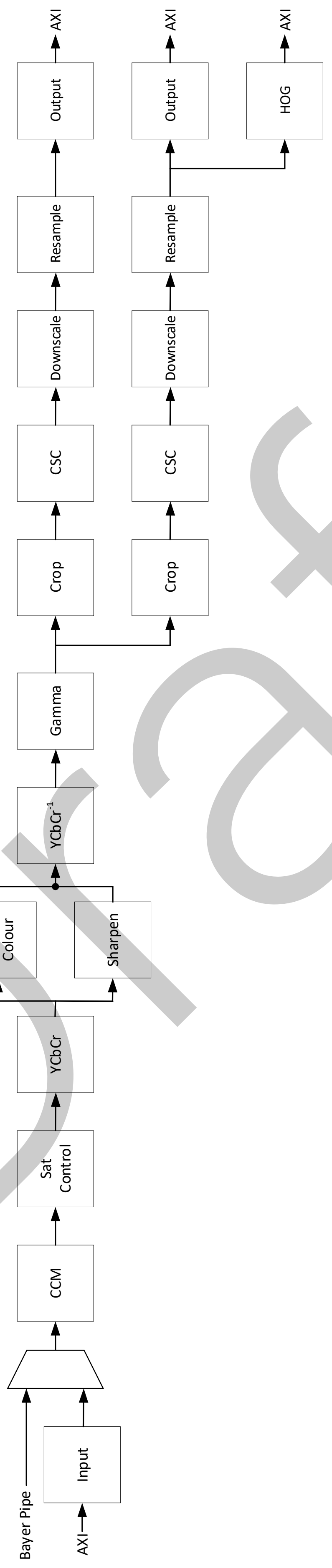

Figure 12: PiSP Back End RGB Pipe

Legend:

- *Input* - pixels can be read from memory instead of from the Bayer Pipe. This is useful for implementing simple rescaling and format conversions of fully-sampled RGB or YUV images.
- *CCM* - Colour Correction Matrix.
- *Sat Control* - Saturation Control block which prevents hue distortion as saturation of different colour channels occurs.
- $YCbCr$ - conversion to YCbCr (or similar) colour space.
- *False Colour + Sharpen* - reduction of false colours, and edge/texture sharpening.
- $YCbCr^{-1}$ - Inverse YCbCr conversion. Converts back to RGB colour space, and should be the inverse transformation of the earlier YCbCr block.
- *Gamma* - apply gamma.
- *CSC* - colour space conversion to desired output colour space.
- *Downscale* - downscale, allowed scale factors are $scale\_factor = 1$, or $2 \leq scale\_factor \leq 8$. Note that the downscale block is not available on both output branches in all hardware variants.
- *Resample* - resampling. Can be used to upscale an image, or to shift the sampling point of pixel data (for example, for YUV420 output).
- *Output* - Output formatter writes image data to memory in a variety of formats, including planar formats, interleaved formats, and so forth.
- *HOG* - Write out HOG (Histogram of Oriented Gradients) features for the image.

We note how the following use cases can be implemented:

1. **Format Conversions**. Simple image format conversions and image resizing are supported by reading the image from memory into the RGB Pipe and changing the output format and/or size in the Output pipe(s). Half resolution image channels (such as in YUV420) are up-sampled by pixel doubling. Depending on the output format, and whether resampling is performed, there may be some very marginal sub-pixel sampling errors in certain of the output channels.
2. **Image Scaling**. The Downscalers do not support downscaling factors (strictly) between 1 and 2, so these cases must be implemented using the Resampling blocks. The Resamplers support arbitrary scale factors (both up- and down-scaling), but are not recommended for downscaling by factors larger than 2, for image quality reasons. Note that the two output branches are free to produce images of completely different sizes. Some hardware variants do not have a downscaler on both output branches.
3. **Non-cosited Output Samples**. Some image formats require some of the output channels to be sampled in a slightly different spatial location to other channels, YUV420 being a case in point. The Resampler blocks allow different pixel offsets to be specified for the three channels, allowing this to achieved. (Note that if the Resampler is being used to downscale, then any such pixel offsets must be scaled up accordingly.)
4. **Computer Vision**. Some versions of the Back End can expedite Computer Vision applications by writing integral images (useful for Haar-based feature detectors, such as Viola-Jones type algorithms) on some hardware variants, and a Histogram of Oriented Gradients (HOG) for each 8x8 cell of an image. Observe that the HOG block is distinct from the Output block on the second branch, so both these outputs can be generated simultaneously (where supported).

7.3 Programmer's Model

7.3.1 Tiles, Tiles, Tiles

Whereas the jobs handled by the Front End consist of requests to process entire frames, a job submitted to the Back End is a request to process one or more tiles. Generally more than one tile is submitted at a time so as to be less reliant on the system's main processor. Such a group of tiles is often referred to as a *batch* of tiles.

Like the Front End, the Back End also operates a 1-deep request queue. This means that the next batch of tiles and their required configuration can be programmed while the previous batch is still being processed. The tiles themselves are calculated and stored by firmware in the processor's memory and their address is supplied to the PiSP Back End which reads them as needed.

It is important to understand that the Back End requires two very different kinds of configuration:

- Global configuration data. This describes how the pixels in a particular image are to be processed and is therefore shared across all the tiles of the image. So for example, LSC (Lens Shading Correction) or Denoise parameters would be common across all tiles from the same frame.
- Tile parameters. Each set of *tile parameters* describes a single tile in the image that is being processed, how many pixels of over-read are present, and how it may change size as it goes through the pipeline. *Ipso facto* the tile parameters, unlike the global configuration data, change from tile to tile. A single set of tile parameters consists of 160 bytes. A *batch* of tiles therefore consists of several sets of *tile parameters*. When a new batch of tiles is submitted it *may* involve a change of global context if the tiles are referring to a new frame, or it may not, when the tiles are merely from further down the same frame. Note how a block such as LSC needs to know where a tile is in the image in order to process it correctly, even when the global LSC configuration does not change. For more information on tile parameters, please see section 7.4.

It should be evident that a batch of tiles does not need to constitute an entire frame, and indeed would usually only be a part of the frame. Processing batches of tiles from *different* frames may be interleaved; whenever there is a change of *frame* then there will normally be a change of (global) configuration, though this lies under the control of firmware.

7.3.2 Tile Address Offsets and Sizes

Tiles have both an input address offset and output address offsets. The input address offset indicates where the first pixel in the tile lies relative to the buffer address specified in the Input block registers. For example, when the entire image buffer is being processed, the top left tile will have an input address offset of zero. There are no particular alignment requirements on the input address offset, although the x and y *pixel* offsets of the first pixel in the tile relative to the top left image pixel at (0,0) should both be even.

Note that the Input block, TDN (Temproal Denoise) Input block and Stitch Input block all share the same input address offset and input pixel offsets.

Each Output branch of the pipeline has its own output address offsets, as the images being output may be in different formats and scaled to different sizes. Each branch actually has up to two output address offsets per tile, covering the use of both interleaved and planar output formats (the third plane, if present, shares the same offset as the second). This time the output address offsets are relative to the buffer addresses in the Output blocks (each Output block can have up to three buffer addresses, when writing fully planar output formats). The output address offsets should always be aligned to *64 bytes* whenever possible, and the width of the output tiles should also be aligned to an exact multiple of 64 bytes as this makes the most efficient use of the memory subsystem.

There are some occasions when this may not be achieveable. For example, if the output image is simply not a multiple of 64 bytes wide then the final column of tiles will require some shorter length writes. In these cases it is still required that the line-to-line *stride* of the output images is a multiple of 64 bytes.

In other cases where very large downscaling factors are being applied then it is again possible for output tiles to be less than 64 bytes wides (especially the U and V planes of 8 bit per pixel 422 and 420 outputs), and subsequent adjacent tiles will also consequently not start at 64 byte aligned output address offsets.

In all cases the supplied firmware will construct the appropriate tiling with the most compliant input and output address offsets.

We note further that no tile passing through the system may be smaller than 16x16 pixels in size. Consequently, the smallest image size that can be process by the PiSP Back End is also 16x16 pixels.

7.3.3 Registers and Interrupts

The Back End is controlled by writing to a number of 32-bit registers (where 64-bit addresses are required, they are supplied as a pair of 32-bit values). These registers are divided into two principal groups:

- *Control and Status Registers* which can be written to to start the PiSP Back End so that it will process another batch of tiles. The status registers can be read to determine what the PiSP has done.
- *Input/Output Configuration Registers* (or just *I/O Registers*) which define where the PiSP Back End reads pixels from and where it writes them to.
- *Processing Configuration Registers* (or just *Configuration Registers*)which control what the PiSP Back End does to pixels once it is running. These constitute the bulk of the registers and are further grouped according to the functional blocks of the Back End pipeline.

Additionally, interrupts are signalled by the Back End which the processor can respond to in order to cause the PiSP Back End to process more tiles, from the same or a different frame. Two kinds of interrupts are generated:

- *Start of Batch Interrupts* or *SOB* interrupts, which are signalled when the Back End has started processing a new batch of tiles (or *job*).
- *End of Batch Interrupts* or *EOB* interrupts, which are signalled when the Back End has completed a batch of tiles (or *job*), including writing all pixel data to memory.

7.3.4 Processing Queue

The Back End implements a similar scheme to the Front End, where the next job can be programmed while the previous job is running. The mechanism is very similar to the Front End where firmware writes to publicly addressable versions of registers and these are copied (latched) internally when a job begins, so that the public versions of the registers can be overwritten while that job is in progress. The only significant difference, as compared to the Front End, is the presence of an extra control bit which stipulates whether a job needs to copy the public I/O and configuration registers when it begins, or whether to use the incumbent values. Thus the register copy can be avoided when the subsequent job is merely another set of tiles from further down the same frame as the previous job (and which should therefore share the same global configuration).

Beyond this the behaviour remains much like the Front End. If the next job has been configured it will start immediately if the Back End is idle. If the Back End was busy it will commence without further intervention once the current job completes. In both cases the *SOB* interrupt will be triggered.

7.3.5 Back End States

The PiSP Back End state machine is simpler than the Front End, consisting only of two states:

1. IDLE - the Back End is idle and will do nothing unless a job is queued, whereupon it will move straight to the ACTIVE state and signal the *SOB* interrupt.
2. ACTIVE - the Back End is actively processing a job. Once the job is complete it will signal the *EOB* interrupt. If no further job has been queued, it will move spontaneously to the IDLE state, or if a job has been queued while it was busy, it will signal the *SOB* interrupt but remain in the ACTIVE state .

Like the Front End, the Back End processing can be stopped by signalling the *abort* command. Any current processing will stop as soon as possible, and any queued job will be abandonned. However, if the PiSP is not IDLE an interrupt will still be raised indicating that the active job, *plus any queued job*, have fully completed (by the value of the *Done* count, which will also be the same as the *Started* count). Note that firmware will not need to process any "intermediate" interrupts wherein these counters might increase one at a time to their final values.

7.3.6 The Back End Status

Besides a *status* registers that returns the current state of the Back End (see section 7.3.5), there is also a single 32-bit *batch status* register. There are no *output status* registers (as the Front End has). Furthermore, the frame counter values are generated by the Back End itself; there is no need to share values produced externally, as was the case with the Front End (sharing values from the camera peripheral). The *batch status* register contains 32 bits, of which only 24 are used, and which can therefore be read atomically, and consists of:

- An 8-bit count of the number of jobs completed.
- An 8-bit count of the number of jobs started.
- An 8-bit count of the number of tiles completed in the current job.

Interrupts are only signalled when *all* the outputs of the tile have finished writing their pixels to memory, so a single status register is sufficient. Counts of completed image lines can be deduced by firmware from the tiling information.

7.4 Tile Parameters

The PiSP Back End processes an image in tiles where each tile is described by a set of *tile parameters*. This is a 160-byte block of data which contains the values that different pipeline blocks need in order to decide where to read pixels from, where write output pixels to, and in some cases exactly how to process them. Each of these sets of *tile parameters* is referred to as a *tile parameter structure*, and they are read one-by-one from a contiguous block of memory as the PiSP works its way through all the tiles it was given.

The fields of the *tile parameter structure* are listed below. In each case we give the size of the field and its offset from the start of the structure, in bytes. Multi-byte values are always stored in little-endian format. Where a field contains multiple entries (for example, for different output branches), the size given is the size of a single entry.

Field	Offset	Size	Description
edge	0	1	Bitfield indicating to the resample blocks whether this tile is to be treated as being at the edge of an image. Bit 0 = left edge, 1 = right edge, 2 = top edge, 3 = bottom edge. Multiple bits may be set, for example when a tile is at a corner. Bits 4 to 7 are ignored.
padding	1	3	These bytes are ignored.
input_addr_offset	4	4	Offset in bytes into the input image from where the top left pixel in the tile is read. This field is used by the Input block both when reading into the Bayer pipe and when reading into the RGB pipe. This value must be at least 4-byte aligned (and may need to be larger if compression or TDN/STITCH outputs are enabled), but for best performance a 64-byte aligned value is recommended.
input_addr_offset2	8	4	Offset in bytes into the 2nd and 3rd image planes from where those planes are read. This value must be at least 4-byte aligned (and may need to be larger - see *input_addr_offset*), but for best performance a 64-byte aligned value is recommended. It is used only when reading into the RGB pipe, and only when the input image has a planar (non-interleaved) format.
input_offset_x	12	2	Horizontal offset in pixels of this tile in the input image.
input_offset_y	14	2	Vertical offset in pixels of this tile in the input image.
input_width	16	2	Width in pixels of this tile as it is read from the input image.
input_height	18	2	Height in pixels of this tile as it is read from the input image.
tdn_input_addr_offset	20	4	Offset in bytes into the TDN input image from where the LTA (long term average) frame is read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
tdn_output_addr_offset	24	4	Offset in bytes into the TDN output image where the new LTA (long term average) frame is written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
stitch_input_addr_offset	28	4	Offset in bytes into the Stitch input image from where the other image is read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Field	Offset	Size	Description
stitch_output_addr_offset	32	4	Offset in bytes into the Stitch output image where the output frame is written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
lsc_grid_offset_x	36	4	Horizontal offset into the LSC table where this tile begins. This is a U5.18 format number in units of the LSC cell width.
lsc_grid_offset_y	40	4	Vertical offset into the LSC table where this tile begins. This is a U5.18 format number in units of the LSC cell height.
cac_grid_offset_x	44	4	Horizontal offset into the CAC table where this tile begins. This is a U3.20 format number in units of the CAC cell width.
cac_grid_offset_y	48	4	Vertical offset into the CAC table where this tile begins. This is a U3.20 format number in units of the CAC cell height.
crop_x_start[n]	52	2	The number of pixels cropped from the left of the tile when passing through the Crop block at the head of output branch n, for $n = 0, 1$.
crop_x_end[n]	56	2	The number of pixels cropped from the right of the tile when passing through the Crop block at the head of output branch n, for $n = 0, 1$.
crop_y_start[n]	60	2	The number of pixels cropped from the top of the tile when passing through the Crop block at the head of output branch n, for $n = 0, 1$.
crop_y_end[n]	64	2	The number of pixels cropped from the bottom of the tile when passing through the Crop block at the head of output branch n, for $n = 0, 1$.
downscale_phase_x[m][n]	68	2	Initial horizontal phase in pixels of colour channel m ($m = 0, 1, 2$) for the Downscaler on output branch n ($n = 0, 1$). This is a U0.12 format value.
downscale_phase_y[m][n]	80	2	Initial vertical phase in pixels of colour channel m ($m = 0, 1, 2$) for the Downscaler on output branch n ($n = 0, 1$). This is a U0.12 format value.
resample_in_width[n]	92	2	Width in pixels of the tile entering the Resample block on output branch n ($n = 0, 1$).
resample_in_height[n]	96	2	Height in pixels of the tile entering the Resample block on output branch n ($n = 0, 1$).
resample_phase_x[m][n]	100	2	Initial horizontal phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch n ($n = 0, 1$). This parameter is in U1.12 format.

