DxO ISP2013
Firmware Integration Guide

Table of contents

Table of figures

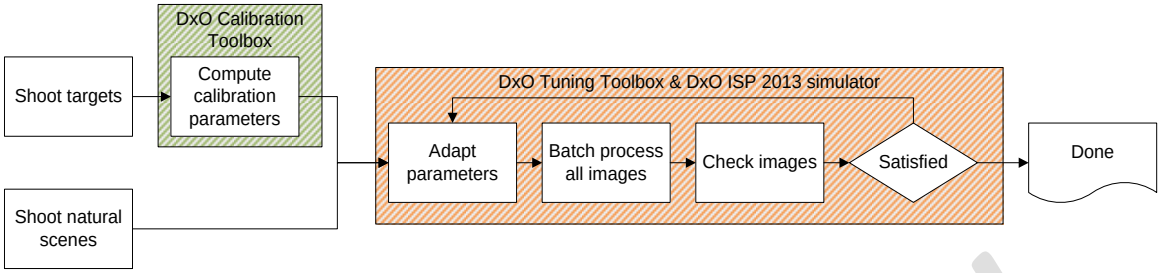


Table of tables

Revision History

Revision	Date	Comments
V1.0	Oct 13, 2013	First version for DxO ISP2013

Table 1: Document Revision History

1 INTRODUCTION

1.1 Purpose

The purpose of the Hardware and Firmware integration guides is to provide all needed views about the DxO ISP Hardware IP and LibDxOISP with detailed guidelines for their integrations in a complete system.

These documents share the first introduction chapters that present an overview of the DxO ISP solution, the image processing features, the system-level supported use cases, as well as the approach to answer the application needs.

The last chapters describe respectively:

- The hardware related specificities: chip level requirements and interfaces, electrical interfaces, deliverables, integration and verification process, and customer configuration dependant data,
- The software related specificities: chip level requirements and interfaces, registers description, deliverables, integration and verification process, and customer configuration dependant data.

1.2 Audience

The audience for these documents is the engineering team that needs to understand the characteristics of DxO ISP from a hardware or software perspective in order to integrate DxO Labs solution in a complete system.

1.3 Use of the present document

The present document is confidential and subject to the terms of the NDA signed between DxO Labs and the recipient of the present document.

The present document is the property of DxO Labs and must be considered as DxO Labs background intellectual property. This document discloses a number of technical characteristics protected by patents filed by DxO Labs, and therefore subject to royalties whatever the product they are used in. The content of this document cannot be used for any purpose without a contract from DxO Labs.

References to third party tools or hardware in the present document are not an authorization or recommendation by DxO Labs to use such third party tool or hardware. The use of any third party tool or hardware is the responsibility of the recipient of the present document.

The present document is preliminary and may contain errors in the description that do not reflect the actual behavior of the product. DxO Labs will update the present document if and when such errors are detected.

1.4 Reference Documents

Ref[1] AHB-Lite Overview, ARM DVI 0044A
Ref[2] AMBA™ Specification (Rev2.0), ARM IHI 0011A or more recent.

Ref[3] DxO_ISP2013-toolboxPresentation
Ref[4] DxO Analyzer Component Edition documentation
Ref[5] DxO_ISP2013-fvtsUserGuide
Ref[6] http://en.wikipedia.org/wiki/SRGB_color_space
Ref[7] http://en.wikipedia.org/wiki/YCbCr
Ref[8] AMBA™ AXI Protocol Specification (Rev1.0), ARM IHI 0022E or more recent.

1.5 Glossary

3A	The combination of Auto Exposure, Auto White Balance and Auto Focus
AE	Auto Exposure
AF	Auto Focus
AHBL	AHB-Lite that is a subset of the full Advanced High-performance Bus (AHB) protocol with respect of the Advanced Microcontroller Bus Architecture (AMBA) which is a registered trademark of ARM Ltd.
API	Application Programming interface (LibDxOISP API and other public APIs)
ASIC	Application-Specific Integrated Circuit
Asserted	Set lo logical “1”
AWB	Auto White Balance
AXI	AMBA Advanced Extensible Interface as specified by ARM Ltd
B&W	Black & White
BIST	Built-In Self-Test
Camera Controls	Software included in the LibDxOISP, and dealing with the shooting parameters, commands and controls to the sensors and associated peripherals (actuators). It includes the following features Auto White Balance + Auto Exposure (with Anti Flickering and Flash control) + Auto Focus.
Chip vendor	DxO Labs customer that integrates DxO ISP solution in a complete system
Capture output	Frame-based streaming output, the supported format are YUV422 and YUV420.
DC Ultra	Synopsys Design Compiler Ultra tool
Deasserted	Set to logical “0”
Display output	Frame-based streaming output, the supported format are RGB, YUV422 and YUV420.
DMA	Direct Memory Access block
DSC	Digital Still Camera
DSLR	Digital Single Lens Reflex
DUT	Device Under Test
DVC	Digital Video Camera = camcorder
DxO ISP	DxO Labs product including DXO image processing algorithms and LibDxOISP
DxO ISP Calibration Toolbox	
DxO ISP Tuning Toolbox	
DxO ISP Toolbox	DxO ISP Calibration Toolbox + DxO ISP Tuning Toolbox
EDA	Electronic Design Automation
EOF	End Of Frame
EOL	End Of Line
EXIF	Exchangeable Image File Format

Face Detection output	Frame-based streaming output, the supported format are YUV422 and YUV420.
Firmware	code executed by the microcontroller (either embedded or external)
FPGA	Field Programmable Gate Array
fps	Frame Per Second
Frame buffer	Availability of remote memory usable by DxO ISP for data storage
FVTS	Functional Verification Test Suite
FW	Firmware
Global ASR	Global Asynchronous Reset
HAL	Hardware Abstraction Layer
HB	Hard block for physical specific implementation of memories, clock gaters, etc.
Host	External microcontroller running among others the DxO ISP firmware
HW	Hardware
ID	IDentifier
Image input	Frame based streaming input, the supported format (raw, YUV422…) depends on the hardware configuration and microcode
IP	Intellectual Property
LibDxOISP	Library running on the MCU which is the sole external interface to DxO ISP hardware IP
LSB	Least Significant Bit
LUT	Look-Up Table
MCU	Micro Controller Unit that runs and is managing the DxO hardwired ISP.
MIPI	Mobile Industry Processor Interface
MIPS	Million of Instructions per Second
MMU	Memory Management Unit
MSB	Most Significant Bit
MTF	Modulation Transfer Function
n/a	not applicable
NoC	Network on Chip
OEM	Original Equipment Manufacturer
PC	Personal Computer
PixStream	DxO Labs pixel-oriented interface with frame formatting (SOF, EOL, EOF) and flow control (Valid, Ready)
RAM	Random Access Memory
ROM	Read only memory
RTL	Register Transfer Level
SDK	Software Development Kit
SoC	System on Chip
Software integrator	In charge of LibDxOISP integration at the system level, it could be the chip vendor, an OEM, or a third party.
STA	Static Timing Analysis
Statistics	An output of the DxO ISP hardware IP available every frame, used by the AE, AWB, AF or another control mechanism running on the MCU.
TCL	Tool Command Language
USB	Universal Serial Bus
VCD	Value Change Dump
Video output	Frame-based streaming output, the supported format are YUV422 and YUV420.

2 FUNCTIONAL DESCRIPTION AND INTERFACES OVERVIEW

2.1 Overview

DxO Labs provides a fully configured image signal processing solution with the DxO ISP2013 product. Note that in this document DxO ISP designates DxO ISP2013 product.

This solution offers a high image processing performance suitable for any camera phone application. It provides on-the-fly processing without the need for a frame buffer based solution for the wide range of needs covering:

- Size-efficient implementations for area-constrained applications such as sensor SoC products,
- High-end DSC class image processing, and advanced camera image processing features

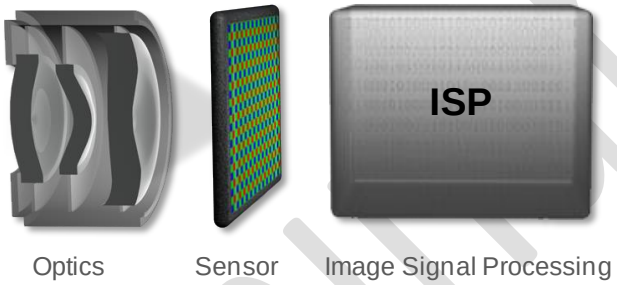


Figure 1 : Main imaging elements

DxO ISP is provided to chip vendors in the form of:

- The DxO ISP Hardware IP which includes DxO Labs highly optimized image hardwired core, as well as all the needed peripherals sub modules dealing with the specificities of the targeted chip vendor application (pre scaler, post scaler, statistics, interfaces, etc.)
- LibDxOISP library as the sole external interface to DxO ISP running on microcontroller.

In addition to the hardware and software, DxO Labs provides the DxO ISP Toolbox for fast camera module integration delivered as two distinct executables:

- DxO ISP Calibration Toolbox for automatically configuring the DxO ISP registers based on objective measurements of the camera module
- DxO ISP Tuning Toolbox for making the subjective adjustments to fit the OEM preferences in terms of image properties.

The DxO ISP has been designed to be flexible. The two axis of flexibility are described below:

- **Axis 1: hardware configurability** via hardware parameters set at design time that is sized for each chip size / performance requirements:
    - The line buffer dimensioning directly impacts the on-the-fly maximal video or still capture resolution.
    - An external frame buffer allows the implementation of a temporal noise filtering.

- **Axis 2: dynamic tunability** via DxO ISP Toolbox calibration and tuning tools:
    - Compute automatically algorithm parameters from shot calibration images and user-defined targets
    - Dynamically tune some high-level algorithm parameters depending on conditions such as gains or illuminants, etc.

2.2 Image algorithms

2.2.1 Overview

DxO ISP hardware is a high performance low power hardwired ISP which provides the following features:

- Raw Bayer to YUV image pipe featuring
  - On-the-fly singlet and couplet defective pixel correction
  - White balance compensation
  - Color lens shading compensation based on calibration
  - Lens shading compensation
  - Gr / Gb unbalance correction
  - Blue fringing reduction
  - Temporal noise reduction
  - DxO xISO monoscale (luma and chroma noise reduction with fine grain preservation)
  - Demosaicing
  - Sharpening / smoothing with automatic noise control
  - Color matrix
  - Tone curve
  - Adaptive lighting
  - Color artefact reduction
- Statistics for Auto-Focus, Auto-Exposure and Auto-White-Balance
- Up to 4 output with independent bicubic or bilinear scaling allowing simultaneous output to display, still codec, video codec, and face detection (optional)

The DxO ISP firmware provides the following features:

- Camera control
  - Auto White Balance,
  - Auto Exposure,
  - Auto Focus.
- DxO ISP hardware control
  - DxO ISP firmware manages completely the DxO ISP hardware registers
  - DxO ISP firmware is active at every frame
  - DxO ISP hardware register settings for the next frame depending upon

- The user settings (such as image size)
- Exposure and white balance
- DxO ISP Calibration Data for the given Device
- Tuning settings

2.2.2 Imaging features

DxO ISP incorporates a number of features, as described in previous sections. The features themselves are not necessarily implemented in a sequential order.

2.2.2.1 Parameters

For each frame, the DxO ISP parameters are chosen depending on the mode and the parameters of the shot. The parameters for most algorithms in DxO ISP depend on the sensor, the noise level, the illuminant and the scene content (for example, bright areas have less visible noise than dark areas). This applies, among others, to detail preservation and MTF enhancement.
Luminance & Color shading correction in DxO ISP is programmable through several 2D interpolated LUT. There is no constraint on the symmetry of the correction. Each color channel has its own correction map.
The color matrix varies with the gain and the illuminant. Gamma correction is independent for each color channel. It is adjusted automatically for each frame depending on Auto Exposure and Auto White Balance.

2.2.2.2 Auto Exposure (AE)

Auto Exposure computes the gain and exposure time to be applied by the sensor in order to adapt to the scene. It includes:

- Automatic backlighting detection and adaptation.
- Automatic 50/60Hz flicker detection and correction: Auto flicker detection runs continuously as part of the AE. It detects whether the ambient light is 50Hz or 60Hz by observing the apparition of banding. Once it is detected, the exposure times is forced to be a multiple of the lighting period. If no flickering is observed, the exposure time is not constrained.
- AE strategies: user can define its own region of interest with associtated weight.
- Manual settings: available to expert users: examples of AE modes supported by DxO ISP are ISO priority, shutter speed priority, full manual.
- Flash control: linked to the AWB, with the supported modes : Auto, On, Off
- Can be linked with a face detection system

2.2.2.3 Auto White Balance (AWB)

Auto White Balance computes the gains to be applied on red and blue channels of the image to adapt to the color of the illuminant.
It includes several heuristics over a large number of windows to detect illuminant in simple cases (one unknown light source), complex cases (several types of unknown light sources – mixed lighting) and flash cases (flash combined with one or more unknown light sources).
Auto white balance uses calibration data to adjust its behavior to sensor characteristics.

2.2.2.4 Autofocus (AF)

The Autofocus uses a measure of sharpness to find the optimum focus. DxO ISP AF is multi-zone and uses a multi-scale approach for faster convergence.
Information from the face detection can be provided to ensure focus on the faces.

2.2.2.5 DxO xISO monoscale

DxO ISP includes advanced breakthrough non linear demosaicing performed simultaneously with denoising allowing very fine details and very small grain, even in very low light in the presence of a significant level of noise. It includes also an advanced chroma noise reduction particularly useful in tungsten or low light environments.

DxO ISP contains several noise reduction algorithms: wide-area colored noise reduction, white noise reduction and color interpolation are noise-aware.

Colored noise reduction works on a very large kernel because the noise has a low spatial frequency. The colored noise reduction parameters vary with gain and white balance coefficients. They are adapted to scene content.

All these algorithms are content-adaptive.

2.2.2.6 Adaptive Lighting

Adaptive Lighting is content-adaptive and makes details in low light parts of the scene become visible.

2.2.2.7 Tuning and calibration

2.2.2.7.1 Overview

DxO Labs provides tools with DxO ISP to ease the calibration and the tuning of the image signal processing when integrating with a new camera module. The entire process can be done off-line and is therefore entirely reproducible, unlike processing images from a live sensor.
The process recommended by DxO Labs is designed around two steps, outlined in following figure: *Calibration and Tuning using DxO ISP Toolbox* below.

- First, a number of shots of targets are made with the camera module. The shots are used to extract measurements of the different characteristics of the camera module. From these measurements, a baseline configuration of the DxO ISP registers is automatically computed. This step requires no human intervention. This step is called calibration, shown in green in the figure.
- Second, natural images are shot. On natural images and on targets, the subjective tradeoffs are adjusted. These are typically tradeoffs between different imaging characteristics such as overall sharpness level, color bias, etc… The tradeoffs depend on the customer's preferences. This step is called tuning, shown in orange in the figure.

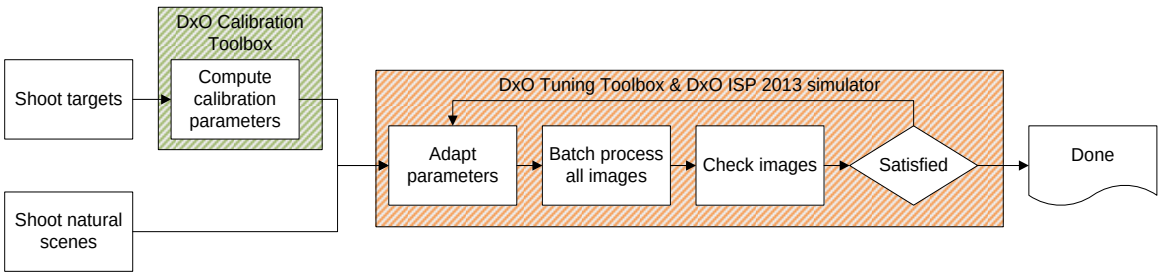


Figure 2: Calibration and Tuning using DxO ISP Toolbox

DxO ISP Toolbox includes:

- DxO ISP Calibration Toolbox,
- DxO Tuning Toolbox.

2.2.2.7.2 DxO ISP Calibration Toolbox

DxO ISP Calibration Toolbox is a command line executable on a Windows PC allowing to:

- Characterize a camera module including the sensor and the lens mounted on the sensor
- Compute the DxO ISP Calibration Data (camera specific base register settings for DxO ISP) necessary for the DxO ISP to be adapted to this camera module

This tool is configured by DxO Labs to match the specific configuration and version of DxO ISP delivered to the chip vendor.

It is provided as an executable binary and is protected by a USB dongle. It runs on a PC under Windows.

The inputs to the DxO ISP Calibration Toolbox are the following:

- A set of RAW images. The images are the captures of measurements targets in various lighting conditions according to recommendations in the DxO ISP 2013 Toolbox documentation
- A configuration file to optionally define some target values to reach

The outputs of the DxO ISP Calibration Toolbox are the following:

- DxO ISP Calibration Data in a C file including code and data - to be compiled and linked with LibDxOISP (Please refer to section 2.5.3)
- The same parameters can also be used in the DxO ISP Tuning Toolbox (Please refer to section below)

2.2.2.7.3 DxO ISP Tuning Toolbox

DxO ISP Tuning Toolbox includes :

- An API allowing to override some of the DxO ISP 2013 registers to tune DxO ISP 2013 on a PC,
- A command line executable that processes a raw image by a simulator of DxO ISP 2013 depending upon DxO ISP 2013 Calibration Data and register settings. This executable is protected by a USB dongle. It runs on a PC under Windows.

The inputs to the DxO ISP Tuning Toolbox are:

- A set of test raw images
- Tuning sliders defined in a configuration file
- DxO ISP 2013 Calibration Data produced by the DxO ISP Calibration Toolbox

The outputs of the DxO ISP Tuning Toolbox shall be:

- A set of RGB images processed by DxO ISP
- Tuning register values

2.2.2.7.3.1 DxO ISP Toolbox documentation

The DxO ISP Toolbox documentation (Please refer to Ref[3]) includes the following information:

- User manual for DxO ISP Calibration Toolbox and DxO ISP Tuning Toolbox

- Guidelines for shooting pictures required for DxO ISP Calibration Toolbox and DxO ISP Tuning Toolbox
- Description of the DxO ISP tuning registers
- Description of the file formats used by the DxO ISP Toolbox for raw image, shooting conditions and DxO ISP Calibration Data
- Recommendations on calibration and tuning evaluation and verification

2.3 Hardware architecture

2.3.1 DxO ISP

DxO Labs developed DxO ISP as a hardwired image core processing which is a configurable.
DxO Labs delivers Silicon IP for image signal processing solution in the form of:

- A proprietary configurable hardwired core processing and associated peripherals such as pre-scaler and post-scaler. It is in charge of all heavy-duty pixel level computation over a frame.

