# DIS Tuning Guide


**Issue** 03

**Date** 2019-06-25

## HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: http://www.hisilicon.com/en/

Email: support@hisilicon.com

# About This Document

## Related Versions

The following table lists the product versions related to this document.

| Product Name | Version |
|---|---|
| Hi3559A | V100ES |
| Hi3559A | V100 |
| Hi3559C | V100 |
| Hi3519A | V100 |
| Hi3556A | V100 |
| Hi3516C | V500 |
| Hi3516D | V300 |
| Hi3516A | V300 |
| Hi3559 | V200 |

**NOTE**

Unless otherwise stated, Hi3559C V100 and Hi3559A V100 contents are consistent.

Unless otherwise stated, Hi3556A V100 and Hi3519A V100 contents are consistent.

Unless otherwise stated, Hi3516D V300, Hi3516A V300, Hi3559 V200, and Hi3516C V500 contents are consistent.

## Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

# Symbol Conventions

The symbols that may be found in this document are defined as follows.

| Symbol | Description |
|---|---|
| DANGER | Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. |
| WARNING | Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. |
| CAUTION | Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury. |
| NOTICE | Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results.<br>NOTICE is used to address practices not related to personal injury. |
| NOTE | Calls attention to important information, best practices and tips.<br>NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration. |

# Change History

Changes between document issues are cumulative. The latest document issue contains all changes made in previous issues.

## Issue 03 (2019-06-25)

This issue is the third official release, which incorporates the following changes:

In section 2.2, the description of **u32Timelag** is modified, and **u32strength** is added.

In section 2.3, the description of Gyro usage is updated.

Section 2.4 is added.

## Issue 02 (2019-03-12)

This issue is the second official release, which incorporates the following changes:

The content related to the Hi3516A V300 is added.

In section 2.2, the description of **u32GyroOutputRange** is modified.

## Issue 01 (2018-12-21)

This issue is the first official release, which incorporates the following changes:

In section 2.2, the description of **u32GyroOutputRange** is modified.

## Issue 00B08 (2018-11-13)

This issue is the eighth draft release, which incorporates the following changes:

In section 2.2, the description of **u32RollingShutterCoef** is modified.

## Issue 00B07 (2018-09-06)

This issue is the seventh draft release, which incorporates the following changes:

The description of Hi3516C V500/Hi3516D V300 is added.

## Issue 00B06 (2018-07-30)

This issue is the sixth draft release, which incorporates the following changes:

Section 2.3.1 is modified.

## Issue 00B05 (2018-01-30)

This issue is the fifth draft release, which incorporates the following changes:

Section 1.3.1 is modified.

## Issue 00B04 (2017-11-15)

This issue is the fourth draft release, which incorporates the following changes:

The description of the Hi3559A V100 is added.

## Issue 00B03 (2017-09-20)

This issue is the third draft release, which incorporates the following changes:

In section 1.2, the descriptions of the GME algorithm are modified.

In section 1.3, figure 1-2 is modified.

In section 2.1, a note is added to u32CropRatio.

## Issue 00B02 (2017-06-30)

This issue is the second draft release, which incorporates the following changes:

In section 2.2, the content related to **u32HorizontalLimit and u32VerticalLimit** is modified.

Section 2.3.5 is modified.

## Issue 00B01 (2017-05-20)

This issue is the first draft release.

# Contents



# Figures



# Tables

# 1 Introduction

Video jitter or instability occurs due to the environment or human factors during video capturing by using a camera. For example, the camera in monitoring scenarios may be affected by vibrations caused by the wind or passed cars, the handheld DV may be affected by human movement, and the event data recorder may be affected by car vibrations. Video jitters need to be eliminated so that the video is more stable and comfortable for watching.

## 1.1 Concept

Digital image stabilization (DIS) is a digital processing process on images.

The motion offset of the current image is calculated by using the DIS algorithm, and then the current image is translated or rotated based on the calculated motion offset to obtain the DIS effect.

## 1.2 Basic Principles of DIS

The DIS is the two-dimensional affine transformation process on images based on the motion offset. Affine transformation includes translation, rotation, scaling, and shearing (parallelogram transformation) of images. The transformation can be indicated by using a 3 x 3 matrix.

$$\begin{bmatrix} \mu \\ \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

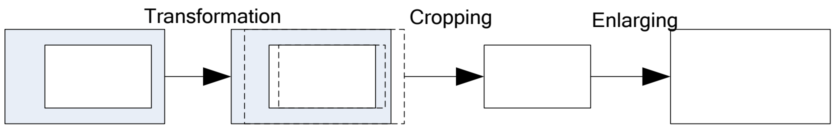
$$x' = \frac{\mu}{\omega}; \quad y' = \frac{\upsilon}{\omega}$$

The 3 x 3 matrix is the motion offset that needs to be calculated in the DIS algorithm. (x, y) indicate the coordinates of the original image, and (x', y') indicate the coordinates after transformation.

