Technical Application Note TAN2005003

Setting a GPIO pin to output a strobe signal pulse pattern
Revised July 5, 2005

1.1. Subject

Technical Application Note (TAN2005003): Setting a GPIO pin to output a strobe signal pulse pattern.

1.2. Applicable Product(s)

The following PGR Imaging Products support this functionality:

- ***Flea*** (all models) with firmware v0.9.0.31 or later
- ***Scorpion*** (all models except SCOR-03NS and SCOR-13SM) with firmware v0.9.0.38 or later

Consult Knowledge Base Article 94 to determine camera firmware versions. The most recent firmware versions can be downloaded from the PGR website at http://www.ptgrey.com/support/downloads/.

1.3. Application Note Description

The purpose of this Technical Application Note is to provide the user with a set of basic instructions on how to configure one of the Applicable Product(s) above to output a strobe pulse pattern on one of its general purpose input/output (GPIO) pins.

By default, a pin that is configured to be a strobe output will output a pulse each time the camera begins integration of an image. The strobe pattern functionality described in this TAN allows users to define the frames for which the camera will output a strobe. This is useful in situations where a strobe should only fire:

- Every Nth frame (e.g. odd frames from one camera and even frames from another camera); or
- N frames in a row out of T (e.g. the last 3 frames in a set of 6); or
- Specific frames within a defined period (e.g. frames 1, 5 and 7 in a set of 8).

> ***NOTE:*** *A strobe pattern can only be defined for those GPIO pins that are configured as strobe outputs using the DCAM 1.31 parallel input/output (PIO) functionality. Pins configured to output a strobe using PGR GPIO_MODE_3 cannot implement a strobe pattern. Refer to* Technical Application Note TAN2005002: Setting a GPIO pin to strobe using DCAM 1.31 Strobe Signal Output registers.

For a full explanation of the DCAM v1.31-specific input/output modes, please consult the "GPIO Control Using DCAM v1.31 Functionality" section of the *PGR IEEE-1394 Digital Camera Register Reference*.

In addition to setting up a strobe signal output, this document also presents an example scenario designed to explain how to:

1. Configure the strobe pattern period.
2. Define a specific strobe pattern.
3. Determine whether a strobe occurred for a specific image.

1.3.1. General Considerations

1.3.1.1. General Purpose Input/Output Pins

The Applicable Product(s) is/are equipped with a set of general purpose input/output (GPIO) pins that can be accessed via the Hirose connector on the back of the camera. Different products may use different Hirose connectors; consult your camera's *Technical Reference* or *Getting Started* manual for part numbers and specifications, GPIO connector pin layouts, and GPIO electrical characteristics.

1.3.1.2. Testing Tools

To configure and test the information presented in this TAN:

1. **Connect the camera's GPIO pins to an oscilloscope or external device.** External devices can include an external light source or LED or other triggerable device. By connecting the appropriate GPIO pins to an external device or oscilloscope, you can verify that the camera is outputting a strobe pulse of the correct delay, duration and pattern. Consult your camera's *Technical Reference* or *Getting Started* manual for:
   a. GPIO connector pin layouts; and
   b. GPIO electrical characteristics.
2. **Access the camera's register space.** The easiest way to try this is using the FlyCap demo software included with the *PGR FlyCapture SDK*. For register definitions and individual bit descriptions, please refer to the "Strobe Signal Output Registers" section of the *PGR IEEE-1394 Digital Camera Register Reference*. The latest versions of PGR FlyCapture and the Register Reference can be downloaded from http://www.ptgrey.com/support/downloads/.

1.3.2. Relevant Camera Registers

- GPIO_STRPAT_CTRL (0x110C)
- GPIO_STRPAT_MASK_PIN_0 (0x1118)
- PIO_DIRECTION (0x11F8)
- FRAME_INFO (0x12F8)
- STROBE_0_INQ (0x1400)
- STROBE_0_CNT (0x1500)

1.3.3. Configuring the Camera

1.3.3.1. Example Overview

For the purposes of this Technical Application Note we will work through the following example strobe pattern scenario for two cameras on the same IEEE-1394 bus:

Parameter	Value
Camera model	FLEA-HIBW-CS
Frame rate	15Hz
Shutter (integration) time	Manual mode, 15ms
Strobe output pin	GPIO0
Strobe output characteristics	High active output, 1ms duration
Strobe output pattern (Cam_A)	Strobe every three (3) frames. The strobe should occur on the third frame. Effective strobe frequency = 15 / 3 = 5Hz.
Strobe output pattern (Cam_B)	Strobe every two (2) frames. The strobe should occur on the second frame. Effective strobe frequency = 15 / 2 = 7.5Hz.

