Frequently Asked Questions

How to use LUT feature.

#5400-0445 Version 1.0.1.1

TOSHIBA TELI CORPORATION

Document in Japanese

このドキュメントは BU/DU シリーズカメラに搭載されている LUT 機能の使用方法について記述したドキュメントです。日本語ドキュメントは英語ドキュメントの後にあります。

End of document in Japanese

This document describes how to use LUT feature in BU / DU series cameras.

1. What is LUT?

LUT (Look Up Table) feature is a feature to convert input signal value to arbitrary value using look up table inside the camera. User application can modify the contents of look up table freely. Typical applications of LUT are gamma correction, image thresholding, positive-negative reversal.

In monochrome camera, LUT Image data conversion will be adopted at the last stage of image signal processing inside the camera. Gamma correction and black level correction are applied to image sensor data before LUT Image data conversion.

In color camera, LUT Image data conversion is applied at the last stage of RGB color space signal processing. Saturation and hue adjustment is applied the output signal LUT Image data conversion.

Bit length of LUT input / output signal depends on camera model. There are 8bit LUT modes and 10 bit LUT models. Please check bit length of LUT input / output signal before using LUT feature.

2. Camera registers used for controlling LUT feature.

The following table shows camera registers used for controlling LUT feature.

Register name	Description
LUTEnable BU: 0x0020603C RegMapBU::ADR_LUT_ENABLE_I BG: 0x0000F000 RegMapBG::ADR_LUT_ENABLE_I	32bit integer register for enabling /disabling LUT feature. value 0: LUT feature is disabled. value 1: LUT feature is enabled.
LUTNumberOfElements BU: 0x002FFFEC BG: Not implemented.	32bit integer register indicating element count of LUT. The element count of LUT is fixed to 1024 in BG series camera.
LUTValueMax BU: 0x002FFFFC BG: Not implemented.	32bit integer register indicating the maximum value writable to 'LUTValue' registers. The element count of BG series camera LUT is 1024.
LUTValue BU: 0x00300000 BG: 0x00010000 RegMapBG::ADR_LUT_VALUE_TOP_I	The main body of look up table. (32bit integer register array)

The following tables show functions, methods, and node name for accessing registers in the above table.

< Controlling camera feature functions> (native C++)

Controlling camera feature functions	Target register	Remarks
GetCamLUTEnable()	LUTEnable	
SetCamLUTEnable()		
	LUTNumberOfElements	Use Cam_ReadReg() to read register value.
	LUTValueMax	Use Cam_ReadReg() to read register value.
GetCamLUTValue()	LUTValue	
SetCamLUTValue()		

<CameraControl class> (.NET framework)

Methods	Target register	Remarks
CameraControl.GetLUTEnable()	LUTEnable	
CameraControl.SetLUTEnable()		
	LUTNumberOfElements	*Use CameraDevice*.ReadRegister() to read register value.
	LUTValueMax	*Use CameraDevice*.ReadRegister() to read register value.
CameraControl.GetLUTValue()	LUTValue	
CameraControl.SetLUTValue()		

<GenICam functions and methods>

Node name	Node type	Target register
“LUTEnable” (XmlFeatures::XF_ID_LUT_ENABLE)	Boolean	LUTEnable register.
“LUTIndex” (XmlFeatures::XF_ID_LUT_INDEX)	Integer	Internal variable for controlling index of LUT to read or write. Application can access to LUTNumberOfElements register by calling Nd_GetIntMax() function to this node.
“LUTValue” (XmlFeatures::XF_ID_LUT_VALIUE)	Integer	Node for accessing to a register in “LUTValue” register array. Application can access to a register specified by LUTIndex node value through this node.

3. Example code for setting LUT data.

This chapter shows example codes for creating gamma correction table in camera LUT whose gamma parameter value is 2.2 and enabling the LUT feature. The argument "`iLenLUT`" is element count of conversion table, "`uiValMax`" is the maximum value allowed in the conversion table.

Error handling codes are omitted in the following sample code to make the procedure of handling LUT feature easier to understand.