Field	Offset	Size	Description
resample_phase_y[m][n]	112	2	Initial vertical phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch n ($n = 0, 1$). This parameter is in U1.12 format.
output_offset_x[n]	124	2	Horizontal offset in pixels where this tile will be written into the output image, for output branch n ($n = 0, 1$).
output_offset_y[n]	128	2	Vertical offset in pixels where this tile will be written into the output image, for output branch n ($n = 0, 1$).
output_width[n]	132	2	Width in pixels of this tile in the output image, for output branch n ($n = 0, 1$).
output_height[n]	136	2	Height in pixels of this tile in the output image, for output branch n ($n = 0, 1$).
output_addr_offset[n]	140	4	Offset in bytes into the output buffer where output branch n ($n = 0, 1$) is to write this tile. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
output_addr_offset2[n]	148	4	Offset in bytes into the output buffer where planes 2 and 3 of this tile are to be written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended. Ignored if the output format is non-planar (interleaved).
output_hog_addr_offset	156	4	Offset in bytes into the HOG output buffer where the results for this tile are to be written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 107: Tile parameter structure. Offsets and sizes are given in bytes.

Tile parameters are explained in more detail below in the discussion of the processing blocks that use them.

7.4.1 Tile Alignment

Note how the tiles should be aligned such that all the *addr_offset* fields are aligned to at least a multiple of 4 bytes. However, when compression is involved alignment will need to be at least 8 bytes, and if any of TDN OUTPUT or STITCH OUTPUT is enabled, then alignment will have to be increased to 16 bytes. For most efficient use of the memory infrastructure, and therefore best performance, these values should ideally be aligned to 64 bytes. This is similar to the restrictions on image buffer base addresses and image *strides*, although with tiles there is no requirement for 128-byte alignment for wallpaper format images.

Observe how 16-byte alignment corresponds to 16, 12 and 8 *pixel* alignment for, respectively, 8bpp, 10bpp and 16bpp images (recall that 10bpp images are written with 3 samples in every 4 byte word). Similar numbers, *mutatis mutandis*, hold for 64-byte alignment.

Firmware is expected to calculate appropriate tilings of an image so as to satisfy these constraints.

7.5 Register Definitions and Tile Parameters

7.5.1 Control Registers

PISP_BE_VERSION				**0x0000**
PiSP Back End version. Note how the fields within this register indicate particular properties and capabilities of this instantiation of the PiSP Back End.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved. Reads as zero.	R	0
25	HOG	Reads as 1 if the final output branch supports HOG output, or 0 if it does not.	R	1
24	INTEGRAL	Reads as 1 if the output branches support integral image output format, or 0 if they do not.	R	0
23:20	DOWNSCALERS	Bitfield that indicates with a 1 in position i that output branch i includes a Downscaler block.	R	2
19:18	NUM_OUTPUTS	One less than the number of output branches.	R	1
17:16	PIX_CLOCK	Base 2 logarithm of the number of pixels per clock processed by the Back End.	R	1
15:8	TILE_WIDTH	One less than the maximum supported tile with divided by 16. That is, $max\_tile\_width/16 - 1$.	R	0x27
7:4	MAJOR_VERSION	PiSP Back End major version number.	R	0
3:0	MINOR_VERSION	PiSP Back End minor version number.	R	1

Table 108: PISP_BE_VERSION register definition.

PISP_BE_CONTROL				**0x0004**
Control register for the PiSP Back End.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	-	Reserved.	-	-
23:16	TILES	Number of tiles to process, which are specified in contiguous memory at the PISP_BE_TILE_ADDR_LO/HI registers. This value should be written at the same moment as the value 1 is written to the QUEUE bit.	W	0
15:4	-	Reserved.	-	-
3	RESET	Sets PISP_BE_BATCH_STATUS register to zero. This bit should only be written to when the Back End is IDLE.	W	0

Continued on next page…

PISP_BE_CONTROL (continued)				**0x0004**
Bits	***Name***	***Description***	***R/W***	***Reset***
2	ABORT	Write a 1 to abort current PiSP Back End processing. The batch currently being processed (if any) and any queued jobs are all abandonned as quickly as possible. If the Back End was not IDLE, an interrupt will be raised to signal that all processing is complete. Other registers are not affected.	W	0
1	COPY	Write a 1 to cause the global configuration registers to be latched and updated internally when the job begins, otherwise write a 0 to keep the same configuration as the previous job. This bit needs to be written on the same operation that writes a 1 to the QUEUE bit.	W	0
0	QUEUE	Write a 1 to queue the new job that has been programmed into the configuration registers.	W	0

Table 109: PISP_BE_CONTROL register definition.

PISP_BE_TILE_ADDR_LO				**0x0008**
Low 32 bits of the address where tile descriptors are read.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_LO	Low 32 bits of address value.	RW	0

Table 110: PISP_BE_TILE_ADDR_LO register definition.

PISP_BE_TILE_ADDR_HI				**0x000C**
High 32 bits of the address where tile descriptors are read.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of address value.	RW	0

Table 111: PISP_BE_TILE_ADDR_HI register definition.

PISP_BE_STATUS				**0x0010**
Status register for the PiSP Back End. Can be polled to determine additional Back End status information.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1	ACTIVE	Reads as 1 when the Back End is busy processing a request and 0 when it is idle (waiting for a new job to be queued).	R	0
0	QUEUED	Reads as 1 when a new Back End job has been queued, and 0 when a new job can be queued.		

Table 112: PISP_BE_STATUS register definition.

PISP_BE_BATCH_STATUS				**0x0014**
Status register for determining progress of the Back End. If necessary, this register can be polled by software.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	-	Reserved.	-	-
23:16	TILES	Number of tiles of the most recently started job that have been completed (meaning that all output pixels have been written to and arrived in memory).	R	0
15:8	STARTED	Count of the number of jobs that have been started. The value here will always be the same as, or 1 larger than (taking account of the modulo 2^8 arithmetic), the value in the DONE field.	R	0
7:0	DONE	Count of the number of jobs that have been completed.	R	0

Table 113: PISP_BE_BATCH_STATUS register definition.

PISP_BE_INTERRUPT_EN				**0x0018**
Interrupt enable register.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved. Reads as zero.	-	-
1	SOB	Set to 1 to enable start of batch (job) interrupts.	RW	0
0	EOB	Set to 1 to enable end of batch (job) interrupts.	RW	0

Table 114: PISP_BE_INTERRUPT_EN register definition.

PISP_BE_INTERRUPT_STATUS				**0x001C**
Interrupt status register. Writing a 1 to a particular bit will clear the corresponding interrupt. Writing 0 to any bit, even reserved ones, has no effect.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1	SOB	Clear start of batch (job) interrupt.	RW	0
0	EOB	Clear end of batch (job) interrupt.	RW	0

Table 115: PISP_BE_INTERRUPT_STATUS register definition.

PISP_BE_AXI				**0x0020**
Configuration for AXI4 auxiliary fields. Controls AXI read/write requests for all tile parameters and image buffers. Only AWQOS and ARQOS may be changed while PiSP Back End is active. Note that the AWPROT and ARPROT fields may be masked by the hardware implemention.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31	Reserved	-		
30:28	AWPROT	Value for write permissions, where used.	RW	0
27:24	AWCACHE	Value for write cache flags, where used.	RW	0
23	MODIFY_ARID	Test bit; should always be clear in normal use.	RW	0
22	ENABLE_BID	Use BID to match responses (should always be set when MINOR_VERSION = 1; ignored by other versions).	RW	0
21	ENABLE_AWID	Vary AWID by channel (should always be set when MI-NOR_VERSION = 1; ignored by other versions).	RW	0
20	BURST_TRIM	Use more efficient write bursts (should always be set when MINOR_VERSION = 1; ignored by other versions).	RW	0
19:16	AWQOS	Value for write QoS field, where used.	RW	0
15	Reserved	-		
14:12	ARPROT	Value for read permissions, where used.	RW	0
11:8	ARCACHE	Value for read cache flags, where used.	RW	0
7:4	Reserved	-		
3:0	ARQOS	Value for read QoS field, where used.	RW	0

Table 116: PISP_BE_AXI register definition.

7.5.2 Input/Output Configuration Registers

These registers define the memory addresses where the PiSP Back End reads pixels from, and where it writes output pixels. They are typically updated by firmware on every frame. The Back End does not output any statistics, though it may generate some "intermediate" images which are fed back into certain blocks on subsequent frames (for example, Temporal Denoise).

Addresses in the Back End are treated in the same way as in the Front End. That is, addresses for reading pixel data in memory must be aligned to a multiple of 4 bytes, and for writing addresses must be a multiple of 16 bytes.

PISP_BE_IO_INPUT_ADDR[n][m] **0x0040**

Memory address from which the Input block (either at the head of the Bayer or RGB pipes) reads pixel data. When $m = 0$ the register gives the low 32 bits of the address, and for $m = 1$ the high 32 bits. n takes the values 0, 1 or 2 for each plane of pixel data that may be read. When the INPUT bit of PISP_BE_BAYER_ENABLE is set, the registers are ignored for $n = 1, 2$. When the INPUT bit of PISP_BE_RGB_ENABLE is set, and a multi-planar format is specified, then data will be read from these addressses.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	ADDRESS	32 bits of the memory address, of which the two least significant bits must be zero when $m = 0$ (that is, the least significant 32 bits of the address).	RW	0

Table 117: PISP_BE_IO_INPUT_ADDR[n][m] register definition.

PISP_BE_IO_TDN_INPUT_ADDR_LO **0x0058**

Low 32 bits of the memory address from which the Temporal Denoise block (TDN) reads pixel data.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	ADDRESS_LO	Low 32 bits of the memory address, of which the low of which the two least significant bits must be zero.	RW	0

Table 118: PISP_BE_IO_TDN_INPUT_ADDR_LO register definition.

PISP_BE_IO_TDN_INPUT_ADDR_HI **0x005C**

High 32 bits of the memory address from which the Temporal Denoise block (TDN) reads pixel data.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 119: PISP_BE_IO_TDN_INPUT_ADDR_HI register definition.

PISP_BE_IO_STITCH_INPUT_ADDR_LO				**0x0060**
Low 32 bits of the memory address from which the Stitch block reads pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_LO	Low 32 bits of the memory address, of which the low of which the two least significant bits must be zero.	RW	0

Table 120: PISP_BE_IO_STITCH_INPUT_ADDR_LO register definition.

PISP_BE_IO_STITCH_INPUT_ADDR_HI				**0x0064**
High 32 bits of the memory address from which the Stitch block reads pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 121: PISP_BE_IO_STITCH_INPUT_ADDR_HI register definition.

Note that images read directly into the RGB pipe (by-passing the Bayer pipe entirely) share their input address registers with the Bayer input address registers PISP_BE_IO_INPUT_ADDR_HI/LO. Both inputs cannot be active at once.

The output address registers can be set to zero to cause the output block in question to drop pixels and not write them to memory. The rest of the pipeline, including signalling interrupts, all operates as normal.

PISP_BE_IO_TDN_ADDR_LO				**0x0068**
Low 32 bits of the memory address to which the Temporal Denoise block (TDN) writes pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 122: PISP_BE_IO_TDN_ADDR_LO register definition.

PISP_BE_IO_TDN_ADDR_HI				**0x006C**
High 32 bits of the memory address to which the Temporal Denoise block (TDN) writes pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 123: PISP_BE_IO_TDN_ADDR_HI register definition.

PISP_BE_IO_STITCH_ADDR_LO				**0x0070**
Low 32 bits of the memory address to which the Stitch block writes pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 124: PISP_BE_IO_STITCH_ADDR_LO register definition.

PISP_BE_IO_STITCH_ADDR_HI				**0x0074**
High 32 bits of the memory address to which the Stitch block writes pixel data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 125: PISP_BE_IO_STITCH_ADDR_HI register definition.

PISP_BE_IO_OUTPUT0_ADDR[n][m]				**0x0078**
Memory address to which Output block 0 writes pixel data. This value specifies the low 32 bits of the address for $m = 0$, and the high 32 bits for $m = 1$. n can take the values 0, 1 or 2 in case the output is to be written in multiple planes (otherwise the registers for higher n values are ignored).				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS	Memory address. Bits 3:0 should be zero when $m = 0$.	RW	0

Table 126: PISP_BE_IO_OUTPUT0_ADDR[n][m] register definition.

PISP_BE_IO_OUTPUT1_ADDR[n][m]				**0x0090**
Memory address to which Output block 1 writes pixel data. This value specifies the low 32 bits of the address for $m = 0$, and the high 32 bits for $m = 1$. n can take the values 0, 1 or 2 in case the output is to be written in multiple planes (otherwise the registers for higher n values are ignored).				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS	Memory address. Bits 3:0 should be zero when $m = 0$.	RW	0

Table 127: PISP_BE_IO_OUTPUT1_ADDR[n][m] register definition.

PISP_BE_IO_HOG_ADDR_LO				**0x00A8**
Low 32 bits of the memory address to which the HOG block writes data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_MID	Bits 31:4 of the memory address.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 128: PISP_BE_IO_HOG_ADDR_LO register definition.

PISP_BE_IO_HOG_ADDR_HI				**0x00AC**
High 32 bits of the memory address to which the HOG block writes data.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	ADDRESS_HI	High 32 bits of memory address.	RW	0

Table 129: PISP_BE_IO_HOG_ADDR_HI register definition.

7.5.3 Global Configuration Registers

These registers affect the global behaviour of the Back End pipeline.

Each Back End block can be enabled or disabled. In general, a disabled block passes its input unmodified to its output, though the Demosaic block, if disabled, duplicates the input pixel to all three outputs. The configuration registers for a disabled block remain accessible, but have no effect until the block is re-enabled.

PISP_BE_GLOBAL_BAYER_ENABLE				**0x00B0**
Register containing enable bits for all the Bayer Pipeline blocks in the Back End.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:23	-	Reserved.	-	-
22	DEMOSAIC	Enable for Demosaic block.	RW	0
21	DEBIN	Enable for Debin block.	RW	0
20	CAC	Enable for CAC (Chromatic Aberration Correction) block.	RW	0
19	TONEMAP	Enable for Tonemap block.	RW	0
18	LSC	Enable for LSC (Lens Shading Correction) block.	RW	0
17	CDN	Enable for CDN (Colour Denoise) block.	RW	0
16	WBG	Enable for WBG (White Balance Gain) block.	RW	0

Continued on next page…

PISP_BE_GLOBAL_BAYER_ENABLE (continued) **0x00B0**

Bits	***Name***	***Description***	***R/W***	***Reset***
15	STITCH_OUTPUT	Enable for Stitch Ouptut block.	RW	0
14	STITCH_COMPRESS	Enable for Stitch Compress block.	RW	0
13	STITCH	Enable for Stitch block.	RW	0
12	STITCH_DECOMPRESS	Enable for Stitch Decompress block.	RW	0
11	STITCH_INPUT	Enable for Stitch Input block.	RW	0
10	BLC	Enable for BLC (Black Level Correction) block.	RW	0
9	SDN	Enable for SDN (Spatial Denoise) block.	RW	0
8	TDN_OUTPUT	Enable for TDN (Temporal Denoise) Output block.	RW	0
7	TDN_COMPRESS	Enable for TDN (Temporal Denoise) Compress block.	RW	0
6	TDN	Enable for TDN (Temporal Denoise) block.	RW	0
5	TDN_DECOMPRESS	Enable for the TDN (Temporal Denoise) Decompress block.	RW	0
4	TDN_INPUT	Enable for TDN (Temporal Denoise) Input block.	RW	0
3	GEQ	Enable for GEQ (Green Equalisation) block.	RW	0
2	DPC	Enable for DPC (Defective Pixel Correction) block.	RW	0
1	DECOMPRESS	Enable for Decompression block.	RW	0
0	INPUT	Enable for Input block.	RW	0

Table 130: PISP_BE_GLOBAL_BAYER_ENABLE register definition.

