Programmer´s Guide for LXC Chunk

AN201517/v1.0/ 2015-10-19

Description

The chunk provides additional information about the respective image (e.g. the timestamp). When using Baumer LXC cameras, this information is encoded into the last eight pixels of the image.

This document describes the process of requesting the chunk information.

Products

All Baumer LXC cameras support the chunk feature.

Preparation

The following requirements must be met in order to request the chunk:

- The Framegrabber SDK must be available for image acquisition.
- The *Camera Link® Config Tool* must be available to activate the chunks in the camera if the Framegrabber SDK does not support chunk activation

Supported Programming Languages

In this document, the process of requesting the chunk is described using a C++ example. Other programming languages can be used.

Table of Contents

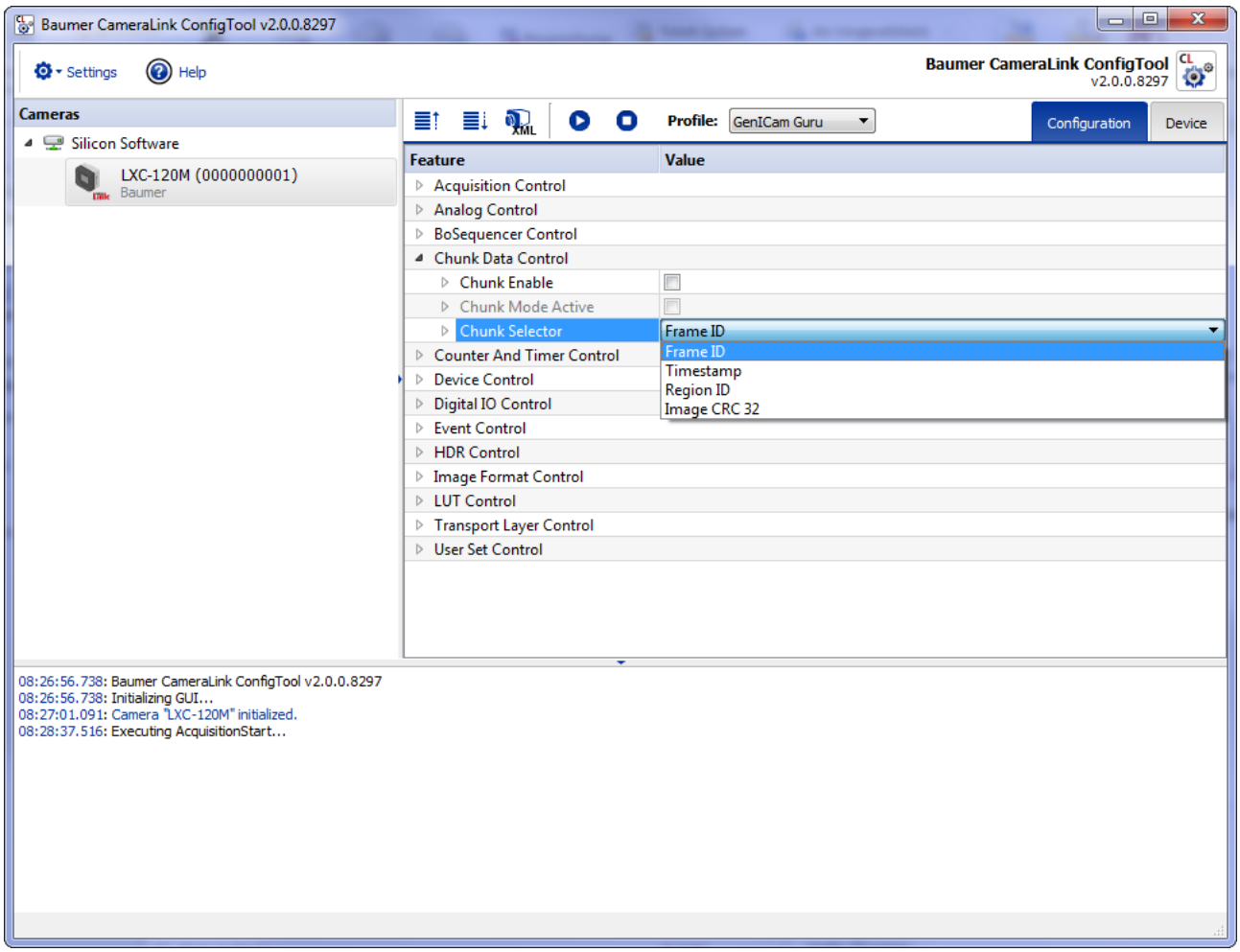
1 General Information

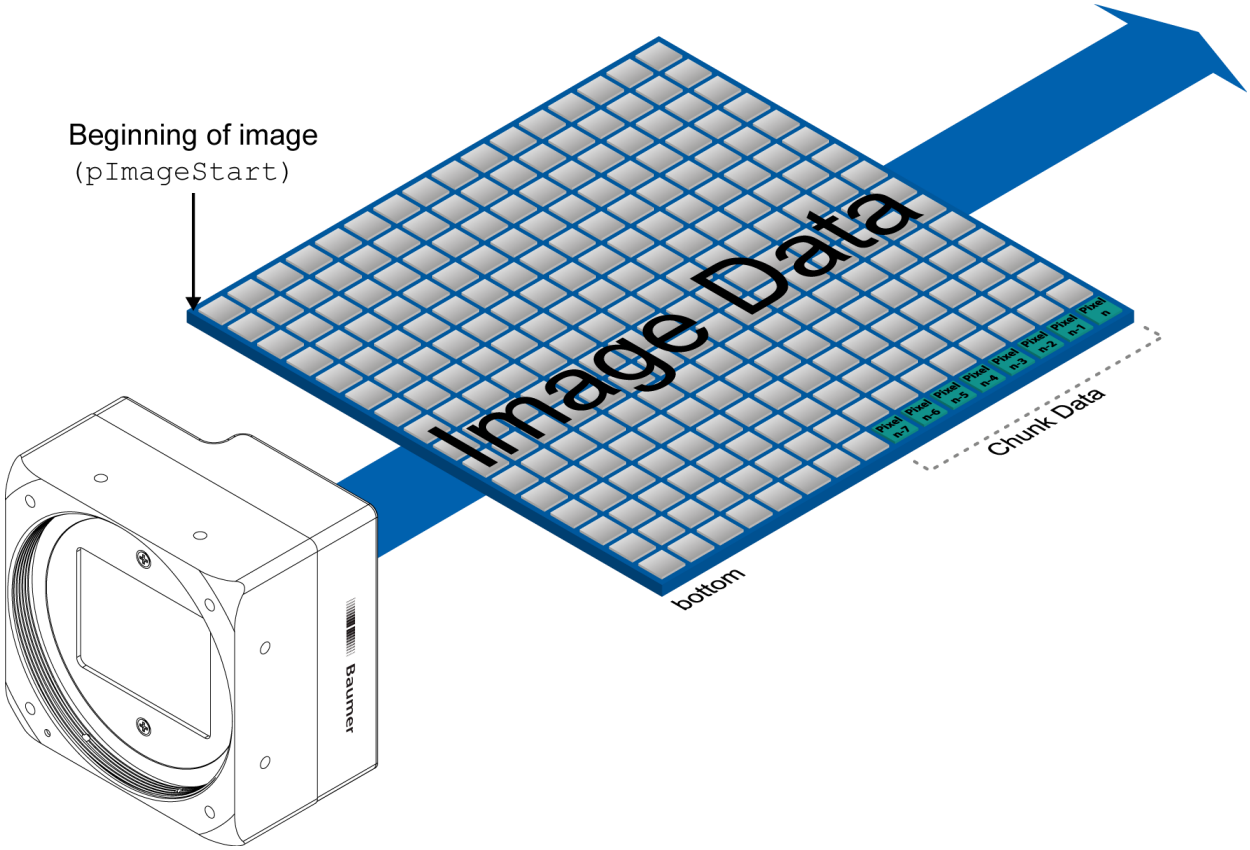
The chunk is encoded into the last eight pixels of each image and contains additional information about the image.

This additional information can include:

Information	Description
CRC32	Delivers a checksum which ensures that the image is transmitted correctly.
