```
#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;
}

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;

#ifdef SOFTWARE_TRIGGER_CAMERA
// Source 7 es un trigger por software
triggerMode.source = 7;
#else
// Disparar la camara 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 configuracion de la camara
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 imagenes
error = cam.StartCapture();
if (error != PGRERROR_OK) {
PrintError( error );
return-1;
}
#ifdef 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++ ) {
```

```
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 camara
error = cam.Disconnect();
if (error != PGRERROR_OK) {
PrintError( error );
return-1;
}
return 0;
}
```

### 7.1.1 StandardExternalTrigger(Mode0)

TriggerMode0isbestdescribedasthestandardexternaltriggermode. Whenthecameraisputintotrigger Mode
thecamerastartsintegrationoftheincominglightfromtriggerinputfalling/risingedge.Theshuttervalue
describesintegrationtime.Noparameterisrequired.Thecameracanbetriggeredinthismodebyusingthe GPIO
asexternaltriggerorbyusingasoftwaretrigger.

Itisnotpossibletotriggerthecameraatfullframerateusing TriggerMode0;however,thisispossibleusing
OverlappedExposureReadoutTrigger(Mode14).

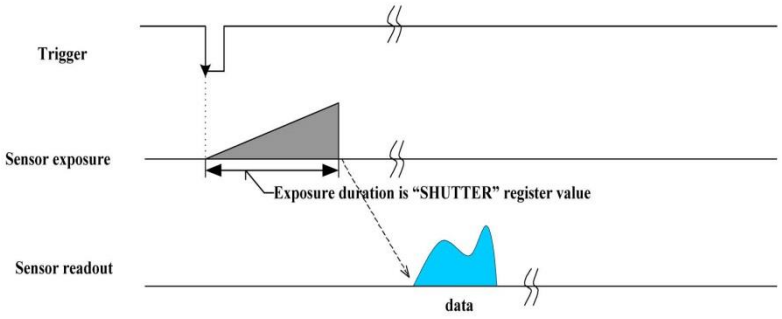



**Figure7.1:TriggerMode0("StandardExternalTriggerMode")**

| Registers—TRIGGER_MODE:830h | | |
|---|---|---|
| Presence | [0] | 1 |
| ON | [6] | 1 |
| Polarity | [7] | Low/High |
| Source | [8-10] | GPIO Pin |
| Value | [11] | Low/High |
| Mode | [12-15] | Trigger_Mode_0 |
| Parameter | [20-31] | None |

### 7.1.2 BulbShutterTrigger(Mode1)

AlsoknownasBulbShuttermode,thecamerastartsintegrationoftheincominglightfromtriggerinput.
Integrationtimeisequaltolowstatetimeoftheexternaltriggerinput.

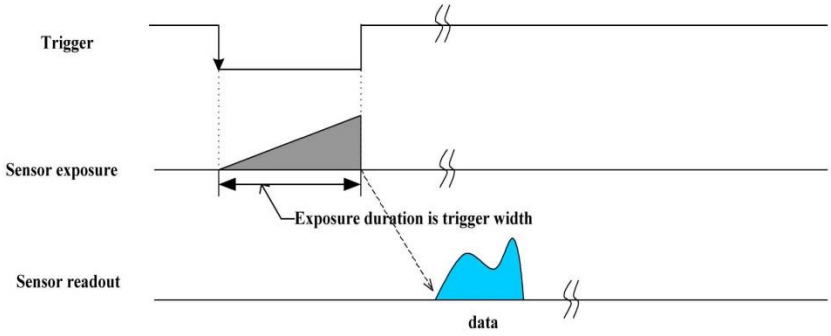



**Figure7.2:TriggerMode1(“BulbShutterMode”)**

| Registers—TRIGGER_MODE:830h | | |
|---|---|---|
| Presence | [0] | 1 |
| ON | [6] | 1 |
| Polarity | [7] | Low/High |
| Source | [8-10] | GPIO Pin |
| Value | [11] | Low/High |
| Mode | [12-15] | Trigger_Mode_1 |
| Parameter | [20-31] | None |