PISP_BE_GLOBAL_RGB_ENABLE **0x00B4**

Register containing enable bits for all the RGB Pipeline blocks in the Back End. Note that the downscale block is not available on both output branches in all hardware variants.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:22	-	Reserved.	-	-
21	HOG	Enable for HOG Ouptut block.	RW	0
20	-	Reserved.	-	-
19	OUTPUT1	Enable for Output block on output branch 1.	RW	0

Continued on next page. . .

PISP_BE_GLOBAL_RGB_ENABLE (continued) **0x00B4**

Bits	*Name*	*Description*	*R/W*	*Reset*
18	OUTPUT0	Enable for Output block on output branch 0.	RW	0
17	-	Reserved.	-	-
16	RESAMPLE1	Enable for Resample block on output branch 1.	RW	0
15	RESAMPLE0	Enable for Resample block on output branch 0.	RW	0
14	-	Reserved.	-	-
13	DOWNSCALE1	Enable for Downscale block on output branch 1.	RW	0
12	DOWNSCALE0	Enable for Downscale block on output branch 0.	RW	0
11	-	Reserved.	-	-
10	CSC1	Enable for CSC (Colour Space Conversion) block on output branch 1.	RW	0
9	CSC0	Enable for CSC (Colour Space Conversion) block on output branch 0.	RW	0
8	GAMMA	Enable for Gamma block.	RW	0
7	YCBCR_INVERSE	Enable for YCbCr inverse conversion block.	RW	0
6	-	Reserved.	-	-
5	SHARPEN	Enable for the Sharpen block.	RW	0
4	FALSE_COLOUR	Enable for False Colour Suppression block.	RW	0
3	YCBCR	Enable for YCbCr conversion block.	RW	0
2	SAT_CONTROL	Enable for Saturation Control block.	RW	0
1	CCM	Enable for CCM (Colour Correction Matrix) block.	RW	0
0	INPUT	Enable for RGB Input block.	RW	0

Table 131: PISP_BE_GLOBAL_RGB_ENABLE register definition.

PISP_BE_GLOBAL_BAYER_ORDER **0x00B8**

Register defining the Bayer Order, and used by all blocks in the Back End pipeline.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:2	-	Reserved.	-	-
1:0	ORDER	Bayer order, either 0x0 (RGGB), 0x1 (GBRG), 0x2 (BGGR) or 0x3 (GRBG).	RW	0

Table 132: PISP_BE_GLOBAL_BAYER_ORDER register definition.

7.5.4 Input Configuration Registers

The Back End Input block can pass pixels either to the start of the Bayer pipe (reading single channel image formats), or it can pass them directly to the start of the RGB pipe, by-passing the Bayer pipe altogether (and reading three channel image formats). The following pixel formats are accepted, and they must be read from memory.

- 16-bit pixel values from memory. The pixels are read in little-endian format and may optionally be left shifted by up to 8 bits.
- Compressed format from memory. This is the PiSP proprietary RAW compression format. These values are simply passed downstream to the Decompression block.
- When used for reading directly into the RGB pipe, various 3-channel image formats are supported (see Table 3).

Its line-to-line stride (pitch) must be a non-negative multiple of 16 bytes. The address of the image is given by the PISP_BE_IO_INPUT_ADDR_LO and PISP_BE_IO_INPUT_ADDR_HI registers.

The registers below define the behaviour of the block.

PISP_BE_INPUT_SIZE **0x00BC**

Dimensions of the input image.

Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being received. This number must be $even$ when being used to read into the Bayer part of the pipe.	RW	0
15:0	WIDTH	Width of the image being received. This number must be $even$ when being used to read into the Bayer part of the pipe.	RW	0

Table 133: PISP_BE_INPUT_SIZE register definition.

PISP_BE_INPUT_FORMAT **0x00C0**

Format definition of the input image. Not all combinations are allowed; the value must correspond to one of the formats listed in Table 2 or Table 3

Bits	***Name***	***Description***	***R/W***	***Reset***
31	-	Reserved.	-	-
30	THREE_CHANNEL	This bit must be 0 for Bayer input, or 1 for RGB/YUV.	RW	0
29	WALLPAPER	Indicates a YUV image is in a 'Wallpaper' layout in memory.	RW	0
28:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE.	Indicates the compression mode used for Bayer input.	-	-
23:21	-	Reserved.	-	-

Continued on next page...

PISP_BE_INPUT_FORMAT (continued)				**0x00C0**
Bits	***Name***	***Description***	***R/W***	***Reset***
20	BPP_32	Indicates that RGB pixels have been padded to 32 bpp (only when MINOR_VERSION ≥ 1).	RW	0
19:16	SHIFT	Pixel values are to be left-shifted by this amount, for $0 \leq SHIFT \leq 8$.	RW	0
15:14	-	Reserved.	-	-
12	ORDER_SWAPPED	Indicates that RGB or YUV components are in an alternate order.	-	-
11:10	-	Reserved.	-	
9:8	SAMPLING	0 = 4:4:4 sampling, 1 = 4:2:2 sampling, 2 = 4:2:0 sampling.	RW	0
7:6	-	Reserved.	-	-
5:4	PLANARITY	0 = fully interleaved, 1 = semi-planar (only the first channel in a separate plane), 2 = fully planar.	RW	0
3:2	-	Reserved.	-	-
1:0	BITS_PER_SAMPLE	Number of bits in each sample. Valid settings are: 0 (8 bps), 1 (10 bps) and 3 (16 bps). The value 2 is reserved and should not be used.		

Table 134: PISP_BE_INPUT_FORMAT register definition.

PISP_BE_INPUT_STRIDE				**0x00C4**
Line to line stride of the first plane pixel data in memory.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 135: PISP_BE_INPUT_STRIDE register definition.

PISP_BE_INPUT_STRIDE2				**0x00C8**
Line to line stride of the second and third planes of pixel data, if the image being read consists of more than one plane of data. When there is only a single plane of data, this register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 136: PISP_BE_INPUT_STRIDE2 register definition.

Tile Parameters

The Input block uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
input_addr_offset	4	4	Offset in bytes into the input image from where the top left pixel in the tile is read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended. This field is used by the Input block both when reading into the Bayer pipe and when reading into the RGB pipe.
input_addr_offset2	8	4	Offset in bytes into the 2nd and 3rd image planes from where those planes are read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended. This field is used only when reading into the RGB pipe, and only when the input image has a planar (non-interleaved) format.
input_offset_x	12	2	Horizontal offset in pixels of this tile in the input image.
input_offset_y	14	2	Vertical offset in pixels of this tile in the input image.
input_width	16	2	Width in pixels of this tile as it is read from the input image.
input_height	18	2	Height in pixels of this tile as it is read from the input image.

Table 137: Tile parameters for the Input block.

The input address offsets listed above are added to the memory address given by PISP_BE_IO_INPUT_ADDR[n][m], where n indicates the colour channel for planar formats, and m indicates the high ($m = 1$) or low ($m = 0$) 32 bits of the address. *input_addr_offset* is added to the address formed by PISP_BE_IO_INPUT_ADDR[0], and *input_addr_offset2* is added to the address formed by PISP_BE_IO_INPUT_ADDR[1] and PISP_BE_IO_INPUT_ADDR[2].

The *input_width* and *input_height* inform the Input block how many pixels it is expected to read. The *input_offset_x/y* values are in fact not needed by the block, but are provided as a courtesy to help debug any problems.

7.5.5 Decompression Configuration Registers

This block should be enabled only if compressed image data is being read from memory by the Input block (so that block's COMPRESSED bit should be set).

PISP_BE_DECOMPRESS **0x00CC**

Parameters for decompression of PiSP compressed RAW image format data.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:26	-	Reserved.	-	-
25:24	MODE	Select decompression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset value added to all pixels after decompression. Normally used to restore black level if compression was configured to subtract it.	RW	0

Table 138: PISP_BE_DECOMPRESS register definition.

Tile Parameters

The Decompression block has no tile parameters.

7.5.6 DPC (Defective Pixel Correction) Configuration Registers

Back End DPC can be slightly more effective than Front End DPC, as it buffers more lines of the image. It finds the MIN..MAX range of values of 8 neighbouring pixels in the same Bayer channel, and extends it by a "margin" which represents the largest acceptable excursion away from neighbouring values. The margin is a function of the MIN and MAX−MIN of the neighbouring pixels. The output is clipped to lie within this range.

PISP_BE_DPC **0x00D0**

Configuration register for Back End DPC block.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:25	-	Reserved.	-	-
24	FOLDBACK	Causes pixels well outside the acceptable excursion to be corrected more aggressively.	RW	0
23:13	-	Reserved.	-	-
12:8	COEFF_RANGE	Coefficient for the range of pixels from this Bayer channel, used to compute the margin.	RW	0
7:5	-	Reserved.	-	-

Continued on next page...

PISP_BE_DPC (continued)				**0x00D0**
Bits	***Name***	***Description***	***R/W***	***Reset***
4:0	COEFF_LEVEL	Coefficient for the darkest neighbouring pixel value, used to compute the margin.	RW	0

Table 139: PISP_BE_DPC register definition.

Tile Parameters

The DPC block has no tile parameters.

7.5.7 GEQ (Green Equalisation) Configuration Registers

The GEQ block corrects imbalance between the two types of green pixel in a Bayer pattern - the greens on red rows and the greens on blue rows. It does this by estimating the difference between a green pixel and its immediate neighbours of the other green type, and then moves this pixel halfway towards them.

This estimate is generated by one of two filters, a "sharper" version of the filter for normal or well-lit environments, or a "softer" version of the filter for low-light environments, and which may cause less amplification of noise. Finally the estimate is limited by a $threshold$ calculated as $threshold = offset + current\_pixel\_value * slope$, where the $offset$ and $slope$ are programmed into the block's configuration. Before limiting the estimate, the $threshold$ itself is clipped to lie within given min and max values.

PISP_BE_GEQ_CONFIG				**0x00D4**
Configuration for the GEQ block.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31	SHARPER	Set to 1 to use the sharper filter for green imbalance estimation.	RW	0
30:26	-	Reserved.	-	-
25:16	SLOPE	Slope value for the threshold calculation.	RW	0
15:0	OFFSET	Offset value for the threshold calculation.	RW	0

Table 140: PISP_BE_GEQ_CONFIG register definition.

PISP_BE_GEQ_MINMAX				**0x00D8**
Limits for the GEQ threshold value.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	MAX	Maximum value the threshold may have	RW	0
15:0	MIN	Minimum value the threshold may have.	RW	0

Table 141: PISP_BE_GEQ_MINMAX register definition.

Tile Parameters

The GEQ block has no tile parameters.

7.5.8 TDN (Temporal Denoise) Input Configuration Registers

The TDN Input block matches the formats of the Back End (Bayer) Input block, except that the image it reads is passed to the TDN block as its LTA (Long Term Average) image.

Its line-to-line stride (pitch) must be a non-negative multiple of 16 bytes. The address of the image is given by the PISP_BE_IO_TDN_INPUT_ADDR_LO and PISP_BE_IO_TDN_INPUT_ADDR_HI registers.

The registers below define the behaviour of the block.

PISP_BE_TDN_INPUT_SIZE				**0x00DC**
Dimensions of the TDN input image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being received.	RW	0
15:0	WIDTH	Width of the image being received.	RW	0

Table 142: PISP_BE_TDN_INPUT_SIZE register definition.

PISP_BE_TDN_INPUT_FORMAT				**0x00E0**
Format definition of the TDN input image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the input is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are left-shifted by this amount, for $0 \leq SHIFT \leq 8$. Normally zero.	RW	0
15:2	-	Reserved.	-	-

Continued on next page...

PISP_BE_TDN_INPUT_FORMAT (continued) **0x00E0**

Bits	*Name*	*Description*	*R/W*	*Reset*
1:0	BITS_PER_SAMPLE	Number of bits in each sample (pixel). Valid settings are: 0 (8 bps) if COMPRESSION_-MODE is nonzero; otherwise when not compressed the value 3 (16 bps) should be used.	RW	0

Table 143: PISP_BE_TDN_INPUT_FORMAT register definition.

PISP_BE_TDN_INPUT_STRIDE **0x00E4**

Line to line stride of pixel data in memory.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 144: PISP_BE_TDN_INPUT_STRIDE register definition.

PISP_BE_TDN_INPUT_STRIDE2 **0x00E8**

This register is ignored.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:0	-	Reserved.	-	-

Table 145: PISP_BE_TDN_INPUT_STRIDE2 register definition.

Tile Parameters

The TDN Input block uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
tdn_input_addr_offset	20	4	Offset in bytes into the TDN input image from where the LTA (long term average) frame is read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 146: Tile parameters for the TDN Input block.

The *tdn_input_addr_offset* is added to the address formed by the PISP_BE_IO_TDN_INPUT_ADDR_-LO/HI registers. There is only a single address offset because the TDN Input image is of necessity in the Bayer format.

7.5.9 TDN (Temporal Denoise) Decompression Configuration Registers

This block should be enabled only if compressed image data is being read from memory by the TDN Input block (so that block's COMPRESSED bit should be set). The TDN Input and Output blocks can cooperate by using compression to reduce the total amount of memory traffic.

PISP_BE_TDN_DECOMPRESS				**0x00EC**
Parameters for decompression of PiSP compressed RAW image format data for TDN input.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select decompression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset value added to all pixels after decompression. Normally used to restore black level if compression was configured to subtract it.	RW	0

Table 147: PISP_BE_TDN_DECOMPRESS register definition.

Tile Parameters

The TDN Decompression block has no tile parameters.

7.5.10 TDN (Temporal Denoise) Configuration Registers

TDN implements a 1-tap IIR temporal filter using the new input frame and the output frame that it generated when it processed the previous input frame (read by the TDN Input block). This previous output frame is referred to as the LTA, or Long Term Average, frame. Every time TDN runs it receives both the new input frame and the previous LTA frame as its input, and outputs a new LTA frame. Normally this new LTA frame is written back to memory (for reading next time) by the TDN Output block.

The TDN filter operates mostly per-pixel, though spatial averaging is employed to get tighter noise estimates, and in turn this reduces the amount of frame-to-frame ghosting. The noise estimate is given by $noise\_constant + noise\_slope * \sqrt{pixel\_value}$.

TDN signals to the next downstream block (SDN, or Spatial Denoise) whether it applied any temporal denoise using the bottom bit (the LSB) of each pixel. When TDN successfully applied temporal denoise this bottom bit is set to 1, and when TDN had to back off because the pixel appears genuinely to be changing, then this bit is set to 0. TDN marks this bit according to whether the amount of temporal denoise applied, given by the blending factor between the new and previous LTA pixels, is greater than a $threshold$. SDN is free to process temporally denoised pixels differently (typically, less aggressively) than undenoised ones.

