Frequently Asked Questions

How to detect frame loss in BU/DU series.

#4300-0310 Version 1.0.0.1

TOSHIBA TELI CORPORATION

Information contained in this document is subject to change without prior notice.

 www.toshiba-teli.co.jp

Document in Japanese

このドキュメントは BU／DU シリーズカメラでフレームロストを検出する方法について記述したドキュメントです。日本語ドキュメントは英語ドキュメントの後にあります。

End of document in Japanese

This document shows how to detect frame loss in BU/DU series camera.

1. How to detect frame loss

Frame loss is a phenomenon that an application failed to get some frames in a series of frames sent from a camera. Application can detect frame loss by checking BlockID of received frame.

BlockID is a number assigned by the camera. The camera will increment BlockID by 1 on sending out a frame, and will send out BlockID as a member of Leader and Trailer of the image streaming data.

Application can detect frame loss by checking that BlockID of a frame is the next number of the previous frame.

Note that a frame previous to current frame, that has not been processed, may exist in ring buffer inside TeliCamAPI, when application uses Strm_ReadCurrentImage() function in TeliCamSDK or ReadCurrentImage() method in TeliCamDNetSDK for getting image.

Please check BlockID of images prior to current frame in the ring buffer inside TeliCamAPI using Strm_LockBuffer() or LockBuffer(), when Block ID of current image is not the next number of BlockID assigned to previously processed image.

The followings are sequence for getting BlockID.

<Gets BlockID from High-Level streaming functions in TeliCamSDK>

BlockID is “ullBlockId” member value of CAM_IMAGE_INFO structure data, which is an argument of Strm_ReadCurrentImage() and Strm_LockBuffer() functions.

<Gets BlockID from Low-Level streaming function in TeliCamSDK>

A) Call Strm_GetStrmReqInfo() to get CAM_STRM_REQUEST_INFO structure data from the CAM_STRM_REQUEST_HANDLE, which is an argument of Strm_DequeueRequest() function.

B) Cast “sU3vInfo.pvLeader” member of CAM_STRM_REQUEST_INFO structure data to U3V_STRM_IMG_LDR*.

C) BlockID is “ullBlockId” member value of U3V_STRM_IMG_LDR*.

<Gets BlockID from streaming methods in TeliCamDNetSDK>

BlockID is “BlockID” member value of CameraImageInfo class instance, which is an argument of ReadCurrentImage() and LockBuffer() methods of CameraStream class instance.

<Gets BlockID from streaming functions in TeliU3vSDK>

A) Cast “pLeader” member of U3V_STRM_REQ structure data to U3V_STRM_LDR*.

B) BlockID is “block_id” member value of U3V_STRM_ LDR*.

2. Hint for prevent frame loss

There are many causes in frame loss. The followings will be useful for preventing frame loss.

- Check USB cable.
  Degradation of cable or connector, incompatibility between camera, USB cable and USB3.0 host controller may be causes of frame loss.

- Check if BufferBusy occurred.
  A buffer in the ring buffer inside TeliCamAPI is locked during “ImageAcquired” callback process or “ImageAcquired” event process. Strm_LockBuffer() function and LockBuffer() method of CameraStream class also locks a buffer in the ring buffer.

  If a buffer is locked for a long time, TeliCamAPI will raise “BufferBusy” event and discard received images, because a buffer that the received image should be written to is protected from overwriting.

  Moving process in “ImageAcquired” callback or “ImageAcquired” event to the other thread or calling Strm_UnlockBuffer() function or UnlockBuffer() method as soon as possible after calling Strm_LockBuffer() function or LockBuffer() method will be useful for preventing “BufferBusy”.

- Increase number of buffers in ring buffer.
  Increasing number of buffers in ring buffer will be useful be useful for preventing “BufferBusy”.
  Application can set number of buffers in ring buffer by specifying number of buffers in arguments of Strm_OpenSimple() function of Open() method of CameraStream class.

- Moves processing in “ImageAcquired” callback or “ImageAcquired” event to the other thread.
  Moving process in “ImageAcquired” callback or “ImageAcquired” event to the other thread will be useful for preventing frame loss.

- Increase number of StreamRequest and call Strm_EnqueueRequest() function immediately.
  On receiving image stream data, device driver will take out a StreamRequest structure from “WaitQueue” inside driver and write stream data in buffer of the StreamRequest structure. If “WaitQueue” is vacant, device driver will discard the received image stream.

  Increasing number of StreamRequest and putting them in “WaitQueue” using Strm_EnqueueRequest() function as soon as possible after data in it has been processed will be useful to keep “WaitQueue” not vacant.

- Make GUI updating cycle longer.
  CPU cost of updating images in PC display is usually not so little. Making image updating cycle longer will be useful to make CPU time for image processing longer.

- Adopt multi-thread processing.
  Most of current CPU has multiple processor cores in a chip. Adopting multi-thread processing will make image processing time shorter, which will be useful for preventing frame loss.