### 7.1.3 SkipFramesTrigger(Mode3)

TriggerMode3allowstheusertoputthecameraintoamodewherethecameraonlytransmitsoneoutofNspecifie
image3.ThisisaninternaltriggermodethatrequiresnoexternalacquisitionWhereNistheparametersetinthe
TriggerMode,thecamerawillissueatriggerinternallyatacycletimethatisNtimesgreaterthanthecurrentframe
rate. AswithTriggerMode0,theShuttervaluedescribesintegrationtime.

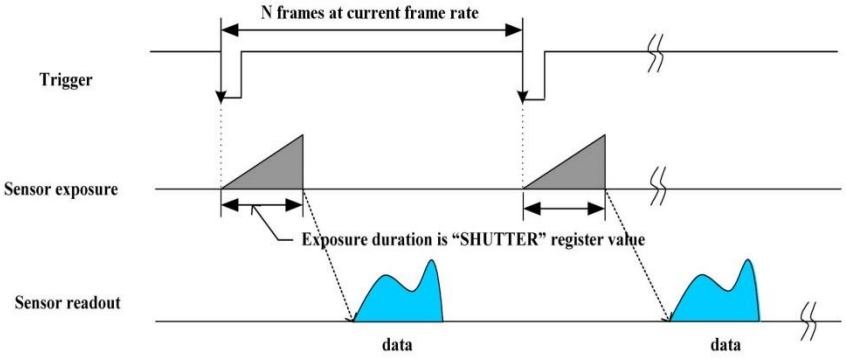



**Figure7.3:TriggerMode3("SkipFramesMode")**

| Registers—TRIGGER_MODE:830h | | |
|---|---|---|
| Presence | [0] | 1 |
| ON | [6] | 1 |
| Polarity | [7] | Low/High |
| Source | [8-10] | GPIO Pin |
| Value | [11] | Low/High |
| Mode | [12-15] | Trigger_Mode_3 |
| Parameter | [20-31] | /N1outofNimagesistransmitted. CycletimeNtimesgreaterthan currentframerate |

### 7.1.4 OverlappedExposureReadoutTrigger(Mode14)

TriggerMode14isavendor-uniquetriggermodethatisverysimilartoTriggerMode0,butallowsfortriggeringatf
framerates.Thismodeworkswellforuserswhowanttodriveexposurestartwithanexternalevent.However,us
whoneedapreciseexposurestartshoulduseTriggerMode0.

Inthefigurebelow,thetriggermaybeoverlappedwiththereadoutofthepreviousimage,continuousshot(free-
running)mode.Ifthetriggerarrivesafterreadoutiscomplete,itwillstartasquicklyastheimagingareacanbecle
Ifthetriggerarrivesbeforetheendofshutterintegration(whenthetriggeris*armed*),itisdropped.If the
triggerarriveswhiletheimageisstillbeingreadoutofthesensor,thestartofexposurewillbedelayeduntiltheney
opportunitytocleartheimagingareawithoutinjectingnoiseintotheoutputimage.Theendofexposurecannotc
beforetheendofthepreviousimagereadout.Therefore,exposurestartmaybedelayedtoensurethis,whichme
priorityisgiventomaintainingtheproperexposuretimeinsteadoftothetriggerstart.