When the first frame passes through TDN, meaning also that there is no previous LTA frame to use, then the block should be put into *reset* mode, by setting the RESET bit. This causes TDN to copy the new frame pixels to the output whilst also clearing the bottom bit, thereby telling SDN that no temporal denoise has been applied.

PISP_BE_TDN_CONFIG1				**0x00F0**
TDN configuration parameters.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	RATIO	The LTA input frame is multiplied by this U2.14 value so that TDN can still operate even when there are (slight) exposure changes happening.	RW	0
15:0	BLACK_LEVEL	Black level value that is subtracted from pixels purely for the generation of a noise estimate (the output pixels themselves do not have their black levels altered).	RW	0

Table 148: PISP_BE_TDN_CONFIG1 register definition.

PISP_BE_TDN_CONFIG_NOISE				**0x00F4**
TDN noise estimation parameters.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	NOISE_SLOPE	U8.8 value use in the $noise\_constant + noise\_slope * \sqrt{pixel\_value}$ noise estimation.	RW	0
15:0	NOISE_CONSTANT	Constant offset value used in the noise estimation.	RW	0

Table 149: PISP_BE_TDN_CONFIG_NOISE register definition.

PISP_BE_TDN_CONFIG2				**0x00F8**
TDN configuration parameters.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:17	-	Reserved.	-	-
16	RESET	When this bit is set TDN does not do any temporal denoise at all and clears the bottom bit of every output pixel.	RW	0
15:0	THRESHOLD	When TDN uses more than this threshold of the LTA pixel (rather than the pixel in the new frame) then TDN marks the bottom bit of the output pixel with a 1; otherwise it marks it with a zero.	RW	0

Table 150: PISP_BE_TDN_CONFIG2 register definition.

Tile Parameters

The TDN block has no tile parameters.

7.5.11 TDN (Temporal Denoise) Compression Configuration Registers

The output of the TDN block, also known as the LTA (or Long Term Average) frame can be compressed by the TDN Compression block before being written to memory.

If this block is enabled the COMPRESSED bit in the TDN Output block should be set. This will cause the output image to be compressed to 8 bits per pixel.

PISP_BE_TDN_COMPRESS **0x00FC**

Parameters for compression of PiSP compressed RAW image format data output by the TDN Output block.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:26	-	Reserved.	-	-
25:24	MODE	Select decompression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset subtracted from pixels before compression. This may improve fidelity when there is a high black level; it is recommended to set it to a multiple of 512 below the lowest expected pixel value.	RW	0

Table 151: PISP_BE_TDN_COMPRESS register definition.

Tile Parameters

The TDN Compression block has no tile parameters.

7.5.12 TDN (Temporal Denoise) Output Configuration Registers

The TDN Output block writes the TDN LTA frame to memory. It supports only two image formats.

- 16-bit pixel values. The pixels are written in little-endian format and may optionally be right shifted by up to 8 bits.
- Compressed format. This is the PiSP proprietary RAW compression format.

Addresses in PiSP have 64 bits, but depending on the target chip some high bits may be ignored. The address at which to write pixel data in memory must be aligned to a multiple of 16 bytes. Its line to line stride must also be a non-negative multiple of 16 bytes.

If both output address registers (ADDR_LO and ADDR_HI) are set to zero, the block silently drops all the pixels without writing them anywhere.

PISP_BE_TDN_OUTPUT_SIZE **0x0100**

Dimensions of the output image.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	HEIGHT	Height of the image being written.	RW	0
15:0	WIDTH	Width of the image being written. Must be even.	RW	0

Table 152: PISP_BE_TDN_OUTPUT_SIZE register definition.

PISP_BE_TDN_OUTPUT_FORMAT **0x0104**

Format definition of the output image.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the output is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are right-shifted by this amount if COMPRESSION_MODE is zero. Normally zero.	RW	0
15:2	-	Reserved.	-	-
1:0	BITS_PER_SAMPLE	Set this to 0 (8 bps) if compressed or 3 (16 bps) if uncompressed.	RW	0

Table 153: PISP_BE_TDN_OUTPUT_FORMAT register definition.

PISP_BE_TDN_OUTPUT_STRIDE **0x0108**

Stride of pixel data in memory.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 154: PISP_BE_TDN_OUTPUT_STRIDE register definition.

PISP_BE_TDN_OUTPUT_STRIDE2				**0x010C**
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 155: PISP_BE_TDN_OUTPUT_STRIDE2 register definition.

Tile Parameters

The TDN Output block uses the following tile parameters.

Field	Offset	Size	Description
tdn_output_addr_offset	24	4	Offset in bytes into the TDN output image where the new LTA (long term average) frame is written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 156: Tile parameters for the TDN Output block.

The *tdn_output_addr_offset* is added to the address formed by the PISP_BE_IO_TDN_OUTPUT_ADDR_-LO/HI registers. There is only a single address offset because the TDN Output image is of necessity in the Bayer format.

7.5.13 SDN(Spatial Denoise) Configuration Registers

The SDN block implements a simplified version of a non-local means (NLM) filter for which it requires only a noise estimate for each pixel, calculated as $noise\_constant + noise\_slope * \sqrt{pixel\_value}$. SDN actually has two pairs of $noise\_constant$ and $noise\_slope$ values, one pair is used for pixels with a 1 as the LSB, and a second pair for pixels with a zero as the LSB. This allows the block to interact better with TDN. A second, usually higher, noise estimate can be used where TDN has failed to denoise pixels, and this helps hide "noise trails" when there is motion between images. When TDN is not enabled, it is usual to program both pairs to the same values.

Finally, SDN can mix a proportion of the original (undenoised) pixel back into the final result, and this can be desirable when it is felt that images look more natural when not over-denoised.

PISP_BE_SDN_CONFIG				**0x0110**
SDN configuration parameters.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	-	Reserved.	-	-
23:16	LEAKAGE	Proportion (out of 256) of the original undenoised value to mix back into the denoised output pixel.	RW	0

Continued on next page...

PISP_BE_SDN_CONFIG (continued)				**0x0110**
Bits	***Name***	***Description***	***R/W***	***Reset***
15:0	BLACK_LEVEL	Black level subtracted from pixels in orer to calculate the noise estimate; the black levels of output pixels are *not* altered.	RW	0

Table 157: PISP_BE_SDN_CONFIG register definition.

PISP_BE_SDN_NOISE				**0x0114**
First pair of SDN noise estimation parameters, used for input pixels with LSB of 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	NOISE_SLOPE	$noise\_slope$ value used for noise estimation.	RW	0
15:0	NOISE_CONSTANT	$noise\_contstant$ value used for noise estimation.	RW	0

Table 158: PISP_BE_SDN_NOISE register definition.

PISP_BE_SDN_NOISE2				**0x0118**
Second pair of SDN noise estimation parameters, used for input pixels with LSB of 0.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	NOISE_SLOPE	$noise\_slope$ value used for noise estimation.	RW	0
15:0	NOISE_CONSTANT	$noise\_contstant$ value used for noise estimation.	RW	0

Table 159: PISP_BE_SDN_NOISE2 register definition.

Tile Parameters

The SDN block has no tile parameters.

7.5.14 BLC (Black Level Correction) Configuration Registers

The BLC block aligns the black levels of all the Bayer channels so that they are the same value. In normal usage, the desired output value is set to zero. In functional terms, the block is otherwise identical to the Front End BLA block.

PISP_BE_BLC_INPUT_RED				**0x011C**
Input black levels for alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	GR	Current black level of green pixels on green-red rows.	RW	0
15:0	R	Current black level of red pixels.	RW	0

Table 160: PISP_BE_BLC_INPUT_RED register definition.

PISP_BE_BLC_INPUT_BLUE				**0x0120**
Input black levels for alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	B	Current black level of blue pixels.	RW	0
15:0	GB	Current black level of green pixels on green-blue rows.	RW	0

Table 161: PISP_BE_BLC_INPUT_BLUE register definition.

PISP_BE_BLC_OUTPUT				**0x0124**
Output black level after alignment.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	-	Reserved.	RW	0
15:0	LEVEL	Output black level of all pixels after alignment.	RW	0

Table 162: PISP_BE_BLC_OUTPUT register definition.

Tile Parameters

The BLC block has no tile parameters.

7.5.15 Stitch Compression Configuration Registers

The input to the Stitch can optionally be compressed before being written to memory.

If this block is enabled the COMPRESSED bit in the Stitch Output block should be set. This will cause the output image to be compressed to 8 bits per pixel.

PISP_BE_STITCH_COMPRESS				**0x0128**
Parameters for compression of PiSP compressed RAW image format data output by the Stitch Output block.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select compression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset subtracted from pixels before compression. This may improve fidelity when there is a high black level; it is recommended to set it to a multiple of 512 below the lowest expected pixel value.	RW	0

Table 163: PISP_BE_STITCH_COMPRESS register definition.

Tile Parameters

The Stitch Compression block has no tile parameters.

7.5.16 Stitch Output Configuration Registers

The Stitch Output block writes the input frame to the Stitch Block to memory. It supports only two image formats.

- 16-bit pixel values. The pixels are written in little-endian format and may optionally be right shifted by up to 8 bits.
- Compressed format. This is the PiSP proprietary RAW compression format.

Addresses in PiSP have 64 bits, but depending on the target chip some high bits may be ignored. The address at which to write pixel data in memory must be aligned to a multiple of 16 bytes. Its line to line stride must also be a non-negative multiple of 16 bytes.

If both output address registers (ADDR_LO and ADDR_HI) are set to zero, the block silently drops all the pixels without writing them anywhere.

PISP_BE_STITCH_OUTPUT_SIZE				**0x012C**
Dimensions of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being written.	RW	0
15:0	WIDTH	Width of the image being written. Must be even.	RW	0

Table 164: PISP_BE_STITCH_OUTPUT_SIZE register definition.

PISP_BE_STITCH_OUTPUT_FORMAT				**0x0130**
Format definition of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the output is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are right-shifted by this amount if COMPRESSION_MODE is zero, for 0 <= SHIFT <= 8. Normally zero.	RW	0
15:2	-	Reserved.	-	-
1:0	BITS_PER_SAMPLE	Set this to 0 (8 bps) if compressed or 3 (16 bps) if uncompressed.	RW	0

Table 165: PISP_BE_STITCH_OUTPUT_FORMAT register definition.

PISP_BE_STITCH_OUTPUT_STRIDE				**0x0134**
Stride of pixel data in memory.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 166: PISP_BE_STITCH_OUTPUT_STRIDE register definition.

PISP_BE_STITCH_OUTPUT_STRIDE2				**0x0138**
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 167: PISP_BE_STITCH_OUTPUT_STRIDE2 register definition.

Tile Parameters

The Stitch Output block uses the following tile parameters.

Field	Offset	Size	Description
stitch_output_addr_offset	24	4	Offset in bytes into the Stitch output image where the other frame is written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 168: Tile parameters for the Stitch Output block.

The *stitch_output_addr_offset* is added to the address formed by the PISP_BE_IO_STITCH_OUTPUT_-ADDR_LO/HI registers. There is only a single address offset because the Stitch Output image is of necessity in the Bayer format.

7.5.17 Stitch Input Configuration Registers

The Stitch Input block matches the formats of the Back End (Bayer) Input block, except that the image it reads is passed to the Stitch block.

Its line-to-line stride (pitch) must be a non-negative multiple of 16 bytes. The address of the image is given by the PISP_BE_IO_STITCH_INPUT_ADDR_LO and PISP_BE_IO_STITCH_INPUT_ADDR_HI registers.

The registers below define the behaviour of the block.

PISP_BE_STITCH_INPUT_SIZE				**0x013C**
	Dimensions of the Stitch input image.			
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image being received.	RW	0
15:0	WIDTH	Width of the image being received.	RW	0

Table 169: PISP_BE_STITCH_INPUT_SIZE register definition.

PISP_BE_STITCH_INPUT_FORMAT				**0x0140**
	Format definition of the Stitch input image.			
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	COMPRESSION_MODE	Set to a nonzero compression mode if the output is compressed.	RW	0
23:20	-	Reserved.	-	-
19:16	SHIFT	Pixel values are left-shifted by this amount, for $0 \leq SHIFT \leq 8$. Normally zero.	RW	0
15:2	-	Reserved.	-	-

Continued on next page...

PISP_BE_STITCH_INPUT_FORMAT (continued)				**0x0140**
Bits	***Name***	***Description***	***R/W***	***Reset***
1:0	BITS_PER_SAMPLE	Number of bits in each sample (pixel). Valid settings are: 0 (8 bps) if COMPRESSION_-MODE is nonzero; otherwise when not compressed the value 3 (16 bps) should be used.	RW	0

Table 170: PISP_BE_STITCH_INPUT_FORMAT register definition.

PISP_BE_STITCH_INPUT_STRIDE				**0x0144**
Line to line stride of pixel data in memory.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 171: PISP_BE_STITCH_INPUT_STRIDE register definition.

PISP_BE_STITCH_INPUT_STRIDE2				**0x0148**
This register is ignored.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:0	-	Reserved.	-	-

Table 172: PISP_BE_STITCH_INPUT_STRIDE2 register definition.

Tile Parameters

The Stitch Input block uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
stitch_input_addr_offset	24	4	Offset in bytes into the Stitch input image where the other image is read. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 173: Tile parameters for the Stitch Input block.

The *stitch_output_addr_offset* is added to the address formed by the PISP_BE_IO_STITCH_OUTPUT_-ADDR_LO/HI registers. There is only a single address offset because the Stitch Output image is of necessity in the Bayer format.

7.5.18 Stitch Decompression Configuration Registers

This block should be enabled only if compressed image data is being read from memory by the Stitch Input block (so that block's COMPRESSED bit should be set). The Stitch Input and Output blocks can cooperate by using compression to reduce the total amount of memory traffic.

PISP_BE_STITCH_DECOMPRESS				**0x014C**
Parameters for decompression of PiSP compressed RAW image format data for Stitch input.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:26	-	Reserved.	-	-
25:24	MODE	Select decompression mode. Legal values are 1, 2 and 3. The normal compression scheme is mode 1.	RW	0
23:16	-	Reserved.	-	-
15:0	OFFSET	Offset value added to all pixels after decompression. Normally used to restore black level if compression was configured to subtract it.	RW	0

Table 174: PISP_BE_STITCH_DECOMPRESS register definition.

Tile Parameters

The Stitch Decompression block has no tile parameters.

7.5.19 Stitch Configuration Registers

The Stitch block merges, or "stitches", together two images of different exposures, and is used in conjunction with the subsequent Tonmap block for generating HDR images.

The block takes two input images. One is the current frame that is travelling down the pipeline, and the other is normally the previous frame, of a different exposure, that has been read by the Stitch Input block. As well as merging these two frames, the Stitch Output block writes the current frame to memory so that it can be re-read by the Stitch Input block on the next frame as the next "previous frame of a different exposure".

Pixels in the long exposure image can, if they are below a *low* threshold value, be used in the stitched output image. If they are above a *high* threshold value then the value from the short exposure image must be used instead, and between these two thresholds the pixels are blended together proportionately. Pixel values from the long exposure image must be multiplied by the *exposure ratio*, so that they match the dynamic range of the short exposure pixels.

One additional problem when stitching images is that of motion. If the (exposure corrected) long and short pixels differ by more than a *motion threshold* then the short pixel will be used instead. This limits the extent to which objects in motion can be suddently "chopped off" if the image there is getting brighter.