This configuration is outlined in Figure 1 below. Multiple cameras on the same bus are automatically synchronized to each other, so with this configuration the strobes of the two cameras should be synchronized at every sixth frame.

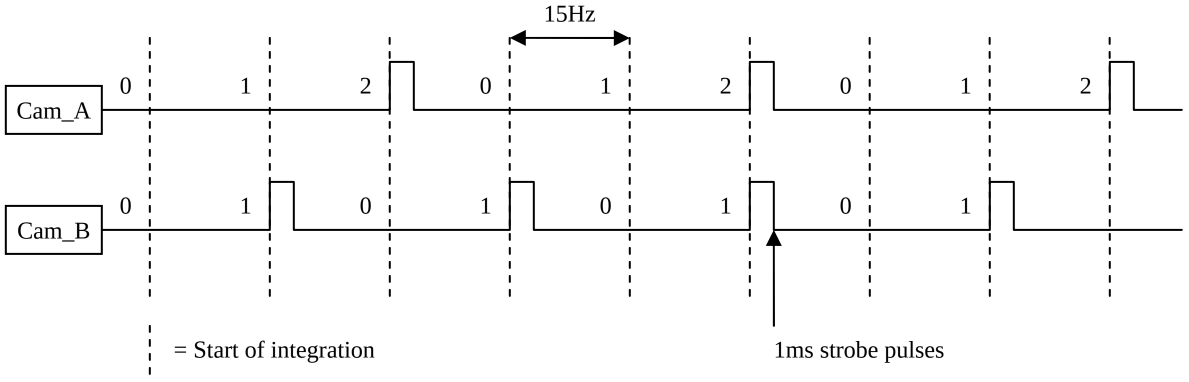

Figure 1: Example of multiple camera strobe pattern synchronization

1.3.3.2. Start the Camera

Using a separate instance of FlyCap for each Flea, begin grabbing images at the frame rate specified above. Put the shutter in manual mode and set the shutter time to be the same for each camera.

1.3.3.3. Configure the Desired Pins to Output a Strobe

Refer to *Technical Application Note TAN2005002: Setting a GPIO pin to strobe using DCAM 1.31 Strobe Signal Output registers* for instructions on configuring each camera to output a 1ms pulse off GPIO0.

1.3.3.4. Configure the Strobe Pattern Period

The programmable strobe pattern period is controlled by the *Count_Period* field of the GPIO_STRPAT_CTRL register 0x110C. The valid values for *Count_Period* are 1 – 16. By default (normal strobe mode), the *Count_Period* is 1 and the camera strobes on every frame.

At the start of integration for each image, the period counter is incremented and the *Current_Count* read-only field is updated. In the example above, Cam_A has a period of three (3). *Current_Count* therefore counts from 0 up to 2, then wraps around back to 0 to start the next period.

To configure the strobe periods for the two cameras in our example we need to set the *Count_Period* by writing the following to register 0x110C:

`(Cam_A) 0x110C = 0x8000 0300`

8	0	0	0	0	3	0	0	Hex
1000	0000	0000	0000	0000	00**11**	0000	0000	Binary
0-7		8-15		16-23		24-31		Bits

`(Cam_B) 0x110C = 0x8000 0200`

8	0	0	0	0	2	0	0	Hex
1000	0000	0000	0000	0000	00**10**	0000	0000	Binary
0-7		8-15		16-23		24-31		Bits

Bit [0] simply indicates the Presence of the strobe pattern feature; our write to this bit is therefore a NoOp. Bits [19-23] control the *Count_Period*, which is 3 for Cam_A and 2 for Cam_B.