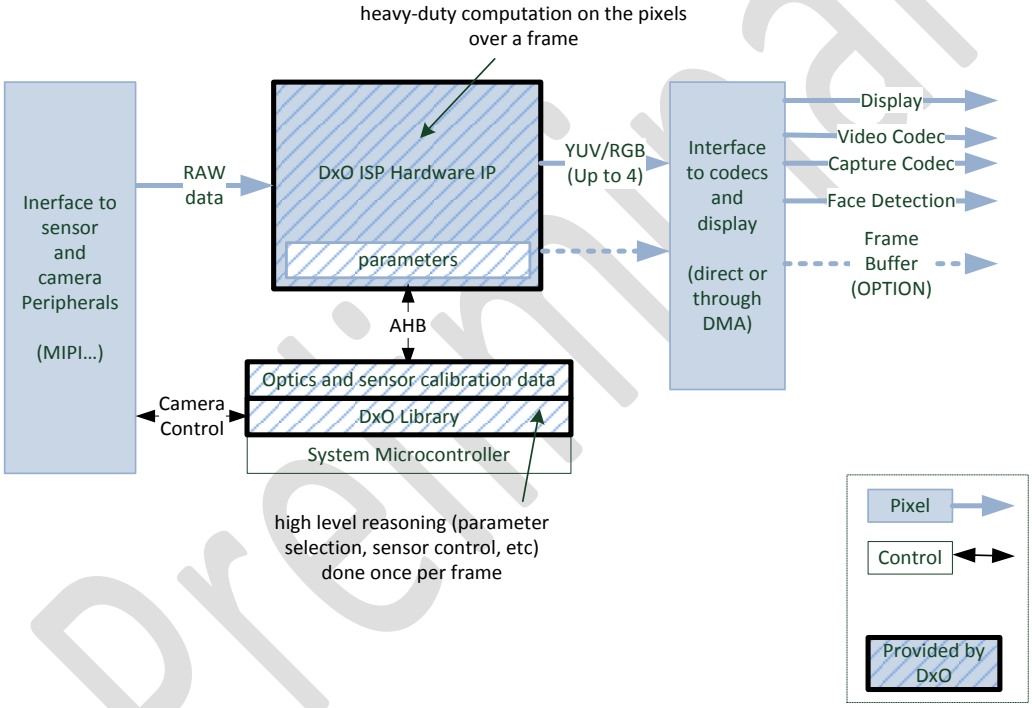


Figure 3: DxO ISP and its environment

The processing is divided into two parts:

- The heavy duty computation on pixels is done on DxO Hardwired ISP
- The high level reasoning (selection of parameters, control of the sensor, auto white balance) is done on the system microcontroller, which is better adapted to this kind of task.

A microcontroller library provided by DxO Labs, LibDxOISP, is used as the sole interface to control DxO ISP from the chip level. The communication is done over an AHBL bus and interruption by libDxOISP. libDxOISP reads status and statistics from DxO Hardwired ISP and set the parameters for the frame.

2.3.2 DxO ISP interfaces

DxO ISP has input/output interfaces for pixels using Pixstream protocol and an AHB Lite interface plus an interrupt output for control and commands.

DxO ISP input interfaces are normally connected to a sensor, but can also be connected to an input DMA for file to file processing or verification.

DxO ISP standard output interfaces are:
- The display output whose data is meant to be used for the screen.
- The video interface used for video recording whose data is meant to be sent to a video codec
- The capture interface whose data is meant to be sent to a JPEG codec.
- The face detection interface whose data is meant to be sent to a face detection external algorithm

DxO ISP is also connected to a frame buffer using DMA through an AXI bus.

2.4 Modes of operation – use cases

The present section describes the use cases supported by DxO Labs product.

In all the figures describing the supported modes of operation, the DxO ISP is the combination of:
- One or several pre-scalers.
- The DxO core image processor.
- One or several post-scalers.
- One Statistics engine.

2.4.1 Streaming

DxO ISP acquires the input pixel stream and apply one single set of image correction and transformation for all outputs.

The DxO ISP computes the statistics and the micro-controller runs the camera controls including:
- Auto Exposure, adaptation of sensor gain and exposure time to scene characteristics, taking into account the output of the anti-flickering.
- Auto White Balance, adaptation of the image pipe to the scene illuminant.
- If applicable, mechanical Auto Focus and adaptation of the focusing distance to the point of interest in the scene.
- Auto Flash, automatic control for flash.
- Correction of Gr/Gb unbalance and crosstalk variation within the field.

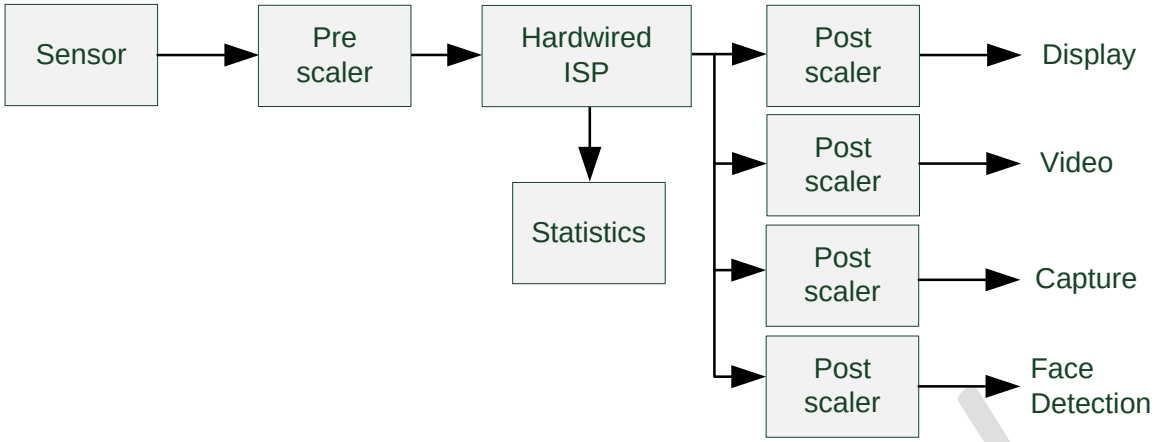


Figure 4: Streaming operating mode

The Display output main characteristics are:

- Output resolution: from Display output minimum to maximum size as defined in DxO ISP delivered configuration. For instance from QCIF to 1080p.
- Output format: a choice among RGB888, RGB565, YUV420 and YUV422.

The Video output main characteristics are:

- Output resolution: from Video output minimum to maximum size as defined in DxO ISP delivered configuration. For instance from QCIF to 2400x1350 (1080p + 10%: to allow the integration of an external video stabilization system)
- Output format: a choice among YUV420 and YUV422.

The Capture output main characteristics are:

- Output resolution: from Capture output minimum to the maximum size as defined in DxO ISP delivered configuration. For instance from QCIF to max sensor size supported.
- Output format: a choice among YUV420 and YUV422.

The Face detection output main characteristics are:

- Output resolution: from Face detection output minimum to the maximum size as defined in DxO ISP delivered configuration. For instance from QCIF to 720p
- Output format: a choice among YUV420 and YUV422.

Please note that each output is optional and can be all disabled, in this case the processing is activated for statistics computing only.

2.4.2 Streaming with temporal noise reduction

This use case is the same as the previous one, except the DxO ISP use the frame buffer to store the previous filtered frame.

Streaming with temporal filtering uses frames N-1 (or a combination of frames before N-1) and N to compute frame N.

In the figure below, as frame N is output from the sensor, frame N-1 is read from memory. Simultaneously, frame N is written to memory. Frame N is output on the activated output.

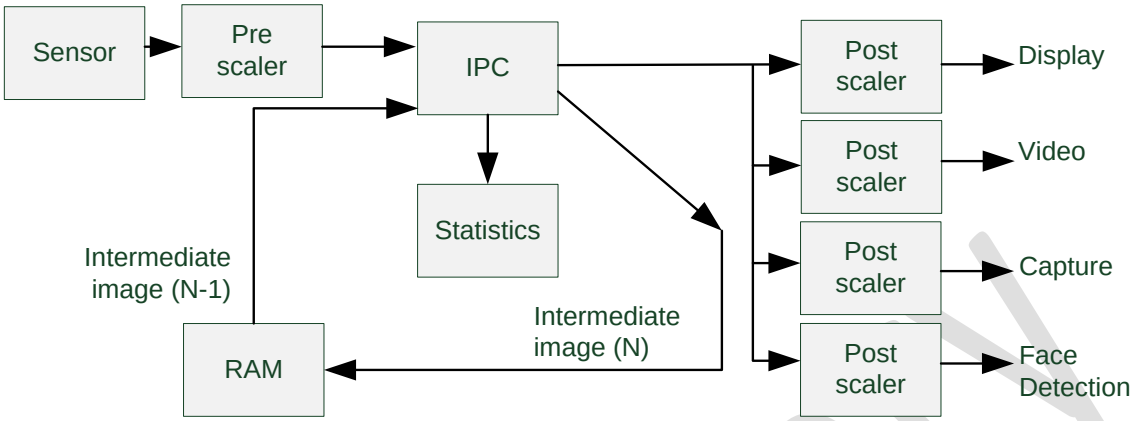


Figure 5: Streaming with temporal noise reduction

2.4.3 RAW data acquisition

The DxO ISP is bypassed and the RAW stream coming from the sensor goes directly to the frame buffer.

Figure 6: RAW data acquisition

- Ouput format: RAW only
- Resolution: up to the maximum sensor image size

2.4.4 Frame buffer sizing

For the temporal noise reduction the frame buffer size depends of the image size processed by the DxO ISP (each pixels are stored on 10bits).

Concerning the RAW data acquisition the frame buffer size corresponds to the image size coming from the sensor.

Please note that DxO Labs delivers a C function to compute accurately the size required for the frame buffer as a function of the use case and the relevant parameters.

2.5 Software architecture

2.5.1 Top level

Camera subsystem contains a DxO Labs library named LibDxOISP with interfaces allowing the chip vendor to plug one or more sensors, and to parameter image treatment algorithms and camera controls.

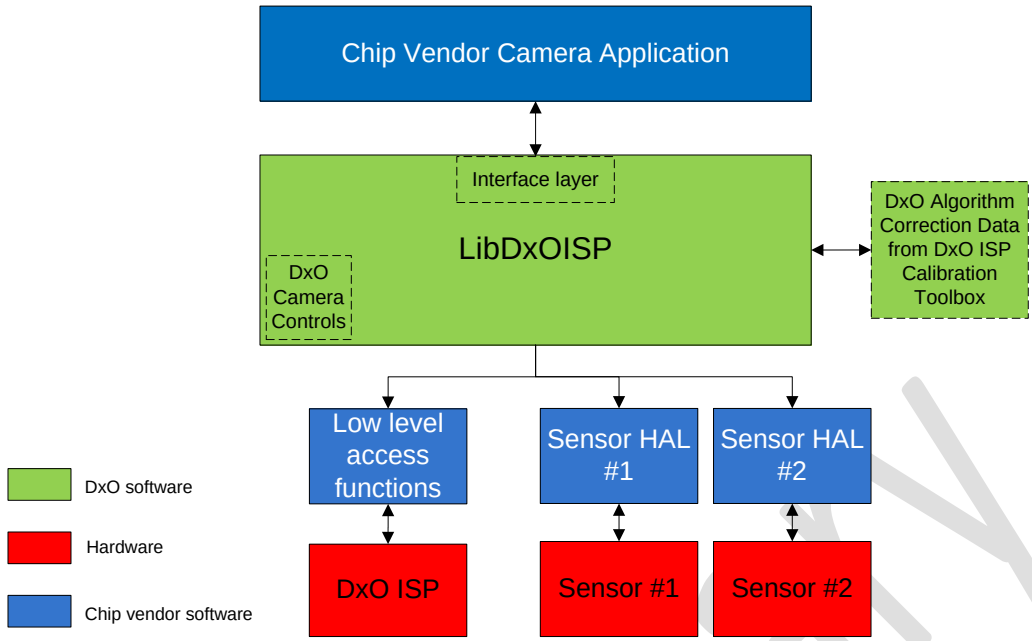


Figure 7: Chip Vendor top level software architecture

The entire LibDxOISP is running on the chip vendor MCU.

2.5.2 DxO ISP State Machine

The LibDxOISP includes a state machine. Each state reflects the on-going operating mode. It results from the command received through the command interface from the camera application running on the MCU. When a new state is set to the firmware, the status is updated right before outputting the first image in this new mode, if relevant.

In the following diagram all state transitions are under the camera application responsibility. LibDxOISP does not change states on its own.

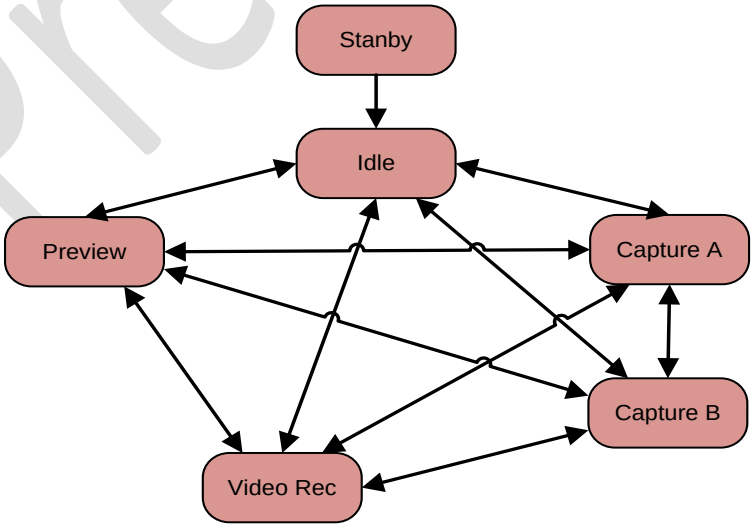


Figure 8: LibDxOISP embedded state machine

State	Description
Standby	State of the DxO ISP after boot.
Idle	State of the DxO ISP after initialization sequence. Once reached, DxO ISP is fully functional.
Preview	Image pipe is used to output image on each activated output. Camera controls are processing. The sensor streams images continuously. The use case can be: - Streaming - Streaming with temporal noise reduction
Capture A	Image pipe is used to output image on capture output and each other activated output. Once a frame has been processed, the sensor returns to idle state, unless another possible state is requested. The use case can be: Streaming (for N frames: see register description to set N)
Capture B	Same as Capture A. See application note below.
Video Rec	Image pipe is used to output image on each activated output. Camera controls are processing. The sensor streams images continuously. The use case can be: - Streaming - Streaming with temporal noise reduction

Table 2: LibDxOISP State Machine – States Overview

Application note:

- Preview and video Rec state differs from camera control perspective.
- Capture is implemented in two states: Capture A and Capture B. These state are equivalent. Transitions from Capture_A to Capture_B or Capture_B to capture_A are checked and lead to image captures. It is the way to perform multiple sequential captures if a new setting is requirement between each capture. Bracketing is the typical use of this feature.
- The switch to CaptureA/B state is directly linked to the camera control convergence time. Camera controls provided by DxO Labs always converge in less than 4 frames (without flash) with a sensor having a reasonable reaction time
- Transitions from Standby state to any other state but Idle are forbidden.
- It is strongly recommended to check that a requested state has been reached by the state machine, by reading it from the status. Once it is done, one may consider requesting another new state.
- Returning to the standby state may be done from any state.

Caveat:
Once state Capture A/B is reported by the DxO ISP status, the processed frame is sent to the capture output. In order to keep the DxO ISP streaming, another state must be immediately requested on the interface. Otherwise, the streaming is stopped.

The toggle delay between Capture A and Capture B states may change. It depends on the precise time the new capture state is claimed.

Please refer to dedicated section for detailed information about commands and status.

2.5.3 Calibration data

2.5.3.1 Difference between calibration & tuning

The calibration process consists in an objective measurement of camera module characteristics:

- It is done using the DxO ISP Calibration Toolbox.
- It produces code and data, that finally will make a CalibrationDataSelection file.

The tuning process consists in subjective adjustments of image rendering to meet the OEM preferences:

- It is done using the DxO ISP Tuning Toolbox.
- It produces a file that includes a list of register offset/value pairs.
- Tuning may be done after software freeze.

The list of (register, value) is set using the LibDxOISP API.

2.5.3.2 Data object from calibration tool box

The DxO ISP Calibration Toolbox produces a C source file. From there on:

- That file is compiled and incorporated to the software.
- Its data and code are used by functions belonging to LibDxOISP.
- The LibDxOISP functions produce correction data matching the camera module characteristics.
- They are called by the firmware on a frame per frame basis during vertical blanking.
- In accordance with the shooting parameters computed by the camera control software (gains, white balance, user settings …), they select the appropriated image processing parameters (filter coefficients, color matrix, tone curve …) for the next frame.

2.6 Software interfaces

The API of the LibDxOISP is mainly composed of structures of registers, for command and status and accessed through functions calls.

2.6.1 Memory mapped register access functions

Registers are accessed through 2 access functions: **DxOISP_CommandSet()** and **DxOISP_StatusGet()**.

2.6.2 Grouping functions

Two types of exchanges exist between the chip vendor camera application and the LibDxOISP:

- Synchronous transfers that have a direct impact on the on-going configuration and/or processing. For example, it corresponds to a full set of commands that have to be all applied in order to keep coherency within the DxO ISP such as the x and y coordinate of a crop.
- Asynchronous transfers that could be initiated regardless the current status the former / following transfers. For example, it corresponds to some tuning settings (command), or capabilities of the ISP (status).

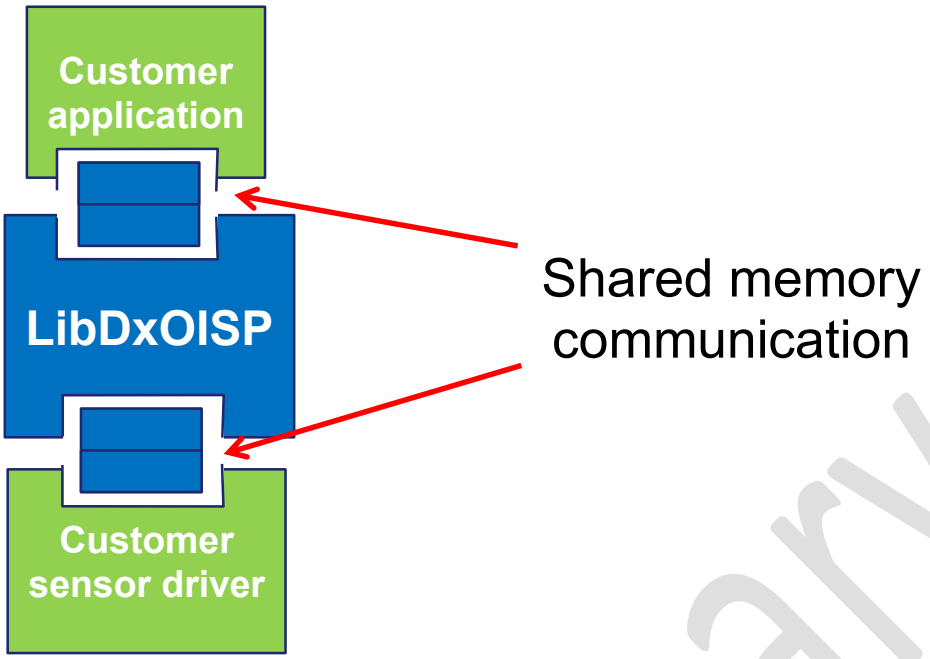


Figure 9: Shared memory access

The features of the synchronous and asynchronous communications are:

1. Synchronous structure (one in each communication direction)
    a. Memory expensive (data structure is replicated 3 times)
    b. Capable of atomic accesses (e.g. crop in x and y)
2. Asynchronous structure (one in each communication direction)
    a. Memory efficient (a single data structure exists)
    b. Adapted for large data (capabilities, configuration)
    c. Useful for quasi-static data, when atomicity is not required or when synchronization can be ensured by other means

Functions **DxOISP_CommandGroupOpen()**, **DxOISP_CommandGroupClose()**, **DxOISP_StatusGroupOpen()** and **DxOISP_StatusGroupClose()** allow guaranteeing atomicity of the accesses (made via set and get functions) done between them.

Actual transfer between private memory and shared memory occur in functions **DxOISP_CommandGroupClose()** and **DxOISP_StatusGroupOpen()**.

2.6.3 Routing interrupt

DxO ISP offers an interrupt interface with one event used to activate **DxOISP_Event()** function on the external microcontroller.

The chip vendor software application must configure the ISP so that the call to function **DxOISP_Event()** occurs early enough before the next frame, in order to guarantee the execution of the function during the vertical blanking of the sensor.

3 CHIP LEVEL REQUIREMENTS AND REGISTER INTERFACES

The following sections describe the camera sub-system software architecture and the way to control the DxO ISP from a remote micro controller through the LibDxOISP delivered software entity.

The main components of the DxO Labs software adaptation layer are:

- The boot / initialization
- The user interface (including the functions and the registers descriptions)
- The interrupt
- The sensor hardware abstraction layer interface (including the functions and the registers descriptions)
- The calibration data for both DxO Labs delivered image processing algorithms and camera controls.