3.1. Example code using camera controlling functions in TeliCamSDK

```
#include <math.h>
#include "TeliCamAPI.h"

using namespace Teli;

bool WriteLUT(CAM_HANDLE hCam, CAM_TYPE eCamType, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;
    // Gets length of look up table in the camera.
    uint32_t uiLenTable = 1024;
    if (CAM_TYPE_U3V == eCamType)
    {
        uiSts = Cam_ReadReg(hCam, 0x002FFFEC, 1, &uiLenTable);
    }

    // Checks if the camera has enough LUT registers to create specified table.
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // Specified table length is too long.
        return TRUE;
    }

    // Calculates data of gamma correction table.
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // Casts iValMax to double type value.
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // Writes the gamma correction table data to LUTValue registers.
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        uiSts = SetCamLutValue(hCam, uiIndex, pauiVal [uiIndex]);
    }

    // Releases temporary variables.
    delete[] pauiVal;

    // Enables LUT feature.
    uiSts = SetCamLutEnable(hCam, true);

    return FALSE;
}
```


3.2. Code using GenICam functions in TeliCamSDK PkgVer3.0.0.1 or later.

```
#include <math.h>
#include "TeliCamAPI.h"
#include "XmlFeatures.h"

using namespace Teli;

bool WriteLUTGen(CAM_HANDLE hCam, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;

    // Gets length of look up table in the camera.
    int64_t lltmp = 0;
    uint32_t uiLenTable = 1024;
    //  Gets the maximum value of LUTIndex node object.
    uiSts = GenApi_GetIntMax(hCam, XmlFeatures::XF_ID_LUT_INDEX, &lltmp);
    uiLenTable = static_cast<uint32_t>(lltmp) + 1;

    // Checks if the camera has enough LUT registers to create specified table.
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // Specified table length is too long.
        return TRUE;
    }

    // Calculates data of gamma correction table.
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // Casts iValMax to double type value.
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // Writes the gamma correction table data to LUTValue registers.
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        // Specifies index of LUTValue registers to write value.
        uiSts = GenApi_SetIntValue(hCam, XmlFeatures::XF_ID_LUT_INDEX,
                                   static_cast<int64_t>(uiIndex));
        // Writes the gamma correction table data to LUTValue register.
        uiSts = GenApi_SetIntValue(hCam, XmlFeatures::XF_ID_LUT_VALUE,
                                    static_cast<int64_t>( pauiVal[uiIndex]));
    }

    // Releases temporary variables.
    delete[] pauiVal;

    // Enables LUT feature.
    uiSts = GenApi_SetBoolValue(hCam, XmlFeatures::XF_ID_LUT_ENABLE, true);

    return FALSE;
}
```


3.3. Code using GenICam functions in TeliCamSDK PkgVer2.3.0.1 or earlier

```
#include <math.h>
#include "TeliCamAPI.h"
#include "XmlFeatures.h"

using namespace Teli;

bool WriteLUTGen(CAM_HANDLE hCam, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;
    CAM_NODE_HANDLE hNode = NULL;
    CAM_NODE_HANDLE hValNode = NULL;

    // Gets length of look up table in the camera.
    int64_t lltmp = 0;
    uint32_t uiLenTable = 1024;
    //  Gets LUTIndex node object.
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_INDEX, &hNode);
    //  Gets the maximum value of LUTIndex node object.
    uiSts = Nd_GetIntMax(hCam, hNode, &lltmp);
    uiLenTable = static_cast<uint32_t>(lltmp) + 1;

    // Checks if the camera has enough LUT registers to create specified table.
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // Specified table length is too long.
        return TRUE;
    }

    // Calculates data of gamma correction table.
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // Casts iValMax to double type value.
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // Writes the gamma correction table data to LUTValue registers.
    // Gets LUTIndex node object.
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_INDEX, &hNode);
    // Gets LUTValue node object.
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_VALUE, &hValNode);
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        // Specifies index of LUTValue registers to write value.
        uiSts = Nd_SetIntValue(hCam, hNode, static_cast<int64_t>(uiIndex));
        // Writes the gamma correction table data to LUTValue register.
        uiSts = Nd_SetIntValue(hCam, hValNode,
                                    static_cast<int64_t>( pauiVal[uiIndex]));
    }

    // Releases temporary variables.
    delete[] pauiVal;

    // Gets LUTEnable node object.
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_ENABLE, &hNode);
    // Enables LUT feature.
    uiSts = Nd_SetBoolValue(hCam, hNode, true);

    return FALSE;
}
```


1. LUT とは

LUT(Look Up Table)機能は、カメラ内蔵のルックアップテーブルを使用して入力画像信号値を任意の値に変換する機能です。ルックアップテーブルの内容はユーザが自由に設定できます。LUT 機能を使用すると、任意補正量でのガンマ補正、二値化、ネガポジ変換など、さまざまな画像変換を行うことができます。

LUT を使用した画像変換処理はモノクロカメラの場合はガンマ補正などの処理を行った後の最終段で処理されます。カラーカメラの場合は Bayer RGB 変換などの処理を行った後、Saturation と Hue の補正処理を行う前の段階に RGB 画像空間上で処理されます。