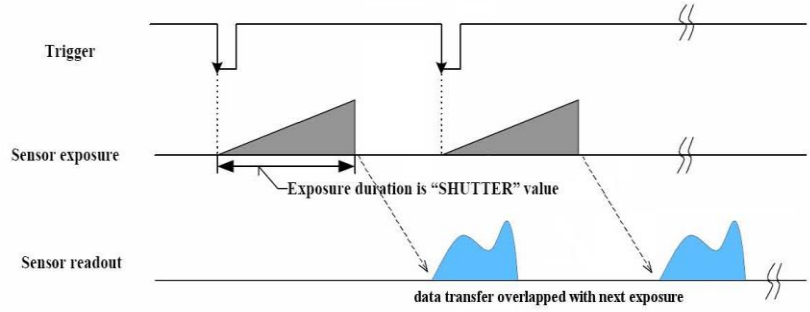



**Figure7.4:TriggerMode14("OverlappedExposure/ReadoutMode")**

| Registers—TRIGGER_MODE:830h | | |
|---|---|---|
| Presence | [0] | 1 |
| ON | [6] | 1 |
| Polarity | [7] | Low/High |
| Source | [8-10] | GPIO Pin |
| Value | [11] | Low/High |
| Mode | [12-15] | Trigger_Mode_14 |
| Parameter | [20-31] | None |

## 7.2 ExternalTriggerTiming

Thetimefromtheexternaltriggerfiringtothestartofshutterisshownbelow:

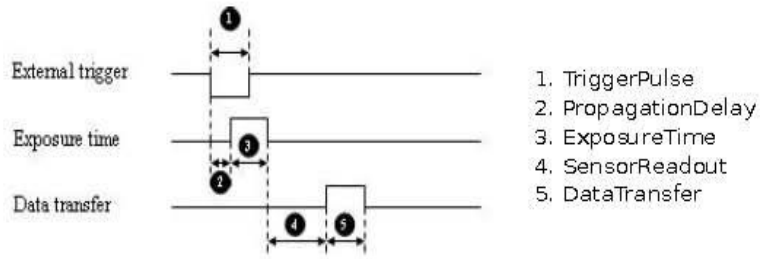



**Figure7.5:Externaltriggertimingcharacteristics**

Itispossibleforuserstomeasurethisthemselvesbyconfiguringoneofthecamera'sGPIOpinstooutputastrobep (seeProgrammableStrobeOutput)andconnectinganoscilliscopeuptotheinputtriggerpinandtheoutputstrol Thecamerawillstrobeeachtimeanimageacquisitionistriggered;thestartofthestrobepulserepresentsthesta exposure.

## 7.3 CameraBehaviorBetweenTriggers

Whenoperatinginexternaltriggermode,thecameraclearschargesfromthesensoratthehorizontalpixelclock determinedbythecurrentframerate.Forexample,ifthecameraissetto10FPS,chargesareclearedoffthesensc ahorizontalpixelclockrateof15KHz.Thisactiontakesplacefollowingshutterintegrationandtheexttriggeris received.Atthatpointthehorizontalclearingoperationisaborted,andafinalclearingoftheentiresensorisperformedpriortoshutterintegrationandtransmission.

## 7.4 ChangingVideoModesWhileTriggering

Youcanchangethevideoformatandmodeofthecamerawhileoperatingintriggermode.Whetherthenewmode thatisrequestedtakeseffectinthenexttriggeredimagedependsonthetimingoftherequestandthetriggermo effect.Thediagrambelowillustratestherelationshipbetweentriggeringandchangingvideomodes.