When transformation is performed on an image, the positions of image pixels are changed. The image edges may exceed the width and height of the original image, and pixels may be translated out of the entire image. Therefore, the image must be cropped and enlarged during

DIS operations. Image edges are cropped based on certain cropping ratio after the transformation operation, and then the cropped image is enlarged based on the original aspect ratio. See Figure 1-1.

**Figure 1-1** DIS schematic diagram




There are three algorithms for calculating the DIS motion offset:

- GME algorithm

  The global motion estimation (GME) algorithm calculates the motion offset of the current frame image to the reference frame image by extracting the image feature points. An image is more stable after being processed by the GME algorithm. The image stabilization effect is good, but smearing appears if large objects move. The reason is that the GME algorithm cannot well differentiate object movement and camera movement, which may result in misjudgment. In low illumination, the image stabilization effect by the GME algorithm may be compromised due to few image feature points.
- Gyroscope (Gyro) algorithm

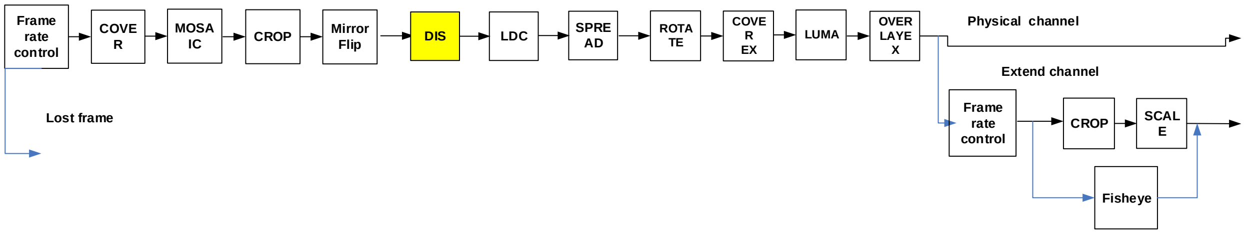
  The Gyro algorithm calculates the motion offset of the current frame by using data generated by the gyroscope, which eliminates the misjudgment issue and ensures the image stabilization effect under low illumination.
- Hybrid algorithm (GME+Gyro)

  The image stabilization effect of the Gyro algorithm is weak due to limitations of the Gyro algorithm itself. In this case, the hybrid algorithm is developed. The hybrid algorithm calculates motion offset by using both the GME and Gyro algorithms.

# 1.3 DIS Implementation

Currently the DIS function is integrated in the VI module, as shown in Figure 1-2.

**Figure 1-2** Functional block diagram of VI channels



**NOTICE**

- DIS can run only on physical channels.
- The DIS video input images can be linear semi-planar420 or single component images. Only uncompressed images are supported.
- After the DIS is enabled, the flip and mirror functions cannot be performed in the VI.
- The aspect ratio for the DIS input image ranges from 16:3 to 16:27.
- During DIS processing, the VGS and GDC modules are required. If multiple modules need to use the VGS or GDC functions, frames may be discarded during DIS processing due to insufficient VGS and GDC performance.

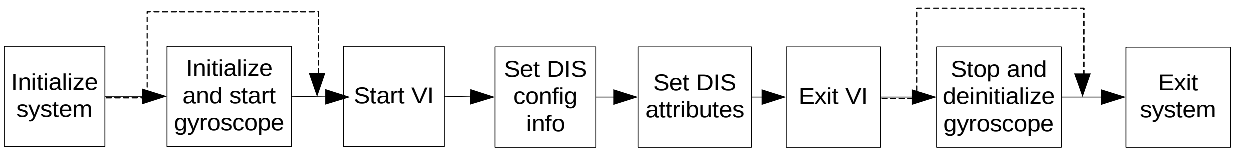
# 2 DIS Development Application

## 2.1 DIS Usage

For details about DIS-related APIs and parameters, see chapter 3 "VI" in the *HiMPP IPC V4.0 Media Processing Software Development Reference*. For details about the implementation, see the DIS sample.

Figure 2-1 shows the DIS usage process.

**Figure 2-1** DIS usage process




## 2.2 Parameter Configuration

Before starting the DIS, you need to configure parameters related to DIS configuration information and attributes. Different parameter values have different impact on the image stabilization effect. This section describes the configurations of several parameters which have important impact on the image stabilization effect.

### u32CropRatio

**u32CropRatio** is the cropping ratio of the DIS output image. The value range is [50, 98], while the value is usually set to 80 (that is, only 80% of the image is output after DIS processing). Assume that the aspect ratio of the input image is 1920 x 1080. Set **u32CropRatio** to **80**, that is, crop 10% of the left, right, upper, and down edges of the input image respectively. The width of the cropped image is 1536 (1920 – 2 x 1920 x 10%), and the height is 864 (1080 – 2 x 1080 x 10%). **Note: If the cropped width or height is not even, then round it down to the nearest even number.**

- When the video input resolution is greater than or equal to 1920 x 1080, the minimum value of **u32CropRatio** is **50**.