LUT の入出力信号のビット数はカメラの機種によって異なり、8bit の機種と 10bit の機種があります。LUT 機能を使用する場合はカメラの LUT 入出力ビット数を確認して使用してください。

2. LUT の制御に使用するカメラレジスタ

LUT の制御に使用するカメラレジスタは以下の表のとおりです。

レジスタ名	内容
LUTEnable BU: 0x0020603C RegMapBU::ADR_LUT_ENABLE_I BG: 0x0000F000 RegMapBG::ADR_LUT_ENABLE_I	LUT 機能の有効無効を設定する 32bit 整数型レジスタです。 値が 1 のとき LUT 有効、0 のとき無効になります。
LUTNumberOfElements BU: 0x002FFFEC BG: レジスタなし	LUT のテーブル要素数を示す 32bit 整数型レジスタです。 BG カメラの場合は 1024 要素に固定されているので、このレジスタは存在しません。
LUTValueMax BU: 0x002FFFFC BG: レジスタなし	LUT のテーブルに設定できる値の最大値を示す 32bit 整数型レジスタです。 BG カメラにはこのレジスタは存在しません。
LUTValue BU: 0x00300000 BG: 0x00010000 RegMapBG::ADR_LUT_VALUE_TOP_I	LUT のテーブルデータ 32bit 整数値配列レジスタです。

TeliCamSDK のカメラ制御関数、GenICam 関数では以下の関数で上記レジスタの読み書きを行うことができます。

【カメラ制御関数】(native C++)

カメラ制御関数	読み書き対象レジスタ	備考
GetCamLUTEnable()	LUTEnable	
SetCamLUTEnable()		
	LUTNumberOfElements	Cam_ReadReg()で読み込み可。
	LUTValueMax	Cam_ReadReg()で読み込み可。
GetCamLUTValue()	LUTValue	
SetCamLUTValue()		

【CameraControl クラス】(.NET framework)

メソッド	読み書き対象レジスタ	備考
CameraControl.GetLUTEnable()	LUTEnable	
CameraControl.SetLUTEnable()		
	LUTNumberOfElements	*CameraDevice*.ReadRegister()メソッドで読み込み可。
	LUTValueMax	*CameraDevice*.ReadRegister()メソッドで読み込み可。
CameraControl.GetLUTValue()	LUTValue	
CameraControl.SetLUTValue()		

【GenICam 関数】

ノード名	ノード型	読み書き対象レジスタ
“LUTEnable” (XmlFeatures::XF_ID_LUT_ENABLE)	Boolean	LUTEnable
“LUTIndex” (XmlFeatures::XF_ID_LUT_INDEX)	Integer	LUTValue の添え字内部変数。 このノードに対し Nd_GetIntMax()を実行することにより LUTNumberOfElements レジスタにアクセス可。
“LUTValue” (XmlFeatures::XF_ID_LUT_VALIUE)	Integer	LUT テーブル値読み書き用ノード。 “LUValue”配列レジスタの LUTIndex ノードに指定したインデックスのレジスタに対して値の読み書きが可能。

3. LUT データ書き込みコード例

LUT にγ値 *2.2* のγ補正テーブルを作成し、LUT を有効にするコード例を掲載します。
引数の `iLenLUT` は実際に使用する変換テーブルの要素数、`uiValMax` はテーブル値の最大値です。
以下のコード例では処理手順をわかりやすくするために各種エラー処理を省略しています。

3.1. カメラ制御関数を使用したコード例

```
#include <math.h>
#include "TeliCamAPI.h"

using namespace Teli;

bool WriteLUT(CAM_HANDLE hCam, CAM_TYPE eCamType, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;
    // カメラに搭載されている LUT レジスタのサイズを取得
    uint32_t uiLenTable = 1024;
    if (CAM_TYPE_U3V == eCamType)
    {
        uiSts = Cam_ReadReg(hCam, 0x002FFFEC, 1, &uiLenTable);
    }

    // 引数に指定されたサイズのテーブルが作成可能か確認。
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // 引数に指定されたサイズが大きすぎるためテーブル作成不可。
        return TRUE;
    }

    // γ値 2.2 のγ補正テーブルデータ作成。
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // テーブル値最大値をダブル型にキャスト。
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // LUT テーブル書き込み
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        uiSts = SetCamLutValue(hCam, uiIndex, pauiVal [uiIndex]);
    }

    // LUT テーブルデータ解放
    delete[] pauiVal;

    // LUT 有効化
    uiSts = SetCamLutEnable(hCam, true);

    return FALSE;
}
```