PISP_BE_STITCH_THRESHOLD				**0x0150**
Thresholds for deciding when to use long or short exposure pixels.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:20	-	Reserved.	-	-
19:16	DIFF_POWER	The low and high thresholds must be separated by a number that is an exact power of 2, given in these bits. When long exposure pixels lie at or above the value LO + (1«DIFF_POWER) then the short exposure pixels are used instead.	RW	0
15:0	LO	Low threshold value. When long exposure pixels are below this value then they are used in place of short exposure pixels.	RW	0

Table 175: PISP_BE_STITCH_THRESHOLD register definition.

PISP_BE_STITCH_CONFIG				**0x0154**
Configuration parameter for the Stitch block.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	MOTION_THRESHOLD_RECIP	Reciprocal of the MOTION_THRESHOLD_256 value. It should not be set larger than 255/MOTION_THRESHOLD_256 + 1 (or 255 if MOTION_THRESHOLD_256 is zero).	RW	0
23:16	MOTION_THRESHOLD_256	Motion threshold above which short exposure pixels are always used, given in units of 256.	RW	0
15	STREAMING_LONG	Set to one if the “streaming input” (the pixels travelling down the pipeline) is the long exposure image. Set to zero if the image read by the Stitch Input block is the long exposure image.	RW	0
14:0	EXPOSURE_RATIO	U0.15 multiplier that will convert long exposure pixels into the same dynamic range as short exposure pixels.	RW	0

Table 176: PISP_BE_STITCH_CONFIG register definition.

Tile Parameters

The Stitch block has no tile parameters.

7.5.20 LSC (Lens Shading Correction) Configuration Registers

The LSC block divides the entire image into a 32x32 grid, and at each vertex of the grid defines separate gains for the red, green and blue channels of the Bayer image. Each of the 32x32 rectangles is referred to as a *cell*.

Pixels passing through this block have the correct gain applied, depending on which of the Bayer channels they are. Where pixels do not lie exactly on a vertex of the grid, bilinear interpolation is used to generate the necessary gains.

PISP_BE_LSC_GRID **0x0158**

Reciprocals of the cell size.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	GRID_STEP_Y	$(1 << 18)/cell\_height$. Note that the value 0 is disallowed here.	RW	0
15:0	GRID_STEP_X	$(1 << 18)/cell\_width$. Note that the value 0 is disallowed here.	RW	0

Table 177: PISP_BE_LSC_GRID register definition.

PISP_BE_LSC_GAINS[n] **0x015C$+4n$**

Jointly-coded RGB gains for each LSC grid vertex, for $n = 0, ..., 1088$.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:30	RANGE	Range containing all R,G,B gains for this vertex. This selects one of [0.5–1.5), [0–2), [0–4), [0–8).	RW	0
29:20	BLUE	Blue gain expressed as a fractional position within the above range.	RW	0
19:10	GREEN	Green gain expressed as a fractional position within the above range.	RW	0
9:0	RED	Red gain expressed as a fractional position within the above range.	RW	0

Table 178: PISP_BE_LSC_GAINS[n] register definition.

Tile Parameters

The LSC block uses the following tile parameters.

Field	Offset	Size	Description
lsc_grid_offset_x	36	4	Horizontal offset into the LSC table where this tile begins. This is a U5.18 format number in units of the LSC cell width.

Field	Offset	Size	Description
lsc_grid_offset_y	40	4	Vertical offset into the LSC table where this tile begins. This is a U5.18 format number in units of the LSC cell height.

Table 179: Tile parameters for the LSC block.

Both values are given as U5.18 format numbers in units of the LSC cell size. So a value of (1«19) would indicate that the first pixel of this tile starts at exactly the beginning of the 2nd cell (out of the 32 available cells).

7.5.21 WBG (White Balance Gain) Configuration Registers

This block applies a fixed gain to each pixel of the Bayer image depending on which of the channels (R, G or B) it is.

PISP_BE_WBG_GAINS0 **0x1260**

Reciprocals of the cell size.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:30	-	Reserved.	-	-
29:16	GAIN_G	U4.10 gain applied to green pixels.	RW	0
15:14	-	Reserved.	-	-
13:0	GAIN_R	U4.10 gain applied to red pixels.	RW	0

Table 180: PISP_BE_WBG_GAINS0 register definition.

PISP_BE_WBG_GAINS1 **0x1264**

Reciprocals of the cell size.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:14	-	Reserved.	-	-
13:0	GAIN_B	U4.10 gain applied to blue pixels.	RW	0

Table 181: PISP_BE_WBG_GAINS1 register definition.

Tile Parameters

The WBG block has no tile parameters.

7.5.22 CDN (Colour Denoise) Configuration Registers

CDN works on the Bayer image and denoises both R-G and B-G. It uses a wide bilateral filter horizontally and an IIR filter vertically. For simplicity, the noise estimate is taken to be constant, and the strength of the IIR filter relative to the FIR bilateral filter can be adjusted. Finally, the result of denoising R-G and B-G is normally used to modify the R and B channels, though optionally some of the change can be assigned into the G channel, should this prove beneficial.

PISP_BE_CDN_CONFIG **0x1268**

CDN configuration.

Bits	Name	Description	R/W	Reset
31:24	G_ADJUST	The proportion of the change that is assigned to the G rather than the R/B channels.	RW	0
23:16	IIR_STRENGTH	Relative strength, out of 256, of the IIR part of the filter.	RW	0
15:0	THRESH	Constant noise estimate.	RW	0

Table 182: PISP_BE_CDN_CONFIG register definition.

Tile Parameters

The CDN block has no tile parameters.

7.5.23 CAC (Chromatic Aberration Correction) Configuration Registers

The PiSP Back End can correct (lateral) chromatic aberration up to about ± 2 pixels vertically and horizontally by resampling the red and blue components. Not unlike LSC, the frame is divided into an 8×8 grid, and at each of the 9×9 vertices horizontal and vertical shifts are specified for both red and blue. In between the vertices, bilinear interpolation is used to construct the necessary shift values. As before, each rectangle within the grid is referred to as a *cell*. Cell size must exceed 16×16 pixels.

Shifts are signed, in units of 1/32 full pixel (1/64 sample of the appropriate component).

PISP_BE_CAC_GRID **0x126C**

Reciprocals of the cell size.

Bits	Name	Description	R/W	Reset
31:16	GRID_STEP_Y	$(1 << 20)/cell\_height$. Note that the value 0 is disallowed here.	RW	0
15:0	GRID_STEP_X	$(1 << 20)/cell\_width$. Note that the value 0 is disallowed here.	RW	0

Table 183: PISP_BE_CAC_GRID register definition.

PISP_BE_CAC_SHIFTS[n]				**0x1270**$+4n$
Pixel shifts for the CAC grid, for $n = 0, ..., 80$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31	-	Reserved.	-	-
30:24	BY	Vertical shift for the blue channel.	RW	0
23	-	Reserved.	-	-
22:16	BX	Horizontal shift for the blue channel.	RW	0
15	-	Reserved.	-	-
14:8	RY	Vertical shift for the red channel.	RW	0
7	-	Reserved.	-	-
6:0	RX	Horizontal shift for the red channel.	RW	0

Table 184: PISP_BE_CAC_SHIFTS[n] register definition.

Tile Parameters

The CAC block uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
cac_grid_offset_x	44	4	Horizontal offset into the CAC table where this tile begins. This is a U3.20 format number in units of the CAC cell width.
cac_grid_offset_y	48	4	Vertical offset into the CAC table where this tile begins. This is a U3.20 format number in units of the CAC cell height.

Table 185: Tile parameters for the CAC block.

Both values are given as U3.20 format numbers in units of the CAC cell size. So a value of (1«21) would indicate that the first pixel of this tile starts at exactly the beginning of the 2nd cell (out of the 8 available cells).

Note: It would be quite usual to program the CAC cell size to be exactly 4 times the LSC cell size (CAC has 8 cells across the image; LSC has 32). In these circumstances the *lsc_grid_offset_x/y* and *cac_grid_offset_x/y* values would be (respectively) identical, notwithstanding the differing interpretations of which bits are fractional.

7.5.24 Debin Configuration Registers

The Debin block is provided for camera modes that produce *binned* camera data. Often this is *2x2 binning*, meaning that pixels in the same Bayer channel are taken in 2x2 blocks and averaged to produce a single output pixel. Repeating this for each Bayer channel means that every 4x4 set of pixels in the original image is reduced to a single 2x2 Bayer quad. Observe that *within* this new Bayer quad the pixels are still only one (original) pixel apart; but between the new quads there are now an extra 2 pixels of spatial separation.

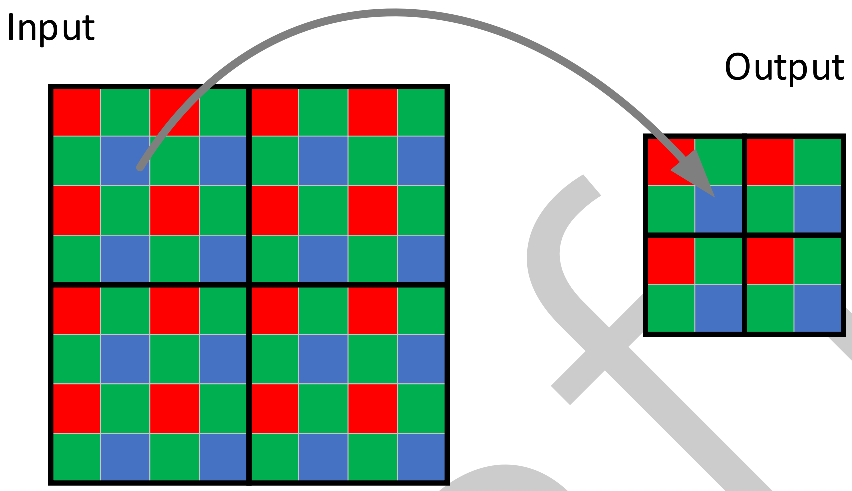

Figure 13: 2x2 binning: every 4x4 input block is converted to a single Bayer quad

When binned camera data like this is passed through a standard demosaicking algorithm - which assumes equal spatial separation between all the pixels - diagonal lines have a tendency to become "jagged". The job of the Debin block is to resample the odd rows and the odd columns to reduce this effect. This is accomplished with a 4-tap FIR filter.

The debinning process can be enabled separately for rows and columns, in case a particular sensor is only binning in one direction.

PISP_BE_DEBIN_COEFFS **0x13B4**

Filter coefficients for debinning.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:24	COEFF3	Fourth filter coefficient in S.7 format.	RW	0
23:16	COEFF2	Third filter coefficient in S.7 format.	RW	0
15:8	COEFF1	Second filter coefficient in S.7 format.	RW	0
7:0	COEFF0	First filter coefficient in S.7 format.	RW	0

Table 186: PISP_BE_DEBIN_COEFFS register definition.

PISP_BE_DEBIN_ENABLE				**0x13B8**
Horizontal and vertical enables for debinning.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:9	-	Reserved.	-	-
8	V_ENABLE	Vertical enable.	RW	0
7:1	-	Reserved.	-	-
0	H_ENABLE	Horizontal enable.	RW	0

Table 187: PISP_BE_DEBIN_ENABLE register definition.

Note: cubic filter coefficients (in this case [-7, 105, 35, -5]) might seem a reasonable choice, but in practice a slightly sharper filter (such as [-13, 111, 41, -11]) can work better.

Tile Parameters

The Debin block has no tile parameters.

7.5.25 Tonemap Configuration Registers

The Tonemap block performs tonemapping on an image. Before doing so, however, it generates a Y (luminance) image Y from the Bayer image B, and separates this into:

1. A low pass Y image, Y_{LP}, using a bilateral filter on Y.
2. A high pass Y signal, given by $Y_{HP} = Y - Y_{LP}$.
3. A colour signal, given by $C = B - Y$.

Next we have a tonemapping operator T which we apply to Y_{LP} to get $T(Y_{LP})$. Finally we reconstruct the output image pixel-wise as

$$B_{out} = T(Y_{LP}) + T(Y_{LP})/Y_{LP} * (C + strength * Y_{HP})$$

where $strength$ is a parameter that can be parameter that can be increased to make the HDR effect look stronger. Note how the way we add back a high pass signal is reminiscent of the standard *unsharp mask* procedure; the use of the bilateral filter (rather than, for example, a plain Gaussian blur) prevents halo-ing at high contrast edges in the final reconstruction.

The threshold for the bilateral filter that generates Y_{LP} is calculated as $detail\_constant + detail\_slope * \sqrt(Y)$, and the filter includes an IIR term vertically. The tone mapping curve is defined by 64 points spaced in the same manner as the Gamma block (*q.v.*), therefore with points concentrated towards the lower end of the range.

PISP_BE_TONEMAP_DETAIL				**0x13BC**
Coefficients for estimating bilateral filter thresholds.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	DETAIL_SLOPE	U8.8 slope value for the threshold calculation.	-	-
15:0	DETAIL_CONSTANT	Constant term for threshold calculation.	RW	0

Table 188: PISP_BE_TONEMAP_DETAIL register definition.

PISP_BE_TONEMAP_STRENGTHS				**0x13C0**
Tonemap configuration strengths.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	STRENGTH	U4.8 $strength$ factor.	-	-
15:12	-	Reserved.	-	-
11:0	IIR_STRENGTH	Relative strength of the vertical IIR filter contribution in U8.4 format.	RW	0

Table 189: PISP_BE_TONEMAP_STRENGTHS register definition.

PISP_BE_TONEMAP_LUT[n]				**0x13C4**$+4n$
Look-up table for tonemap curve, for $n = 0, ..., 63$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SLOPE	Slope of this segment of the tonemap curve.	RW	0
15:0	POSITION	y (output) position of tonemap curve for this x (input) value.	RW	0

Table 190: PISP_BE_TONEMAP_LUT[n] register definition.

Note: the x value of point n in the tonemap curve is given by:

- if $n < 32$, $n * 512$
- otherwise if $n < 48$, $(n - 32) * 1024 + 16384$
- otherwise $(n - 48) * 2048 + 32768)$

Tile Parameters

The Tonemap block has no tile parameters.

7.5.26 Demosaic Configuration Registers

The Demosaic block performs an edge-sensitive adaptive demosaicking operation, with built-in false colour suppression (which is additional to the False Colour block in the RGB pipe).

The algorithm will attempt to sharpen the resulting image using information from other Bayer channels, though this behaviour can be limited where necessary (for example, in very noisy conditions, or where chromatic aberration destroys the correlation between colour channels).

PISP_BE_DEMOSAIC_CONFIG				**0x14C4**
	Demosaic configuration.			
Bits	***Name***	***Description***	***R/W***	***Reset***
31:10	-	Reserved.	-	-
9:8	FC_MODE	Built-in false colour suppression mode. 0 = none, 1 = medium, 2 = stronger, 3 = behaviour undefined.	RW	0
7:0	SHARPER	Use other Bayer channels to increase sharpness. 0 = disable this effect, larger values incease sharpness.	RW	0

Table 191: PISP_BE_DEMOSAIC_CONFIG register definition.

Tile Parameters

The Demosaic block has no tile parameters.

7.5.27 RGB Input Configuration Registers

The RGB Input block is shared with the (Bayer) Input block, and its use in this mode is signalled by setting the INPUT bit in the PISP_BE_GLOBAL_RGB_ENABLE register. It has no separate registers in its own right.

7.5.28 CCM (Colour Correction Matrix) Configuration Registers

For an RGB triple x, this block calculates the output triple y as

$$y = Mx + C$$

for a 3x3 matrix M and a triple of offsets C.

