How to handle chunk data

Background
Chunk data is supported in recent JAI cameras (such as GO-2400-PGE and GO-5100-PGE) and in the JAI SDK (3.0.1 or later). Users can refer to the camera manual to check whether a particular camera model supports chunk data or not. This document describes how to handle chunk data in supported cameras using the JAI SDK and Control Tool.

Chunk Data
Chunk Data itself is defined in the GigE Vision and USB3 Vision standards. It has a broad meaning.

For JAI cameras, chunk data is metadata of each image captured by the camera. The metadata is attached to each image stream. By using this chunk data, users can get a variety of information when the image is captured, such as, ROI, exposure time, and so on.

Supported Chunk Data Items
The type of Chunk Data that is supported for a specific type of camera is listed in the camera's operating manual.

An example of this, which will be used for this document, can be found in the Appendix section of this Technical Note.

Enabling and Selecting Chunk Data
To attach metadata to an image stream, the user should enable chunk data for the camera and select the desired chunk data items.

1) Enabling Chunk Data

In the Control Tool section labeled "m) Chunk Data Control," set "Chunk Mode Active" to "True". This enables Chunk Data.

2) Selecting Chunk Data

In the "m) Chunk Data Control" section, use the "Chunk Selector" to select a chunk data item and then select "True" in the "Chunk Enable" field. This will enable the selected chunk data item. Repeat the process to enable additional items.

Figure 1 shows the enabling of "Offset X". "Offset Y" has also been enabled. Thus, when images are captured, the "Chunk Offset X" and "Chunk Offset Y" fields are updated.

Technical Note

How to handle chunk data

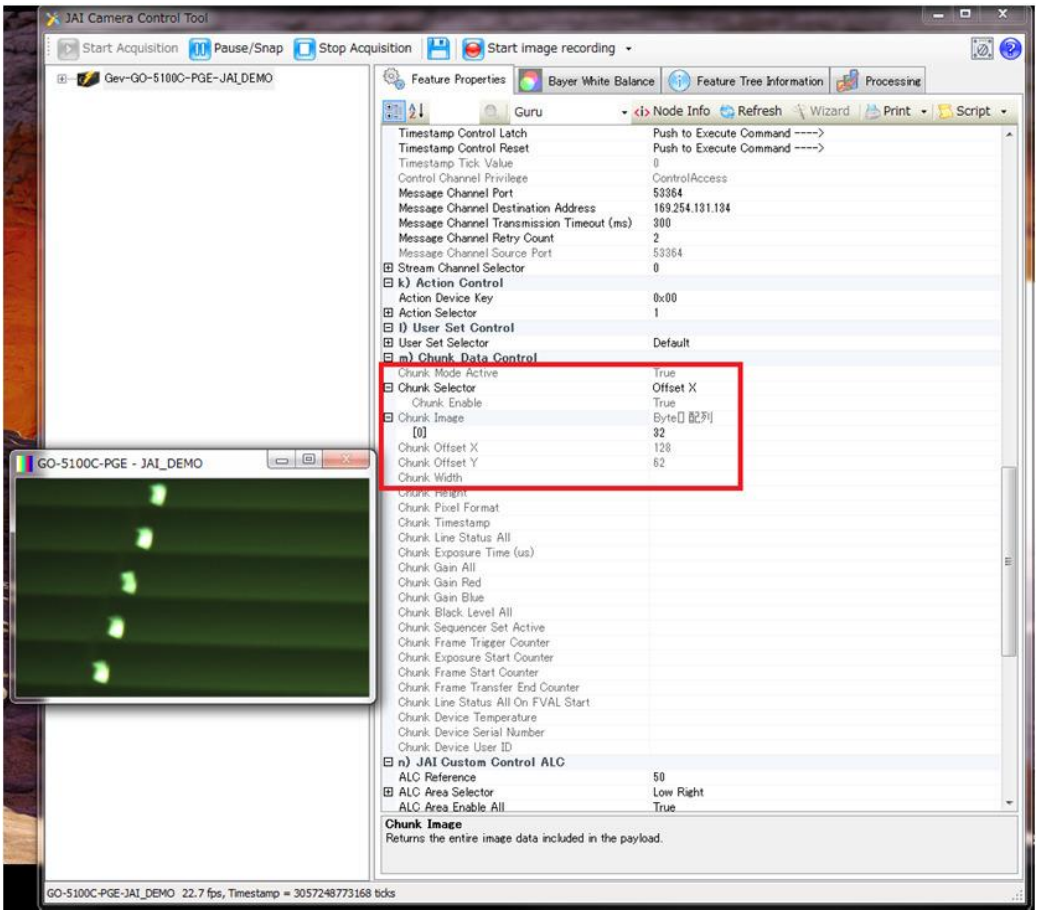

Figure 1 – Enabling and selecting Chunk Data

Acquiring chunk data in an application

In this document, we will introduce two methods: the “callback method” and the “expert method.”