Document in Japanese

1. フレームロストの検出方法

フレームロストは、アプリケーションが一連のカメラ送出画像の中の一部のフレームの受信に失敗する事象のことです。受信した画像の BlockID を確認することによりフレームロストの発生を検出することができます。

BlockID はカメラが発番する番号です。カメラは 1 枚の画像を送るたびに BlockID の値を 1 だけ増やし、画像データと一緒に、画像ストリームの Leader と Trailer の両方の BlockID メンバーとして創出されます。

アプリケーションはフレームの BlockID が前回フレームの BlockID と連番になっていることを確認することによりフレームロストの検出を行うことができます。

TeliCamSDK の Strm_ReadCurrentImage()関数または TeliCamDNetSDK の ReadCurrentImage() メソッドをご使用の場合、カレントフレームの直前のフレームがまだ処理されていない状態のまま TeliCamAPI 内のリングバッファ内に存在するかもしれないことに注意してください。

カレントフレームの BlockID が前回処理したフレームの BlockID と連番になっていない場合、Strm_LockBuffer()関数または LockBuffer()メソッドを使用して リングバッファ内のカレントフレームに先立って受信された画像の BlockID も確認してください。

以下に BlockID の取得の仕方を記載します。

<TeliCamSDK 高水準ストリーム関数の場合>

Strm_ReadCurrentImage()関数又は Strm_LockBuffer()関数の引数として取得した CAM_IMAGE_INFO 構造体の ullBlockId メンバーの値が BlockID 値です。

< TeliCamSDK 低水準ストリーム関数の場合>

A) Strm_DequeueRequest() 関数で取得した CAM_STRM_REQUEST_HANDLE を Strm_GetStrmReqInfo()関数の引数として与えて、 CAM_STRM_REQUEST_INFO 構造体を取得する。

B) CAM_STRM_REQUEST_INFO 構造体の “sU3vInfo.pvLeader” メンバーをキャストして U3V_STRM_IMG_LDR*型にしたポインタを作成する。

C) U3V_STRM_IMG_LDR*が指す構造体の“ullBlockId”メンバーの値が BlockID 値です。

< TeliCamDNetSDK の CameraStream クラスメソッドの場合>

CameraStream オブジェクトの ReadCurrentImage()又は LockBuffer()メソッドで取得した CameraImageInfo オブジェクトの “BlockID”メンバーの値が BlockID 値です。

< TeliU3VSDK ストリーム関数の場合>

A) U3V_STRM_REQ 構造体の“pLeader”メンバーを U3V_STRM_LDR*型にキャストしたポインタを作成する。

B) U3V_STRM_ LDR*が指す構造体の“block_id” メンバーの値が BlockID 値です。

2. フレームロストを防ぐためのヒント

フレームロストの発生要因はさまざまな要因があります。以下にフレームロスト防止に効果があると思われる項目を示します。

- USB ケーブルを確認する。

USB ケーブルや USB コネクタの劣化、カメラと USB ケーブルと USB3.0 ホストコントローラの相性問題がフレームロストを誘発する可能性があります。

- “BufferBusy”が発生していないか確認する。
  “ImageAcquired”コールバック又は“ImageAcquired”イベントハンドラー実行中はリングバッファ内の処理対象のバッファがロックされます。Strm_LockBuffer()関数、CameraStream クラスの LockBuffer() メソッドを実行したときも処理対象のバッファがロックされます。

  バッファが長期間ロックされ続けると、受信画像をバッファに書き込めないため TeliCamAPI は“BufferBusy”イベントを発生させ、受信した画像を破棄します。

  “ImageAcquired”コールバック 又は“ImageAcquired”イベントの処理の別スレッドへの移動、Strm_LockBuffer()関数又は LockBuffer()メソッド実行後できる限り速やかに Strm_UnlockBuffer()関数又は UnlockBuffer()メソッドを実行することにより“BufferBusy”の発生を抑制できます。

- リングバッファのバッファ数を増やす。
  リングバッファのバッファ数を増やすと“BufferBusy”抑制に効果があります。

  Strm_OpenSimple()関数または CameraStream クラスの Open()メソッドで引数にバッファ数を指定すると、リングバッファ数を変更することができます。

- Moves processing in “ImageAcquired” callback to the other thread.
  “ImageAcquired”コールバック 又は“ImageAcquired”イベントの処理を別スレッドへ移動すると、フレームロストの発生を抑制できます。

- StreamRequest 数を増やし、受信画像処理後は速やかに Strm_EnqueueRequest()実行。
  画像ストリームを受信するとデバイスドライバは ドライバ内の“WaitQueue” から StreamRequest 構造体を取り出し、StreamRequest 構造体内のバッファに受信データを書き込みます。“WaitQueue”が空の場合、ドライバは受信した画像データ廃棄します。

  StreamRequest 構造体の数を増やし、受信画像の処理が完了し次第 Strm_EnqueueRequest()を実行して StreamRequest 構造体を “WaitQueue”に戻すことにより、“WaitQueue”が空になる危険を化抑制できます。

- 画面の更新周期を長めにする。
  PC の画面に表示する画像の更新には、通常はそれなりの CPU コストがかかります。画面の更新周期を長めにすることにより、画像処理に使用できる CPU 時間を増やすことができ、フレームロストの抑制にも効果があります。

- マルチスレッド処理の適用。
  最近の CPU の多くはチップ内に複数のプロセッサコアを内蔵しています、マルチスレッド処理を採用することにより画像処理時間を短縮でき、フレームロスト発生の抑制に効果があります。

End of document in Japanese

3. Others

3.1. Revision History

Date	Version	Description
2016/05/24	1.0.0	Created the initial version

3.2. Disclaimer

The disclaimer of this document is described in “License Agreement TeliCamSDK Eng.pdf” in TeliCamSDK installation folder.
Make sure to read this Agreement carefully before using it.
Refer to TeliCamSDK installation folder/Documents/License folder