3.2. GenICam 関数を使用したコード例 (TeliCamSDK PkgVer3.0.0.1 以降)

```
#include <math.h>
#include "TeliCamAPI.h"
#include "XmlFeatures.h"

using namespace Teli;

bool WriteLUTGen(CAM_HANDLE hCam, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;

    // カメラに搭載されている LUT レジスタのサイズを取得
    int64_t lltmp = 0;
    uint32_t uiLenTable = 1024;
    // LUTIndex ノードの最大値を取得
    uiSts = GenApi_GetIntMax(hCam, XmlFeatures::XF_ID_LUT_INDEX, &lltmp);
    uiLenTable = static_cast<uint32_t>(lltmp) + 1;

    // 引数に指定されたサイズのテーブルが作成可能か確認。
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // 引数に指定されたサイズが大きすぎるためテーブル作成不可。
        return TRUE;
    }

    // γ値 2.2 のγ補正テーブルデータ作成。
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // テーブル値最大値をダブル型にキャスト。
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // LUT テーブル書き込み
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        // テーブルインデックスの指定
        uiSts = GenApi_SetIntValue(hCam, XmlFeatures::XF_ID_LUT_INDEX,
                                   static_cast<int64_t>(uiIndex));
        // テーブル値の書き込み
        uiSts = GenApi_SetIntValue(hCam, XmlFeatures::XF_ID_LUT_VALUE,
                                   static_cast<int64_t>( pauiVal[uiIndex]));
    }
    // LUT テーブルデータ解放
    delete[] pauiVal;

    // LUT 有効化
    uiSts = GenApi_SetBoolValue(hCam, XmlFeatures::XF_ID_LUT_ENABLE, true);

    return FALSE;
}
```


3.3. GenICam 関数を使用したコード例 (TeliCamSDK PkgVer2.3.0.1 以前)

```
#include <math.h>
#include "TeliCamAPI.h"
#include "XmlFeatures.h"

bool WriteLUTGen(CAM_HANDLE hCam, int32_t iLenLUT, uint32_t uiValMax)
{
    CAM_API_STATUS uiSts = CAM_API_STS_SUCCESS;
    CAM_NODE_HANDLE hNode = NULL;
    CAM_NODE_HANDLE hValNode = NULL;

    // カメラに搭載されている LUT レジスタのサイズを取得
    int64_t lltmp = 0;
    uint32_t uiLenTable = 1024;
    // LUTIndex ノードを取得
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_INDEX, &hNode);
    // LUTIndex ノードの最大値を取得
    uiSts = Nd_GetIntMax(hCam, hNode, &lltmp);
    uiLenTable = static_cast<uint32_t>(lltmp) + 1;

    // 引数に指定されたサイズのテーブルが作成可能か確認。
    if(static_cast<uint32_t>(iLenLUT) < uiLenTable)
    {
        // 引数に指定されたサイズが大きすぎるためテーブル作成不可。
        return TRUE;
    }

    // γ値 2.2 のγ補正テーブルデータ作成。
    uint32_t *pauiVal = new uint32_t[uiLenTable];
    // テーブル値最大値をダブル型にキャスト。
    double dVMax = static_cast<double>(uiValMax);

    double dGamma = 2.2;
    double dGammaR = 1 / dGamma;

    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        pauiVal[uiIndex] = min(uiValMax,
                               (uint32_t)(pow(uiIndex / dVMax, dGammaR) * dVMax));
    }

    // LUT テーブル書き込み
    // LUTIndex ノード取得
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_INDEX, &hNode);
    // LUTValue ノード取得
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_VALUE, &hValNode);
    for(uint32_t uiIndex= 0; uiIndex < uiLenTable; ++uiIndex)
    {
        // テーブルインデックスの指定
        uiSts = Nd_SetIntValue(hCam, hNode, static_cast<int64_t>(uiIndex));
        // テーブル値の書き込み
        uiSts = Nd_SetIntValue(hCam, hValNode,
                                    static_cast<int64_t>( pauiVal[uiIndex]));
    }
    // LUT テーブルデータ解放
    delete[] pauiVal;

    // LUTEnable ノード取得
    uiSts = Nd_GetNode(hCam, XmlFeatures::XF_ID_LUT_ENABLE, &hNode);
    // LUT 有効化
    uiSts = Nd_SetBoolValue(hCam, hNode, true);

    return FALSE;
}
```

End of document in Japanese

4. Others

4.1. Revision History

Date	Version	Description
2016/06/13	1.0.0	Created the initial version
2018/11/20	1.0.1	Added example code using new GenICam functions

4.2. Disclaimer

The disclaimer of this document including example code is described in “License Agreement TeliCamSDK Eng.pdf” in TeliCamSDK installation folder.
Make sure to read this Agreement carefully before using it.
Refer to TeliCamSDK installation folder/Documents/License folder