- When the video input resolution is less than 1920 x 1080, the minimum value of **u32CropRatio** is **70**.

## enMode

The DIS algorithm involves the concept of degree of freedom (DOF).

- From the user's perspective, the DOF concept indicates the three-dimensional space. For the X, Y, and Z axes, each axis can be translated and rotated. That is, there are a total of six motions, which are known as the 6-axis image stabilization.
- From the algorithm's perspective, the DOF concept indicates the number of operators used in the 3 x 3 matrix of affine transformation.

The affine transformation operation varies according to the DOF numbers.

Differences between 4DOF and 6DOF are as follows:

The 4DOF algorithm uses four operators, and it mainly implements translation, rotation, and scaling on images. Compared with the 6DOF algorithm, the 4DOF algorithm has less two operators. Fewer operators lead to fewer misjudgments and prevent smearing phenomena on the background caused by large moving objects. The rolling shutter phenomenon is obvious.

The 6DOF algorithm uses six operators, and it mainly implements translation, rotation, scaling, aspect ratio conversion, and shearing on images. It has good image stabilization effect and can calibrate parallelograms. However, the shortcoming is that smearing phenomena on the background may occur.

## enMotionLevel

**enMotionLevel** is the camera motion level, including the low, normal and high levels.

- Low motion level: The camera moves slightly.
- Normal motion level: The camera moves normally.
- High motion level: The camera moves greatly.

This parameter is generally set to the normal level. The motion level needs to be adjusted according to the actual situation.

**NOTE**

Hi3516C V500 does not support Low mode.

## enPdtType

**enPdtType** indicates the product type supported by the DIS. Currently, three kinds of products are supported, including IPC, DV and UAV products. Adjust the product type according to the actual product.

## bCameraSteady

**bCameraSteady** indicates the enable switch for determining whether the camera is fixed and static. This parameter takes effect only when IPC products are used. In case of DV and UAV products, this parameter is invalid, and the default value is **HI_FALSE**.

## as32RotationMatrix

**as32RotationMatrix** is a 3 x 3 rotation matrix, **MATRIX_NUM** is **9**. This parameter is used for the direction conversion between the Gyro sensor coordinate system and the image coordinate system. The algorithm is based on the image coordinate system. Different installation positions of Gyro correspond to different coordinate directions. Therefore, the direction conversion is required. In addition, ensure that the Gyro chip is parallel or perpendicular to the image sensor.

The direction conversion between the Gyro coordinate system and the image coordinate system is implemented through the rotation matrix. Assume that the Gyro data is (Xg, Yg, Zg), and the Gyro data used by the algorithm is (Xa, Ya, Za).

$$\begin{bmatrix} Xa \\ Ya \\ Za \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & l \end{bmatrix} \begin{bmatrix} Xg \\ Yg \\ Zg \end{bmatrix}$$

The nine parameters of the rotation matrix **as32RotationMatrix** [MATRIX_NUM] correspond to [a, b, c, d, e, f, g, h, l], respectively. For the calculation details of the nine parameters, see section 2.3.3 "Adaptation of Gyro and Image Coordinate Direction."

## u32GyroOutputRange

**u32GyroOutputRange** is the angle range of the Gyro data. To improve the precise, the value range is multiplied by 100, that is, [0, 200000]. The current range used by the Gyro is also multiplied by 100, that is, 250 x 100 = 25000.

## u32GyroDataBitWidth

**u32GyroDataBitWidth** is the bit width of the Gyro data. The value range is [0, 32]. The bit width of the current Gyro data is 15 bits, that is, the output data value of the Gyro is [-32768, 32768].

## u32MovingSubjectLevel

**u32MovingSubjectLevel** is used to determine whether an object is moving in front of the camera. Its value ranges from 0 to 6. This parameter is used to prevent smearing on the background. Background smearing and image stabilization effect must be balanced.

- A smaller value indicates higher stability during the motion process but proneness to smearing.
- A larger value indicates poorer image stabilization effect during the motion process but fewer smearing issues.

## u32RollingShutterCoef

**u32RollingShutterCoef** is used to calibrate the rolling shutter strength. The value range is [0, 1000]. This parameter applies to the scenario where the camera is moving towards one direction for a long time, for example, shooting the view outside the window on a train. For the rolling shutter caused by the repeated jitter, the algorithm adaptively detects and corrects the phenomenon. You are advised to set this parameter to **0**.

## u32ViewAngle

This parameter is not supported by Hi3559A V100/Hi3519A V100/Hi3516C V500.