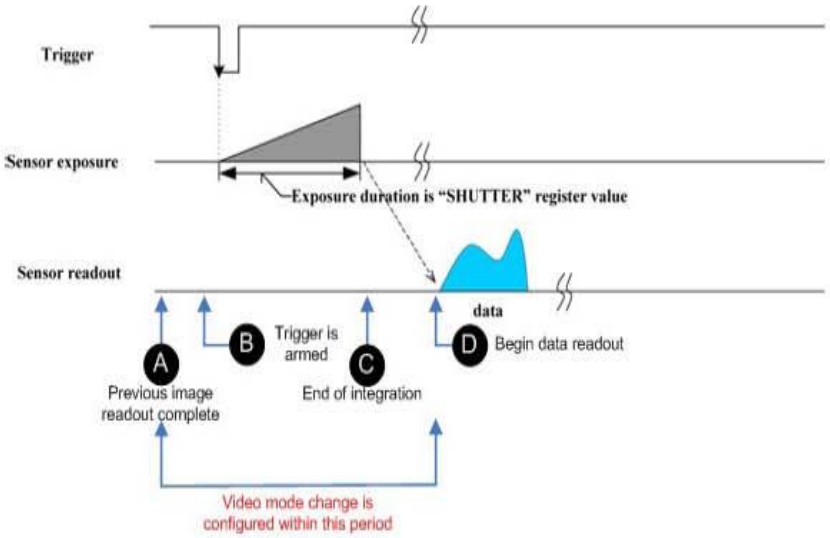



**Figure7.6:RelationshipBetweenExternalTriggeringandVideoModeChangeRequest**

WhenoperatinginStandardExternalTrigger(Mode0)orinBulbShutterTrigger(Mode1),videomodechangerec madebeforepointAonthediagramarehonoredinthenexttriggeredimage.Thecameramaywill requestmadeafterpointAinthenexttriggeredimage.Butthisattemptmayormaynotsucceed,inwhichcasethe requestishonoredonetriggeredimagelaterwhen beforepointA.Theresultisthat,inmostcases,thereisadelayofonetriggeredimageforavideomoderequest,ma beforetheconfigurationperiod,totakeeffect.

## 7.5 AsynchronousSoftwareTriggering

Shutterintegrationcanbeinitiatedbyasoftwaretriggerviа SOFTWARE_TRIGGER:62Ch.

Thetimefromasoftwaretriggerinitiationtothestartofshutterisshownbelow:

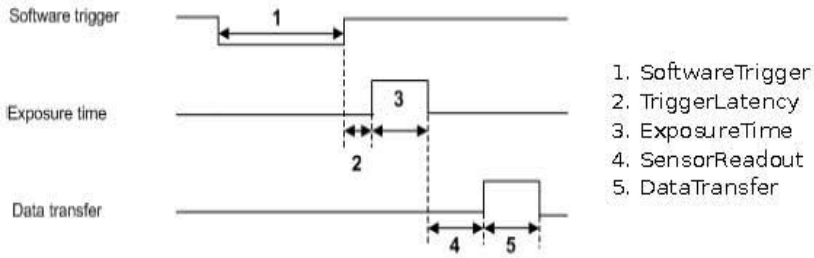



**Figure7.7:Softwaretriggertiming**

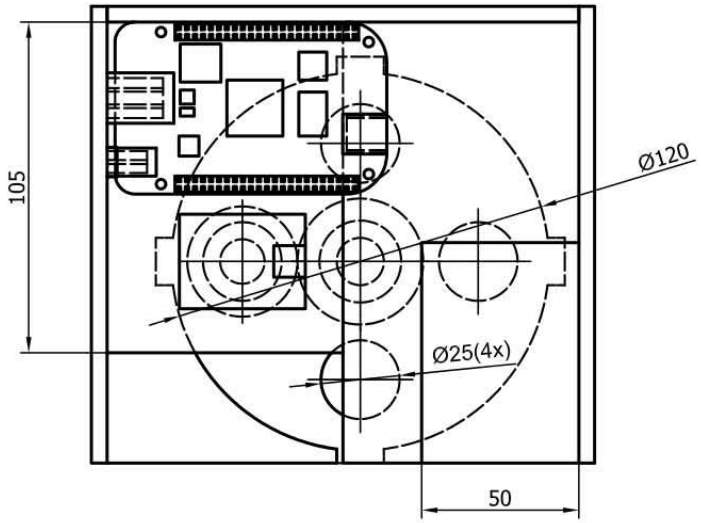
Thetimefromwhenthesoftwaretriggeriswrittenonthecameratowhenthestartofintegrationoccurscanonlyb approximated.Wethenaddthetriggerlatency(timefromthetriggerpulsetothestartofintegration)tothis.