In the following register map sections (Please refer to sections 3.2.3 and 3.2.10), commands registers and status registers are described. Commands registers are write only for the camera application and give access to all functionalities of the DxO ISP. The status registers are read only for the camera application and give information about the state and capabilities of the DxO ISP.

For both the command and the status registers, there are a synchronous and an asynchronous part. The synchronous part uses a group functionality to guarantee an atomic access where needed whereas the asynchronous part is accessed without atomic considerations, leaving the synchronization responsibility to the chip vendor (typically, such synchronization is simply not needed).

For each register, the encoding tells whether the register uses a signed (s) or an unsigned (u) representation. If the value is a plain integer, the width (in bits) is given (for example: 8u, 16u, 8s, 16s). If the value needs a fractional part, the width of integer part (in bits) and the width of the fractional part are separated by a decimal point. For example: 12.4s, 12 bits are for the integer part (from -2048 to 2047) and 4 bits for the fractional part (from 0/16 to 15/16).

3.1 Camera subsystem architecture overview

Camera subsystem contains a DxO Labs library named LibDxOISP with interfaces allowing the chip vendor to drive one or more sensors, and to customize image treatment algorithms and camera controls.

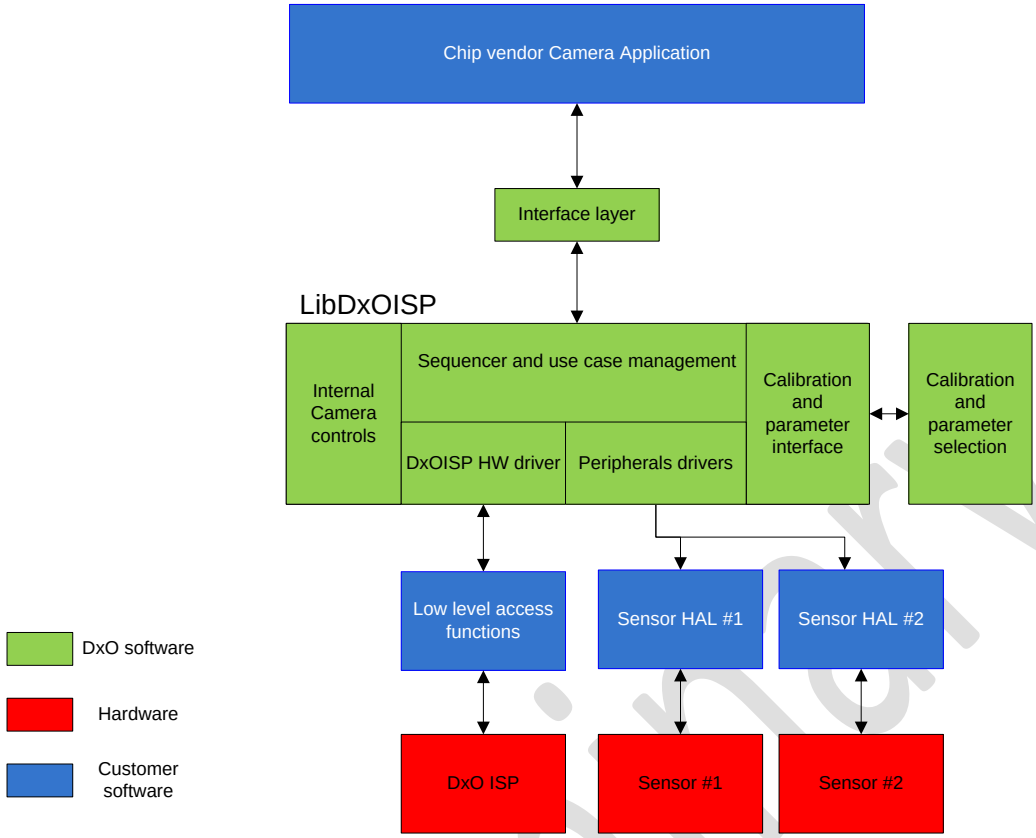


Figure 10: Camera Subsystem architecture

The LibDxOISP contains:

- The DxO Labs camera controls (LibCameraControl)
- Interface to the calibration files (generated by the calibration toolbox)
- Driver for DxO ISP hardware
- Sequencer and use cases management

The chip vendor is in charge of writing:

- Camera application based on the interface layer
- Sensor hardware abstraction layer according to the interface defined with the LibDxOISP

3.2 Interfaces

3.2.1 Boot

The camera application must first call function DxOISP_Init() in order to use LibDxOISP.

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_Init(void);
```

DESCRIPTION

This function initializes all fields described in the register map (Please refer to section 3.2.3) with default values. It is forbidden to call any other DxOISP function before calling this one, including DxOISP_Event().

3.2.2 User Interface description

DxO ISP is driven through a register interface and using access functions. There are two types of registers: command registers and status registers.
Accesses are done with functions DxOISP_CommandSet and DxOISP_StatusGet. Some accesses are synchronous. Then, grouping function calls are required. Other accesses are asynchronous. It depends on the location of the register is in the register map (as explained in the section 2.6).

3.2.2.1 DxOISP_CommandSet

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_CommandSet (uint16_t usOffset, uint16_t usSize, void* pBuf);
```

DESCRIPTION

usOffset	a 16 bits unsigned value providing the offset in the logic mapping of command registers. This value can be obtained from DxOISP.h as an offset in a structure pointed by pBuf. It is recommanded to use the macro DxOISP_CMD_OFFSET() with the field name to obtain the offset. This macro is defined in file “DxOISP.h”.
usSize	a 16 bits value providing the length in bytes of the data buffer.
pBuf	a pointer to the data buffer holding the value(s) to upload to the command registers.

DxOISP_CommandSet writes usSize bytes from the data buffer pointed to by pBuf to the consecutive command registers starting at usOffset.

EXAMPLE USAGE

```
DxOISP_CommandSet(
      DxOISP_CMD_OFFSET(stSync.stControl)
,     sizeof(UserRamZone.stSync)
,     (void*)&UserRamZone.stSync.stControl
);
```

3.2.2.2 DxOISP_StatusGet

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_StatusGet(uint16_t usOffset, uint16_t usSize, void* pBuf);
```

DESCRIPTION

usOffset	a 16 bits unsigned value providing the offset in the logic mapping of status registers. It is recommended to use the macro DxOISP_STATUS_OFFSET() with the field name to obtain the offset. This macro is defined in file “DxOISP.h”.
usSize	a 16 bits value providing the length in bytes of the data buffer.
pBuf	a pointer to a buffer that will contain the data downloaded from the status registers.

DxOISP_StatusGet reads usSize bytes from the consecutive status registers starting at usOffset to the data buffer pointer to by pBuf.

EXAMPLE USAGE

```
DxOISP_StatusGet(
```

```
        DxOISP_STATUS_OFFSET(stSync.stSystem.eMode
,       sizeof(uint8_t)
,       &ucStatusMode
);
```

3.2.2.3 Grouping functions

It may be required that one or several commands are applied as a single transaction to ensure that no partial modification is applied when processing a frame.

For example, a change in frame size is expressed as a change of the registers holding the width and the height of the frame. These two (or more) registers need being modified simultaneously.

This is done using the “open” and “close” functions described below.

All calls to the DxOISP_CommandSet() function performed between a call to DxOISP_CommandGroupOpen() and a subsequent call to DxOISP_CommandGroupClose() are guaranteed to be applied simultaneously.

The same mechanism applies to retrieve status, such as “Exif”[1] parameters that should be coherent whenever they are acquired by the user. In this case the user will use DxOISP_StatusGroupOpen() function before the DxOISP_StatusGet() call sequence needed to retrieve the data. Finally, DxOISP_StatusGroupClose() is called.

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_CommandGroupOpen(void);
```

DESCRIPTION

The DxOISP_CommandGroupOpen function is used to start a group of atomic updates to command registers.
All calls to DxOISP_CommandSet() made between a call to DxOISP_CommandGroupOpen() and a subsequent call to DxOISP_CommandGroupClose() are applied simultaneously.

SYNOPSIS

```
#include "DxOISP.h"
uint32_t DxOISP_CommandGroupClose(void);
```

DESCRIPTION

The DxOISP_CommandGroupClose() function is used to end a group of atomic updates to command registers.
All calls to DxOISP_CommandSet() made between a call to DxOISP_CommandGroupOpen() and a subsequent call to DxOISP_CommandGroupClose() are applied simultaneously.

A nominal sequence to write commands looks like:

- DxOISP_CommandGroupOpen ()

[1] EXIF is a standard for recording metadata information in image file. It stands for Exchangeable Image File format (see www.exif.org). In practice, it contains for example, the exposure time, the aperture, whether the flash was used or not, etc…

- DxOISP_CommandSet ()
- DxOISP_CommandSet ()
- …
- DxOISP_CommandGroupClose ()

DxOISP_CommandGroupClose() returns an index. The commands will be taken into account at the frame referenced by the that index.
The index is only meaningful if the DxOISP status read by the camera application is different from DxOISP_MODE_STANDBY and DxOISP_MODE_IDLE. When the DxO ISP status is DxOISP_MODE_STANDBY or DxOISP_MODE_IDLE, the returned value always is 1 (because we are not streaming frames). The index of the first frame is 0.

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_StatusGroupOpen(void);
```

DESCRIPTION

The DxOISP_StatusGroupOpen function is used to guarantee the atomicity of the group of status elements.
All calls to DxOISP_StatusGet() made between a call to DxOISP_StatusGroupOpen() and a subsequent call to DxOISP_StatusGroupClose() refer to data retrieved simultaneously.

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_StatusGroupClose(void);
```

DESCRIPTION

The DxOISP_StatusGroupClose function is used to guarantee the atomicity of a group of status elements.
All calls to DxOISP_StatusGet() made between a call to DxOISP_StatusGroupOpen() and a subsequent call to DxOISP_StatusGroupClose() refer to data sampled simultaneously.

A nominal sequence to read status looks like:

- DxOISP_StatusGroupOpen()
- DxOISP_StatusGet()
- DxOISP_StatusGet()
- …
- DxOISP_StatusGroupClose()

3.2.3 User interface registers

3.2.3.1 Synchronous Command Section – General Commands

Name:	**eMode**		
fullName:	ts_DxOIspSyncCmd.stControl.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_MODE_STANDBY		
Holds the mode the user wants to put the ISP in. Possible values are: • DxOISP_MODE_STANDBY • DxOISP_MODE_IDLE			

- DxOISP_MODE_PREVIEW
- DxOISP_MODE_CAPTURE_A
- DxOISP_MODE_CAPTURE_B
- DxOISP_MODE_VIDEOREC

The DxO ISP state machine is described at section 2.5.2.

Name:	ucSourceSelection		
fullName:	ts_DxOIspSyncCmd.stControl.ucSourceSelection		
Encoding:	8u	Unit:	enumerated
Default:	0		

ucSourceSelection selects the sensor plugged onto the DxO ISP (depending on the external hardware, this can be either a sensor or a DMA or any other hardware object behaving like a sensor). Each sensor port has a unique ID. The maximum value depends of the number sensor HAL built and linked with the DxOISP firmware delivery.

Name:	ucCalibrationSelection		
fullName:	ts_DxOIspSyncCmd.stControl.ucCalibrationSelection		
Encoding:	8u	Unit:	enumerated
Default:	0		

Select the calibration data used for camera control and image pipe. The maximum value depends of the number of calibration data built and linked with the DxOISP firmware library.

Name:	ucNbCaptureFrame		
fullName:	ts_DxOIspSyncCmd.stControl.ucNbCaptureFrame		
Encoding:	8u	Unit:	none
Default:	1		

Number of consecutive frames acquired when DxOISP_MODE_CAPTURE_A or DxOISP_MODE_CAPTURE_B modes are requested

Name:	isTNREnabled		
fullName:	ts_DxOIspSyncCmd.stSystem. isTNREnabled		
Encoding:	8u	Unit:	boolean
Default:	0		

If set to 1 the temporal noise reduction is enabled.
This mode is available only in mode DxOISP_MODE_PREVIEW and DxOISP_MODE_VIDEOREC.

Name:	eImageOrientation		
fullName:	ts_DxOIspSyncCmd.stControl.eImageOrientation		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_FLIP_OFF_MIRROR_OFF		

Sets the image orientation of the active input. The DxO ISP firmware will program the relevant sensor accordingly to output the image in the desired orientation. The meaning of orientation is explained by Figure 11: image orientation illustration. Possible values are:

- DxOISP_FLIP_OFF_MIRROR_OFF
- DxOISP_FLIP_OFF_MIRROR_ON
- DxOISP_FLIP_ON_MIRROR_OFF
- DxOISP_FLIP_ON_MIRROR_ON

Why modifying orientation?

1- The most commun need for such feature is to compensate the assembly constraint of the sensor module into the mobile phone. A sensor may be assembled backwards and image output by the sensor need to be oriented accordingly to display the scene as the user sees it.
2- In case of double sensor mobile phone (usually a back side high resolution sensor for the camera application and a front side lower resolution one for video conference calls), the application needs to correct the mirroring effect occurred when displaying the image.
3- Some mobile have an unfoldable and twistable main display. In order to display the scene in the same manner what ever the display position, the camera application must controls the orientation of the image and thus set accordingly that register to the DxO ISP.

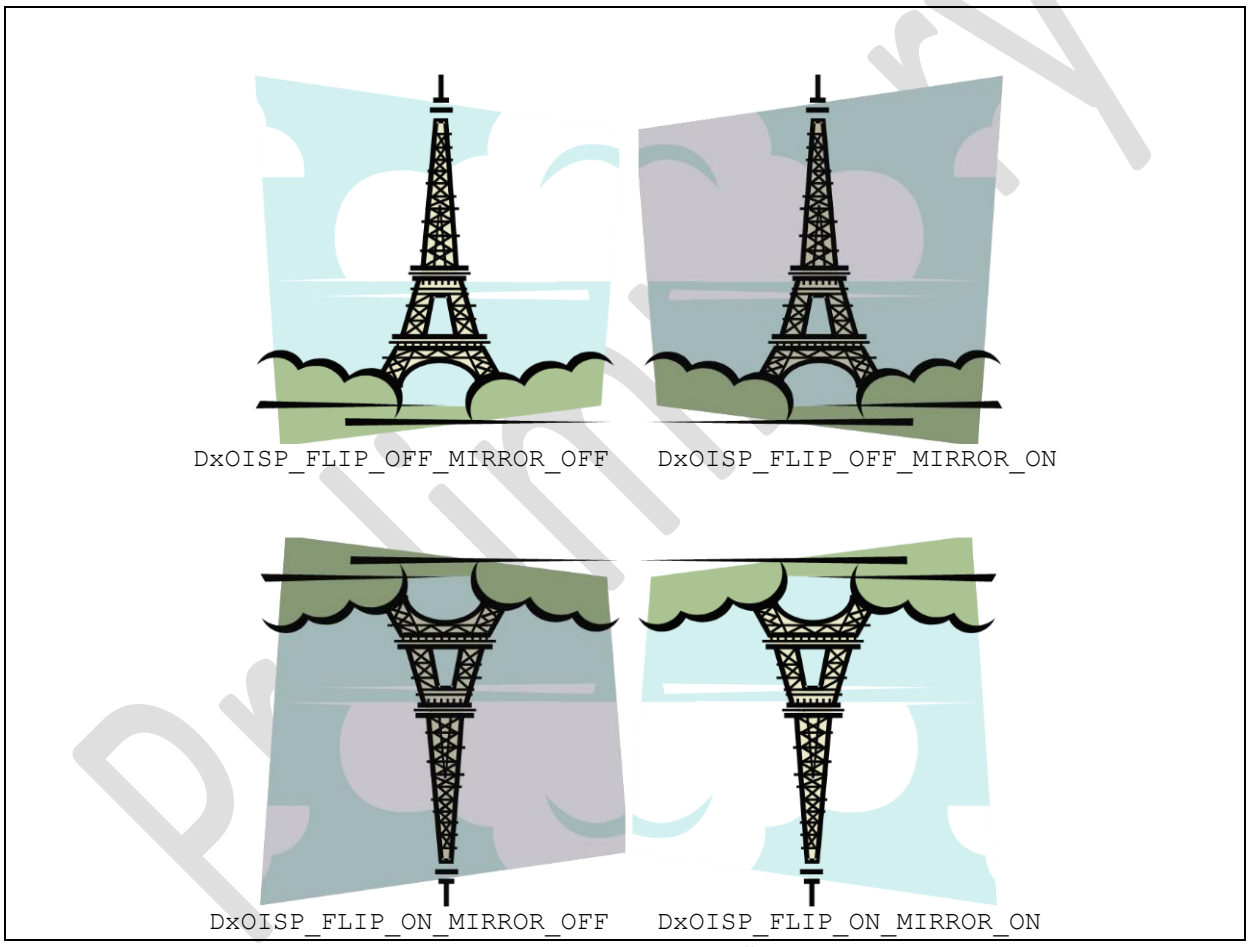


Figure 11: image orientation illustration

Name:	**ucInputDecimation**		
fullName:	ts_DxOIspSyncCmd.stSystem. ucInputDecimation		
Encoding:	8u	Unit:	none
Default:	1		
Decimation factor applied before image processing on the sensor or in the DxO ISP depending on the sensor capabilities.			

Name:	usFrameRate		
fullName:	ts_DxOIspSyncCmd.stControl.usFrameRate		
Encoding:	12.4u	Unit:	Hz
Default:	30 << 4		
Sets the target frame rate. LibDxOISP achieves a frame rate always less than or equal to the target frame rate, and tries to be as close as possible to it, combining the hardware and sensor configurations.			

Name:	usZoomFocal		
fullName:	ts_DxOIspSyncCmd.stControl.usZoomFocal		
Encoding:	8.8u	Unit:	none
Default:	1 << 8		
Sets the optical zoom if any. Register value is the ratio between focal distance and sensor diagonal size expressed in 8.8. The value is sent directly to the sensor hardware abstraction library.			

Name:	usFrameToFrameLimit		
fullName:	ts_DxOIspSyncCmd.stControl.usFrameToFrameLimit		
Encoding:	0.16u	Unit:	none
Default:	0xFFFF		
Sets the perceived speed motion of the digital zoom to go from the current image out size to the new ordered one. It is expressed as the ratio of the maximum frame to frame distance of any frame corner relative to width/height (horizontal distance is relative to width, vertical distance is relative to height). The trajectory of each corner is guaranteed to be a straight line and steps are as large as possible while enforcing the above mentioned rule. When set to maximum value (0xFFFF), new frame size is reached in one frame. In case of state transition with a modification of the input crop (registers ts_DxOIspSyncCmd.stControl.stOutput[0 to 4].stCrop), this register must be set to 0xFFFF. If usFrameToFrameLimit is different from 0xFFFF, the sensor remains in the same configuration and every new crop is performed by the the ISP excepted if the new crop is larger than the sensor's one. Once a new crop position is reached by the ISP, this crop can be applied on the sensor by setting the usFrameToFrameLimit to 0xFFFF.			

The following registers deal with the geometry of the various DxO ISP outputs. The DxO ISP supports the following outputs:

- A display ouput, used by the equipment screen.
- A video codec output, used by the video encoder of the equipement.
- A capture output, used by the JPEG encoder of the equipement.
- A face detection output, used by the face detection part of the equipement.

Every output may be set separately. The camera application may request any width and height on every output as long as the total number of pixels (width*height) does not exceed the statically configured bandwidth capabilities and the largest programmed image width does not exceed the statically configured maximum image width.
When the crop of the various outputs differ, a global crop, computed as the union of all active crops, is applied to the sensor.

Below are described the settings for any output.
outputIdx is a can be the following enumerate value:

- DxOISP_OUTPUT_DISPLAY
- DxOISP_OUTPUT_VIDEO
- DxOISP_OUTPUT_CAPTURE
- DxOISP_OUTPUT_FD

Name:	isEnabled		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].isEnabled		
Encoding:	8u	Unit:	Boolean
Default:	0		
If set to 1 the corresponding output is enabled. Set to 0 to disable the output.			