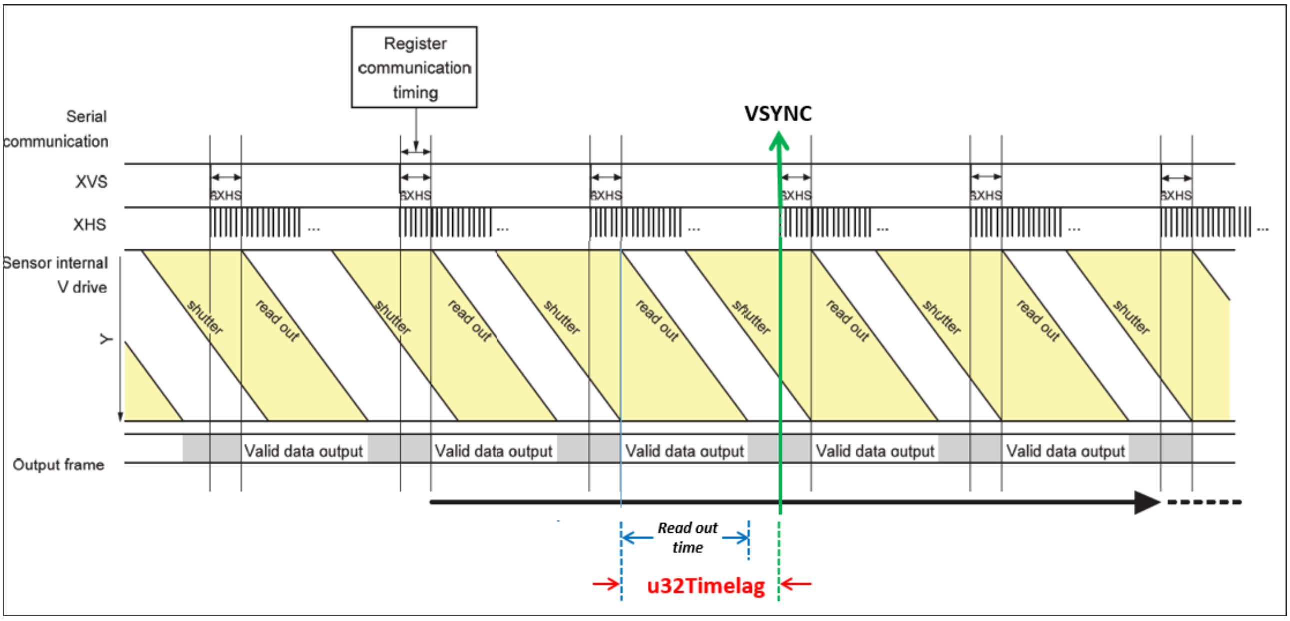
## u32TimeLag

**u32Timelag** indicates the time difference between the readout time **(t_readout)** of the last line of the current frame and the Vsync (**t_vsync**) of the next frame. The unit is µs.

u32Timelag = t_readout – t_vsync

With a normal sensor timing configured, the parameter value is close to the readout time and is **negative**. In frame reduction mode, the absolute value of this parameter increases with the extension of the blanking interval. Figure 2-2 shows the position of **u32Timelag** in the sensor timing.

**Figure 2-2** u32Timelag in the sensor timing




## u32HorizontalLimit and u32VerticalLimit

**u32HorizontalLimit** and **u32VerticalLimit** indicate the horizontal and vertical offset limits respectively. The value range is [0, 1000]. When a large moving object causes the horizontal offset of background smearing to exceed a certain amplitude, the DIS is not performed. The offset amplitude can be calculated as follows: **2047** x **u32HorizontalLimit**/1000.

This parameter must be used in conjunction with the **bCameraSteady** and takes effect only when **bCameraSteady** is set to **HI_TRUE**. When **bCameraSteady** is set to **HI_FALSE**, the value is set to **1000** by default.

## bStillCrop

**bStillCrop** is used to disable the DIS image stabilization effect while retaining the original cropping ratio for output. After this function is enabled, the DIS output picture has no image stabilization effect, but the cropping ratio of the output picture is the same as that of the output picture with the image stabilization effect. Typically this parameter is set to **HI_FALSE**. It can be set to **HI_TRUE** if required.

### u32Strength

In low illumination, the edge jitter of moving objects appears to be more obvious when DIS is enabled. In a scene with low illumination and intense motion, the edge of a moving object (or, motion edge) becomes blurry due to the prolonged shutter time. The blurriness of a motion edge changes periodically with the jitter periodicity. With DIS enabled, the periodic change of the motion edge is highlighted in contrast with the stabilized motion body in the enlarged image.

**u32strength** controls the Gyro-based DIS strength. It is valid only for Gyro-based DIS. The maximum value is **1024**.

**NOTICE**

In the development, the default value of this parameter should also be set to **1024**. You are advised to retain the default value of **u32strength**. Decreasing its value compromises the DIS effect.

For details about how to use **u32Strength**, see 2.3.7 "Gyro-based DIS Strength Applications."

# 2.3 Gyro

The main purpose of using Gyro in image stabilization is as follows:

- Use local DIS to improve the DIS effect.

