#include "stdafx.h"
#include<time.h>
#include<sys/timeb.h>
#ifdef LINUX
#include <unistd.h>
#endif
#include "FlyCapture2.h"

using namespace FlyCapture2;
void PrintError( Error error ) {
  error.PrintErrorTrace();
}

bool CheckSoftwareTriggerPresence( Camera* pCam ) {
  const unsigned int k_triggerInq = 0x530;
  Error error;
  unsigned int regVal = 0;
  error = pCam -> ReadRegister( k_triggerInq, &regVal );
  if (error != PGRERROR_OK) {
    PrintError( error );
    return false;
  }
  if( ( regVal & 0x10000 ) != 0x10000 ) {
    return false;
  }
  return true;
}

bool PollForTriggerReady( Camera* pCam ) {
  const unsigned int k_softwareTrigger = 0x62C;
  Error error;
  unsigned int regVal = 0;
  do {
    error = pCam->ReadRegister( k_softwareTrigger, &regVal );
    if (error != PGRERROR_OK) {
      PrintError( error );
      return false;
    }
  }
  while ( (regVal >> 31) != 0 );
  return true;
}

bool FireSoftwareTrigger( Camera* pCam ) {
  const unsigned int k_softwareTrigger = 0x62C;
  const unsigned int k_fireVal = 0x80000000;
  Error error;
  error = pCam->WriteRegister( k_softwareTrigger, k_fireVal );
  if (error != PGRERROR_OK) {
    PrintError( error );
    return false;
  }
  return true;
}
return true;

} 

int main(int argc, char** argv) {
int k_numImages = 100;
int timeex;
struct timeb startT, finishT;
unsigned int seconds, milliseconds;

// Define condiciones de las capturas de la camara
char name[256]= "multiespectral";
char format[256]="pgm";
printf("\n");
printf("TESIS_Anthony_Camara_Multiespectral_2014\n");
#undef SOFTWARE_TRIGGER_CAMERA
Error error;
BusManager busMgr;
unsigned int numCameras;
error = busMgr.GetNumOfCameras(&numCameras);
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
printf( "Number of cameras detected: %u\n", numCameras );
if ( numCameras < 1 ) {
    printf( "Insufficient number of cameras... exiting\n" );
    return-1;
}
PGRGuid guid;
error = busMgr.GetCameraFromIndex(0, &guid);
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
Camera cam;
// Conectarse a la camara
error = cam.Connect(&guid);
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
printf("Camara conectada\n");
// Encender la camara
const unsigned int k_cameraPower = 0x610;
const unsigned int k_powerVal = 0x80000000;
error = cam.WriteRegister( k_cameraPower, k_powerVal );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
const unsigned int millisecondsToSleep = 100;
unsigned int regVal = 0;
unsigned int retries = 10;

// Esperar que la camara termine de encender
do {
    #if defined(WIN32) || defined(WIN64)
        Sleep(millisecondsToSleep);
    #else
        usleep(millisecondsToSleep * 1000);
    #endif
    error = cam.ReadRegister(k_cameraPower, &regVal);
    if (error == PGRERROR_TIMEOUT)
    }
else if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
    }
    retries--;
} while ((regVal & k_powerVal) == 0 && retries > 0);

// Verificar errores de limites de tiempo
if (error == PGRERROR_TIMEOUT) {
    PrintError( error );
    return-1;
    }

// Obtener la informacion de la camara
CameraInfo camInfo;
error = cam.GetCameraInfo(&camInfo);
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
    }

#ifndef SOFTWARE_TRIGGER_CAMERA
    // Verificar trigger externo
    TriggerModeInfo triggerModeInfo;
    error = cam.GetTriggerModeInfo( &triggerModeInfo );
    if (error != PGRERROR_OK) {
        PrintError( error );
        return-1;
        }
        if ( triggerModeInfo.present != true ) {
            printf( "Camera does not support external trigger! Exiting...\n" );
            return-1;
        }
    #endif
    // Obtener configuracion actual del trigger
    TriggerMode triggerMode;
    error = cam.GetTriggerMode( &triggerMode );
    if (error != PGRERROR_OK) {
        PrintError( error );
        return-1;
        }
        // Configurar la camara al trigger modo 0
        triggerMode.onOff = true;
        triggerMode.mode = 0;
triggerMode.parameter = 0;

#else
// Source 7 es un trigger por software
triggerMode.source = 7;
#endif

// Disparar la cámara de forma externa por el pin 2.
triggerMode.source = 2;
triggerMode.polarity = 0;
#endif

error = cam.SetTriggerMode( &triggerMode );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}

bool retVal = PollForTriggerReady( &cam );
if ( !retVal ) {
    printf("\nError polling for trigger ready!\n");
    return-1;
}