Name:	eFormat		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].eFormat		
Encoding:	8u	Unit:	Enumerated
Default:	DxOISP_FORMAT_YUV422		

Sets the required format for the considered output. For all outputIdx values, possible eFormat values are:

- DxOISP_FORMAT_YUV422
- DxOISP_FORMAT_YUV422_NV
- DxOISP_FORMAT_YUV420

Note: YUV422_NV corresponds to a YUV422 format but both U have the same value, and both V have the same value too.

If outputIdx is DxOISP_OUTPUT_DISPLAY, possible additonal eFormat values are:

- DxOISP_FORMAT_RGB565
- DxOISP_FORMAT_RGB888

If outputIdx is DxOISP_OUTPUT_CAPTURE another possible additional value is DxOISP_FORMAT_RAW. This is a particular case used to grab pixels coming from the sensor directly into the frame buffer. This is useful to debug or during calibration process.
Note: In this case no pixels are sent to the output Capture, as pixels goes to the frame buffer, so memory must be allocated for it.

Name:	eOrder		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].eOrder		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_YUV422ORDER_YUYV		

Sets the order for the pixels on the considered output for YUV formats.

If eFormat is DxOISP_FORMAT_YUV422 or DxOISP_FORMAT_YUV422_NV, possible values for eOrder are:

- DxOISP_YUV422ORDER_YUYV
- DxOISP_YUV422ORDER_YYUV
- DxOISP_YUV422ORDER_UYYV
- DxOISP_YUV422ORDER_UVYY
- DxOISP_YUV422ORDER_UYVY
- DxOISP_YUV422ORDER_YUVY

- DxOISP_YUV422ORDER_YVYU
- DxOISP_YUV422ORDER_YYVU
- DxOISP_YUV422ORDER_VYYU
- DxOISP_YUV422ORDER_VUYY
- DxOISP_YUV422ORDER_VYUY
- DxOISP_YUV422ORDER_YVUY

If eFormat is DxOISP_FORMAT_YUV420, possible values for eOrder are:

- DxOISP_YUV420ORDER_YYU
- DxOISP_YUV420ORDER_YUY
- DxOISP_YUV420ORDER_UYY
- DxOISP_YUV420ORDER_YYV
- DxOISP_YUV420ORDER_YVY
- DxOISP_YUV420ORDER_VYY

For YYU, YUY and UYY based ordering value, U samples are output on even lines and V samples are output on odd lines. For YYV, YVY and VYY based orderings, V samples are output on even lines and U samples are output on odd lines.

If eFormat is DxOISP_FORMAT_RGB888, possible values for eOrder are:

- DxOISP_RGBORDER_RGB
- DxOISP_RGBORDER_RBG
- DxOISP_RGBORDER_GRB
- DxOISP_RGBORDER_GBR
- DxOISP_RGBORDER_BGR
- DxOISP_RGBORDER_BRG

If eFormat is DxOISP_FORMAT_RGB565, consistant values for eOrder are:

- DxOISP_RGBORDER_RGB
- DxOISP_RGBORDER_BGR

The next examples illustrate the role of that parameter:
If eOrder is DxOISP_YUV422ORDER_YYUV, all lines have the following pixel order:

- 1st pixel: Y
- 2nd pixel: Y
- 3rd pixel: U
- 4th pixel: V

If eOrder is DxOISP_RGBORDER_RGB, all lines have the following pixel order:

- 1st pixel: R
- 2nd pixel: G
- 3rd pixel: B

If eOrder is DxOISP_YUV420ORDER_YYU, all lines have the following pixel order:

- 1st pixel: Y
- 2nd pixel: Y
- 3rd pixel: U

Name:	eEncoding		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].eEncoding		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_ENCODING_YUV_601FS		

Sets YUV format of the considered output. This register is relevant only if eFormat either is DxOISP_FORMAT_YUV422, DxOISP_FORMAT_YUV422_NV or DxOISP_FORMAT_YUV420.
Possible values are:

- DxOISP_ENCODING_YUV_601FS
- DxOISP_ENCODING_YUV_601FU
- DxOISP_ENCODING_YUV_601CU
- DxOISP_ENCODING_YUV_601CS
- DxOISP_ENCODING_YUV_709FS
- DxOISP_ENCODING_YUV_709FU
- DxOISP_ENCODING_YUV_709CU
- DxOISP_ENCODING_YUV_709CS

F stands for full-range and means that the output will range from 0 to 255 for Y and from-128 to 127 for U and V. This is what most JPEG encoders expect.
C stands for clipped range and means the output will range from 16 to 235 for Y and from-112 to +112 for U and V.
U and S stands for unsigned and signed. The unsigned representations add 128 to U and V.

Hence, the possible ranges for the different values are:

- DxOISP_ENCODING_YUV_601FS: Y in [0, 255], U and V in [-128, 127].
- DxOISP_ENCODING_YUV_601FU: Y, U and V in [0, 255].
- DxOISP_ENCODING_YUV_601CU: Y in [16, 235], U and V in [16, 240].
- DxOISP_ENCODING_YUV_601CS: Y in [16, 235], U and V in [-112, 112].
- DxOISP_ENCODING_YUV_709FS: Y in [0, 255], U and V in [-128, 127].
- DxOISP_ENCODING_YUV_709FU: Y, U and V in [0, 255].
- DxOISP_ENCODING_YUV_709CU: Y in [16, 235], U and V in [16, 240].
- DxOISP_ENCODING_YUV_709CS: Y in [16, 235], U and V in [-112, 112].

See Ref[6] and Ref[7] for more information on these formats.

Name:	usXAddrStart		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].stCrop.usXAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		

Sets the X-address of upper left corner of the considered output crop into the visible pixel array of the active input. Value must be even and less than usXAddrEnd.

Name:	usYAddrStart		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].stCrop.usYAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		

Sets the Y-address upper left corner of the considered output crop into the visible pixel array of the active input. Value must be even and less than usYAddrEnd.

Name:	usXAddrEnd		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].stCrop.usXAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the X-address of lower right corner of the considered output crop into the visible pixel array of the active input. Value must be odd, greater than usXAddrStart and less than sensor width. Caveats: • crop width must be superior or equal to 4 pixels. • 0xFFFF is not a valid usXAddrEnd value.			

Name:	usYAddrEnd		
fullName:	ts_DxOIspSyncCmd.stControl. stOutput[outputIdx].stCrop.usYAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the Y-address of lower right corner of the considered output crop into the visible pixel array of the active input. Value must be odd, greater than usYAddrStart and less than sensor height. Caveats: • crop height must be superior or equal to 4 pixels. • 0xFFFF is not a valid usXAddrEnd value.			

Name:	usSizeX		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].usSizeX		
Encoding:	16u	Unit:	pixels
Default:	640		
Sets the width of the considered output. Must be even. Note that the digital zoom factor is computed from size and crop information.			

Name:	usSizeY		
fullName:	ts_DxOIspSyncCmd.stControl.stOutput[outputIdx].usSizeY		
Encoding:	16u	Unit:	pixels
Default:	480		
Sets the height of the considered output. Must be even. Note that the digital zoom factor is computed from size and crop information.			

Additional note: usSizeX and usSizeY combined with the crop in the active input define the zoom aspect ratio on the display output. When these values are modified, LibDxOISP, taking into account the value usFrameToFrameLimit, performs a smooth zoom, in or out, based on a geometrical suite (i.e. constant zoom ratio from one frame to the next).

```
void
getCropForZoomFactor(
    const uint16_t usImageDimension
,   const float    fZoomRatio
,   uint16_t     * usXorYAddrStart
,   uint16_t     * usXorYAddrEnd
) {
    uint16_t usCenter = usImageDimension/2;
    *usXorYAddrStart = usCenter * (1.0 - 1.0 / fZoomRatio);
```

```
    *usXorYAddrEnd    = usCenter * (1.0 + 1.0 / fZoomRatio);
}
```

Name:	uiFrameBufferBaseAddr		
fullName:	ts_DxOIspSyncCmd.stControl.uiFrameBufferBaseAddr		
Encoding:	32u	Unit:	
Default:	0		
Start address of the memory allocated for the frame buffer.			

Name:	uiFrameBufferSize		
fullName:	ts_DxOIspSyncCmd.stControl.uiFrameBufferSize		
Encoding:	32u	Unit:	
Default:	0		
Size of the memory allocated for the frame buffer. As this size depends of the use case, a function is provided to compute it (see section 3.2.7)			

3.2.3.2 Synchronous Command Section – Camera control interface

The registers below offer the camera application a way to drive the camera controls. The LibDxOISP camera controls manage six different servo-controls (mechanical autofocus, flash control, anti-flickering, auto-exposure and auto white-balance). Every sub-functionality of the camera controls can be configured independently.

Below are detailed settings for these controls.

3.2.3.2.1 Anti-Flickering related

Name:	eMode		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAfk.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AFK_MODE_AUTO		
Sets the anti-flickering mode. Possible values are: • DxOISP_AFK_MODE_AUTO • DxOISP_AFK_MODE_MANUAL When set to auto, LibDxOISP automatically detects the illumination source frequency and cancels flickering effects if possible.			

Name:	eFrequency		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAfk.stForced.eFrequency		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AFK_FREQ_0HZ		
When eMode is DxOISP_AFK_MODE_MANUAL, the camera application forces the frequency of the illumination source to cancel. Possible values are: • DxOISP_AFK_FREQ_0HZ • DxOISP_AFK_FREQ_50HZ • DxOISP_AFK_FREQ_60HZ Unless set to DxOISP_AFK_FREQ_0HZ, LibDxOISP will cancel flickering effects in case of a light of the given frequency, if possible.			

3.2.3.2.2 Auto-Exposure related

Name:	eMode		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AE_MODE_AUTO		

Sets the auto-exposure mode. The auto-exposure ensures a proper brightness of the subject (by adjusting sensor analog gains, sensor digital gains, and exposure time). It can either be entirely handled by LibDxOISP or manually set. Possible values are:

- DxOISP_AE_MODE_AUTO
- DxOISP_AE_MODE_MANUAL
- DxOISP_AE_MODE_ISO_PRIORITY
- DxOISP_AE_MODE_ISO_PRIORITY_NO_BANDING
- DxOISP_AE_MODE_SHUTTER_SPEED_PRIORITY

Application note:

Auto-exposure preset :	Description :
DxOISP_AE_MODE_ISO_PRIORITY	Fix gain – AE selects a combination exposure time+aperture
DxOISP_AE_MODE_ISO_PRIORITY_NO_BANDING	Fix gain (with flexibility to avoid banding effect) – AE selects a combination exposure time+aperture
DxOISP_AE_MODE_SHUTTER_SPEED_PRIORITY	Fix exposure time - AE selects a combination gain+aperture

Table 4: Auto-exposure preset description

Name:	cEvBias		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stControl.cEvBias		
Encoding:	3.5s	Unit:	2x (1 stop)
Default:	0.0		

Applies a bias on the target exposure in order to get an image that is darker or brighter. Range is [-3.0 : 3.0]. This value is taken into account when eMode is set to any value other than DxOISP_AE_MODE_MANUAL

Name:	isLock		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stControl.isLock		
Encoding:		Unit:	Boolean
Default:	0		

If set to 1, the auto exposure is locked to the latest estimated exposure values.
If flash mode is ON, flash will trigger but without a new estimation of AE.
If flash mode is AUTO, the flash will trigger or not depending whether the flash was reputated as needed when AE lock was activated:

- If AE lock is activated when AE says that we will need flash, then flash will trigger whatever are the conditions at the moment of the capture (unless AE lock is released of course)
- If AE lock is activated when AE estimates that no flash is required (in daylight for example), then there will never have flash until AE lock is released

Name:	uiMaxExposureTime		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stControl.uiMaxExposureTime		
Encoding:	16.16u	Unit:	ms
Default:	0xFFFFFFFF		
Sets the maximum exposure time the camera control can set in preview,or video mode. This may be useful to ensure a minimum frame rate in case of lowlight scenes. This value is taken into account when eMode is set to any value but DxOISP_AE_MODE_MANUAL.			

Name:	uiMaxExposureTimeCapture		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stControl.uiMaxExposureTimeCapture		
Encoding:	16.16u	Unit:	ms
Default:	0xFFFFFFFF		
Sets the maximum exposure time that the camera control can set in capture mode. This value is taken into account when eMode is set to any value but DxOISP_AE_MODE_MANUAL.			

Name:	uiExposureTime		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.uiExposureTime		
Encoding:	16.16u	Unit:	ms
Default:	0x888		
Sets the current exposure time. This value is taken into account when eMode is set to DxOISP_AE_MODE_MANUAL or DxOISP_AE_MODE_SHUTTER_SPEED_PRIORITY.			

Name:	usAperture		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.usAperture		
Encoding:	8.8	Unit:	2x (1 stop)
Default:	2.8 << 8		
Sets the current aperture (F number). This value is taken into account when eMode is set to DxOISP_AE_MODE_MANUAL.			

Name:	usIspGain		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.usIspGain		
Encoding:	8.8u	Unit:	None
Default:	1.0 << 8		
Sets the digital gain that will be applied by the ISP algorithm. This value is taken into account when eMode is set to DxOISP_AE_MODE_MANUAL or DxOISP_AE_MODE_ISO_PRIORITY.			

Name:	usIspTcGain		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.usIspTcGain		
Encoding:	8.8u	Unit:	None
Default:	1.0 << 8		
Sets the gain that will be applied in the tone curve by the ISP algorithm. This value is taken into account when eMode is set to DxOISP_AE_MODE_MANUAL or DxOISP_AE_MODE_ISO_PRIORITY.			

Name:	usAnalogGain[0] … usAnalogGain[3]		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.usAnalogGain[4]		
Encoding:	8.8u	Unit:	None
Default:	1.0 << 8		
Sets the analog gain for each single channel of the sensor. These values are passed to the sensor hardware abstraction library. Values are under camera application's responsibility. 0, 1, 2 and 3 channels respectively are DxOISP_CHANNEL_GR, DxOISP_CHANNEL_R, DxOISP_CHANNEL_B and DxOISP_CHANNEL_GB channels. These values are taken into account when eMode is set to DxOISP_AE_MODE_MANUAL or DxOISP_AE_MODE_ISO_PRIORITY If ISO100 is targetted, then the recommended values for the analog gains is 1.0 << 8 for all four values.			

Name:	usDigitalGain[0] … usDigitalGain[3]		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stForced.usDigitalGain[4]		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Sets the digital gain for each single channel of the sensor set as active input. These values are passed to the sensor hardware abstraction library. Values are under camera application's responsibility. 0, 1, 2 and 3 channels respectively are DxOISP_CHANNEL_GR, DxOISP_CHANNEL_R, DxOISP_CHANNEL_B and DxOISP_CHANNEL_GB channels. These values are taken into account when eMode is set to DxOISP_AE_MODE_MANUAL or DxOISP_AE_MODE_ISO_PRIORITY			

The following registers make it possible to define up to six rectangles of interest in the image. The five last rectangles are dedicated to the face detection and the first may be used to define a default region of interest on the entire image.

One can also use the regions of interest to define the metering modes. For example, if we use coordinates relative to the size of the image that goes from (0, 0) to (1, 1):

- Spot: define one rectangle from about (1/3, 1/3) to (2/3, 2/3) with weight 1.
- Center-weighted: define 3 rectangles:
  - (0, 0) to (1, 1) with weight 1
  - (1/4, 1/4) to (3/4, 3/4) with weight 2
  - (3/8, 3/8) to (5/8, 5/8) with weight 3
- Average: define one rectangle from (0, 0) to (1, 1) with weight 1.

The sum of all weights cannot bypass 255.

Note: all pixel addresses must be given after flipping/mirroring appliance by the sensor.

Name:	stFaces[6].usXAddrStart		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stRoi.stFaces[6].usXAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		
Sets the X-address of upper left corner of the considered rectangle of interest. Value must be even and less than usXAddrEnd.			

Name:	stFaces[6]. usYAddrStart		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stRoi.stFaces[6].usYAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		
Sets the Y-address of upper left corner of the considered rectangle of interest. Value must be even and less than usYAddrEnd.			

Name:	stFaces[6].usXAddrEnd		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stRoi.stFaces[6].usXAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the X-address of lower right corner of the rectangle of interest. Value must be odd, greater than usXAddrStart and less than sensor width (default value is not a valid value).			

Name:	stFaces[6].usYAddrEnd		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stRoi.stFaces[6].usYAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the Y-address of lower right corner of the rectangle of interest. Value must be odd, greater than usYAddrStart and less than sensor height (default valule is not a valid value).			

Name:	ucWeights[6]		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAe.stRoi.ucWeights[6]		
Encoding:	8u	Unit:	none
Default:	0		
Set the weight of the considered region of interest. Use the value 0 if the considered region is not used.			

3.2.3.2.3 Auto White-Balance related

The white balance ensures that colors of the subject are not altered by the illumination source. Automatic white-balance is advised in most illumination conditions. Nevertheless, camera application can be set to other preset depending on the chosen type of light source or scene conditions.

Name:	eMode		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAwb.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AWB_MODE_AUTO		
Sets the auto white-balance mode. White balance may be automatic, manual or preset. Possible values are: • DxOISP_AWB_MODE_AUTO • DxOISP_AWB_MODE_MANUAL • DxOISP_AWB_MODE_PRESET			

Name:	ePreset		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAwb.stForced.ePreset		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AWB_PRESET_DAYLIGHT		
Sets a predefined white-balance correction to apply. This value is taken into account when eMode is set to DxOISP_AWB_MODE_PRESET. Possible presets are: • DxOISP_AWB_PRESET_HORIZON • DxOISP_AWB_PRESET_TUNGSTEN • DxOISP_AWB_PRESET_TL84 • DxOISP_AWB_PRESET_COOLWHITE • DxOISP_AWB_PRESET_D45 • DxOISP_AWB_PRESET_DAYLIGHT • DxOISP_AWB_PRESET_CLOUDY • DxOISP_AWB_PRESET_SHADE			

Application note: color temperatures for each auto white-balance preset:

Preset	Color Temperature (K)
AUTO	2400°- 8000°
HORIZON	2400°
TUNGSTEN	3000°
TL84	4100°
COOLWHITE	4100°
D45	4500°
DAYLIGHT	5000°
CLOUDY	6500°
SHADE	8000°

Table 3: Preset to Color Temperature equivalence

Name:	usWbRedScale		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAwb.stForced.usWbRedScale		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Sets a forced scale on the red color relative to green. This is applied as a digital gain in the ISP algorithm. It does not affect the exposure settings. This scale is taken into account when eMode is set to DxOISP_AWB_MODE_MANUAL.			

Name:	usWbBlueScale		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAwb.stForced.usWbBlueScale		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Sets a forced scale on the blue color relative to green. This is applied as a digital gain kn the ISP algorithm. It does not affect the exposure settings. This scale is taken into account when eMode is set to DxOISP_AWB_MODE_MANUAL..			

3.2.3.2.4 Flash related

A flash, when present, can be entirely driven by the DxO ISP. Automatic flash control is advised in most conditions. A manual mode is offered to set the flash power if estimated by the camera application on its own.

Name:	eMode		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stFlash.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_FLASH_MODE_AUTO		
Sets the flash monitoring mode. Possible values are: • DxOISP_FLASH_MODE_AUTO • DxOISP_FLASH_MODE_MANUAL • DxOISP_FLASH_MODE_OFF • DxOISP_FLASH_MODE_ON If DxOISP_FLASH_MODE_OFF is selected, the flash will be never used. If DxOISP_FLASH_MODE_ON is selected, the flash will be used in capture and video mode. The flash is kept OFF in preview mode except if the DxOISP_FLASH_MODE_MANUAL mode is used.			