  Local DIS can perform reverse correction on the image jitter according to the lens distortion characteristics. In the case of a large distortion, a better DIS effect can sill be achieved. Local DIS provides satisfactory DIS performance on inconsistent local jitters.
- Prevent smearing phenomena on the background.

  In many cases, the GME algorithm cannot determine whether the foreground or the camera is moving. For example, if there is a large object moving before a static camera, the algorithm may regard the foreground moving as camera moving by mistake and start anti-jitter, causing the smearing phenomena on the background. The Gyro can reflect the motion state of camera. Adding a Gyro can better remedy the defect mentioned above.
- Realize image stabilization in scenarios with low illumination or few feature points.

  In scenarios with low illumination, the GME algorithm cannot collect feature points due to the dark image background. Therefore the image stabilization effect cannot be achieved in low illumination conditions. However, the usage of Gyro can solve the problem with greatest ease.

## 2.3.1 Gyro Algorithm Process

The process of Gyro algorithm is as follows:

**Step 1** Before using the Gyro related algorithms, ensure that the board is attached with an available Gyro chip.

**Step 2** Before enabling the DIS function, load the drivers motionsensor_chip, motionsensor_mng, and hi_spi, and ensure that motionsensor starts to work and generates data.

Note the following:

- You must load the motionsensor_mng driver before the motionsensor_chip driver. After loading the motionsensor_mng driver each time, reload the motionsensor_chip driver.
- The process of starting motionsensor is as follows: Initialize the motionsensor -> Set motionsensor parameters -> Start the motionsensor.
- The process of stopping motionsensor is as follows: Stop the motionsensor -> Deinitialize the motionsensor. You are advised to start the motionsensor before enabling the DIS and setting the DIS mode to DIS_MODE_GYRO, while stop motionsensor after stopping the VI.

**Step 3** Adapt the directions of the Gyro coordinate system and the image coordinate system. Correctly configure the rotation matrix **as32RotationMatrix** [MATRIX_NUM].

**Step 4** Determine the camera calibration parameters and **u32Timelag**.

**Step 5** Initialize and start the Gyro, and then enable the DIS.

**Step 6** Disable the DIS, stop the Gyro, and then exit the system.

**----End**

For details, see the Gyro usage related content in the DIS sample code file.

## 2.3.2 Gyro Drive Reference Code

The SDK contains the driver code of Bosch Gyro. For Gyro of other models, see the driver code of Bosch Gyro and make modification as required.

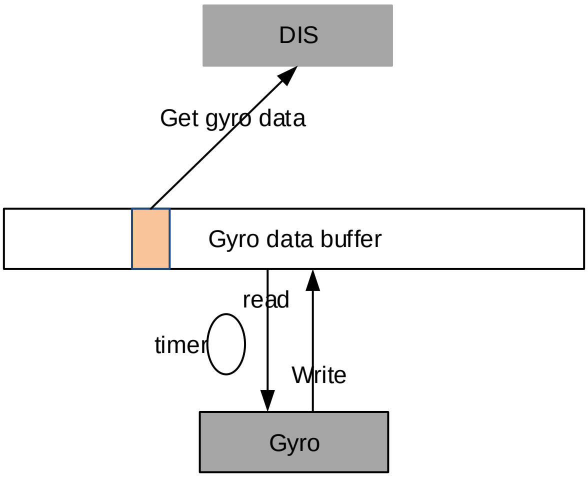
The code is stored in the following directory:

```
motionsensor driver: \drv\extdrv\motionsensor\
Sample: \mpp\sample\dis\
```

For the usage of the code, run the command **make** in the **extdrv** directory. Then you can obtain the **ko** file in the **\mpp\ko** directory. The **\mpp \ko\load** script does not load the Gyro driver by default. Modify the script based on the actual situation.

Figure 2-3 shows the principle of obtaining Gyro data by DIS.

**Figure 2-3** Principle of obtaining Gyro data by DIS




The Gyro data is stored in the allocated Gyro Data buffer. After the Gyro driver is started, an internal timer in the driver is started, continuously reads data from the Gyro FIFO, timestamps each group of data, and writes the data to the Gyro Data buffer. Based on the start and end timestamps of each frame, the DIS driver obtains Gyro data of corresponding periods from the Gyro Data buffer for image stabilization processing.

### 2.3.3 Adaptation of Gyro and Image Coordinate Direction

When installing the Gyro chip, ensure that the chip is installed in standard mode, that is, keep the chip parallel or perpendicular to the image sensor.

For the usage of Gyro algorithm, the camera mobile information is provided by the Gyro. Therefore, Gyro data accuracy is crucial. Different Gyro installation positions correspond to different directions of coordinate system.

When using DIS Gyro related algorithms, you must correctly adapt the Gyro and image coordinate directions.

To clearly describe the image coordinate system, mobile phone screens are used, as shown in Figure 2-4.

- The Z-axis is perpendicular to the image. The Z-axis direction that points to human eyes is positive.
- X-axis and Y-axis are in horizontal and vertical direction and correspond to the width and height of the image, respectively.