RegionID	Delivers the ID of the Region (Multi-ROI)from which the image originates.
FrameID	Delivers a unique ID for the image in the form of a number.
Timestamp	Delivers a timestamp for each image.

A maximum of two entries can be added to the chunk data via the "Chunk Selector" / "Chunk Selector" feature (see screenshot *Camera Link® ConfigTool v2*):





Chunk Data

The chunk data is assigned to the last 8 pixels as follows, depending on the settings:

Combination	Pixel n-7	Pixel n-6	Pixel n-5	Pixel n-4	Pixel n-3	Pixel n-2	Pixel n-1	Pixel n
1	Bit 0	Timestamp						Bit 63
2	Bit 0	CRC32		Bit 31	Bit 0	RegionID		Bit 31
3	Bit 0	CRC32		Bit 31	Bit 0	FrameID		Bit 31
4	Bit 0	FrameID		Bit 31	Bit 0	RegionID		Bit 31
5	Bit 0	RegionID		Bit 31	Bit 0	0		Bit 31
6	Bit 0	FrameID		Bit 31	Bit 0	0		Bit 31
7	Bit 0	CRC32		Bit 31	Bit 0	0		Bit 31

2 Request Chunk with C++

Example 1: Reading out chunk data

Mono 8: The camera transfers the image data in 8 bit format. The Framegrabber SDK delivers 8 bit image data to the application program:

```
__int64 GetChunkNodeValueMono8(unsigned char* pImageStart, unsigned int uNumberOfPixels)
{
	unsigned char*		pEndOfImage   = pImageStart + uNumberOfPixels;
	unsigned char*		pStartOfChunk = pEndOfImage - 8;
	__int64			iChunkValue   = *((__int64*)pStartOfChunk);

	return iChunkValue;
}
```