Note that when MINOR_VERSION = 0, the maximum offset range is $[-FSD, +FSD)$. For later versions, it is extended to $[-4FSD, +4FSD)$. In all versions, the coefficient range is just below ± 8.0.

PISP_BE_CCM_MATRIX[n]				**0x14C8$+4n$**
Matrix coefficients $2n$ and $2n+1$ for $n = 0, 1, 2, 3, 4$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0
29:16	COEFF1	Coefficient $2n+1$ of the matrix in S4.10 format. For $n = 4$ this field is *ignored*.	RW	0
15:14	-	Reserved.	-	0
13:0	COEFF0	Coefficient $2n$ of the matrix in S4.10 format.	RW	0

Table 192: PISP_BE_CCM_MATRIX[n] register definition.

PISP_BE_CCM_OFFSETS[n]				**0x14DC$+4n$**
Offsets n for $n = 0, 1, 2$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:0	OFFSET	Offset in S19.10 format. When MINOR_VERSION = 0, bits 28:27 are ignored and should match the sign bit 26.	RW	0

Table 193: PISP_BE_CCM_OFFSETS[n] register definition.

Tile Parameters

The CCM block has no tile parameters.

7.5.29 Saturation Control Configuration Registers

The colour gains in an imaging pipeline can cause the different colour channels to saturate at different places in the image. When these clip it causes the *hue* of the colour to change. If this is undesirable the dynamic range of the pixel values in the pipeline can be reduced by 1 or 2 bits, and then scaled up again in the Saturation Control block. This block performs this scaling in such a way that *all* the colours are scaled down proportionately when they may clip. In this way, the *hue* of the original colour is preserved, though at a lesser level of colour *saturation*.

PISP_BE_RGB_SAT_CONTROL_SHIFT				**0x14E8**
Left-shift to apply to colour channels.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:18	-	Reserved.	-	-
17:16	B_SHIFT	Left-shift to apply to blue pixels, taking the value 0, 1 or 2.	RW	0
15:10	-	Reserved.	-	-

Continued on next page...

PISP_BE_RGB_SAT_CONTROL_SHIFT (continued)				**0x14E8**
Bits	***Name***	***Description***	***R/W***	***Reset***
9:8	G_SHIFT	Left-shift to apply to green pixels, taking the value 0, 1 or 2.	RW	0
7:2	-	Reserved.	-	-
1:0	R_SHIFT	Left-shift to apply to red pixels, taking the value 0, 1 or 2.	RW	0

Table 194: PISP_BE_RGB_SAT_CONTROL_SHIFT register definition.

Tile Parameters

The Saturation Control block has no tile parameters.

7.5.30 YCbCr Conversion Configuration Registers

This block converts the RGB pixels into a YCbCr colour space, using a standard 3x3 matrix. Functionally it is identical to the CCM block. Further down the pipeline, the YCbCr Inverse block should be programmed to the inverse transform.

PISP_BE_YCBCR_MATRIX[n]				**0x14EC$+4n$**
Matrix coefficients $2n$ and $2n+1$ for $n = 0, 1, 2, 3, 4$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0
29:16	COEFF1	Coefficient $2n+1$ of the matrix in S4.10 format. For $n = 4$ this field is *ignored*.	RW	0
15:14	-	Reserved.	-	0
13:0	COEFF0	Coefficient $2n$ of the matrix in S4.10 format.	RW	0

Table 195: PISP_BE_YCBCR_MATRIX[n] register definition.

PISP_BE_YCBCR_OFFSETS[n]				**0x1500$+4n$**
Offsets n for $n = 0, 1, 2$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:0	OFFSET	Offset in S19.10 format. When MINOR_VERSION = 0, bits 28:27 are ignored and should match the sign bit 26.	RW	0

Table 196: PISP_BE_YCBCR_OFFSETS[n] register definition.

Tile Parameters

The YCbCr Conversion block has no tile parameters.

7.5.31 Sharpening Configuration Registers

The Sharpening block applies high pass 5x5 FIR filters to the image in order to calculate a value that can be added back to the initial pixel values to sharpen them. It operates only on the first image channel (which represents luminance, being after the YCbCr Conversion block). Five distinct filters are applied. The first four are usually programmed to detect horizontal, vertical and both orientations of diagonal edges, and final filter is programmed to be isotropic, responding to "speckle" rather than edge detail.

The filter coefficients are numbers in S6 format, therefore lying from -32 to 31 (inclusive). The coefficients must obey the rule that the sum of the positive filter coefficients must be no greater than 31, and that the sum of the negative filter coefficients must be no less than -32.

The filter results are thresholded (to avoid picking up image noise), scaled and combined before being added back to the image.

The Sharpening block does not actually apply the calculated delta value directly to the image. Instead it passes the value, along with the original pixel, to the Sharpen and False Colour Combining block to do this after some further processing.

The sharpening filters are relatively difficult to tune so the block contains masks that can be set to force the output pixels to black or white depending on which of the directional filters responded (other pixels are output as grey).

PISP_BE_SHARPEN_FILTER0[n] **0x150C**$+4n$

Coefficients for filter 0, for $n = 0, 1, ..., 6$

Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	-
29:24	COEFF3	Coefficient $4n + 3$ in S6 format. This field is *unused* when $n = 6$.	RW	0
23:22	-	Reserved.	-	-
21:16	COEFF2	Coefficient $4n + 2$ in S6 format. This field is *unused* when $n = 6$.	RW	0
15:14	-	Reserved.	-	-
13:8	COEFF1	Coefficient $4n + 1$ in S6 format. This field is *unused* when $n = 6$.	RW	0
7:6	-	Reserved.	-	-
5:0	COEFF0	Coefficient $4n$ in S6 format.	RW	0

Table 197: PISP_BE_SHARPEN_FILTER0[n] register definition.

PISP_BE_SHARPEN_FILTER1[n]				**0x1528**$+4n$
Coefficients for filter 1, for $n = 0, 1, ..., 6$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	-
29:24	COEFF3	Coefficient $4n + 3$ in S6 format. This field is *unused* when $n = 6$.	RW	0
23:22	-	Reserved.	-	-
21:16	COEFF2	Coefficient $4n + 2$ in S6 format. This field is *unused* when $n = 6$.	RW	0
15:14	-	Reserved.	-	-
13:8	COEFF1	Coefficient $4n + 1$ in S6 format. This field is *unused* when $n = 6$.	RW	0
7:6	-	Reserved.	-	-
5:0	COEFF0	Coefficient $4n$ in S6 format.	RW	0

Table 198: PISP_BE_SHARPEN_FILTER1[n] register definition.

PISP_BE_SHARPEN_FILTER2[n]				**0x1544**$+4n$
Coefficients for filter 2, for $n = 0, 1, ..., 6$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	-
29:24	COEFF3	Coefficient $4n + 3$ in S6 format. This field is *unused* when $n = 6$.	RW	0
23:22	-	Reserved.	-	-
21:16	COEFF2	Coefficient $4n + 2$ in S6 format. This field is *unused* when $n = 6$.	RW	0
15:14	-	Reserved.	-	-
13:8	COEFF1	Coefficient $4n + 1$ in S6 format. This field is *unused* when $n = 6$.	RW	0
7:6	-	Reserved.	-	-
5:0	COEFF0	Coefficient $4n$ in S6 format.	RW	0

Table 199: PISP_BE_SHARPEN_FILTER2[n] register definition.

PISP_BE_SHARPEN_FILTER3[n]				**0x1560**$+4n$
Coefficients for filter 3, for $n = 0, 1, ..., 6$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	-
29:24	COEFF3	Coefficient $4n+3$ in S6 format. This field is *unused* when $n = 6$.	RW	0
23:22	-	Reserved.	-	-
21:16	COEFF2	Coefficient $4n+2$ in S6 format. This field is *unused* when $n = 6$.	RW	0
15:14	-	Reserved.	-	-
13:8	COEFF1	Coefficient $4n+1$ in S6 format. This field is *unused* when $n = 6$.	RW	0
7:6	-	Reserved.	-	-
5:0	COEFF0	Coefficient $4n$ in S6 format.	RW	0

Table 200: PISP_BE_SHARPEN_FILTER3[n] register definition.

PISP_BE_SHARPEN_FILTER4[n]				**0x157C**$+4n$
Coefficients for filter 4, for $n = 0, 1, ..., 6$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	-
29:24	COEFF3	Coefficient $4n+3$ in S6 format. This field is *unused* when $n = 6$.	RW	0
23:22	-	Reserved.	-	-
21:16	COEFF2	Coefficient $4n+2$ in S6 format. This field is *unused* when $n = 6$.	RW	0
15:14	-	Reserved.	-	-
13:8	COEFF1	Coefficient $4n+1$ in S6 format. This field is *unused* when $n = 6$.	RW	0
7:6	-	Reserved.	-	-
5:0	COEFF0	Coefficient $4n$ in S6 format.	RW	0

Table 201: PISP_BE_SHARPEN_FILTER4[n] register definition.

PISP_BE_SHARPEN_THRESHOLD0				**0x1598**
Thresholds for filter 0.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	SLOPE	Slope in U1.11 format for multiplying the current pixel to calculate filter response threshold.	-	-
15:0	OFFSET	Offset used for calculating filter response threshold.	RW	0

Table 202: PISP_BE_SHARPEN_THRESHOLD0 register definition.

PISP_BE_SHARPEN_SCALE0				**0x159C**
Scale for filter 0 response.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:0	SCALE	Scale factor in U3.9 format for multiplying the thresholded filter response.	RW	0

Table 203: PISP_BE_SHARPEN_SCALE0 register definition.

PISP_BE_SHARPEN_THRESHOLD1				**0x15A0**
Thresholds for filter 1.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	SLOPE	Slope in U1.11 format for multiplying the current pixel to calculate filter response threshold.	-	-
15:0	OFFSET	Offset used for calculating filter response threshold.	RW	0

Table 204: PISP_BE_SHARPEN_THRESHOLD1 register definition.

PISP_BE_SHARPEN_SCALE1				**0x15A4**
Scale for filter 1 response.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:0	SCALE	Scale factor in U3.9 format for multiplying the thresholded filter response.	RW	0

Table 205: PISP_BE_SHARPEN_SCALE1 register definition.

PISP_BE_SHARPEN_THRESHOLD2				**0x15A8**
Thresholds for filter 2.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	SLOPE	Slope in U1.11 format for multiplying the current pixel to calculate filter response threshold.	-	-
15:0	OFFSET	Offset used for calculating filter response threshold.	RW	0

Table 206: PISP_BE_SHARPEN_THRESHOLD2 register definition.

PISP_BE_SHARPEN_SCALE2				**0x15AC**
Scale for filter 2 response.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:0	SCALE	Scale factor in U3.9 format for multiplying the thresholded filter response.	RW	0

Table 207: PISP_BE_SHARPEN_SCALE2 register definition.

PISP_BE_SHARPEN_THRESHOLD3				**0x15B0**
Thresholds for filter 3.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	SLOPE	Slope in U1.11 format for multiplying the current pixel to calculate filter response threshold.	-	-
15:0	OFFSET	Offset used for calculating filter response threshold.	RW	0

Table 208: PISP_BE_SHARPEN_THRESHOLD3 register definition.

PISP_BE_SHARPEN_SCALE3				**0x15B4**
Scale for filter 3 response.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:0	SCALE	Scale factor in U3.9 format for multiplying the thresholded filter response.	RW	0

Table 209: PISP_BE_SHARPEN_SCALE3 register definition.

PISP_BE_SHARPEN_THRESHOLD4				**0x15B8**
Thresholds for filter 4.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	-	-
27:16	SLOPE	Slope in U1.11 format for multiplying the current pixel to calculate filter response threshold.	-	-
15:0	OFFSET	Offset used for calculating filter response threshold.	RW	0

Table 210: PISP_BE_SHARPEN_THRESHOLD4 register definition.

PISP_BE_SHARPEN_SCALE4				**0x15BC**
Scale for filter 4 response.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:12	-	Reserved.	-	-
11:0	SCALE	Scale factor in U3.9 format for multiplying the thresholded filter response.	RW	0

Table 211: PISP_BE_SHARPEN_SCALE4 register definition.

PISP_BE_SHARPEN_POSITIVE_CONFIG				**0x15C0**
Strength and limit value for positive sharpening.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LIMIT	Maximum allowed possible sharpening value (prior to applying positive transfer function).	RW	0
15:12	-	Reserved.	-	-
11:0	STRENGTH	Factor in U3.9 format to scale sharpening strength.	RW	0

Table 212: PISP_BE_SHARPEN_POSITIVE_CONFIG register definition.

PISP_BE_SHARPEN_POSITIVE_FUNC[n]				**0x15C4**$+4n$
Gain factor applied to positive sharpening response derived from pixel value, for $n = 0, 1, 2, 3$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	GAIN1	Gain $2n + 1$ in U6.10 format.	RW	0
15:0	GAIN0	Gain $2n$ in U6.10 format.	RW	0

Table 213: PISP_BE_SHARPEN_POSITIVE_FUNC[n] register definition.

PISP_BE_SHARPEN_POSITIVE_FUNC[4]				**0x15D4**
Final gain factor applied to positive sharpening response derived from pixel value and final limit.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LIMIT	Limit for sharpening response after application of gain factor.	RW	0
15:0	GAIN0	Gain 8 in U6.10 format.	RW	0

Table 214: PISP_BE_SHARPEN_POSITIVE_FUNC[4] register definition.

PISP_BE_SHARPEN_NEGATIVE_CONFIG				**0x15D8**
Strength and limit value for negative sharpening.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LIMIT	Maximum allowed possible sharpening value (prior to applying negative transfer function).	RW	0
15:12	-	Reserved.	-	-
11:0	STRENGTH	Factor in U3.9 format to scale sharpening strength.	RW	0

Table 215: PISP_BE_SHARPEN_NEGATIVE_CONFIG register definition.

PISP_BE_SHARPEN_NEGATIVE_FUNC[n]				**0x15DC$+4n$**
Gain factor applied to negative sharpening response derived from pixel value, for $n = 0, 1, 2, 3$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	GAIN1	Gain $2n + 1$ in U6.10 format.	RW	0
15:0	GAIN0	Gain $2n$ in U6.10 format.	RW	0

Table 216: PISP_BE_SHARPEN_NEGATIVE_FUNC[n] register definition.

PISP_BE_SHARPEN_NEGATIVE_FUNC[4]				**0x15EC**
Final gain factor applied to negative sharpening response derived from pixel value and final limit.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	LIMIT	Limit for sharpening response after application of gain factor.	RW	0
15:0	GAIN0	Gain 8 in U6.10 format.	RW	0

Table 217: PISP_BE_SHARPEN_NEGATIVE_FUNC[4] register definition.

PISP_BE_SHARPEN_MASKS				**0x15F0**
Masks to allow tuning of sharpen filters.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	GREY	When any bits of BLACK or WHITE are set, and no filter exceeds its threshold value, then pixels are output with this grey level (out of 255).	RW	0
23:21	-	Reserved.	-	-
20:16	BLACK	Output pixel is set to black when the strongest filter response i has bit i set to 1 in this mask.	RW	0
15:13	-	Reserved.	-	-
12:8	WHITE	Output pixel is set to white when the strongest filter response i has bit i set to 1 in this mask.	RW	0
7:5	-	Reserved.	-	-
4:0	ENABLE	Set bit i to 1 to enable filter i, and zero to disable it.	RW	0

Table 218: PISP_BE_SHARPEN_MASKS register definition.

Tile Parameters

The Sharpening block has no tile parameters.