Name:	usPower		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stFlash.stForced.usPower		
Encoding:	16u	Unit:	None
Default:	0		
Forces the flash intensity to be sent to the sensor HAL. This value is taken into account when eMode is set to DxOISP_FLASH_MODE_MANUAL. 0 means that flash is off.			

3.2.3.2.5 Mechanical Autofocus related

A mechanical autofocus can be driven by the DxO ISP. Focus position can be selected automatically by the DxO ISP or manually set.

Name:	eMode		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AF_MODE_AUTO		
Sets the mechanical auto-focus mode. Functionality is enabled when set to auto or disabled when set to manual. The oneshot mode is used according to the event register (see description below), in this case the search focus is launched only once, and then the focus is locked to the best position found. The hyperfocal mode is used to set the focus to its minimal position. Possible values are: • DxOISP_AF_MODE_AUTO • DxOISP_AF_MODE_ONESHOT • DxOISP_AF_MODE_MANUAL • DxOISP_AF_MODE_HYPERFOCAL			

Name:	eEvent		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.eEvent		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AF_EVENT_NO_EVENT		
Possible values are: • DxOISP_AF_EVENT_NO_EVENT • DxOISP_AF_EVENT_UPDATE_FOCUS_A • DxOISP_AF_EVENT_UPDATE_FOCUS_B This register is used only when the stAf.eMode is set to DxOISP_AF_MODE_ONESHOT. An update of the focus is launched every time this register is changed from DxOISP_AF_EVENT_NO_EVENT to DxOISP_AF_EVENT_UPDATE_FOCUS_A or DxOISP_AF_EVENT_UPDATE_FOCUS_B and every time this register toggles between DxOISP_AF_EVENT_UPDATE_FOCUS_A and DxOISP_AF_EVENT_UPDATE_FOCUS_B.			

Name:	usPosition		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stForced.usPosition		
Encoding:	16u	Unit:	none
Default:	0		
Sets the position of the mechanical auto-focus. This value is taken into account when eMode is set to DxOISP_AF_MODE_MANUAL. When applicable, this value is sent the sensor hardware abstraction library. The value corresponds to the AF register. Consequently, it depends of the hardware used.			

The following registers make it possible to define up to six rectangles of interest in the image. The five last rectangles are dedicated to the face detection and the first can be used to define a default region of interest on the entire image.

Note: all pixel addresses must be given after flipping/mirroring applied by the sensor.

Name:	stFaces[6].usXAddrStart		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stRoi.stFaces[6].usXAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		
Sets the X-address of upper left corner of the considered rectangle of interest. Value must be even and less than usXAddrEnd.			

Name:	stFaces[6]. usYAddrStart		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stRoi.stFaces[6].usYAddrStart		
Encoding:	16u	Unit:	pixels
Default:	0		
Sets the Y-address of upper left corner of the considered rectangle of interest. Value must be even and less than usYAddrEnd.			

Name:	stFaces[6].usXAddrEnd		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stRoi.stFaces[6] usXAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the X-address of lower right corner of the considered rectangle of interest. Value must be odd, greater than usXAddrStart and less than sensor width.			

Name:	stFaces[6].usYAddrEnd		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stRoi.stFaces[6].usYAddrEnd		
Encoding:	16u	Unit:	pixels
Default:	0xFFFF		
Sets the Y-address of lower right corner of the considered rectangle of interest. Value must be odd, greater than usYAddrStart and less than sensor height.			

Name:	ucWeights[6]		
fullName:	ts_DxOIspSyncCmd.stCameraControl.stAf.stRoi.ucWeights[6]		
Encoding:	8u	Unit:	None
Default:	0		
Set the weights of the regions of interest. Use the value 0 if the region is not used.			

3.2.3.3 Asynchronous Command Section

3.2.3.3.1 System parameters

Name:	uiSystemClock		
fullName:	ts_ DxOIspAsyncCmd.stSystem.uiSystemClock		
Encoding:	16.16u	Unit:	MHz
Default:	50 << 16		
This is the frequency of the DxO ISP. This information is used to set timer generating periodic interrupt when the sensor is not streamintg and to compute the horizontal blanking of the sensor.			

Name:	uiMaxBandwidth		
fullName:	ts_ DxOIspAsyncCmd.stSystem.uiMaxBandwidth		
Encoding:	24.4u	Unit:	MBit/s
Default:	0		
This is the maximum instantaneous bandwidth tolerated onto all the DxO ISP interfaces (input frame buffer + output frame buffer + various outputs). This information allows the DxO ISP firmware to reduce the pixel rate of the sensor to prevent bandwidth saturation (by modifying the horizontal blanking of the sensor). Default value 0 disables the limitation.			

Name:	usMinInterframeTime		
fullName:	ts_ DxOIspAsyncCmd.stSystem.usMinInterframeTime		
Encoding:	8.8u	Unit:	ms
Default:	1 << 8		
This time represents the execution time of the DxOISP_Event() function. This means that DxOISP_Event() must be called at least usMinInterframeTime milliseconds before the start of the next frame.			

3.2.3.3.2 Tuning parameters

Tuning parameters have relevancy or efficiency based on different parameters. For tuning registers, the table below indicates how to interpret the tuning cursor description.

Variability	Key	Values
coarseGain	cg	DxOISP_COARSE_GAIN_LOWLIGHT DxOISP_COARSE_GAIN_STANDARD DxOISP_COARSE_GAIN_HIGHLIGHT
coarseIlluminant	ci	DxOISP_COARSE_ILLUMINANT_HORIZON DxOISP_COARSE_ILLUMINANT_TUNGSTEN DxOISP_COARSE_ILLUMINANT_FLUO DxOISP_COARSE_ILLUMINANT_OUTDOOR DxOISP_COARSE_ILLUMINANT_FLASH
exposureTime	e	DxOISP_EXPOSURE_LOW DxOISP_EXPOSURE_MEDIUM DxOISP_EXPOSURE_HIGH
Fine Illuminant	fi	DxOISP_FINE_ILLUMINANT_HORIZON DxOISP_FINE_ILLUMINANT_TUNGSTEN DxOISP_FINE_ILLUMINANT_TL84 DxOISP_FINE_ILLUMINANT_COOLWHITE DxOISP_FINE_ILLUMINANT_D45 DxOISP_FINE_ILLUMINANT_DAYLIGHT DxOISP_FINE_ILLUMINANT_CLOUDY DxOISP_FINE_ILLUMINANT_SHADE DxOISP_FINE_ILLUMINANT_FLASH
Fine gain	fg	DxOISP_FINE_GAIN_ISORANGE1 DxOISP_FINE_GAIN_ISORANGE2 DxOISP_FINE_GAIN_ISORANGE3 DxOISP_FINE_GAIN_ISORANGE4 DxOISP_FINE_GAIN_ISORANGE5

Table 4: Tuning Cursor ranges description

Name:	stLensCorrection.ucLensShading[cg][ci]		
fullName:	ts_DxOIspAsyncCmd.stLensCorrection.ucLensShading[ ][ ]		
Encoding:	8u	Unit:	boolean
Default:	0x80		
Modulates the amount of amplification at correction time for the R, G and B channels simultaneously to get the same modulation on all channels: • 0: 50% remaining Fall-Off • 0xFF: 0 % remaining Fall-Off			

Name:	stLensCorrection.ucSharpness[ci]		
fullName:	ts_DxOIspAsyncCmd.stLensCorrection.ucSharpness[ ][ ]		
Encoding:	8u	Unit:	none
Default:	0x80		
Sharpening intensity:			

- 0: OFF
- 0x80: medium0xFF high

Name:	stSensorDefectCorrection.ucBrightPixels [e]		
fullName:	ts_DxOIspAsyncCmd.stSensorDefectCorrection.ucBrightPixels[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		

Intensity of bright pixels correction:

- 0: OFF
- 0xFF: maximum pixels are detected as defective pixels

Name:	stSensorDefectCorrection.ucDarkPixels[e]		
fullName:	ts_DxOIspAsyncCmd.stSensorDefectCorrection.ucDarkPixels[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		

Intensity of dark pixels correction:

- 0: OFF
- 0xFF: maximum pixels are detected as defective pixels

Name:	stNoiseAndDetails.ucSegmentation[fg]		
fullName:	ts_DxOIspAsyncCmd.stNoiseAndDetails.ucSegmentation[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		

Modulates the amount of edges detected or discarded:

- 0: maximum of homogeneous areas detected.
- 0xFF: maximum of texture detected.

Name:	stNoiseAndDetails.ucGrain[fg]		
fullName:	ts_DxOIspAsyncCmd.stNoiseAndDetails.ucGrain[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		

Modulates the noise/texture trade-off:

- 0: maximum of denoising
- 0xFF: maximum of texture preserved

Name:	stNoiseAndDetails.ucLuminanceNoise[fg]		
fullName:	ts_DxOIspAsyncCmd. stNoiseAndDetails.ucLuminanceNoise[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		

Intensity of denoising:

- 0: OFF
- 0xFF: maximum denoising

Name:	stNoiseAndDetails.ucChrominanceNoise[fg]		
fullName:	ts_DxOIspAsyncCmd. stNoiseAndDetails.ucChrominanceNoise[ ]		
Encoding:	8u	Unit:	none
Default:	0x80		
Intensity of colored noise reduction: • 0: OFF • 0xFF: maximum denoising			

Name:	stColor.cWbBiasRed[fi]		
fullName:	ts_DxOIspAsyncCmd.stColor.cWbBiasRed[ ]		
Encoding:	8s	Unit:	none
Default:	0		
Red bias on preset: • -128:-10% bias • 0: 0% bias • 127: +10% bias			

Name:	stColor.cWbBiasBlue[fi]		
fullName:	ts_DxOIspAsyncCmd.stColor.cWbBiasBlue[ ]		
Encoding:	8s	Unit:	none
Default:	0		
Blue bias on preset: • -128:-10% bias • 0: 0% bias • 127: +10% bias			

Name:	stColor.cSaturation [cg][ci]		
fullName:	ts_DxOIspAsyncCmd.stColor.cSaturation[ ]		
Encoding:	8s	Unit:	none
Default:	0		
Changes the amount of saturation: • -128: -50% saturation • 0: no additional saturation • 127: 50% saturation			

Name:	stExposure.cBrightness		
fullName:	ts_DxOIspAsyncCmd.stExposure.cBrightness		
Encoding:	8s	Unit:	none
Default:	0		
Brightness adjustment: • -128: average brightness of the image is divided by 2 • 0: average brightness of the image is untouched • 127: average brightness of the image is multiplied by 2			

Name:	stExposure.cContrast		
fullName:	ts_DxOIspAsyncCmd.stExposure.cContrast		
Encoding:	8s	Unit:	none
Default:	0		
Contrast adjustment: • -128: contrast of the image is divided by 2 • 0: contrast of the image is untouched • 127 : contrast of the image is multiplied by 2			

Name:	stExposure.ucBlackLevel[cg][ci]		
fullName:	ts_DxOIspAsyncCmd.stExposure.ucBlackLevel[ ][ ]		
Encoding:	8u	Unit:	none
Default:	0x80		
• 0: OFF • 0xFF: strong			

Name:	stExposure.ucWhiteLevel[cg][ci]		
fullName:	ts_DxOIspAsyncCmd.stExposure.ucWhiteLevel[ ][ ]		
Encoding:	8u	Unit:	none
Default:	0x80		
• 0: OFF • 0xFF: strong			

Name:	stExposure.ucGamma[cg][ci]		
fullName:	ts_DxOIspAsyncCmd.stExposure.ucGamma[ ][ ]		
Encoding:	8u	Unit:	none
Default:	0x80		
• 0: OFF • 0xFF: Strong			

Name:	isDebugEnable		
fullName:	ts_DxOIspAsyncCmd.isDebugEnable		
Encoding:	8u	Unit:	boolean
Default:	0		
Enables the dump of commands received by the DxO ISP firmware. This dump is only available with firmware compiled in debug mode.			

3.2.3.4 Synchronous Status Section – Global status

Status allows knowing frame by frame the global state of the LibDxOISP and the parameters used to process images. Command may not be taken into account instantaneously for some reasons (sensor latency, camera control convergence constraints…). For that status reports the current state of the sytem to the camera application.

Name:	eMode		
fullName:	ts_DxOIspSyncStatus.stSystem.eMode		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_MODE_STANDBY		

Returns the current mode of the ISP. Possible values are:

- DxOISP_MODE_STANDBY
- DxOISP_MODE_IDLE
- DxOISP_MODE_PREVIEW
- DxOISP_MODE_CAPTURE_A
- DxOISP_MODE_CAPTURE_B
- DxOISP_MODE_VIDEOREC

DxO ISP state machine is described in section 2.5.2.

Name:	isTNREnabled		
fullName:	ts_DxOIspSyncStatus.stSystem. isTNREnabled		
Encoding:	8u	Unit:	boolean
Default:	0		

If set to 1 the temporal noise reduction is enabled.

	usFrameRate		
fullName:	ts_DxOIspSyncStatus.stSystem.usFrameRate		
Encoding:	12.4u	Unit:	Hz
Default:	0		

Returns current frame rate. This value is relevant if eMode is any value but DxOISP_MODE_STANDBY or DxOISP_MODE_IDLE

Name:	uiFrameCount		
fullName:	ts_DxOIspSyncStatus.stSystem.uiFrameCount		
Encoding:	32.0u	Unit:	none
Default:	0		

Returns the frame index of the current frame. This index is used across all status registers.
This register is set to 0xFFFFFFF when ts_DxOIspSyncStatus.stSystem.eMode is DxOISP_MODE_IDLE or DxOISP_MODE_STANDBY.
When statistics for the current frame are being updated in the asynchronous section of the registers, the register temporarily holds value 0xFFFFFFFF.

Name:	ucErrorCode		
fullName:	ts_DxOIspSyncStatus.stSystem.ucErrorCode		
Encoding:	8u	Unit:	Bit field
Default:	DxOISP_NO_ERROR		

Returns an error status. Possible values are (see enumerate te_DxOISPError):

- DxOISP_NO_ERROR (value 0)
- DxOISP_PIXEL_STUCK (value 1) means that pixels are stucked in the ISP, and a hardware reset has been applied
  Before this hardware reset the following callback is called by the LibDxOISP:
  void DxOISP_PixelStuckHandler(void);

It allows camera application to have time to dump hardware status register and in particular the following function (provided by the LibDxOISP):

```
void DxOISP_DumpStatus (void);
```

This function print interesting status registers value of the DxO ISP hardware.

- DxOISP_SHORT_FRAME (value 2) means that the DxO ISP received a frame shorter than expected and generated black lines in order to fill missing pixels. It is strongly suggestes to drop such frames

3.2.3.5 Synchronous Status Section – Image and output information

Name:	**usAnalogGain[0] … usAnalogGain[3]**		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.usAnalogGain[4]		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Returns analog gains applied by the active sensor to each channel. Channels 0, 1, 2 and 3 respectively are channels Gr, R, B and Gb. Not relevant when active input is the frame buffer.			

Name:	**usDigitalGain[0] … usDigitalGain[3]**		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.usDigitalGain[4]		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Returns digital gains applied by the active sensor to each channel. Channels 0, 1, 2 and 3 respectively are channels Gr, R, B and Gb. Not relevant when active input is the frame buffer.			

Name:	**usIspDigitalGain**		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.usIspDigitalGain		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Returns the digital gain applied by the ISP algorithm.			

Name:	**uiExposureTime**		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.uiExposureTime		
Encoding:	16.16u	Unit:	ms
Default:	0		
Returns the exposure time for the current frame. Not relevant when active input is the frame buffer.			

Name:	**usAperture**		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.usAperture		
Encoding:	8.8u	Unit:	2x (1 stop)
Default:	0		
Returns the current aperture of the active sensor (expressed in f number). Not relevant when active input is the frame buffer.			

Name:	usCurrentLuminance		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAe.usCurrentLuminance		
Encoding:	16u	Unit:	Greyscale levels
Default:	0		
Returns the exposure level as seen by the AE. This value can be used to find out whether particular commands to the sensor have been applied. When stable, the values should converge towards the same target.			

Name:	eSelectedPreset		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAwb.eSelectedPreset		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AWB_PRESET_HORIZON		
Returns the preset selected for white balance correction when ts_DxOIspSyncCmd.stCameraControl.stAwb.eMode is set to DxOISP_AWB_MODE_AUTO or DxOISP_AWB_MODE_PRESET. When set to DxOISP_AWB_MODE_MANUAL, this register is not relevant.			

Name:	usWbRedScale		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAwb.usWbRedScale		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Returns the scale on red channel applied by the white balance correction. This value is relevant for all values of ts_DxOIspSyncCmd.stCameraControl.stAwb.eMode register.			

Name:	usWbBlueScale		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAwb.usWbBlueScale		
Encoding:	8.8u	Unit:	none
Default:	1.0 << 8		
Returns the scale on blue channel applied by the white balance correction. This value is relevant for all values of ts_DxOIspSyncCmd.stCameraControl.stAwb.eMode register.			

Name:	eConvergence		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAf.eConvergence		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_AF_OFF		

Returns the current status of the auto focus algorithm. In case of fixed focus sensor the value DxOISP_AF_OFF is used. Value DxOISP_AF_FOCUSING indicates the auto focus is still searching the optimal focus position for the current scene. Value DxOISP_AF_FOCUSED is set when this optimal focus is reached.
This register is relevant only if ts_DxOIspSyncCmd.stCameraControl.stAf.eMode setting is set to DxOISP_AF_MODE_AUTO.
Possible values are:

- DxOISP_AF_OFF
- DxOISP_AF_FOCUS_FAILED
- DxOISP_AF_FOCUSING
- DxOISP_AF_FOCUSED

Name:	usPosition		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAf.usPosition		
Encoding:	16u	Unit:	none
Default:	0		
Returns the current mechanical autofocus position. This register is relevant only if ts_DxOIspSyncCmd.stCameraControl.stAf.eMode is set to DxOISP_AF_MODE_AUTO or DxOISP_AF_MODE_MANUAL. The value corresponds to the AF register. It depends on the hardware used.			

Name:	uiSharpness		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAf.uiSharpness		
Encoding:	32u	Unit:	none
Default:	0		
Returns the current level of sharpness as seen by the AF. This value is intended for debug purposes only.			

Name:	isRequiredForCapture		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stFlash.isRequiredForCapture		
Encoding:	8u	Unit:	boolean
Default:	0		
Returns 1 if the auto exposure detects that the flash will be necessary in case of a capture. Returns 0 otherwise. This information is updated in any Auto Exposure mode but DxOISP_AE_MODE_MANUAL			

Name:	usPower		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stFlash.usPower		
Encoding:	16.0u	Unit:	None
Default:	0		
Returns the value sent by the DxOISP firmware to the sensor HAL.			

Name:	eFrequency		
fullName:	ts_DxOIspSyncStatus.stImageInfo.stAfk.eFrequency		
Encoding:	8u	Unit:	Enumerated
Default:	0		
Returns the frequency that has been detected by the anti-flickering algorithm if ts_DxOIspSyncCmd.stCameraControl.stAfk.eMode is set to DxOISP_AFK_MODE_AUTO. Otherwise returns the forced frequency. Possible values are: • DxOISP_AFK_FREQ_0HZ • DxOISP_AFK_FREQ_50HZ • DxOISP_AFK_FREQ_60HZ			

Name:	ucSensorDownSamplingX		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucSensorDownsamplingX		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the down sampling factor applied by the sensor on X axis (including binning and skipping).			

Name:	ucSensorDownSamplingY		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucSensorDownsamplingY		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the down sampling factor applied by the sensor on Y axis (including binning and skipping).			

Application note:
A downsampling can be a skipping, a binning or a combination of both.

Name:	ucSensorBinningX		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucSensorBinningX		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the binning applied by the sensor on X axis.			

Name:	ucSensorBinningY		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucSensorBinningY		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the binning applied by the sensor on Y axis.			

Name:	ucISPDownSamplingX		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucISPDownsamplingX		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the down sampling factor applied by the DxO ISP on X axis (including binning and skipping).			

