The ISG sample GUI http://isgcameras.com/pdf-support/Allegro_USB3-Vision_Docs/IsgU3VGuiSample.zip provides example code that demonstrates the basic steps that are required in order to run the camera. Here is a brief overview of some basic features covered in the sample code, as well as some extensions to it.

Setting values for camera features

The "start" button shows the settings to bring up the camera in free running mode by calling function OnBnClickedButtonStartCamera. Inside the function there are examples of using the GenAPI interface for controlling features that have integer values:

```
    CCommandPtr theCmdPtr;
    CIntegerPtr theIntPtr;


    theIntPtr = ISGCamera->_GetNode("Width");
    theImageWidth = (uint32_t)theIntPtr->GetValue(false,false);

    theIntPtr = ISGCamera->_GetNode("Height");
    theImageHeight = (uint32_t)theIntPtr->GetValue(false,false);
```

As well as an enumeration value:

```
    theEnumPtr = ISGCamera->_GetNode("PixelFormat");
    pixVal =  theEnumPtr->GetIntValue();
```

Formats such as mono8 and rgb8 can be selected and properly be displayed by the sample viewer.

Not shown in the sample code are advanced features such as gain and exposure. These are floating point features and their nodes return GenICam defined CFloatPtr types as shown here:

```
        CFloatPtr theFloatVarPtr;

        theFloatVarPtr = myProjDialog->ISGCamera->_GetNode("Gain");
        theFloatVarPtr = myProjDialog->ISGCamera->_GetNode("ExposureTime");
```

The CFloatPtr class, provides the ability to query the minimum and maximum allowed values in much the same way as can be done for the integer type

```
        maxGain = theFloatVarPtr->GetMax();
        minGain = theFloatVarPtr->GetMin();
```

If the application were to set a value that is out of range, the GenAPI DLL will treat this as an exception condition. Such a condition can be captured as a text string. This exception handling

provides detailed information about the error and can be displayed using try-catch around your expression as shown here:

```
CFloatPtr theFloatVarPtr;
double integrationFloatVal = 1000000000.00;  // out of range value
theFloatVarPtr = ISGCamera->_GetNode("ExposureTime");

try
{
  theFloatVarPtr->SetValue(integrationFloatVal);
}
catch(exception& e)
{
  cout << e.what() << endl;
}
```

In this case, the out of range value resulted in the following report:

Value 1000000000.000000 must be smaller than or equal 5000000.000000. :
OutOfRangeException thrown in node 'ExposureTime' while calling 'ExposureTime.SetValue() ' (file 'FloatT.h', line 85)

Changing the camera mode from free running to software triggered

As mentioned earlier, the start button shows how to run the camera in its default, free running mode. Below is a sequence that demonstrates how to capture frames via software trigger following power up of the camera:

Note that the AcquisitionMode should be setup prior to executing AcquisitionStart, as this start command will lock some of the features of the camera that cannot be modified during running state.

```
CCommandPtr theCmdPtr;
CEnumerationPtr theEnumPtr;

int status;
unsigned char* buf = NULL;
bool triggerActive = true;
uint32_t frameSize = 2048 * 2048;
uint64_t acqMode;

theEnumPtr = ISGCamera->_GetNode("AcquisitionMode");
acqMode = 0; // single frame
theEnumPtr->SetIntValue(acqMode);

       status = theU3VPort->InitStream(frameSize,true,1,0);// 0 = infinite timeout
if (status == -1)
{
       cout << " init stream failed " << endl;
       return;
}

theCmdPtr = ISGCamera->_GetNode("AcquisitionStart");
theCmdPtr->Execute();  // start camera

theCmdPtr =  ISGCamera->_GetNode("TriggerSoftware");
    theCmdPtr->Execute(true);
```

```
status = theU3VPort->GetNextFrame(&buf,1000);// 1 second timeout
```

Once all frames are grabbed in a loop, and you are ready to stop camera, execute the following:

```
theU3VPort->HaltStream(triggerActive);

SetDlgItemText(IDC_BUTTON_START_CAMERA,"Start");

theCmdPtr = ISGCamera->_GetNode("AcquisitionStop");
theCmdPtr->Execute();  // stop camera
```