The DIS algorithm is based on the image coordinate system. Figure 2-5 shows the coordinate system of the ICM20690 Gyro.

**Figure 2-4** Image coordinate system

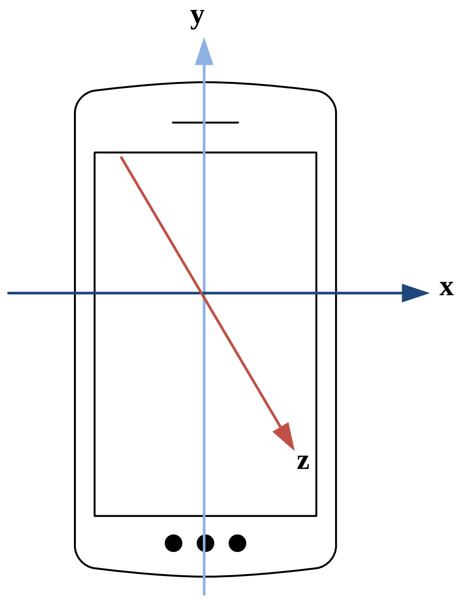
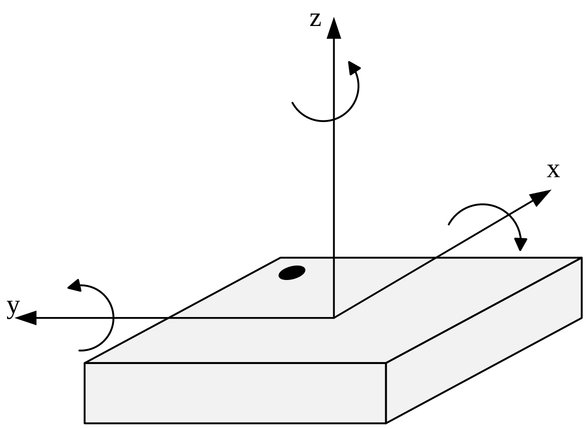



**Figure 2-5** Gyro coordinate system




Here we introduce how to convert coordinate system directions for Gyros installed in two different positions. Conversation in other installation positions can be analogized. The black dot indicates the position of the Gyro chip pin 1.

- Gyro installation position 1

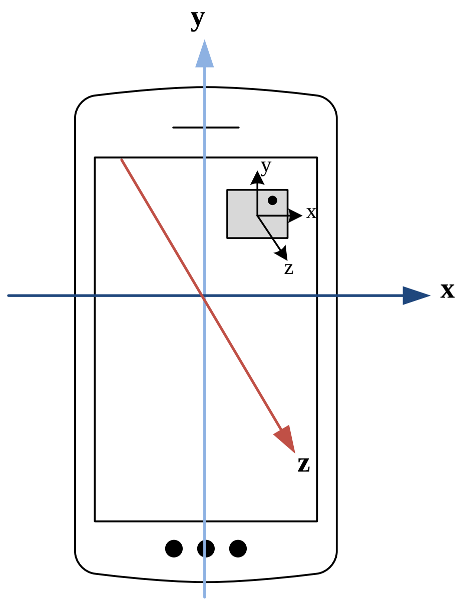
  If the Gyro installation position is shown in Figure 2-6, the direction of the Gyro coordinate system is consistent with that of the image coordinate system. The data obtained by the Gyro is (Xg, Yg, Zg) and the Gyro data used by the algorithm is (Xa, Ya, Za). In this case, the relationship is as follows:

$$\begin{bmatrix} Xa \\ Ya \\ Za \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xg \\ Yg \\ Zg \end{bmatrix} = \begin{bmatrix} Xg \\ Yg \\ Zg \end{bmatrix}$$

Therefore, set the rotation matrix in DIS Config as a unit matrix, that is $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

The nine parameters of **32RotationMatrix**[MATRIX_NUM] correspond to [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1], respectively.

**Figure 2-6** Gyroscope installation position 1




- Gyro installation position 2

If the Gyro installation position is shown in Figure 2-7, the directions of the Gyro coordinate system and the image coordinate system are inconsistent and require conversion. The data obtained by the Gyro is (Xg, Yg, Zg) and the Gyro data used by the algorithm is (Xa, Ya, Za). In this case, the conversion relationship is as follows:

$$\begin{bmatrix} Xa \\ Ya \\ Za \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xg \\ Yg \\ Zg \end{bmatrix} = \begin{bmatrix} Yg \\ -Xg \\ Zg \end{bmatrix}$$

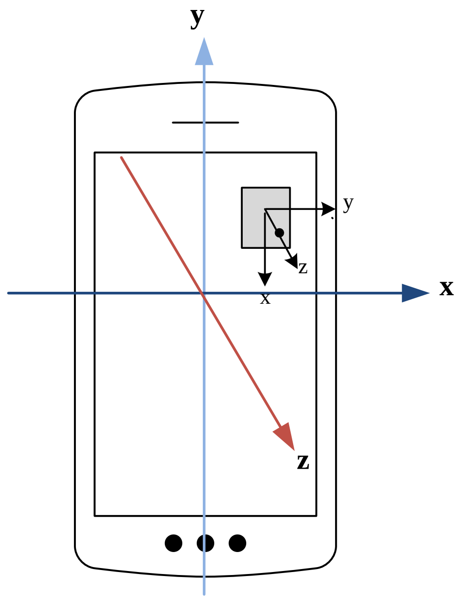
Therefore, set the rotation matrix in DIS Config as follows:

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The nine parameters of **as32RotationMatrix**[MATRIX_NUM] correspond to [0, 1, 0, –1, 0, 0, 0, 0, 0, 1], respectively.