Name:	ucISPDownSamplingY		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucISPDownsamplingY		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the down sampling factor applied by the DxO ISP on Y axis (including binning and skipping).			

Name:	ucISPBinningX		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucISPBinningX		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the binning applied by the DxO ISP on X axis.			

Name:	ucISPBinningY		
fullName:	ts_DxOIspSyncStatus.stImageInfo.ucISPBinningY		
Encoding:	8.0u	Unit:	None
Default:	1		
Returns the binning applied by the DxO ISP on Y axis.			

Name:	usPedestal		
fullName:	ts_DxOIspSyncStatus.stImageInfo.usPedestal		
Encoding:	16u	Unit:	None
Default:	Capability of the sensor with ID 0		
Minimal value of a pixel (dark level). It may change from one frame to another.			

Name:	usZoomFocal		
fullName:	ts_DxOIspSyncStatus.stImageInfo.usZoomFocal		
Encoding:	8.8u	Unit:	none
Ratio between focal distance and sensor diagonal size.			

Name:	isEnabled		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].isEnabled		
Encoding:	8u	Unit:	boolean
Default:	0		
Return 1 if the output is enabled.			

Name:	eFormat		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].eFormat		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_FORMAT_YUV422		

Returns the current format for the considered output. For all outputIdx values, possible values are:

- DxOISP_FORMAT_YUV422
- DxOISP_FORMAT_YUV422_NV
- DxOISP_FORMAT_YUV420

If outputIdx is DxOISP_OUTPUT_DISPLAY, possible additonal values are:

- DxOISP_FORMAT_RGB565
- DxOISP_FORMAT_RGB888

If outputIdx is DxOISP_OUTPUT_CAPTURE another possible additional value is DxOISP_FORMAT_RAW.

Name:	eOrder		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].eOrder		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_YUV422ORDER_YUYV		

Returns the order for the pixels of the considered output.

If eFormat is DxOISP_FORMAT_YUV422 or DxOISP_FORMAT_YUV422_NV, possible values for are:

- DxOISP_YUV422ORDER_YUYV
- DxOISP_YUV422ORDER_YYUV
- DxOISP_YUV422ORDER_UYYV
- DxOISP_YUV422ORDER_UVYY
- DxOISP_YUV422ORDER_UYVY
- DxOISP_YUV422ORDER_YUVY
- DxOISP_YUV422ORDER_YVYU
- DxOISP_YUV422ORDER_YYVU
- DxOISP_YUV422ORDER_VYYU
- DxOISP_YUV422ORDER_VUYY
- DxOISP_YUV422ORDER_VYUY
- DxOISP_YUV422ORDER_YVUY

If eFormat is DxOISP_FORMAT_YUV420, possible values are:

- DxOISP_YUV420ORDER_YYU
- DxOISP_YUV420ORDER_YUY
- DxOISP_YUV420ORDER_UYY
- DxOISP_YUV420ORDER_YYV
- DxOISP_YUV420ORDER_YVY
- DxOISP_YUV420ORDER_VYY

For YYU, YUY and UYY based ordering value, U samples are output on even lines and V samples are output on odd lines. For YYV, YVY and VYY based orderings, V samples are output on even lines and U samples are output on odd lines.

If eFormat is DxOISP_FORMAT_RGB888, possible values are:

- DxOISP_RGBORDER_RGB
- DxOISP_RGBORDER_RBG
- DxOISP_RGBORDER_GRB
- DxOISP_RGBORDER_GBR
- DxOISP_RGBORDER_BGR
- DxOISP_RGBORDER_BRG

If eFormat is DxOISP_FORMAT_RGB565, possible values are:

- DxOISP_RGBORDER_RGB
- DxOISP_RGBORDER_BGR

Name:	eEncoding		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].eEncoding		
Encoding:	8u	Unit:	enumerated
Default:	DxOISP_ENCODING_YUV_601FS		

Returns the current encoding for the considered output. This register is relevant only if eFormat either is DxOISP_FORMAT_YUV422, DxOISP_FORMAT_YUV422_NV or DxOISP_FORMAT_YUV420.
Possible values are:

- DxOISP_ENCODING_YUV_601FS
- DxOISP_ENCODING_YUV_601FU
- DxOISP_ENCODING_YUV_601CU
- DxOISP_ENCODING_YUV_601CS
- DxOISP_ENCODING_YUV_709FS
- DxOISP_ENCODING_YUV_709FU
- DxOISP_ENCODING_YUV_709CU
- DxOISP_ENCODING_YUV_709CS

F stands for full-range and means that the output will range from 0 to 255 for Y and from-128 to 127 for U and V. This is what most JPEG encoders expect.
C stands for clipped range and means the output will range from 16 to 235 for Y and from-112 to +112 for U and V.
U and S stands for unsigned and signed. The unsigned representations add 128 to U and V.

Hence, the possible ranges for the different values are:

- DxOISP_ENCODING_YUV_601FS: Y in [0, 255], U and V in [-128, 127].
- DxOISP_ENCODING_YUV_601FU: Y, U and V in [0, 255].
- DxOISP_ENCODING_YUV_601CU: Y in [16, 235], U and V in [16, 240].
- DxOISP_ENCODING_YUV_601CS: Y in [16, 235], U and V in [-112, 112].
- DxOISP_ENCODING_YUV_709FS: Y in [0, 255], U and V in [-128, 127].
- DxOISP_ENCODING_YUV_709FU: Y, U and V in [0, 255].
- DxOISP_ENCODING_YUV_709CU: Y in [16, 235], U and V in [16, 240].
- DxOISP_ENCODING_YUV_709CS: Y in [16, 235], U and V in [-112, 112].

See Ref[6] and Ref[7] for more information on these formats.

Name:	usXAddrStart		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].stCrop.usXAddrStart		
Encoding:	16.0u	Unit:	pixels
Default:	0		

Returns the X-address of upper left corner of the considered output crop into the visible pixel array of the active input.

Name:	usYAddrStart		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].stCrop.usYAddrStart		
Encoding:	16.0u	Unit:	pixels
Default:	0		

Returns the Y-address of upper left corner of the considered output crop into the visible pixel array of the active input.

Name:	usXAddrEnd		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].stCrop.usXAddrEnd		
Encoding:	16.0u	Unit:	pixels
Default:	Width of the sensor with ID 0 minus one		
Returns the X-address of lower right corner of the considered output crop into the visible pixel array of the active input.			

Name:	usYAddrEnd		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].stCrop.usYAddrEnd		
Encoding:	16.0u	Unit:	pixels
Default:	Height of the sensor with ID 0 minus one		
Returns the Y-address of lower right corner of the considered output crop into the visible pixel array of the active input.			

Name:	usSizeX		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].usSizeX		
Encoding:	16.0u	Unit:	pixels
Default:	640		
Returns the pixel width of the display output.			

Name:	usSizeY		
fullName:	ts_DxOIspSyncStatus.stOutput[outputIdx].usSizeY		
Encoding:	16.0u	Unit:	pixels
Default:	480		
Returns the pixel height of the display output.			

3.2.3.6 Asynchronous Status Section – Image Signal Processing capabilities

Name:	stMaxSize[].usX		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stHardware.stMaxSize[outputIdx].usX		
Encoding:	16.0u	Unit:	Pixels
Default:	Depends on the hardware configuration.		
Maximum width on the corresponding output.			

Name:	stMaxSize[].usY		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stHardware. stMaxSize[outputIdx].usY		
Encoding:	16.0u	Unit:	Pixels
Default:	Depends on the hardware configuration.		
Maximum height on the corresponding output.			

Typical values for stMaxSize	
Display	1920x1080
Video	2400x1350
Capture	Max sensor size
Face detection	1280x720

Name:	ucRawPixelWidth		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stHardware.ucRawPixelWidth		
Encoding:	8u	Unit:	Bits
Default:	Depends on the hardware configuration.		
Maximum pixel width supported by the DxO ISP.			

Name:	isAfk		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stCameraControl.isAfk		
Encoding:	8u	Unit:	Boolean
Default:	1		
Returns 1 if anti-flickering algorithm is present.			

Name:	isAe		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stCameraControl.isAe		
Encoding:	8u	Unit:	Boolean
Default:	1		
Returns 1 if auto-exposure algorithm is present.			

Name:	isAwb		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stCameraControl.isAwb		
Encoding:	8u	Unit:	Boolean
Default:	1		
Returns 1 if auto white-balance algorithm is present.			

Name:	isFlash		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stCameraControl.isFlash		
Encoding:	8u	Unit:	Boolean
Default:	1		
Returns 1 if flash management is present.			

Name:	isAf		
fullName:	ts_DxOIspAsyncStatus.stIspCapabilities.stCameraControl.isAf		
Encoding:	8u	Unit:	Boolean
Default:	1		
Returns 1 if mechanical auto-focus algorithm is present.			

3.2.3.7 Asynchronous Status Section – Sensor capabalities

Following registers give capabilities of the selected sensor.

Name:	enableInterframeCmdOnly		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.enableInterframeCmdOnly		
Encoding:	8u	Unit:	Boolean
Default:	Capability of the sensor with ID 0		
Reports the capability of the sensor to receive commands during the interframe only (value 1) or at any time during the frame (value 0).			

The ability to receive commands at any time allows the reduction of the interframe time. However, some sensors are instable in such a situation and must not be used that way.

Name:	ucPixelWidth		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucPixelWidth		
Encoding:	8u	Unit:	Bit
Default:	Capability of the sensor with ID 0		
Reports the bitwidth of one pixel output by the sensor.			

Name:	ePhase		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ePhase		
Encoding:	8u	Unit:	Enumerated
Default:	Capability of the sensor with ID 0		
Reports the phase of the sensor. Possible values are: • DXOISP_SENSOR_PHASE_GRBG (Gr, R, B, Gb) • DXOISP _SENSOR_PHASE_RGGB (R, Gr, Gb, B) • DXOISP _SENSOR_PHASE_BGGR (B, Gb, Gr, R) • DXOISP _SENSOR_PHASE_GBRG (Gb, B, R, Gr)			

Name:	isPhaseStatic		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.isPhaseStatic		
Encoding:	8u	Unit:	Boolean
Default:	Capability of the sensor with ID 0		
Return 1 if the sensor bayer phase is still the same whatever the orientation (flip, mirror)			

Name:	eNaturalOrientation		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.eNaturalOrientation		
Encoding:	8u	Unit:	Enumerated
Default:	Capability of the sensor with ID 0		
Reports the value to write to register ts_DxOIspCmd.ts_DxOIspSyncCmd.stControl.eImageOrientation in order to set the orientation of the camera to “natural”, i.e. without any flip or mirror. Possible values are: • DXOISP_FLIP_OFF_MIRROR_OFF • DXOISP_FLIP_OFF_MIRROR_ON • DXOISP_FLIP_ON_MIRROR_OFF • DXOISP_FLIP_ON_MIRROR_ON			

Name:	usBayerWidth		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usBayerWidth		
Encoding:	16u	Unit:	Bayers
Default:	Capability of the sensor with ID 0		
Reports native sensor width.			

Name:	usBayerHeight		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usBayerHeight		
Encoding:	16u	Unit:	Bayers
Default:	Capability of the sensor with ID 0		
Reports native sensor height.			

Name:	usMinHorBlanking		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usMinHorBlanking		
Encoding:	16u	Unit:	Pixels
Default:	Capability of the sensor with ID 0		
Reports minimum horizontal blanking supported by the sensor.			

Name:	usMinVertBlanking		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usMinVertBlanking		
Encoding:	16u	Unit:	Lines
Default:	Capability of the sensor with ID 0		
Reports vertical blanking supported by the sensor.			

Name:	ucMaxHorDownsamplingFactor		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucMaxHorDownsamplingFactor		
Encoding:	8u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports maximum horizontal down sampling that can be provided by the sensor (taking into account skipping and binning capabilities).			

Name:	ucMaxVertDownsamplingFactor		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucMaxVertDownsamplingFactor		
Encoding:	8u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports maximum vertical down sampling that can be provided by the sensor (taking into account skipping and binning capabilities).			

Name:	uiMaxPixelClock		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.uiMaxPixelClock		
Encoding:	16.16u	Unit:	MHz
Default:	Capability of the sensor with ID 0		
Reports maximum pixel clock supported by the sensor.			

Name:	ucMaxDelayImageFeature		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usMaxDelayImageFeature		
Encoding:	8u	Unit:	Frames
Default:	Capability of the sensor with ID 0		
Reports maximum delay to reach a stable crop, downsampling, blanking configuration.			

Name:	ucMaxDelayImageQuality		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.usMaxDelayImageQuality		
Encoding:	8u	Unit:	Frames
Default:	Capability of the sensor with ID 0		
Reports maximum delay to reach a stable image quality. Applies all commands except image feature relative ones			

Name:	ucNbUpperDummyLines		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucNbUpperDummyLines		
Encoding:	8u	Unit:	Pixels
Default:	Capability of the sensor with ID 0		
Reports the number of dummy lines received by the DxO ISP at the beginning of the image. These lines will be cropped in input by the DxO ISP.			

Name:	ucNbLeftDummyColumns		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucNbLeftDummyColumns		
Encoding:	8u	Unit:	Pixels
Default:	Capability of the sensor with ID 0		
Reports the number of dummy columns received by the DxO ISP at the left of the image. These columns will be cropped in input by the DxO ISP.			

Name:	ucNbLowerDummyLines		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucNbLowerDummyLines		
Encoding:	8u	Unit:	Pixels
Default:	Capability of the sensor with ID 0		
Reports the number of dummy lines received by the DxO ISP at the end of the image. These lines will be cropped in input by the DxO ISP.			

Name:	ucNbRightDummyColumns		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stImageFeature.ucNbRightDummyColumns		
Encoding:	8u	Unit:	Pixels
Default:	Capability of the sensor with ID 0		
Reports the number of dummy columns received by the DxO ISP at the right of the image. These columns will be cropped in input by the DxO ISP.			

Name:	usIso		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stShootingParam.usIso		
Encoding:	16u	Unit:	ISO
Default:	Capability of the sensor with ID 0		
Reports the real ISO value of the sensor for an analog gain and a digital gain set to 1.			

Name:	usMaxAGain		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stShootingParam.usMaxAGain		
Encoding:	8.8u	Unit:	none
Default:	Capability of the sensor with ID 0		
Reports maximum analog gain supported by the sensor.			

Name:	usMaxDGain		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stShootingParam.usMaxDGain		
Encoding:	8.8u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports maximum digital gain supported by the sensor.			

Name:	usMaxPower		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stFlash.usMaxPower		
Encoding:	16u	Unit:	Cd
Default:	Capability of the sensor with ID 0		
Reports maximum power supported by the flash measured in lux at 1m on optical axis. Returns 0 if the sensor has no flash.			

Name:	usMin		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAf.usMin		
Encoding:	16u	Unit:	none
Default:	Capability of the sensor with ID 0		
Reports the minimum possible value of the AF register. It depends on the hardware being used.			

Name:	usMax		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAf.usMax		
Encoding:	16u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports the maximum possible value of the AF register. It depends of the hardware being used. If usMax is equal to usMin then there is no auto focus.			

Name:	usMinDefocus		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAf.stPerUnit.usMinDefocus		
Encoding:	16u	Unit:	none
Default:	Capability of the sensor with ID 0		
Reports the per unit minimal defocus position			

Name:	usMidDefocus		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAf.stPerUnit.usMidDefocus		
Encoding:	16u	Unit:	none
Default:	Capability of the sensor with ID 0		
Reports the per unit intermediate position to define the direction of a new search			

Name:	usMaxDefocus		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAf.stPerUnit.usMaxDefocus		
Encoding:	16u	Unit:	none
Default:	Capability of the sensor with ID 0		
Reports the per unit maximal defocus position. If set to 0, the pre calibrated values are used.			

Name:	usMin		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stZoomFocal.usMin		
Encoding:	8.8u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports the minimum zoom focal. This is a ratio between focal distance and sensor diagonal size.			

Name:	usMax		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stZoomFocal.usMax		
Encoding:	8.8u	Unit:	None
Default:	Capability of the sensor with ID 0		
Reports the maximum zoom focal. This is a ratio between focal distance and sensor diagonal size. If usMin is equal to usMax then there is no zoom.			

Name:	usSensorDiagSize		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stZoomFocal.usSensorDiagSize		
Encoding:	8.8u	Unit:	millimeter
Default:	Capability of the sensor with ID 0		
Reports the sensor diagonal size			

Name:	usMin		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAperture.usMin		
Encoding:	8.8u	Unit:	2x (1 stop)
Default:	Capability of the sensor with ID 0		
Reports the minimum aperture as an f number			

Name:	usMax		
fullName:	ts_DxOIspAsyncStatus.stSensorCapabilities.stAperture.usMax		
Encoding:	8.8u	Unit:	2x (1 stop)
Default:	0 Capability of the sensor with ID 0		
Reports the maximum aperture as an f number. If usMin is equal to usMax, then there is no variable aperture.			

3.2.3.8 Use case summary

The following table explains the different use case /mode available with the associated difference.

AE exposure/gain curve: The calibration toolbox allows to set 3 differents curves: preview, video, capture. They are used according to the mode selected except if the auto exposure is set in manual mode.

Use case	Mode	AE exposure/gain curve	Available output	Flash (Auto mode)
Streaming	Preview	Preview	All	OFF
	Video	Video	All	ON or OFF during all video

	CaptureA/B	Capture	All	ON or OFF
Streaming with TNR	Preview	Preview	All	OFF
	Video	Video	All	ON or OFF during all video
RAW acquisition	CaptureA/B	Capture	None (data go in FB)	ON or OFF

Note:

- For DxOISP_MODE_CAPTURE_A and DxOISP_MODE_CAPTURE_B the streaming is available during ucNbCaptureFrame frames. After that the streaming is stopped.
- Transition to mode DxOISP_MODE_CAPTURE_A or DxOISP_MODE_CAPTURE_B is done only when sensor and camera control have converged to a correctly exposed image. It can take several frames in particular when the flash is needed.

3.2.4 DxOISP_Event()

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_Event(void);
```

DESCRIPTION

DxOISP_Event must be called when DxO ISP sends an interruption.

The function DxOISP_Event does the following actions:

- **Read new commands**
- **Set new sensor settings**
- **Get statistics from DxO ISP**
- **Get new correction data**
- **Setup DxO ISP for the next frame**
- Update status registers

All actions up to the setup of DxO ISP (**in bold in the above list**) must be achieved before the end of the sensor vertical blanking. It means, there is a real-time constraint. The update of the status registers can be carried out during the time of the next frame.

In states DxOISP_MODE_STANDBY and DxOISP_MODE_IDLE, the DxOISP_Event() function will only read new commands and launch associated action. There is not real-time constraint.

3.2.5 DxOISP_PostEvent()

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_PostEvent(void);
```

DESCRIPTION

Function DxOISP_PostEvent () must be called after the return of the function DxOISP_Event ().

This function can be executed during the time of the frame. Timing constraint for this function is lower than for DxOISP_Event(). Consequently, it may be executed as a lower priority task.

3.2.6 Internal data flow

Prior to the description of each internal interface (sensor hardware abstraction library, camera control, and calibration data selection), it is important to have an overview of the relationships between these different parts. The following figure shows the different interactions between them.