Reading this register, depending on quickly the read returns, should show the *Current_Count* value cycling through its range of values: 0, 1, 2 for Cam_A; and 0, 1 for Cam_B.

1.3.3.5. Define the Strobe Pattern

Following the example above, we want Cam_A to strobe on the third frame only (*Current_Count* = 2) and Cam_B to strobe on the second frame only (*Current_Count* = 1). To do this, we need to configure the *Enable_Mask* in the GPIO_STRPAT_MASK_PIN_0 register 0x1118.

In general, when the *Current_Count* equals N the GPIO pin will only output a strobe if Bit [N] of the *Enable_Mask* is set to '1'. By default, all of the bits in the *Enable_Mask* are set to '1' (a read of 0x1118 would display 0x8000FFFF) so that the strobe occurs for every frame.

In our example above, we want Cam_A to strobe when the *Current_Count* equals 2. We therefore need to set Bit [2] of the *Enable_Mask*, which is actually Bit [18] of the GPIO_STRPAT_MASK_PIN_0 register, to '1'. To ensure a strobe does not occur when *Current_Count* is equal to 0 or 1, we need to set Bits [0-1] to '0'. We want Cam_B to strobe when the count equals 1, so we need to set Bit [1] (Bit [17] of register 0x1118) to '1' and Bit [0] to '0'. Our register writes would therefore look like this:

`(Cam_A) 0x1118 = 0x8000 3FFF`

8	0	0	0	3	F	F	F	Hex
1000	0000	0000	0000	**001**1	1111	1111	1111	Binary
0-7		8-15		16-23		24-31		Bits

`(Cam_B) 0x1118 = 0x8000 7FFF`

8	0	0	0	7	F	F	F	Hex
1000	0000	0000	0000	**01**11	1111	1111	1111	Binary
0-7		8-15		16-23		24-31		Bits

At this point, the strobe pulses coming off each camera should look very similar to those in Figure 1 above. If they are not aligned correctly, simply adjust the *Enable_Mask* until the pulses are synchronized.

1.3.3.6. Matching Strobe Pulses to Images

Many customers who are designing custom software applications to control the cameras and acquire images want to be able to know whether a given image would have a strobe pulse associated with it. For example, the strobe may be used to turn on a small LED or lighting system that illuminates the scene for a specific image. It may be useful to know whether a grabbed image, which the user now has access to in memory, is one that should be illuminated. In the same way, it would also be useful to know at what point in the pattern we are at, which might allow us to make various camera changes in anticipation of the next strobe.

The easiest way to accomplish this is to embed the value of the 32-bit GPIO_STRPAT_CTRL register 0x110C into the image by setting Bit [24] of the FRAME_INFO register 0x12F8. This register allows the user to control the types of frame-specific information that is embedded into the first several pixels of the image.

***NOTE**: Consult the* PGR IEEE-1394 Digital Camera Register Reference *for a detailed description of the FRAME_INFO register. Writing values different than those below could cause unexpected results.*

Following the example above, we would therefore do the following for both cameras:

```
0x12F8 = 0x8000 0080
```

8	0	0	0	0	0	8	0	Hex
1000	0000	0000	0000	0000	0000	**1**000	0000	Binary
0-7		8-15		16-23		24-31		Bits

Once this is done, the first four pixels (or 4 bytes of image data) represent the 32-bit (4-byte) value of the GPIO_STRPAT_CTRL register. From this, the application can parse the image data to look for the *Current_Count* Bits [28-31] to determine whether or not the current image should have a strobe associated with it. Following our example above, and assuming the user is programming with the PGR FlyCapture SDK, we would therefore have:

```
FlyCaptureImage     imageCam_A;
FlyCaptureContext   contextCam_A;
unsigned char       ucCurrentCount;

// Insert code here to create a context, initialize the camera,
// configure registers and start the cameras

flycaptureGrabImage2( contextCam_A, &imageCam_A );
ucCurrentCount = image.pData[3] // Byte[4] or bits[28-31] of register 0x110C
if( ucCurrentCount == 2 )
{
      // A strobe should have occurred for this image – do some stuff
}
```