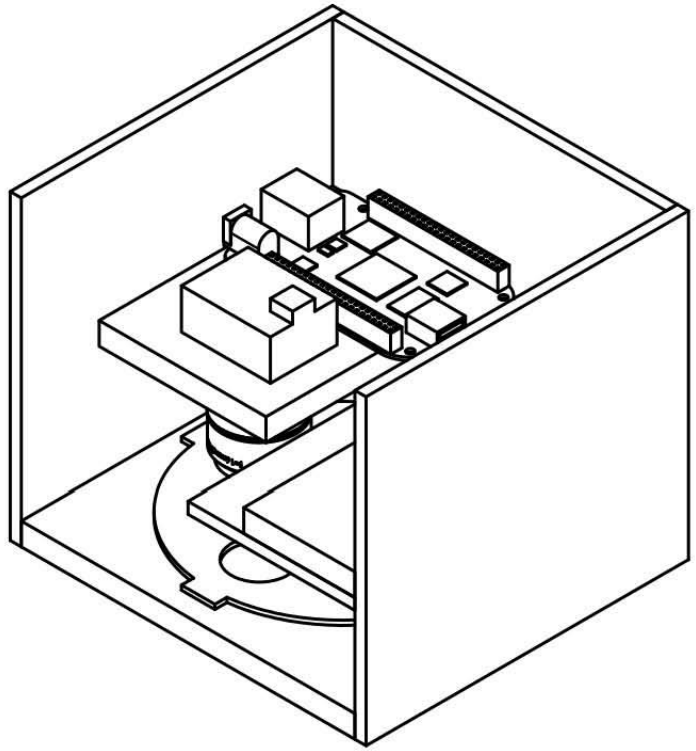
*Thistimingissolelyfromthecameraperspective.Itisvirtuallyimpossible topredictthetimingfromtheuserperspectiveduetolatencies inthe processingofcommandsonthehostPC.*

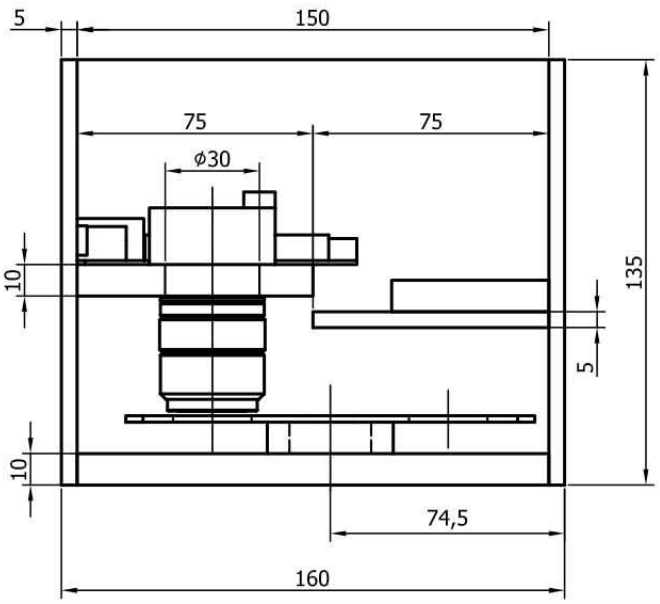
## 7.6 IsochronousDataTransfer

Isochronoustransmissionisthetransferofimageddatafromthecameratothe PC in a continual stream that is regu by an internal clock. Isochronous transfers on the bus guarantee timely delivery of data, but not necessarily integ data.

For more information about isochronous transmission, including packet formats and bandwidth requirements, see IsochronousPacketFormat.






| | | |
|---|---|---|
| PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU<br>FACULTAD DE CIENCIAS E INGENIERIA – ESPECIALIDAD: ING. ELECTRONICA | | |
| METODO DE PROYECCION | DISEÑO CHASIS | ESCALA<br>1:2 |
| 20100866 | GONZALES HUISA, ANTHONY RICARDO | FECHA:<br>12/12/2014 |
| ANEXO 3 | | LAMINA:<br>A4 |