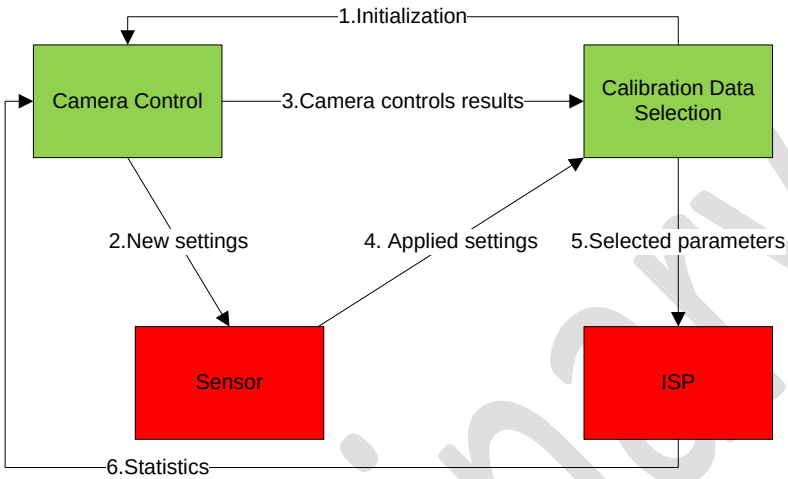


Figure 12: Internal data flow

Except the first step (initialization of the camera control with data coming from the calibration data selection part), all other actions are executed for each frame:

- Camera control results give new parameters for the sensor (such as exposure related parameters) and its actuator if any.
- LibCameraControl (please refer to section 2.5.1) can control image pipe algorithm like the dynamic contrast enhancement, so some results have an impact on the parameters selected by the library calibration data.
- Sensor hardware abstraction library report settings that the sensor will effectively apply for the next frame.
- The LibCalibrationDataSelsection (please refer to section 2.5.1) with information of step 3 and 4 select image pipe data that the DxO IPC will use on the next frame.
- DxO ISP computes statistics on the frame while it applies a RAW to YUV image pipe. These statistics will be used by camera control library for the next frame.

3.2.7 DxOISP_ComputeFrameBufferSize

DxO provides a function helping camera application to compute the memory needed to use frame buffer.

SYNOPSIS

```
#include "DxOISP.h"
void DxOISP_ComputeFrameBufferSize(
    const ts_DxOIspSyncCmd * pstIspSyncCmd
,         uint32_t         * uiBufferSize
);
```

DESCRIPTION

pstIspSyncCmd: a pointer to a structure of type ts_DxOIspSyncCmd that contains all commands the camera application will send to the DxO ISP.

uiBufferSize: return the memory required for the frame buffer in Bytes.

3.2.8 DxOISP_Printf

DxO ISP firmware calls the function DxOISP_Printf to output debug information. This function can be called in case of assertion, or in case of debug (using dbg version of the DxO ISP firmware). The camera application must implement this function to redirect information to the right place.

SYNOPSIS

```
#include "DxOISP.h"
int DxOISP_Printf(const char* format, …);
```

This function uses the same interface than the classic C function: printf
An example of implementation is provided in the file main.c

3.2.9 Sensor hardware abstraction layer

Sensor hardware abstraction layer interface is quite similar to the user interface, with a list of registers accessed with functions DxOISP_SensorCommandSet() and DxOISP_SensorStatusGet(). As for the user interface, a grouping mechanism is used to guarantee coherency of commands and status.

This interface is designed to support more than one sensor. Access is done with the same set of functions by changing the sensor ID.

The following figure explains in which frame timing camera controls are used. The latency between the statistics computation and the camera controls action on both the sensor and the DxO ISP is two frames.

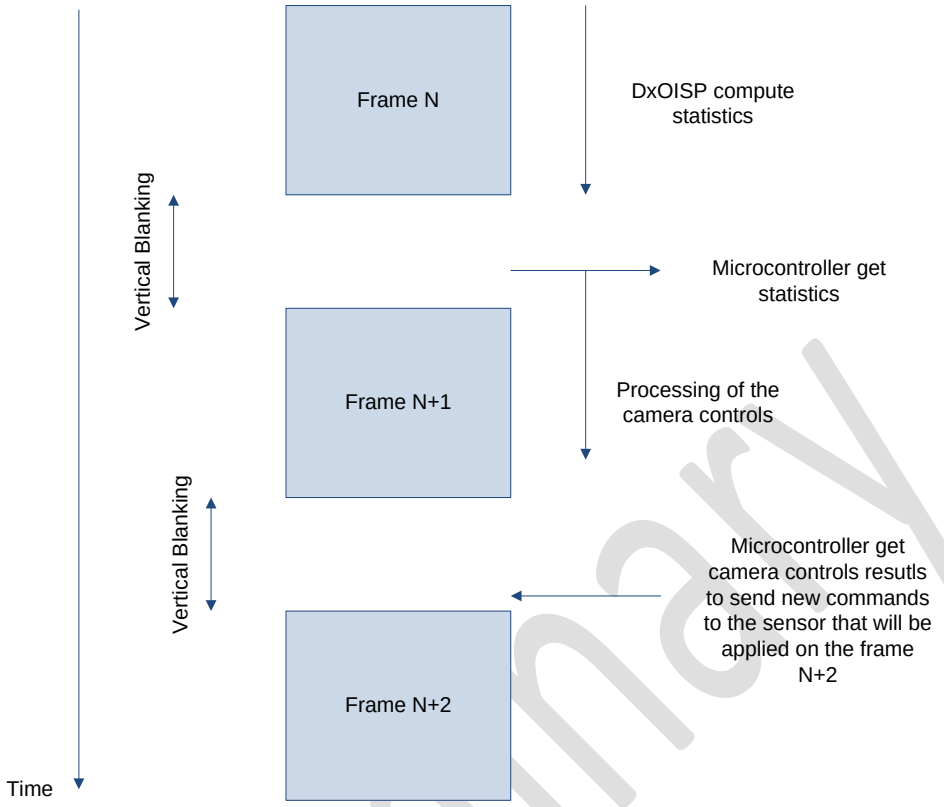


Figure 13: Camera Controls frame timing

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorInitialize(uint8_t ucSensorID);
```

DESCRIPTION

ucSensorID the identifier of the sensor. This is the value contained in the register ucSourceSelection (Please refer to section 3.2.3.1)

The goal of function DxOISP_SensorInitialize() is to initialize the communication interface between the sensor hardware abstraction library and the LibDxOISP.

This function is called by the LibDxOISP in the two following cases:

- when the state machine goes from STANDBY to IDLE state.
- every time the state machine returns to the IDLE state and the sensorID differs from the previous used one.

Consequently, if more than one sensor are connected to the DxO ISP, the switch of sensor (through writing a value into command register ts_DxOIspSyncCmd.stControl. ucSourceSelection) must occur at the same time the idle state is requested.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorUninitialize(uint8_t ucSensorID);
```

DESCRIPTION

ucSensorID the identifier of the sensor.

The goal of function DxOISP_SensorUninitialize() is to reset the communication interface between the sensor hardware abstraction library and the LibDxOISP.

This function is called by the LibDxOISP in the two following cases:

- when the state machine goes to STANDBY state.
- every time the state machine returns to the IDLE state and the sensorID differs from the previous used one.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorCommandSet(
    uint8_t    ucSensorID
,   uint16_t   usOffset
,   uint16_t   usSize
,   void     * pBuf
);
```

DESCRIPTION

ucSensorID	the identifier of the sensor.
usOffset	a 16 bits unsigned value providing the offset in the logic mapping of command registers.
usSize	16 bits value providing the length of the data buffer.
pBuf	a pointer to the data buffer to upload to the command registers.

Function DxOISP_SensorCommandSet() is supposed to write usSize bytes from the data buffer pointed to by pBuf to the consecutive command registers starting at usOffset.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorCommandGroupOpen(uint8_t ucSensorID);
```

DESCRIPTION

ucSensorID the identifier of the sensor.

Function DxOISP_SensorCommandGroupOpen() is used to guarantee atomic application of a group of updates to the sensor registers.
All changes related to calls to DxOISP_SensorCommandSet() made between a call to DxOISP_SensorCommandGroupOpen() and a subsequent call to DxOISP_SensorCommandGroupClose() will be applied simultaneously.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorCommandGroupClose(uint8_t ucSensorID);
```

DESCRIPTION

ucSensorID the identifier of the sensor.

Function DxOISP_SensorCommandGroupClose() is used to guarantee atomic application of a group of updates to the sensor registers.
All changes related to calls to DxOISP_SensorCommandSet() made between a call to DxOISP_SensorCommandGroupOpen() and a subsequent call to DxOISP_SensorCommandGroupClose() will be applied simultaneously.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorStatusGet(
    uint8_t    ucSensorId
,   uint16_t   usOffset
,   uint16_t   usSize
,   void     * pBuf
);
```

DESCRIPTION

ucSensorID	the identifier of the sensor.
usOffset	a 16 bits unsigned value providing the offset in the logic mapping of status registers.
usSize	a 16 bits value providing the length in bytes of the data buffer.
pBuf	a pointer to the data buffer containing the status registers.

Fucntion DxOISP_SensorStatusGet() reads usSize bytes from the consecutive status registers starting at usOffset to the data buffer pointed to by pBuf.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorStatusGroupOpen(uint8_t ucSensorId);
```

DESCRIPTION

ucSensorID	the identifier of the sensor.

Function DxOISP_SensorStatusGroupOpen() is used to guarantee atomic retrieval of a group of sensor status registers.
All calls to DxOISP_SensorStatusGet() made between a call to DxOISP_SensorStatusGroupOpen() and a subsequent call to DxOISP_SensorStatusGroupClose() will refer to a consistent set of data.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorStatusGroupClose(uint8_t ucSensorId);
```

DESCRIPTION

ucSensorID	the identifier of the sensor.

Function DxOISP_SensorStatusGroupClose() is used to guarantee atomic retrieval of a group of sensor status registers.
All calls to DxOISP_SensorStatusGet() made between a call to DxOISP_SensorStatusGroupOpen() and a subsequent call to DxOISP_SensorStatusGroupClose() will refer to a consistent set of data.

SYNOPSIS

```
#include "DxOISP_SensorAPI.h"
void DxOISP_SensorFire (uint8_t ucSensorId);
```

DESCRIPTION

ucSensorID the identifier of the sensor.

This function will be called by the LibDxOISP every time it is required by the sensor hardware abstraction library (through the setting of the register isSensorFireNeeded. Please refer to section 3.2.10.1.2). Some sensors need to be stopped and restarted later to apply particular settings. In such a case the sensor is stopped when sensor hardware abstraction library applies new settings and is restarted when this function is called (typically after the setup of the DxO ISP in function DxOISP_Event).

3.2.10Sensor/Actuator hardware abstraction library register interfaces

3.2.10.1 Register map

3.2.10.1.1 Sensor command

Name:	isSensorStreaming		
fullName:	ts_SENSOR_Cmd.isSensorStreaming		
Encoding:	8u	Unit:	boolean
Value 1 enables the streaming of the sensor. Value 0 stops it.			

Name:	usXBayerstart		
fullName:	ts_SENSOR_Cmd.stImageFeature.usXBayerstart		
Encoding:	16u	Unit:	Bayers
Sets the upper left corner of the crop into the visible pixel array of the active input. The coordinates use the physical space coordinates, before downsampling.			

Name:	usYBayerstart		
fullName:	ts_SENSOR_Cmd.stImageFeature.usYBayerstart		
Encoding:	16u	Unit:	Bayers
Sets the upper left corner of the crop into the visible pixel array of the active input. The coordinates use the physical space coordinates, before downsampling.			

Name:	usXBayerEnd		
fullName:	ts_SENSOR_Cmd.stImageFeature.usXBayerEnd		
Encoding:	16u	Unit:	Bayers
Sets the bottom right corner of the crop into the visible pixel array of the active input. The coordinates use the physical space coordinates, before downsampling.			

Name:	usYBayerEnd		
fullName:	ts_SENSOR_Cmd.stImageFeature.usYBayerEnd		
Encoding:	16u	Unit:	Bayers
Sets the bottom right corner of the crop into the visible pixel array of the active input. The coordinates use the physical space coordinates, before downsampling.			

Name:	eOrientation		
fullName:	ts_SENSOR_Cmd.stImageFeature.eOrientation		
Encoding:	8u	Unit:	enumerated
Sets the orientation of the image. Possible values are: • DxOISP_FLIP_OFF_MIRROR_OFF • DxOISP_FLIP_OFF_MIRROR_ON • DxOISP_FLIP_ON_MIRROR_OFF • DxOISP_FLIP_ON_MIRROR_ON			

Name:	uiMinHorBlanking		
fullName:	ts_SENSOR_Cmd.stImageFeature.uiMinHorBlanking		
Encoding:	32u	Unit:	pixels
Sets the minimal horizontal blanking required by the application.			

Name:	usMinVertBlanking		
fullName:	ts_SENSOR_Cmd.stImageFeature.usMinVertBlanking		
Encoding:	8.8u	Unit:	ms
Sets the minimal vertical blanking required by the application.			

Name:	usMaxFrameRate		
fullName:	ts_SENSOR_Cmd.stImageFeature.usMaxFrameRate		
Encoding:	12.4u	Unit:	Hz
Sets the target frame rate. This frame rate must be respected by the sensor if exposure time and hardware capabilities allow it. Otherwise, achieved frame rate must be lower than targeted one.			

Name:	usHorDownsamplingFactor		
fullName:	ts_SENSOR_Cmd.stImageFeature.usHorDownsamplingFactor		
Encoding:	8u	Unit:	none
Sets the horizontal down sampling factor requested by the application. It may be a binning, a skipping, or a combination of both. If a choice is possible, binning is preferred.			

Name:	usVertDownsamplingFactor		
fullName:	ts_SENSOR_Cmd.stImageFeature.usVertDownsamplingFactor		
Encoding:	8u	Unit:	none
Sets the vertical downsampling factor requested by the application. It may be a binning, a skipping, or a combination of both. If a choice is possible, binning is preferred.			

Name:	uiExposureTime		
fullName:	ts_SENSOR_Cmd.stShootingParam.uiExposureTime		
Encoding:	16.16u	Unit:	ms
Sets the exposure time. This parameter has priority over the target frame rate for the computation of horizontal and vertical blankings.			

Name:	usAGain[4]		
fullName:	ts_SENSOR_Cmd.stShootingParam.usAGain[4]		
Encoding:	8.8u	Unit:	none
Sets the analog gain for Gr, R, B and Gb channels. If the sensor supports only a global analog gain, the gain for Gr channel must be applied globally.			

Name:	usDGain[4]		
fullName:	ts_SENSOR_Cmd.stShootingParam.usDGain[4]		
Encoding:	8.8u	Unit:	none
Sets the digital gain for Gr, R, B and Gb channels. If the sensor supports only a global digital gain, the gain for Gr channel must be applied globally.			

Name:	usFlashPower		
fullName:	ts_SENSOR_Cmd.usFlashPower		
Encoding:	16.0u	Unit:	None
Sets the sensor's flash power driving register. ZOBZOB: TBD			

Name:	usAfPosition		
fullName:	ts_SENSOR_Cmd.usAfPosition		
Encoding:	16u	Unit:	None
Sets the sensor's AF register. Its value depends on the hardware used.			

Name:	usZoomFocal		
fullName:	ts_SENSOR_Cmd.usZoomFocal		
Encoding:	8.8u	Unit:	none
Sets the ratio between focal distance and sensor diagonal size.			

Name:	usAperture		
fullName:	ts_SENSOR_Cmd.usAperture		
Encoding:	8.8u	Unit:	2x (1 stop)
Sets the sensor aperture (in f number).			

Name:	uiFrameCount		
fullName:	ts_SENSOR_Cmd.uiFrameCount		
Encoding:	32u	Unit:	none
Sets the reference frame counter. It is sent by the DxO ISP firmware. As the sensor HAL can be called at any time and since one cannot rely on having such a piece of information from the sensor, it may be useful to know whether the sensor HAL is called for a new frame or during the same frame. It also can be used to maintain a history of commands sent to the sensor.			

3.2.10.1.2 Sensor status

Name:	isSensorFireNeeded		
fullName:	ts_SENSOR_Status.isSensorFireNeeded		
Encoding:	8u	Unit:	boolean
Indicates whether function DxOISP_SensorFire() must be called to launch the streaming of the sensor. If needed, value is 1. Otherwise, it is 0 (zero).			

Name:	stAppliedCmd		
fullName:	ts_SENSOR_Status.stAppliedCmd		
Encoding:	n.a.	Unit:	n.a.
Reports settings really applied by the sensor for the next frame. Because sensors usually have some hardware latency and internal constraints, the applied commands may differ from the required ones. Providing the set of really applied commands allows the ISP to take these constraints into account. Additional fields allow the distinction between deviations due to latency (that may improve over the following frames) and those due to the sensor intrinsic limitations (that will not improve over the following frames).			

Name:	eCurrentPhase		
fullName:	ts_SENSOR_Status.eCurrentPhase		
Encoding:	8u	Unit:	enumerated
Reports the phase of the image output of the sensor. It may depend on the orientation. Possible values are: • DXOISP_SENSOR_PHASE_GRBG (Gr, R, B, Gb) • DXOISP_SENSOR_PHASE_RGGB (R, Gr, Gb, B) • DXOISP_SENSOR_PHASE_BGGR (B, Gb, Gr, R) • DXOISP_SENSOR_PHASE_GBRG (Gb, B, R, Gr)			

Name:	ucHorSkippingFactor		
fullName:	ts_SENSOR_Status.ucHorSkippingFactor		
Encoding:	8u	Unit:	none
Reports the horizontal skipping applied by the sensor. It must be greater than or equal to 1. Value 1 means no skipping is applied. Value 2 means every second pixel along the X axis is transmitted and so on. This register does not take into account any horizontal binning.			

Name:	ucVertSkippingFactor		
fullName:	ts_SENSOR_Status.ucVertSkippingFactor		
Encoding:	8u	Unit:	None
Reports the vertical skipping applied by the sensor. It must be greater than or equal to 1. Vaue 1 means no skipping is applied. Value 2 means every second pixel along the Y axis is transmitted and so on. This register does not take into account any vertical binning.			

Name:	isFrameInvalid		
fullName:	ts_SENSOR_Status.isFrameInvalid		
Encoding:	8u	Unit:	none
Value 1 reports that current frame should be dropped for any reason (geometry not stabilized, exposure not valid, …). This typically occurs when sliding state machine from one state to another that changes drastically the sensor settings.			

Name:	isImageQualityStable		
fullName:	ts_SENSOR_Status.isImageQualityStable		
Encoding:	8u	Unit:	boolean

Value 1 reports that registers other than geometry related ones (crop, down sampling, mirror/flip and blanking) will not change any further. This addresses among others, registers related to gain or exposure time.

Example:
LibDxOISP changes the analog gain of the sensor at frame N. We assume a sensor latency of 2 frames for that action. The next lines illustrate on a frame by frame basis the evolution of the requested and read analog gain:
Frame N: Command: 256, Status: 256, isImageQualityStable = 1
Frame N+1: Command: 384, Status: 256, isImageQualityStable = 0
Frame N+2: Command: 384, Status: 256, isImageQualityStable = 0
Frame N+3: Command: 384, Status: 384, isImageQualityStable = 1

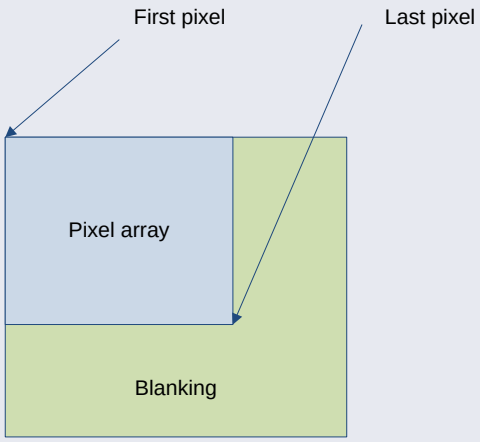
Ty pically this register is checked during the transition to the mode DxOISP_MODE_CAPTURE_A or DxOISP_MODE_CAPTURE_B to ensure to take an image correctly exposed.

Name:	uiPixelClock		
fullName:	ts_SENSOR_Status.uiPixelClock		
Encoding:	16.16	Unit:	MHz

Pixel clock on the DxO ISP Sensor input interface.

Name:	uiFrameValidTime		
fullName:	ts_SENSOR_Status.uiFrameValidTime		
Encoding:	16.16	Unit:	ms

Reports the time from the first to the last pixel out of the sensor. The following diagram illustrates simply that:



Name:	usPedestal		
fullName:	ts_SENSOR_Status.usPedestal		
Encoding:	16u	Unit:	None
Reports the minimal value of a pixel (dark level). It may vary according to the sensor configuration.			