// Obtener configuración de la cámara
FC2Config config;
error = cam.GetConfiguration( &config );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}

// Configurar tiempo de grabación
config.grabTimeout = 50000;

// Set the camera configuration
error = cam.SetConfiguration( &config );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}

// Camara esta lista para capturar imágenes
error = cam.StartCapture();
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}

#endif SOFTWARE_TRIGGER_CAMERA
if ( !CheckSoftwareTriggerPresence( &cam ) ) {
    printf( "SOFT_ASYNC_TRIGGER not implemented on this camera! Stopping application\n");
    return-1;
}
#else
printf( "Waiting for Pulse at %d.\n", triggerMode.source );
#endif

Image image;
for ( int imageCount=0; imageCount < k_numImages; imageCount++ ) {

Tesis publicada con autorización del autor
No olvide citar esta tesis
ftime(&startT);
PollForTriggerReady( &cam);

// Graba imagen
error = cam.RetrieveBuffer( &image );
if (error != PGRERROR_OK) {
    PrintError( error );
    break;
}
printf(".\n");
// Crea una imagen convertida
Image convertedImage;
// Convierte la imagen en bruto
error = image.Convert( PIXEL_FORMAT_MONO8, &convertedImage );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
// Crea un nombre unico para las imagenes
char filename[512];
sprintf( filename, "%s%d.%s", name, imageCount+1,format);

if (imageCount >= 0) {
    error = convertedImage.Save( filename );
    if (error != PGRERROR_OK) {
        PrintError( error );
        return-1;
    }
    printf( "Picture #%d saved!!\n",imageCount+1);
    ftime(&finishT);
    seconds = finishT.time - startT.time - 1;
    milliseconds = (1000 - startT.millitm) + finishT.millitm;
    timeex = milliseconds + seconds * 1000;
    printf("Esta foto demoro %d \n",timeex);
}

// Deja de capturar imagenes
error = cam.StopCapture();
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
// Apaga el trigger
triggerMode.onOff = false;
error = cam.SetTriggerMode( &triggerMode );
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
// Desconecta la cámara
error = cam.Disconnect();
if (error != PGRERROR_OK) {
    PrintError( error );
    return-1;
}
return 0;
}
7.1.1 Standard External Trigger (Mode 0)

Trigger Mode 0 is best described as the standard external trigger mode. When the camera is put into Trigger Mode, the camera starts integration of the incoming light from the external trigger, but the external trigger is not used to control the exposure. The exposure is determined by the Shutter value. It is not possible to trigger the camera at a specific time using Trigger Mode 0, however, this is possible using Overlapped Exposure Readout Trigger (Mode 14).

![Diagram of Trigger Mode 0](image)

**Figure 7.1: Trigger Mode 0 ("Standard External Trigger Mode")**

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence</td>
<td>[0] Presence, 1</td>
</tr>
<tr>
<td>ON</td>
<td>[6] ON, 1</td>
</tr>
<tr>
<td>Polarity</td>
<td>[7] Polarity, Low/High</td>
</tr>
<tr>
<td>Source</td>
<td>[8-10] Source, GPIO Pin</td>
</tr>
<tr>
<td>Mode</td>
<td>[12-15] Mode, Trigger_Mode_0</td>
</tr>
<tr>
<td>Parameter</td>
<td>[20-31] Parameter, None</td>
</tr>
</tbody>
</table>

Register **TRIGGER_MODE**: 830h
7.1.2 BulbShutterTrigger (Mode1)

Also known as BulbShutter, the camera starts integration of the incoming light from the trigger input. Integration time is equal to low state time of the external trigger input.

Figure 7.2: TriggerMode1 ("BulbShutterMode")

<table>
<thead>
<tr>
<th>Registers</th>
<th>TRIGGER_MODE:830h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence</td>
<td>[0] 1</td>
</tr>
<tr>
<td>ON</td>
<td>[6] 1</td>
</tr>
<tr>
<td>Polarity</td>
<td>[7] Low/High</td>
</tr>
<tr>
<td>Source</td>
<td>[8-10] GPIO Pin</td>
</tr>
<tr>
<td>Value</td>
<td>[11] Low/High</td>
</tr>
<tr>
<td>Mode</td>
<td>[12-15] Trigger_Mode_1</td>
</tr>
<tr>
<td>Parameter</td>
<td>[20-31] None</td>
</tr>
</tbody>
</table>
7.1.3 SkipFramesTrigger (Mode 3)

TriggerMode3 allows the user to put the camera into a mode where it only transmits one out of N specific images. This is an internal trigger mode that requires no external parameters. In TriggerMode3, the camera will issue a trigger internally at a cycle time that is N times greater than the current frame rate. As with TriggerMode0, the Shutter value describes integration time.