**Figure 2-7** Gyro installation position 2




### 2.3.4 Lens Calibration Parameters and u32Timelag Determination

The data structures of the lens calibration parameters (CAMERA_CALIB_DIS_PARAM_S) and MPI board parameters (LDCV2_ATTR_S) are identical. You can directly configure the values generated by lens calibration to the corresponding MPI parameters.

For details about **u32Timelag**, see the calculation of **u32Timelag** in section 2.2 "Parameter Configuration."

### 2.3.5 Gyro Initialization and Startup

Initializing the Gyro is mainly for allocating MMZ memory to store the Gyro data.

The Gyro output data in stored in a ring buffer, and the algorithm reads the Gyro data from the buffer based on the frame interrupts. The X-, Y-, and Z-axis data correspond to the timestamps.

**Figure 2-8** Buffer data

| | | | | |
|---|---|---|---|---|
| x | y | z | temp | timestamp |

The buffer allocated from the MMZ is used to store the following data: X-, Y-, and Z-axis data of the gyroscope, temperature data and timestamp.

The data structure length of timestamp is 8 bytes, and the data structure length of X-, Y-, and Z-axis data or temperature data is 4 bytes.

When a frame interrupt occurs, data query and collection from the buffer are performed based on the start and end timestamps. Find the Gyro data that meets the conditions and then transmit the data to the DIS algorithm.

For the details about buffer assignment and size, see the sample code file.

### 2.3.6 Gyro and Accelerometer Configuration

- You are advised to set the measurement range of the Gyro to **250** for the IPC or **1000** for the DV (10-bit decimal precision).
- You are advised to set the Gyro data sampling rate (ODR) to **1000** (10-bit decimal precision) and the bit width of Gyro output data to 15 bits. The Gyro data range is [–32768, +32768].
- You are advised to set the measurement range of the accelerometer to **16** (10-bit decimal precision).
- You are advised to set the data sampling rate (ODR) of the accelerometer to **1000** (10-bit decimal precision), and the bit width of the accelerometer output to 15 bits. The accelerometer data range is [–32768, +32768].

### 2.3.7 Gyro-based DIS Strength Applications

The following applications apply only to Gyro-based DIS. The maximum DIS strength is **1024**. In the development, the default DIS strength should also be set to **1024**.

You are advised to use the exposure limiting policy first and then use the **u32Strength** attenuation as required.

1. Limiting the maximum exposure time in the AE route:

   It is recommended that the maximum exposure time be less than or equal to 10 ms. You can also limit the maximum exposure time in the AE route.

   Impact on the image effect: With the tradeoff between the DIS effect, brightness, and NR, the jitter of the motion edge to the human eye is reduced radically.

   - In normal illumination (for example, daytime at outdoor): The normal mode and DIS mode are not affected.
   - In low illumination (for example, bright indoors while dark outside): The DIS effect is improved, and the image is dark or more noise is found.
   - In extreme low illumination: The DIS effect is improved, the image is very dark (with the maximum gain multiplier unchanged), or obvious noise is found (with the maximum gain multiplier increased).

**NOTICE**

- When only the exposure time is limited without changing the gain, the image may be too dark in low illumination.
- When the exposure time is limited and the gain multiplier is increased, the image is not darken, but the noise becomes obvious.

2. Adjusting the DIS strength adaptively based on the exposure time:

   Have the DIS strength attenuate based on the exposure time **ExpTime**.

   Set adaptive attenuation according to a ratio in the case of over 10 ms exposure time. Set the DIS strength in the case of 10 ms exposure time to the maximum (**1024)**. Set the DIS strength in the case of 30 ms exposure time to the minimum (**1**). Set the DIS strength in the case of 10–30 ms exposure time to smoothly-transitioned values.

   Impact on the image effect: The image brightness and NR are given the priority, so that the jitter of motion edge is less obvious to the human eye at the cost of the DIS effect, but the jitter of motion edges is not addressed radically.

   - In normal illumination (for example, daytime at outdoor): The normal mode and DIS mode are not affected.
   - In low illumination (for example, bright indoors while dark outside): The DIS effect is compromised to reduce the contrast between the stableness of the motion body and the jitter of the motion edge.
   - In extreme low illumination: The DIS produces zero effect, equivalent to the scene with DIS disabled.