Mono 10: The camera transfers the image data in 10 bit format. The Framegrabber SDK delivers 16 bit image data to the application program:

```
__int64 GetChunkNodeValueMono10(unsigned short* pImageStart, unsigned int uNumberOfPixels)
{
	unsigned short*		pEndOfImage   = pImageStart + uNumberOfPixels;
	unsigned short*		pStartOfChunk = pEndOfImage - 8;

	// Every pixel is represented as 16 bit
	// The chunk data is stored in the upper 8 bits of every pixel
	// (which itself uses 10   bits)
	__int64			iChunkValue = 0;
	unsigned short*		pValuePtr = pStartOfChunk;
	for (int i = 0; i < 8; i++) {
		// Extract the next 16 bits and copy it to 64 bit
		__int64 iCurPixel = *pValuePtr;
		pValuePtr++;

		// As Mono10 consists of 10 bits per pixel
		// (where the upper 8 bits are used as chunk) we extract these 8 bits.
		iCurPixel = (iCurPixel >> 2) & 0xFF;

		// Copy these 8 bits to the extracted chunk value
		iChunkValue = (iCurPixel << (i * 8)) | iChunkValue;
	}

	return iChunkValue;
}
```

Once all of the chunk data has been extracted from the image (`iChunkValue`), the individual pieces of chunk data can be accessed as follows:

```
void SplitChunkInformation(__int64 iChunkValue, __int32* pUpperDWORD, __int32* pLowerDWORD)
{
	*pUpperDWORD = (iChunkValue & 0xffffffff00000000) >> 32;
	*pLowerDWORD = (iChunkValue & 0x00000000ffffffff);
}
```

Example 2: Calculating the CRC32 (Mono 8)

```
typedef unsigned int       uint32_t;

// ---------------------------------------------------------------------------------------
void CreateTable_CRC32(uint32_t* pTable)
{
	// Calculate CRC for all 256 bytes previously
	for (uint32_t iValue = 0; iValue <= 255; iValue++)
	{
		uint32_t iCrc32 = iValue;

		// CRC-32 Bitmask, inverted bit sequence
		const int CRC32MASK = 0xEDB88320;
		for (int i = 0; i < 8; i++)
		{
			if (iCrc32 & 1)
				iCrc32 = (iCrc32 >> 1) ^ CRC32MASK;
			else
				iCrc32 = (iCrc32 >> 1);
		}

		// save value
		pTable[iValue] = iCrc32;
	}
}

// ---------------------------------------------------------------------------------------
uint32_t crc32_Mono8(unsigned char* pImageStart, unsigned int uNumberOfPixels)
{
	// Create table for CRC32 (to improve performance while calculating CRC value)
	static uint32_t	s_CrcTable[256];
	static  bool		s_CrcTableInitialized = false;
	if (!s_CrcTableInitialized) {
		s_CrcTableInitialized = true;
		CreateTable_CRC32(s_CrcTable);
	}

	// ------------------------
	// Calculate CRC32 of image
	// ------------------------
	uint32_t iCrc32 = ~0;		// Initially set all bits to 1

	// -- Iterate over whole image (except last 8 pixels) --
	unsigned char*		pEndOfImage = pImageStart + uNumberOfPixels;
	unsigned char*		pStartOfChunk = pEndOfImage - 8;

	unsigned char* pCurPixel = pImageStart;
	while (pCurPixel < pStartOfChunk)
	{
		// Extract current pixel value
		unsigned char nValue = *pCurPixel;


		// Update CRC32 value
		iCrc32 = s_CrcTable[(iCrc32 ^ nValue) & 0xFF] ^ (iCrc32 >> 8);

		// Go to next pixel
		pCurPixel++;
	}
```

```
        // Finally invert all bits
        iCrc32 = ~iCrc32;

        return iCrc32;
}
```

Notice

For other image formats, the upper eight bits from each pixel are used and must be extracted for the CRC32 calculation.

The lower bits are not included in the CRC32 calculation.

3 Keep in mind/Special cases

-

4 Downloads

-

5 Support

In case of any questions or for troubleshooting please contact our support team.

Worldwide

Baumer Optronic GmbH
Badstrasse 30 · DE-01454 Radeberg
Phone +49 3528 4386 845
support.cameras@baumer.com

6 Disclaimer

All other product and company names mentioned are trademarks or registered trademarks of their respective owners.

Revisions in the course of technical progress and possible errors reserved.

Baumer
Passion for Sensors

Baumer Optronic GmbH
Badstrasse 30 · DE-01454 Radeberg
Phone +49 3528 4386 0 · Fax +49 3528 4386 86
sales@baumeroptronic.com · www.baumer.com