Callback Method

This is the easiest way to acquire chunk data. The JAI SDK parses the chunk data from the data stream and attaches it to the camera’s node map. The user just needs to read the camera’s node map in a callback function.

The example code shown below can be found in the StreamCallbackSampleDig.cpp sample included with the JAI SDK:

```
void CStreamCallbackSampleDlg::StreamCBFunc(J_tIMAGE_INFO * pAqImageInfo)

{
        // Test of chunk data
        // Try to read one of the chunks from the test camera
        int64_t chunkDataValue = 0;
        J_STATUS_TYPE err = J_Camera_GetValueInt64(m_hCam, "ChunkTimestamp", &chunkDataValue);
        ....
}
```

As coded in the sample, just reading the camera node by JCameraGetValuexxx() is enough. The data in the camera node is valid until the callback function returns.

Expert Method

If a user wants the application to control the parsing of the stream data, the code can be written to directly access the stream data. A variety of API calls are supported in the JAI SDK, such as:

J_DataStream_GetBufferChunkData()

J_Camera_AttachChunkData()

J_Camera_UpdateChunkData()

J_Camera_DetachChunkData()

Sample code can be found in the StreamThreadSample. To use the Expert Method, the programmer should be familiar with the chunk data format.

Appendix

Supported chunk data in a GO-5100-PGE camera.

Feature	Genicam Name	Chunk ID	Data Size (bytes)	Interface	Notes
Image	ChunkImage	1000h	-	IRegister	
OffsetX	ChunkOffsetX	2000h	4	IInteger	
OffsetY	ChunkOffsetY	2001h	4	IInteger	
Width	ChunkWidth	2002h	4	IInteger	
Height	ChunkHeight	2003h	4	IInteger	
ExposureTime	ChunkExposureTime	2004h	4	IFloat	Timed only
GainAll	ChunkGainAll	2005h	4	IFloat	
GainRed	ChunkGainRed	2006h	4	IFloat	Color only
GainBlue	ChunkGainBlue	2007h	4	IFloat	Color only
BlackLevelAll	ChunkBlackLevelAll	2008h	4	IFloat	
BlackLevelRed	ChunkBlackLevelRed	2009h	4	IFloat	Color only
BlackLevelBlue	ChunkBlackLevelBlue	200Ah	4	IFloat	Color only
BinningHV_LUTEnable	ChunkBinningHV_LUTEnable	200Bh	4	IInteger	[0]: H-Binning Enable (Mono only) [1]: V-Binning Enable (Mono only) [2]: Binning MODE (Mono only) [3]: LUT Enable [4]: Sensor V-Binning (Mono only) [5]: Sensor H-Binning (Mono only)
SequencerSetActive	ChunkSequencerSetActive	200Ch	4	IInteger	
FrametriggerCounter	ChunkFrametriggerCounter	200Eh	4	IInteger	
ExposureStartCounter	ChunkExposureStartCounter	200Fh	4	IInteger	
FrameStartCounter	ChunkFrameStartCounter	2010h	4	IInteger	
FrameTransferEndCounter	ChunkFrameTransferEndCounter	2011h	4	IInteger	
PixelFormat	ChunkPixelFormat	2012h	4	Ienumeration	
LineStatusAll	ChunkLineStatusAll	2013h	4	IInteger	[0]: Line1 - TTL Out 1 [1]: Line2 - Opt Out 1 [2]: Line3 - Opt Out 2 [3]: Line4 - TTL In 1 [4]: Line5 - Opt In 1 [5]: Line6 - Opt In 2 [6]: Line7 - CC1/CXP In [7]: Line8 - TTL Out 2(Option) [8]: Line9 - TTL Out 3(Option) [9]: Line10 - TTL In 2(Option) [10]: Line11 - LVDS In(Option) [11]: TimeStampReset [12]: Nand0_In_1 [13]: Nand0_In_2 [14]: Nand1_In_1 [15]: Nand1_In_2
Timestamp	ChunkTimestamp	2014h	8	IInteger	
LineStatusAllOnFVALStart	ChunkLineStatusAllOnFVALStart	2016h	4	IInteger	same as LineStatusAll

DeviceSerialNumber	ChunkDeviceSerialNumber	2017h	64	IString	
DeviceUserID	ChunkDeviceUserID	2018h	64	IString	
DeviceTemperature	ChunkDeviceTemperature	2019h	4	IFloat	
UserData	ChunkUserData	201Ah	128	IRegister	

End.

Revision History

Revision	Date	Changes
1	2017/2/28	New release