3. Adaptive solution (recommended):

   The solution involves a tradeoff between the DIS effect, image brightness, NR, and motion edge jitter stabilization.

   When the DIS is enabled, perform the following based on the Gyro motion information of the gyroscope:

   - In scenes with intense motion: Limit the exposure time and increase the gain multiplier. Reduce the DIS strength when the exposure time and gain reach the maximum.
   - In scenes with mild motion: Restore the original exposure time and gain.

   Optimize the AE and NR parameters as required to address the motion blurriness and noise issue.

   Expected results:

   - In scenes with normal illumination: The results in all aspects are optimal.
   - In scenes with mild motion (including static scenes): The results in all aspects are optimal.
   - In scenes with low illumination and intense motion: The DIS effect is improved. The brightness is slightly compromised or slightly more noise is found.
   - In scenes with extremely low illumination and intense motion: The brightness and NR are improved, but the DIS effect is compromised.

# 2.4 Lens Calibration

## 2.4.1 Checkerboard Calibration

### 2.4.1.1 Calibration Tool

For details, see section 2.5.10 "2.5.10 HiPQ DIS Calibration Tool" in the *HiSilicon PQ Tools User Guide.*

### 2.4.1.2 Sending the Results to the Board

LDCV2_ATTR_S needs to be configured for sending the results to the board. For details about the LDCV2 attributes, see section 2.4.1 "Basic Data Structures" in the *HiMPP V4.0 Media Processing Software Development.*

## 2.4.2 FOV Calibration

### 2.4.2.1 Background

In the zoom application outdoor, because the focal lengths of the FOV changes, it is complex to calibrate the focal lengths by using LDCV2 calibration with the lens. You can convert the FOV of a given lens to the LDCV2 parameters instead of using the checkerboard calibration with the lens.

FOV conversion is convenient and simple, but requires specific conditions. If the conversion result is not ideal, you should resort to the lens checkerboard calibration.

An implementation sample is provided for the FOV conversion to LDCV2 parameters.

### 2.4.2.2 FOV Conversion to LDCV2

- Inputs: image width, image height, FOV type, and FOV
- Outputs: LDCV2 parameters

**Table 2-1** Value range of LDCV2 parameters converted from the FOV

| LDCV2 Parameter | Value Range | Description |
|---|---|---|
| s32FocalLenX | [6400, 117341700] | Valid focal length of the lens in the horizontal direction |
| s32FocalLenY | [6400, 117341700] | Valid focal length of the lens in the vertical direction |
| s32CoorShiftX | W/2*100 | X-coordinate of the optical center. **W** indicates the image width. |
| s32CoorShiftY | H/2*100 | Y-coordinate of the optical center. **H** indicates the image width. |
| as32DstJunPt[0] | 800000 | Lens distortion coefficient |
| as32DstJunPt[1] | 800000 | Lens distortion coefficient |
| s32SrcJunPt | 800000 | Lens distortion coefficient |
| as32SrcCaliRatio[0][0] | 10000 | Lens distortion coefficient |

| LDCV2 Parameter | Value Range | Description |
|---|---|---|
| as32SrcCaliRatio[0][1] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[0][2] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[0][3] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[1][0] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[1][1] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[1][2] | 0 | Lens distortion coefficient |
| as32SrcCaliRatio[1][3] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[0][0] | 100000 | Lens distortion coefficient |
| as32DstCaliRatio[0][1] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[0][2] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[0][3] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[1][0] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[1][1] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[1][2] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[1][3] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[2][0] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[2][1] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[2][2] | 0 | Lens distortion coefficient |
| as32DstCaliRatio[2][3] | 0 | Lens distortion coefficient |
| s32MaxDu | 1048576 | Lens distortion coefficient |

**W** and **H** indicate the image width and height, for example, for 2160p, **W** is **3840** and **H** is **2160**.

### 2.4.2.3 Precautions for FOV Conversion

FOV conversion can be considered as a special method for lens calibration. Under specific conditions, it provides higher efficiency than lens calibration. Pay attention to the following:

- Center overlap. When designing the structure, ensure that the optical center of the sensor and the center of the lens overlap.
- FOV range. FOV conversion applies mainly to a lens with long focus and an FOV within (0°, 20°). A lens with short focus or a wide-angle lens is not recommended.
- Distortion. The lens should have no obvious distortion. The distortion rate of barrel distortion should not exceed –10% and the distortion rate of pincushion distortion should not exceed 5%.
- FOV accuracy. Provide an accurate FOV. You are advised to keep the error within 5%.

Otherwise, the converted parameters may not meet the expectation. In this case, you can use the lens checkerboard calibration (model calibration or production line calibration) instead to improve the accuracy.

### 2.4.2.4 FOV Conversion Sample

Find the sample code in **mpp/sample/dis**.

You can perform encapsulation yourself to implement dynamic calling and smooth switching.