7.5.32 False Colour Configuration Registers

The False Colour block provides additional false colour filtering, over and above what is already available in the Demosaic block. It performs a type of 3x3 median filtering operation, where the neighbouring pixels are actually 1, 2 or 3 pixels distant from the central pixel.

This block works only on the Cb and Cr channels and, rather like the Sharpening block, doesn't *make* any changes to the values but passes them on to the Sharpening and False Colour Combining block.

PISP_BE_FALSE_COLOUR_CONFIG				**0x15F4**
False colour configuration				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reserved.	-	-
1:0	DISTANCE	Distance of neighbouring pixels, either 1 or 2.	RW	0

Table 219: PISP_BE_FALSE_COLOUR_CONFIG register definition.

Tile Parameters

The False Colour block has no tile parameters.

7.5.33 Sharpening and False Colour Combining Configuration Registers

This block receives “recommendations” as to the changes that the Sharpening and False Colour block wish to make. It performs a small amount of extra processing and then applies the changes:

- When False Colour is desaturating a pixel, it can increase Y a little so that the pixel does not get quite so dark.
- When a pixel is being darkened (being on the dark side of an edge being sharpened), the Cb and Cr signals can be reduced to prevent the pixels from becoming more “colourful”.

PISP_BE_SH_FC_COMBINE_CONFIG				**0x15F8**
Sharpening and False Colour combine configuration				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:24	-	Reserved.	-	-
23:16	C2_FACTOR	Controls amount of brightening of a pixel when the Cr channel is being desaturated. 0 = no effect, 255 = maximum effect.	RW	0
15:8	C1_FACTOR	Controls amount of brightening of a pixel when the Cb channel is being desaturated. 0 = no effect, 255 = maximum effect.	RW	0
7:0	Y_FACTOR	Controls amount of desaturation of pixels being darkened. 0 = no effect, 255 = maximum effect.	RW	0

Table 220: PISP_BE_SH_FC_COMBINE_CONFIG register definition.

7.5.34 YCbCr Inverse Conversion Configuration Registers

This block converts the pixel data back into the RGB colour space. It should be programmed with the inverse transform to the one used in the YCbCr Conversion block.

Note that when MINOR_VERSION = 0, the offset range is sufficient to invert ‘full-range’ YCbCr, but may be insufficient for some ‘reduced-range’ representations.

PISP_BE_YCBCR_INV_MATRIX[n]				**0x15FC**$+4n$
Matrix coefficients $2n$ and $2n+1$ for $n = 0, 1, 2, 3, 4$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0
29:16	COEFF1	Coefficient $2n+1$ of the matrix in S4.10 format. For $n = 4$ this field is *ignored*.	RW	0
15:14	-	Reserved.	-	0
13:0	COEFF0	Coefficient $2n$ of the matrix in S4.10 format.	RW	0

Table 221: PISP_BE_YCBCR_INV_MATRIX[n] register definition.

PISP_BE_YCBCR_INV_OFFSETS[n]				**0x1610**$+4n$
Offsets n for $n = 0, 1, 2$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:0	OFFSET	Offset in S19.10 format. When MINOR_VERSION = 0, bits 28:27 are ignored and should match the sign bit 26.	RW	0

Table 222: PISP_BE_YCBCR_INV_OFFSETS[n] register definition.

Tile Parameters

The YCbCr Inverse Conversion block has no tile parameters.

7.5.35 Gamma Configuration Registers

The Gamma block applies the same piecewise linear transfer function to all the pixel values. The look-up table that defines this function is denser towards the bottom of the range where more accuracy is required.

PISP_BE_GAMMA_LUT[n]				**0x161C**$+4n$
Look-up table for gamma curve, for $n = 0, ..., 63$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0
29:16	SLOPE	Slope of this segment of the gamma curve.	RW	0
15:0	POSITION	y (output) position of gamma curve for this x (input) value.	RW	0

Table 223: PISP_BE_GAMMA_LUT[n] register definition.

Note: the x value of point n in the tonemap curve is given by:

- if $n < 32$, $n * 512$
- otherwise if $n < 48$, $(n - 32) * 1024 + 16384$
- otherwise $(n - 48) * 2048 + 32768)$

7.5.36 Crop 0 Configuration Registers

The Crop block on output branch 0 has no global configuration registers.

Tile Parameters

The Crop block on output branch 0 uses the following tile parameters.

Field	Offset	Size	Description
crop_x_start[0]	52	2	The number of pixels cropped from the left of the tile when passing through the Crop block at the head of output branch 0.
crop_x_end[0]	56	2	The number of pixels cropped from the right of the tile when passing through the Crop block at the head of output branch 0.
crop_y_start[0]	60	2	The number of pixels cropped from the top of the tile when passing through the Crop block at the head of output branch 0.
crop_y_end[0]	64	2	The number of pixels cropped from the bottom of the tile when passing through the Crop block at the head of output branch 0.

Table 224: Tile parameters for the Crop block on output branch 0.

The indicated numbers of pixels are cropped, respectively, from the left, right, top and bottom of the tile. The width of the tile leaving the block is reduced by a total of $crop\_x\_start[0] + crop\_x\_end[0]$ pixels, and its height is reduced by $crop\_y\_start[0] + crop\_y\_end[0]$ pixels.

7.5.37 Crop 1 Configuration Registers

The Crop block on output branch 1 has no global configuration registers.

Tile Parameters

The Crop block on output branch 1 uses the following tile parameters.

Field	Offset	Size	Description
crop_x_start[1]	54	2	The number of pixels cropped from the left of the tile when passing through the Crop block at the head of output branch 1.
crop_x_end[1]	58	2	The number of pixels cropped from the right of the tile when passing through the Crop block at the head of output branch 1.
crop_y_start[1]	62	2	The number of pixels cropped from the top of the tile when passing through the Crop block at the head of output branch 1.
crop_y_end[1]	66	2	The number of pixels cropped from the bottom of the tile when passing through the Crop block at the head of output branch 1.

Table 225: Tile parameters for the Crop block on output branch 1.

The indicated numbers of pixels are cropped, respectively, from the left, right, top and bottom of the tile. The width of the tile leaving the block is reduced by a total of $crop\_x\_start[1] + crop\_x\_end[1]$ pixels, and its height is reduced by $crop\_y\_start[1] + crop\_y\_end[1]$ pixels.

7.5.38 CSC (Colour Space Conversion) 0 Configuration Registers

The CSC block on output branch 0 is functionally identical to the CCM block, implementing a colour space conversion with a 3x3 matrix.

PISP_BE_CSC0_MATRIX[n]				**0x171C$+4n$**
		Matrix coefficients $2n$ and $2n+1$ for $n = 0, 1, 2, 3, 4$		
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0
29:16	COEFF1	Coefficient $2n+1$ of the matrix in S4.10 format. For $n = 4$ this field is *ignored*.	RW	0
15:14	-	Reserved.	-	0
13:0	COEFF0	Coefficient $2n$ of the matrix in S4.10 format.	RW	0

Table 226: PISP_BE_CSC0_MATRIX[n] register definition.

PISP_BE_CSC0_OFFSETS[n]				**0x1730$+4n$**
		Offsets n for $n = 0, 1, 2$		
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:0	OFFSET	Offset in S19.10 format. When MINOR_VERSION = 0, bits 28:27 are ignored and should match the sign bit 26.	RW	0

Table 227: PISP_BE_CSC0_OFFSETS[n] register definition.

Tile Parameters

The CSC blocks have no tile parameters.

7.5.39 CSC (Colour Space Conversion) 1 Configuration Registers

The CSC block on output branch 1 is functionally identical to the CCM block, implementing a colour space conversion with a 3x3 matrix.

PISP_BE_CSC1_MATRIX[n]				**0x173C$+4n$**
		Matrix coefficients $2n$ and $2n+1$ for $n = 0, 1, 2, 3, 4$		
Bits	***Name***	***Description***	***R/W***	***Reset***
31:30	-	Reserved.	-	0

Continued on next page...

PISP_BE_CSC1_MATRIX[n] (continued)				**0x173C**$+4n$
Bits	***Name***	***Description***	***R/W***	***Reset***
29:16	COEFF1	Coefficient $2n+1$ of the matrix in S4.10 format. For $n = 4$ this field is *ignored*.	RW	0
15:14	-	Reserved.	-	0
13:0	COEFF0	Coefficient $2n$ of the matrix in S4.10 format.	RW	0

Table 228: PISP_BE_CSC1_MATRIX[n] register definition.

PISP_BE_CSC1_OFFSETS[n]				**0x1750**$+4n$
Offsets n for $n = 0, 1, 2$				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:0	OFFSET	Offset in S19.10 format. When MINOR_VERSION = 0, bits 28:27 are ignored and should match the sign bit 26.	RW	0

Table 229: PISP_BE_CSC1_OFFSETS[n] register definition.

Tile Parameters

The CSC blocks have no tile parameters.

7.5.40 Downscale 0 Configuration Registers

The Downscale block on output branch 0 performs downscaling using a “trapezoidal” filter, with independent downscaling factors in the x and y directions. It can downscale by a factor f satisfying $f = 1$ or $2 \leq f \leq 8$ (downscaling by a factor of 1 is useful when downscaling in only one direction).

In the description below, $input\_width$ and $input\_height$ refer to the dimensions of the whole image (not just a tile) prior to downscaling, and $output\_width$ and $output\_height$ again refer to the dimensions of the whole image after downscaling.

Note that the downscale block on output branch 0 is not available in all hardware variants.

PISP_BE_DOWNSCALE0_SCALE				**0x175C**
Downscale factors.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SCALE_FACTOR_V	Vertical scale factor. Should be set to $(4096 * input\_height)/output\_height$.	RW	0
15:0	SCALE_FACTOR_H	Horizontal scale factor. Should be set to $(4096 * input\_width)/output\_width$.	RW	0

Table 230: PISP_BE_DOWNSCALE0_SCALE register definition.

PISP_BE_DOWNSCALE0_RECIP				**0x1760**
Downscale reciprocals.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:16	SCALE_RECIP_V	Vertical reciprocal factor. Should be set to $(4096 * output\_height)/input\_height$.	RW	0
15:13	-	Reserved.	-	-
12:0	SCALE_RECIP_H	Horizontal reciprocal factor. Should be set to $(4096 * output\_width)/input\_width$.	RW	0

Table 231: PISP_BE_DOWNSCALE0_RECIP register definition.

Tile Parameters

The Downscale block on output branch 0 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
downscale_phase_x[0][m]	68	2	Initial horizontal phase in pixels of colour channel m $(m = 0, 1, 2)$ for the Downscaler on output branch 0. This is a U0.12 format value.
downscale_phase_y[0][m]	80	2	Initial vertical phase in pixels of colour channel m $(m = 0, 1, 2)$ for the Downscaler on output branch 0. This is a U0.12 format value.

Table 232: Tile parameters for the Downscale block on output branch 0.

The initial phases need to be programmed correctly for each tile to avoid slight discontinuities or "creases" in the output image where tiles fail to join accurately. When the downscaling factor is exactly 1, then the phase corresponding to that direction must be set to zero.

7.5.41 Downscale 1 Configuration Registers

The Downscale block on output branch 1 is functionally identical to the one on output branch 0.

PISP_BE_DOWNSCALE1_SCALE				**0x1764**
Downscale factors.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SCALE_FACTOR_V	Vertical scale factor. Should be set to $(4096 * input\_height)/output\_height$.	RW	0
15:0	SCALE_FACTOR_H	Horizontal scale factor. Should be set to $(4096 * input\_width)/output\_width$.	RW	0

Table 233: PISP_BE_DOWNSCALE1_SCALE register definition.

PISP_BE_DOWNSCALE1_RECIP				**0x1768**
Downscale reciprocals.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:29	-	Reserved.	-	-
28:16	SCALE_RECIP_V	Vertical reciprocal factor. Should be set to $(4096 * output\_height)/input\_height$.	RW	0
15:13	-	Reserved.	-	-
12:0	SCALE_RECIP_H	Horizontal reciprocal factor. Should be set to $(4096 * output\_width)/input\_width$.	RW	0

Table 234: PISP_BE_DOWNSCALE1_RECIP register definition.

Tile Parameters

The Downscale block on output branch 1 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
downscale_phase_x[1][m]	74	2	Initial horizontal phase in pixels of colour channel m ($m = 0, 1, 2$) for the Downscaler on output branch n ($n = 0, 1$). This is a U0.12 format value.
downscale_phase_y[1][m]	86	2	Initial vertical phase in pixels of colour channel m ($m = 0, 1, 2$) for the Downscaler on output branch n ($n = 0, 1$). This is a U0.12 format value.

Table 235: Tile parameters for the Downscale block on output branch 1.

These parameters are functionally identical to the ones for the Downscale block on output branch 0.

7.5.42 Resample 0 Configuration Registers

The Resample block on output branch 0 resamples the image to a different output size with a 6-tap FIR polyphase filter using 16 phases. The filter can resample an image to a smaller size (useful when the scale factor is between 1 and 2 which disallows the Downscale block), or to a larger size. Downscaling is recommended not to exceed a factor of 4x, and upscaling is recommended not to exceed a factor of 16x.

In the description below, $input\_width$ and $input\_height$ refer to the dimensions of the whole image (not just a tile) prior to resampling, and $output\_width$ and $output\_height$ again refer to the dimensions of the whole image after resampling.

PISP_BE_RESAMPLE0_SCALE				**0x176C**
Resample scale factors.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	SCALE_FACTOR_V	Vertical scale factor. Should be set to $(input\_height - 1) * 4096/(output\_height - 1)$.	RW	0
15:0	SCALE_FACTOR_H	Horizontal scale factor. Should be set to $(input\_width - 1) * 4096/(output\_width - 1)$.	RW	0

Table 236: PISP_BE_RESAMPLE0_SCALE register definition.

PISP_BE_RESAMPLE0_COEFFS[n][m]				**0x1770**$+12n + 4m$
Resample filter coefficients, for $n = 0, 1, ..., 15$ and $m = 0, 1, 2$.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:28	-	Reserved.	RW	0
27:16	COEFF1	Coefficient $2m + 1$ for the filter with phase n. Format is S2.10.	RW	0
15:12	-	Reserved.	RW	0
11:0	COEFF0	Coefficient $2m$ for the filter with phase n. Format is S2.10.	RW	0

Table 237: PISP_BE_RESAMPLE0_COEFFS[n][m] register definition.

Tile Parameters

The Resample block on output branch 0 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
resample_in_width[0]	92	2	Width in pixels of the tile entering the Resample block on output branch 0.
resample_in_height[0]	96	2	Height in pixels of the tile entering the Resample block on output branch 0.
resample_phase_x[0][m]	100	2	Initial horizontal phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch 0. This parameter is in U1.12 format, thereby permitting negative values.
resample_phase_y[0][m]	112	2	Initial vertical phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch 0. This parameter is in U1.12 format, thereby permitting negative values.

Field	Offset	Size	Description
output_width[0]	132	2	Width in pixels of this tile in the output image, for output branch 0.
output_height[0]	136	2	Height in pixels of this tile in the output image, for output branch 0.

Table 238: Tile parameters for the Resample block on output branch 0.

Like the Downscaler blocks, the Resampler needs to know the initial phases of the first output sample in each tile to prevent slight discontinuities or "creases". Additionally, an initial phase value greater than 4095 will instruct the Resampler to start one sample to the right and use an initial phase of (resample_phase_[x|y] - 4096).

The Resample block also needs to know the exact size of the input tile it is expecting, and of the output tile it is to produce. Note that the dimensions of the tile that is output by the Resampler is the same as the size that is output by the downstream Output block.