Name:	enableInterframeCmdOnly		
fullName:	ts_SENSOR_Status.stSensorCapabilities.enableInterframeCmdOnly		
Encoding:	8u	Unit:	Boolean
Reports the capability of the sensor to receive commands during the interframe only (value 1) or at any time during the frame (value 0). The ability to receive commands at any time allows the reduction of the interframe time. However, some sensors are instable in such a situation and must not be used that way.			

Name:	ucPixelWidth		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ucPixelWidth		
Encoding:	8u	Unit:	Bit
Reports the bitwidth of one pixel output by the sensor.			

Name:	ePhase		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ePhase		
Encoding:	8u	Unit:	enumerated
Reports the native phase of a the sensor. Possible values are: • DXOISP_SENSOR_PHASE_GRBG (Gr, R, B, Gb) • DXOISP_SENSOR_PHASE_RGGB (R, Gr, Gb, B) • DXOISP_SENSOR_PHASE_BGGR (B, Gb, Gr, R) • DXOISP_SENSOR_PHASE_GBRG (Gb, B, R, Gr)			

Name:	isPhaseStatic		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.isPhaseStatic		
Encoding:	8u	Unit:	Boolean
Reports whether the phase of the Bayer array depends of the sensor configuration or not, in particular the flip and mirror parameters.			

Name:	eNaturalOrientation		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.eNaturalOrientation		
Encoding:	8u	Unit:	Enumerated
Reports the value to write to register ts_DxOIspCmd.ts_DxOIspSyncCmd.stControl.eImageOrientation in order to set the orientation of the camera to “natural”, i.e. without any flip or mirror. Possible values are: • DXOISP_FLIP_OFF_MIRROR_OFF • DXOISP_FLIP_OFF_MIRROR_ON • DXOISP_FLIP_ON_MIRROR_OFF • DXOISP_FLIP_ON_MIRROR_ON			

Name:	usBayerWidth		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usBayerWidth		
Encoding:	16u	Unit:	bayers
Reports native sensor width.			

Name:	usBayerHeight		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usBayerHeight		
Encoding:	16u	Unit:	bayers
Reports native sensor height.			

Name:	usMinHorBlanking		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMinHorBlanking		
Encoding:	16u	Unit:	pixels
Reports minimum horizontal blanking supported by the sensor.			

Name:	usMinVertBlanking		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMinVertBlanking		
Encoding:	16u	Unit:	lines
Reports minimum vertical blanking supported by the sensor.			

Name:	usMaxHorDownsamplingFactor		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMaxHorDownsamplingFactor		
Encoding:	16u	Unit:	none
Reports maximum horizontal down sampling that can be provided by the sensor (taking into account skipping and binning capabilities).			

Name:	usMaxVertDownsamplingFactor		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMaxVertDownsamplingFactor		
Encoding:	16u	Unit:	none
Reports maximum vertical down sampling that can be provided by the sensor (taking into account skipping and binning capabilities).			

Name:	uiMaxPixelClock		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.uiMaxPixelClock		
Encoding:	16.16u	Unit:	MHz
Reports maximum pixel clock supported by the sensor.			

Name:	ucMaxDelayImageFeature		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMaxDelayImageFeature		
Encoding:	8u	Unit:	frames
Reports a maximum delay to reach a stable crop, downsampling, blanking configuration.			

Name:	ucMaxDelayImageQuality		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.usMaxDelayImageQuality		
Encoding:	8u	Unit:	frames
Reports maximum delay to reach a stable image quality. Applies all commands except image feature relative ones.			

Name:	ucNbUpperDummyLines		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ucNbUpperDummyLines		
Encoding:	8u	Unit:	Lines
Reports the number of dummy lines sent by the sensor to the DxO ISP at the beginning of the image.			

Name:	ucNbLeftDummyColumns		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ucNbLeftDummyColumns		
Encoding:	8u	Unit:	Pixels
Reports the number of dummy columns sent by the sensor to the DxO ISP at the left of the image.			

Name:	ucNbLowerDummyLines		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ucNbLowerDummyLines		
Encoding:	8u	Unit:	Lines
Reports the number of dummy lines sent by the sensor to the DxO ISP at the end of the image.			

Name:	ucNbRightDummyColumns		
fullName:	ts_SENSOR_Status.stCapabilities.stImageFeature.ucNbRightDummyColumns		
Encoding:	8u	Unit:	Pixels
Reports the number of dummy columns sent by the sensor to the DxO ISP at the right of the image.			

Name:	usIso		
fullName:	ts_SENSOR_Status.stCapabilities.stShootingParam.usIso		
Encoding:	16u	Unit:	ISO
Reports the real ISO value of the sensor for an analog gain and a digital gain set to 1.			

Name:	usMaxAGain		
fullName:	ts_SENSOR_Status.stCapabilities.stShootingParam.usMaxAGain		
Encoding:	8.8u	Unit:	none
Reports maximum analog gain supported by the sensor.			

Name:	usMaxDGain		
fullName:	ts_SENSOR_Status.stCapabilities.stShootingParam.usMaxDGain		
Encoding:	8.8u	Unit:	none
Reports maximum digital gain supported by the sensor.			

Name:	usMaxPower		
fullName:	ts_SENSOR_Status.stCapabilities.stFlash.usMaxPower		
Encoding:	16u	Unit:	
Reports maximum power supported by the flash. Returns 0 if the sensor has no flash.			

Name:	usMin		
fullName:	ts_SENSOR_Status.stCapabilities.stAf.usMin		
Encoding:	16u	Unit:	none
Reports the minimum possible value of the AF register. It depends on the hardware being used.			

Name:	usMax		
fullName:	ts_SENSOR_Status.stCapabilities.stAf.usMax		
Encoding:	16u	Unit:	none
Reports the maximum possible value of the AF register. It depends of the hardware being used.			

Name:	usMinDefocus		
fullName:	ts_SENSOR_Status.stCapabilities.stAf.stPerUnit.usMinDefocus		
Encoding:	16u	Unit:	none
Reports the per unit minimal defocus position.			

Name:	usMidDefocus		
fullName:	ts_SENSOR_Status.stCapabilities.stAf.stPerUnit.usMidDefocus		
Encoding:	16u	Unit:	none
Reports the per unit intermediate position to define the direction of a new search			

Name:	usMaxDefocus		
fullName:	ts_SENSOR_Status.stCapabilities.stAf.stPerUnit.usMaxDefocus		
Encoding:	16u	Unit:	none
Reports the per unit maximal defocus position.			

Name:	usMin		
fullName:	ts_SENSOR_Status.stCapabilities.stZoomFocal.usMin		
Encoding:	8.8u	Unit:	none
Reports the minimum zoom focal. This is a ratio between focal distance and sensor diagonal size			

Name:	usMax		
fullName:	ts_SENSOR_Status.stCapabilities.stZoomFocal.usMax		
Encoding:	8.8u	Unit:	none
Reports the maximum zoom focal. This is a ratio between focal distance and sensor diagonal size. If usMin is equal to usMax then there is no zoom.			

Name:	usSensorDiagSize		
fullName:	ts_SENSOR_Status.stCapabilities.stZoomFocal.usSensorDiagSize		
Encoding:	8.8u	Unit:	millimeter
Reports the sensor diagonal size			

Name:	usMin		
fullName:	ts_SENSOR_Status.stCapabilities.stAperture.usMin		
Encoding:	8.8u	Unit:	2x (1 stop)
Reports the minimum aperture as an f number			

Name:	usMax		
fullName:	ts_SENSOR_Status.stCapabilities.stAperture.usMax		
Encoding:	8.8u	Unit:	2x (1 stop)
Reports the maximum aperture as an f number. If usMin is equal to usMax, then there is no variable aperture.			

3.2.10.2 Functionalities

As DxO ISP supports only one sensor at a time, the same data area can be used for the register map. Function DxOISP_SensorInitialize() will be used to initialize this data area.

In the case of a file to file, a DMA can be plugged in input of the DxO ISP and can be seen as a sensor. Consequently, a “sensor” hardware abstraction library that actually drives the DMA must be developed in order to handle the “file to file” use case.

4 DELIVERABLES

4.1 Deliverables summary

The DxO ISP firmware set of deliverables is composed of:

- Documentation
  - The present DxO ISP Firmware integration guide describing the DxO ISP principles and how to integrate it in the chip vendor complete system.
  - The FVTS user guide describing the verification bench, the chip vendor has to settle up and how to run the test vectors.
  - The FVTS vectors description describing vectors ans respective goals of each
  - The sensor HAL developer guide.
- Software product entities
  - The DxO ISP firmware (in ELF format) objects compiled for the chip vendors platforms. For each platform are provided:
    - A debug (dbg) release:
      - includes assertions and specific debug print
    - An integration (opt) release:
      - includes assertion with compilation optimization
    - A production (pro) release:
      - no assertion checked
  - The C source file instantiating calibration data.
  - The compiled calibration data object.
  - The interface C declaration header file:
    - DxOISP.h
    - DxOISP_SensorAPI.h
    - DxOISP_ahblAccess.h

Example code

  - A fake camera application code interfacing the LibDxOISP to perform a preview.
  - A full sensor HAL C code for the Omnivision OV8820 sensor.
  - Prototypes and example for ahbl accesses: files DxOISP_ahblAccess.h and DxOISP_ahblAccess.c.
- Verification Suite
  - A set of test vectors to run at chip vendor's side.
  - A set of common tools for verification bench adaptation.

Each software delivery will contain these items in order to ensure at chip vendor's side coherency and integrity of the items used. The delivery will be composed of two items:

- File DxOISP_SwDelivery.tgz: the delivery itself
- File DxOISP_SwDelivery.md5: the associated checksum for integrity check.

4.2 Firmware object

The libDxOISP objects are split into 2 files:

- DxOISP_Firmware.o

- calibData.o

DxOISP_Firmware.o object file contains the entire ISP firmware code (and data) except the calibration of the embedded algorithms (camera control module and image processing).

calibData.o object file contains all calibration data for camera controls and image processing for all supported sensor. This file is provided in the delivery and must be used in association with all test vectors provided.

calibData.o can be rebuilt from the following source files:

- ccParamsTabSelection.c
- calibDataTabSelection.c
- ccParams_<calib_name>.c
- data_<calib_name>.c

ccParamsTabSelection.c and calibDataTabSelection.c (example files are provided in the LibDxOISP delivery) contain table of pointers to functions of calibration selection. ccParamsTabSelection.c corresponds the camera control calibration. calibDataTabSelection.c corresponds to the calibration of image pipe

ccParams.c and data.c contain each one a function of calibration respectively for the camera control and for the image processing. These files are usually generated by the chip vendor using the ISP calibration toolbox.

4.3 ISP Calibration toolbox usage

The ISP Calibration toolbox generates a data.c file that contains calibration data. It corresponds to the image pipe used by the DxO ISP.

For each sensor, such data.c file must be generated.
In addition, the calibration toolbox generates a file named ctlParams.c that contains the calibration of the camera controls. As for the previous file, one such file must be used per sensor.

The DxO ISP supports multiple calibration data for multiple sensors.

4.3.1 Compilation of generated source file

For the compilation, the following header files are required:

- DxOISP.h
- syms.h
- glb.h

To compile generated C file, one must pass the following options on the command line:

```
-DDXO_PRINTF=DxOISP_Printf
-DDATA_GLOBAL_HEADER=firmware/include/DxOISP.h
-DDATA_NAMESPACE=<calibration_name>
-DPROC_NAMESPACE=generic
-DDATA_ROOT_DIR=firmware/correctionData/include
-DDATA_SYMS_H_DIR=firmware/correctionData/include
```

Note: the DATA_NAMESPACE allows to differenciate names of function called to obtain calibration data.

Each object file defines one function that selects correction data. These functions are called through a table of function pointers defined in file corrDataTabSelection. That file has to be modified according to the generated calibration compiled and linked with the others object.

4.3.2 Compilation of the camera control file

To compile the ctlParams.c files, one must compile the ccParams.c file.

This file includes the ctlParams.c file generated for one calibration. The customer must compile one ccParams.c file per generated ctlParams.c file. The name of the function defined in ccParams.c must be modified for each calibration data.

Functions defined in ccParams*.o files are called through a table of pointers to functions defined in file ccParamsTabSelection.c.
This file has to be modified according to the generated calibration.

4.3.3 Link

Once all previous files are compiled, you must link these objects together to get file calibData.o.

This is a partial link, for that use this kind of command line:

```
ld -r
    data_OV8825.o
    ccParams_OV8825.o
    …
    ccParamsTabSelection.o
    calibDataTabSelection.o
    -o calibData.o
```

Finally calibData.o is linked with DxOISP_Firmware.o to build the complete DxOISP firmware.
A Makefile is provided to illustrate this production workflow. The custumer can execute it directly in the firmware delivery as an example.

5 INTEGRATION AND VERIFICATION

5.1 Checking the integrity of the delivered version

For integrity checking of the delivered part, every single package (generally a zipped file) is provided with an md5 checksum.

5.2 Checking the version and configuration

The LibDxOISP provides a function DxOISP_getVersion returning information related to the product and related to a calibration.

SYNOPSIS

```
typedef struct {
      uint32_t       uiHwRevisionId;
      const char*    pcFirmware;
      const char*    pcCalib;
} ts_DxOISPReleaseInfo;

void DxOISP_GetVersion(
      uint8_t               ucCalibrationSelection
,     ts_DxOISPReleaseInfo* pstReleaseInfo
);
```

DESCRIPTION

`ucCalibrationSelection`: The identifier of the calibration used.

`pstReleaseInfo`: A pointer to a structure containing 2 strings of character, one giving firmware information (pcFirmware), one giving calibration information (pcCalib) and hardware identifier (uiHwRevisionId) allowing to distinguish the different hardware deliveries.

Firmware information:

The string is formatted as:
"DxOISP_FW_<toolchain>+<svn_revision>+<date_of_build>+<majorId>+<minorId>+<calib_rev>"

Example:
`"DxOISP_FW_arm -Linux-gnueabi+rev_72153+date_20131015+major_2+minor_1+calibRev_1.0.11"`

<toolchain>	identify the toolchain used to build DxOISP_Firmware.o
<svn_revision>	revision subversion at build time
<date_of_build>	date of build
<majorId>	identitfy the API version
<miniorId>	increment for each evolution (new fonctionnality, bug fix, …)
<calib_rev>	identify the compatiliby with the calibration toolbox

Calibration information:

The string is formatted as:
"<revision>+<date>+<sensor_name>"

Example:
"rev_1.0.11+DATE_20131015+SENSOR_OV8825"

<revision>	identify the revision of the calibration toolbox
<date>	date of generation of the calibration
<sensor_name>	name of the sensor which must be used with this calibration

5.3 Setting communication between LibDxOISP and DxO ISP IP

In order to ease integration and allow flexibility, DxO Labs offers to the chip vendor to define its own AHBL access functions.
The functions called by LibDxOISP to access hardware registers of DxOSIP through AHBL bus are declared below (prototypes are declared in file DxOISP_ahblAccess.h)

```
void DxOISP_setRegister(const uint32_t offset, uint32_t v);

uint32_t DxOISP_getRegister(const uint32_t offset);

void DxOISP_getMultiRegister(const uint32_t offset, uint32_t* p, uint32_t
nElem
);

void  DxOISP_setCopyRegister(const uint32_t offset, const uint32_t* p,
uint32_t nElem);
```

Figure 14: AHBL access function prototypes

An implementation example is provided in file DxOISP_ahblAcces.c.

Caveat: the following considerations must be taken into account when defining these functions:

1- AHBL access are word oriented (32 bit access).
2- AHBL address are byte oriented.
3- The function DxOISP_getMultiRegister gives a read access to memory with its own access incrementer embedded, so Offset value must not be incremented
4- The function DxOISP_setCopyRegister gives a write access to memory withtout incrementer embedded so the Offset value must be incremented

5.4 Linking with its own sensor HAL

The chip vendor must link the LibDxOISP with the compiled sensor HAL. The expected interface between LibDxOISP and the sensor object is defined into the file DxOISP_SensorAPI.h (please refer to to DxOISP-FW-delivery.tar.bz2 package).

In addition to the product items themselves, DxO Labs provides an example of sensor hardware abstraction layer. This example is located in file DxOISP_SensorAPI.c. This POSIX C file is an example of code that offers a unique interface not depending of the number of sensor currently connected to DxO ISP.

Each function is implemented as the canvas here after:

```
void DxOISP_SensorInitialize(uint8_t ucSensorId)
{
    S_TabFunc_Initialize[ucSensorId]();
}

static void (*S_TabFunc_Initialize [])(void) = {
#ifdef SENSOR_ONE_PLUGGED
    DxOSensor_SensorOne_Initialize,
#endif
#ifdef SENSOR_TWO_PLUGGED
    DxOSensor_SensorTwo_Initialize,
#endif
}
```

Figure 15: sensor HAL function canvas

Figure 15: sensor HAL function canvas illustrate an example for function DxOISP_Initialize(). The same canvas is applied for all other functions of the sensor interface.
That function merely calls the appropriate initialization function by pointer access.

Caveat: to use that canvas, the index of the function must match the sensor Id, used in fact as an access index in the function pointers array.

Exemple:
In the case of an OV8820 sensor connected as sensor 0 and again an OV8820 sensor connected as sensor 1, the chip vendor must define the array as following:

```
static void (*S_TabFunc_Initialize [])(void) = {
    DxOSensor_OV8820_Initialize,
    DxOSensor_OV8820_Initialize,
}
```

Figure 16: sensor initialization in case of two sensos connected

In addition to file DxOISP_SensorAPI.c, DxO Labs provides the sensor HAL example for sensor OV8820, containing the definition of DxOSensor_ OV8820_Initialize().

5.5 Functional integration verification

5.5.1 FVTS Overview

The FVTS (Functional Verification Test Suite) is a workflow delivered by DxO Labs to its customers that performs pre- and post-silicon verification. It includes test vectors intended to be replayed at the chip vendor's side and checked against provided references.

A complete description of the FVTS and the required development lying with chip vendor are detailed in Ref[5] document.

Dxo Labs FVTS provides means:

- to run file to file system level verification vectors on different environments and to check the results against the references for verification and non-regression purposes
- to verify that logic added outside the DxO ISP does not affect the correct working of the DxO ISP

Predefined verification vectors are delivered by DxO Labs:

- A verification vector can be any sequence of images and LibDxOISP accesses that have an impact on the DxO ISP functionality and processing. The verification vectors are provided with reference images corresponding to the expected outputs. The reference images are produced by DxO Labs using a bit accurate C model of the DxO ISP (and verified against RTL simulations and FPGA runs). These vectors are used as acceptance criteria for DxO Labs deliverables for file to file FPGA bench.
- These vectors must be passed by the chip vendor on file to file chip level FPGA or accelerated simulation verification bench prior to tape out and checked against references to verify that logic added outside the DxO ISP does not affect the correct working of the DxO ISP.
- The verification suite covers a sub-set of the entire range of the user settings. Each test corresponds to single or multiple sequences of stimuli in order to ensure a correct

behavior of the product. Through these scenarios, the complete set of requirements and functional features are validated.

5.5.2 Bench requirements

To perform the functional verification test suite, the chip vendor must develop a bench fulfilling a number ofrequirements. The bench must include a host providing:

- A mean to compile and execute a C programming language vector
- a mean to monitor the test execution onto the DUT.
- a mean to send images to the DxO ISP.
- a mean to store images after processing completion from the DxO ISP.
- a mean to fulfill DxO ISP memory requirement dealing with frame buffer.

Many functions with prototypes defined by DxO must be developed by customer to full fil these means. The detail behavior of every single function is defined in Ref[5] document.

5.5.3 FVTS test passing

A FVTS test vector passes when the main C test vector exits without any errors and when all collected images during the execution have subsequently been asserted against provided references by DxO.