![Diagram showing Trigger, Sensor exposure, and Sensor readout]

Figure 7.3: TriggerMode3 (“SkipFramesMode”)

<table>
<thead>
<tr>
<th>Registers—TRIGGER_MODE: 0x30h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence [0]</td>
</tr>
<tr>
<td>ON [6]</td>
</tr>
<tr>
<td>Polarity [7]</td>
</tr>
<tr>
<td>Source [8-10]</td>
</tr>
<tr>
<td>Value [11]</td>
</tr>
<tr>
<td>Parameter [20-31]</td>
</tr>
</tbody>
</table>
7.1.4 Overlapped Exposure Readout Trigger (Mode 14)

TriggerMode14 is a vendor:unique:trigger:mode that is very similar to TriggerMode0, but allows for triggering at full frame rates. This mode works well for users who want to drive exposure restart with an external event. However, users who need a precise exposure restart should use TriggerMode0.

In the figure below, the trigger may be overlapped with the readout. In continuous-shot (free-running) mode, if the trigger arrives after readout is complete, it will start as quickly as the imaging area can be cleared. If the trigger arrives before the end of shutter integration, the trailing edge of the trigger signal is sampled on the re-clock edge to determine if the trigger arrives while the image is still being read out of the sensor. The start of exposure will be delayed until the next opportunity to clear the imaging area without injecting noise into the output image. The end of exposure cannot be before the end of the previous image readout. Therefore, exposure restart may be delayed to ensure this, which means priority is given to maintaining the proper exposure time instead of the trigger start.

![TriggerMode14 Diagram](image)

Figure 7.4: TriggerMode14 (“Overlapped Exposure Readout Mode”)

<table>
<thead>
<tr>
<th>Registers — TRIGGER MODE: 830h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence</td>
</tr>
<tr>
<td>ON</td>
</tr>
<tr>
<td>Polarity</td>
</tr>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Mode</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
</tbody>
</table>
7.2 External Trigger Timing

The time from the external trigger firing to the start of shutter is shown below:

1. Trigger Pulse
2. Propagation Delay
3. Exposure Time
4. Sensor Readout
5. Data Transfer

Figure 7.5: External trigger timing characteristics

It is possible for users to measure this themselves by configuring one of the camera's GPIO pins to output a strobe (see Programmable Strobe Output) and connecting an oscilloscope to the input trigger pin and the output strobe. The camera will strobe each time an image acquisition is triggered; the start of the strobe pulse represents the start of exposure.

7.3 Camera Behavior Between Triggers

When operating in external trigger mode, the camera clears charges from the sensor at the horizontal pixel clock determined by the current frame rate. For example, if the camera is set to 10 FPS, charges are cleared off the sensor at 100 kHz. An action takes place following shutter integration: external triggers received at that point the horizontal clearing operation is performed and a final clearing of the entire sensor is performed prior to shutter integration and transmission.
7.4 Changing Video Modes While Triggering

You can change the video format and mode of the camera while operating in trigger mode. Whether the new mode that is requested takes effect in the triggered image depends on the timing of the request and the trigger mode effect. The diagram below illustrates the relationship between triggering and changing video modes.

![Diagram showing relationship between triggering and video mode change](image)

**Figure 7.6: Relationship Between External Triggering and Video Mode Change Request**

When operating in Standard External Trigger (Mode 0) or in Bulb Shutter Trigger (Mode 1), video mode changes made before point A on the diagram are honored in the triggered image. The attempt to change a request made after point A in the triggered sequence may or may not succeed, in which case the request is honored on the triggered image for the exposure time before point A. The result that, in most cases, the reordering delay for a triggered image for a video mode request, ms, before the configuration period takes effect.
7.5 AsynchronousSoftwareTriggering

Shutter integration can be initiated by a software trigger via SOFTWARE_TRIGGER/6ZCh.

The time from software trigger initiation to the start of shutter is shown below.

![Software trigger diagram](image)

Figure 7.7: Software trigger timing

The time from when the software trigger is written on the camera to when the start of integration occurs can only be approximated. We then add the trigger latency time from the trigger pulse on the start of integration to this.

*This timing is solely from the camera perspective. It is virtually impossible to predict timing from the user perspective or to tolerate the processing of commands on the host PC.*

7.6 IsochronousDataTransfer

Isochronous transmission is the transfer of image data from the camera to the PC in a continuous stream that is regulated by an internal clock. Isochronous transfer ensures the delivery of image data in a timely and consistent manner.

For more information about isochronous transmission, packet formats, and bandwidth requirements, refer to the PointGreyPacketFormat.
DISEÑO CHASIS

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERÍA – ESPECIALIDAD: ING. ELECTRÓNICA

MÉTODO DE PROYECCIÓN

20100866 GONZALES HUISA, ANTHONY RICARDO
FECHA: 12/12/2014
LAMINA: A4

ANEXO 3