7.5.43 Resample 1 Configuration Registers

The Resample block on output branch 1 is functionally identical to the Resample block on output branch 1.

PISP_BE_RESAMPLE1_SCALE **0x1830**

Resample scale factors.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:16	SCALE_FACTOR_V	Vertical scale factor. Should be set to $(input\_height - 1) * 4096/(output\_height - 1)$.	RW	0
15:0	SCALE_FACTOR_H	Horizontal scale factor. Should be set to $(input\_width - 1) * 4096/(output\_width - 1)$.	RW	0

Table 239: PISP_BE_RESAMPLE1_SCALE register definition.

PISP_BE_RESAMPLE1_COEFFS[n][m] **0x1834**$+12n + 4m$

Resample filter coefficients, for $n = 0, 1, ..., 15$ and $m = 0, 1, 2$.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:28	-	Reserved.	RW	0
27:16	COEFF1	Coefficient $2m + 1$ for the filter with phase n. Format is S2.10.	RW	0
15:12	-	Reserved.	RW	0

Continued on next page...

PISP_BE_RESAMPLE1_COEFFS[n][m] (continued) **0x1834**$+12n + 4m$

Bits	***Name***	***Description***	***R/W***	***Reset***
11:0	COEFF0	Coefficient $2m$ for the filter with phase n. Format is S2.10.	RW	0

Table 240: PISP_BE_RESAMPLE1_COEFFS[n][m] register definition.

Tile Parameters

The Resample block on output branch 1 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
resample_in_width[1]	94	2	Width in pixels of the tile entering the Resample block on output branch 1.
resample_in_height[1]	98	2	Height in pixels of the tile entering the Resample block on output branch 1.
resample_phase_x[1][m]	106	2	Initial horizontal phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch 1. This parameter is in U1.12 format, thereby permitting negative values.
resample_phase_y[1][m]	118	2	Initial vertical phase in pixels of colour channel m ($m = 0, 1, 2$) for the Resampler on output branch 1. This parameter is in U1.12 format, thereby permitting negative values.
output_width[1]	134	2	Width in pixels of this tile in the output image, for output branch 1.
output_height[1]	138	2	Height in pixels of this tile in the output image, for output branch 1.

Table 241: Tile parameters for the Resample block on output branch 1.

Like the Downscaler blocks, the Resampler needs to know the initial phases of the first output sample in each tile to prevent slight discontinuities or "creases". Additionally, an initial phase value greater than 4095 will instruct the Resampler to start one sample to the right and use an initial phase of (resample_phase_[x|y] - 4096).

The Resample block also needs to know the exact size of the input tile it is expecting, and of the output tile it is to produce. Note that the dimensions of the tile that is output by the Resampler is the same as the size that is output by the downstream Output block.

7.5.44 Output 0 Configuration Registers

The Output blocks write the results of PiSP processing to the memory address given by the PISP_BE_IO_OUTPUT0_ADDR_LO/HI registers. The block is able to flip the output image horizontally or vertically, and the output format is controlled by the registers below.

PISP_BE_OUTPUT0_SIZE				**0x18F4**
Dimensions of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image.	RW	0
15:0	WIDTH	Width of the image.	RW	0

Table 242: PISP_BE_OUTPUT0_SIZE register definition.

PISP_BE_OUTPUT0_FORMAT				**0x18F8**
Format definition of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31	-	Reserved.	-	-
30	THREE_CHANNEL	Write a 1 to output a three channel image, otherwise only the first channel is output.	RW	0
29	WALLPAPER	Set to 1 to write the image in “wallpaper roll” foramt.	RW	0
28	INTEGRAL	Placeholder for *integral image*. As no existing PiSP version supports integral images, this bit should be zero.	RW	0
27:22	-	Reserved.	-	-
21	X_VALUE	When BPP_32 is set, set this bit to 1 to fill padding bytes with the value 0xFF.	RW	0
20	BPP_32	Set to 1 to pad RGB 444 pixels to 32 bpp (only when MINOR_VERSION $\geq$ 1).	RW	0
19:16	SHIFT	Pixel values are right-shifted by this amount, for $0 \leq SHIFT \leq 8$ (only if BITS_PER_SAMPLE indiciates 16 bps).	RW	0
15:13	-	Reserved.	-	-

Continued on next page...

PISP_BE_OUTPUT0_FORMAT (continued) **0x18F8**

Bits	*Name*	*Description*	*R/W*	*Reset*
12	ORDER	Only when PLANARITY is fully interleaved: When SAMPLING indicates 422, this bit causes the image to be written out as UYVY (i.e. the first channel always written second) rather than as YUYV. When SAMPLING indicates 444 and BPP_32 is set, it inserts pading at the lowest byte address rather than the highest.	RW	0
11:10	-	Reserved.	-	-
9:8	SAMPLING	0 = 444 sampling (like YUV444), 1 = 422 sampling, 2 = 420 sampling.	RW	0
7:6	-	Reserved.	-	-
5:4	PLANARITY	0 = fully interleaved, 1 = semi-planar (only the first channel in a separate plane), 2 = fully planar.	RW	0
1:0	BITS_PER_SAMPLE	Number of bits in each sample. Valid settings are: 0 (8 bps), 1 (10 bps) and 3 (16 bps). The value 2 is unused.		

Table 243: PISP_BE_OUTPUT0_FORMAT register definition.

PISP_BE_OUTPUT0_STRIDE **0x18FC**

Line to line stride of the first plane.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 244: PISP_BE_OUTPUT0_STRIDE register definition.

PISP_BE_OUTPUT0_STRIDE2 **0x1900**

Line to line stride of planes other than the first (ignored when the format is fully interleaved).

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 245: PISP_BE_OUTPUT0_STRIDE2 register definition.

PISP_BE_OUTPUT0_TRANSFORM				**0x1904**
Output transform specification.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reseverd.	-	-
1	VFLIP	Vertically mirror the output image.	RW	0
0	HFLIP	Horizontally mirror the output image.	-	0

Table 246: PISP_BE_OUTPUT0_TRANSFORM register definition.

PISP_BE_OUTPUT0_CLIP				**0x1908**
Clip output values from the first channel to the given range.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Upper limit.	RW	0
15:0	LO	Lower limit.	-	0

Table 247: PISP_BE_OUTPUT0_CLIP register definition.

PISP_BE_OUTPUT0_CLIP2				**0x190C**
Clip output values from the second and third channels to the given range.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Upper limit.	RW	0
15:0	LO	Lower limit.	-	0

Table 248: PISP_BE_OUTPUT0_CLIP2 register definition.

Tile Parameters

The Output block on branch 0 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
output_offset_x[0]	124	2	Horizontal offset in pixels where this tile will be written into the output image, for output branch 0.
output_offset_y[0]	128	2	Vertical offset in pixels where this tile will be written into the output image, for output branch 0.
output_width[0]	132	2	Width in pixels of this tile in the output image, for output branch 0.

Field	Offset	Size	Description
output_height[0]	136	2	Height in pixels of this tile in the output image, for output branch 0.
output_addr_offset[0]	140	4	Offset in bytes into the output buffer where output branch 0 is to write this tile. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
output_addr_offset2[0]	148	4	Offset in bytes into the output buffer where planes 2 and 3 of this tile are to be written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended. Ignored if the output format is non-planar (interleaved).

Table 249: Tile parameters for the Output block on branch 0.

The output address offsets listed above are added to the memory address given by PISP_BE_IO_-OUTPUT0_ADDR[n][m], where n indicates the colour channel for planar formats, and m indicates the high ($m = 1$) or low ($m = 0$) 32 bits of the address. *output_addr_offset* is added to the address formed by PISP_BE_IO_OUTPUT0_ADDR[0], and *output_addr_offset2* is added to the address formed by PISP_BE_IO_OUTPUT0_ADDR[1] and PISP_BE_IO_OUTPUT0_ADDR[2].

The *output_width* and *output_height* inform the Input block how many pixels it is expected to read. The *output_offset_x/y* values are in fact not needed by the block, but are provided as a courtesy to help debug any problems.

7.5.45 Output 1 Configuration Registers

The Output block on branch 1 is functionally identical to the Output block on branch 0.

PISP_BE_OUTPUT1_SIZE				**0x1910**
Dimensions of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HEIGHT	Height of the image.	RW	0
15:0	WIDTH	Width of the image.	RW	0

Table 250: PISP_BE_OUTPUT1_SIZE register definition.

PISP_BE_OUTPUT1_FORMAT				**0x1914**
Format definition of the output image.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31	-	Reserved.	-	-
30	THREE_CHANNEL	Write a 1 to output a three channel image, otherwise only the first channel is output.	RW	0
29	WALLPAPER	Set to 1 to write the image in “wallpaper roll” foramt.	RW	0
28	INTEGRAL	Placeholder for *integral image*. As no existing PiSP version supports integral images, this bit should be zero.	RW	0
27:22	-	Reserved.	-	-
21	X_VALUE	When BPP_32 is set, set this bit to 1 to fill padding bytes with the value 0xFF.	RW	0
20	BPP_32	Set to 1 to pad RGB 444 pixels to 32 bpp (only when MINOR_VERSION $\geq$ 1).	RW	0
19:16	SHIFT	Pixel values are right-shifted by this amount, for $0 \leq SHIFT \leq 8$.	RW	0
15:13	-	Reserved.	-	-
12	ORDER	Only when PLANARITY is fully interleaved: When SAMPLING indicates 422, this bit causes the image to be written out as UYVY (i.e. the first channel always written second) rather than as YUYV. When SAMPLING indicates 444 and BPP_32 is set, this inserts padding at the lowest byte address rather than the highest.	RW	0
11:10	-	Reserved.	-	-
9:8	SAMPLING	0 = 444 sampling (like YUV444), 1 = 422 sampling, 2 = 420 sampling.	RW	0
7:6	-	Reserved.	-	-
5:4	PLANARITY	0 = fully interleaved, 1 = semi-planar (only the first channel in a separate plane), 2 = fully planar.	RW	0
1:0	BITS_PER_SAMPLE	Number of bits in each sample. Valid settings are: 0 (8 bps), 1 (10 bps) and 3 (16 bps). The value 2 is unused.		

Table 251: PISP_BE_OUTPUT1_FORMAT register definition.

PISP_BE_OUTPUT1_STRIDE				**0x1918**
Line to line stride of the first plane.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 252: PISP_BE_OUTPUT1_STRIDE register definition.

PISP_BE_OUTPUT1_STRIDE2				**0x191C**
Line to line stride of planes other than the first (ignored when the format is fully interleaved).				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 253: PISP_BE_OUTPUT1_STRIDE2 register definition.

PISP_BE_OUTPUT1_TRANSFORM				**0x1920**
Output transform specification.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:2	-	Reseverd.	-	-
1	VFLIP	Vertically mirror the output image.	RW	0
0	HFLIP	Horizontally mirror the output image.	-	0

Table 254: PISP_BE_OUTPUT1_TRANSFORM register definition.

PISP_BE_OUTPUT1_CLIP				**0x1924**
Clip output values from the first channel to the given range.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Upper limit.	RW	0
15:0	LO	Lower limit.	-	0

Table 255: PISP_BE_OUTPUT1_CLIP register definition.

PISP_BE_OUTPUT1_CLIP2				**0x1928**
Clip output values from the second and third channels to the given range.				
Bits	***Name***	***Description***	***R/W***	***Reset***
31:16	HI	Upper limit.	RW	0
15:0	LO	Lower limit.	-	0

Table 256: PISP_BE_OUTPUT1_CLIP2 register definition.

Tile Parameters

The Output block on branch 1 uses the following tile parameters.

Field	**Offset**	**Size**	**Description**
output_offset_x[1]	126	2	Horizontal offset in pixels where this tile will be written into the output image, for output branch 1.
output_offset_y[1]	130	2	Vertical offset in pixels where this tile will be written into the output image, for output branch 1.
output_width[1]	134	2	Width in pixels of this tile in the output image, for output branch 1.
output_height[1]	138	2	Height in pixels of this tile in the output image, for output branch 1.
output_addr_offset[1]	144	4	Offset in bytes into the output buffer where output branch 1 is to write this tile. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.
output_addr_offset2[1]	152	4	Offset in bytes into the output buffer where planes 2 and 3 of this tile are to be written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended. Ignored if the output format is non-planar (interleaved).

Table 257: Tile parameters for the Output block on branch 1.

The output address offsets listed above are added to the memory address given by PISP_BE_IO_-OUTPUT1_ADDR$[n][m]$, where n indicates the colour channel for planar formats, and m indicates the high $(m = 1)$ or low $(m = 0)$ 32 bits of the address. *output_addr_offset* is added to the address formed by PISP_BE_IO_OUTPUT1_ADDR$[0]$, and *output_addr_offset2* is added to the address formed by PISP_BE_IO_OUTPUT1_ADDR$[1]$ and PISP_BE_IO_OUTPUT1_ADDR$[2]$.

The *output_width* and *output_height* inform the Input block how many pixels it is expected to read. The *output_offset_x/y* values are in fact not needed by the block, but are provided as a courtesy to help debug any problems.

7.5.46 HOG (Histogram of Oriented Gradients) Output Configuration Registers

The HOG block converts the image into the “histogram of oriented gradients” form. The image is divided into 8x8 cells, and from each of these cells it creates a histogram of gradients of either 9 or 18 bins, depending on whether signed or unsigned gradients are required (as is the convention, every bin contains a spread of angles of 20 degrees).

The histograms are written to memory in raster scan order. The histogram normalisation step, conventionally performed by HOG algorithms upon the histograms, is *not* carried out by the block. It may be performed subsequently by software, if required.

The histogram is constructed only for a single channel. The three incoming channels can be mixed if necessary.

PISP_BE_HOG_CONFIG **0x192C**

HOG configuration. Note that the channel mixing is performed as follows:

$$\begin{aligned} mix0 &= mix0\ ?\ mix0+1 : 0 \\ mix1 &= mix1\ ?\ mix1+1 : 0 \\ mix2 &= mix2\ ?\ mix2+1 : 0 \\ out &= (mix0 \cdot in_0 + mix1 \cdot in_1 + mix2 \cdot in_2) \gg 8 \end{aligned}$$

and $mix0 + mix1 + mix2 \leqslant 256$.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:24	CHANNEL_MIX2	U0.8 proportion of the third channel to use.	RW	0
23:16	CHANNEL_MIX1	U0.8 proportion of the second channel to use.	RW	0
15:8	CHANNEL_MIX0	U0.8 proportion of the first channel to use.	RW	0
7:1	-	Reserved.	-	-
0	SIGNED	Set to 0 for unsigned gradients, 1 for signed gradients.	-	0

Table 258: PISP_BE_HOG_CONFIG register definition.

PISP_BE_HOG_STRIDE **0x1930**

Stride in bytes between an 8x8 feature block and the 8x8 feature block directly below it in the image.

Bits	*Name*	*Description*	*R/W*	*Reset*
31:4	STRIDE_MID	Bits 31:4 of the stride in bytes.	RW	0
3:0	-	Reserved. Do not write any non-zero value.	-	0

Table 259: PISP_BE_HOG_STRIDE register definition.

Tile Parameters

The HOG Output block uses the following tile parameters.

Field	Offset	Size	Description
output_hog_addr_offset	156	4	Offset in bytes into the HOG output buffer where the results for this tile are to be written. This value must be at least 16-byte aligned, but for best performance a 64-byte aligned value is recommended.

Table 260: Tile parameters for the HOG Output block.

The output address offset listed above is added to the memory address given by PISP_BE_IO_HOG_-ADDR_LO/HI.