Frequently Asked Questions

How to use Sequential Shutter.

#5400-0439　Version 1.0.1.1

TOSHIBA TELI CORPORATION

Document in Japanese

このドキュメントはシーケンシャルシャッタ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 about how to use Sequential Shutter function.

1. What is Sequential Shutter?

Sequential Shutter is a function for capturing images of a target object under multiple camera settings, for example, changing exposure time, gain, image offset, and so on.

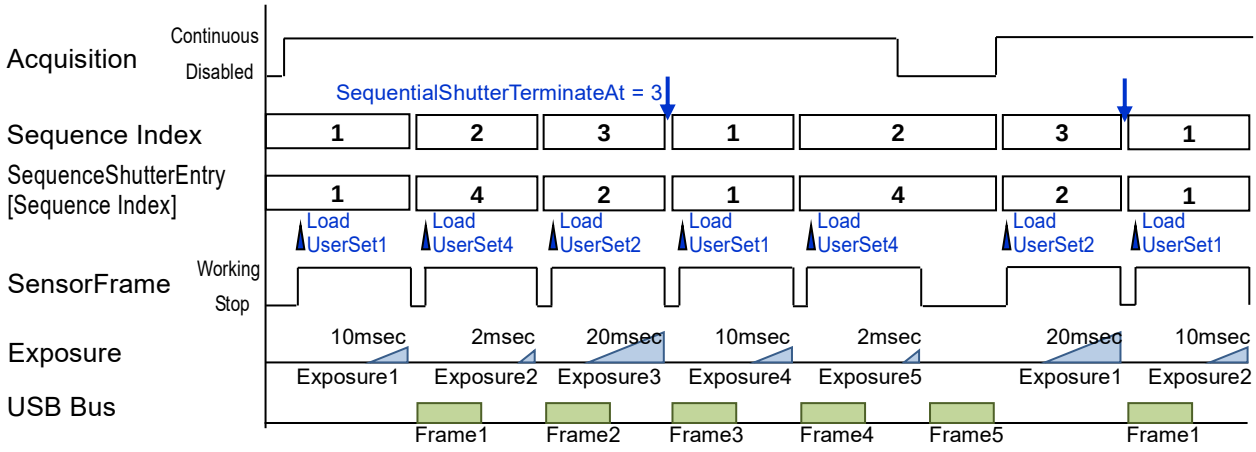
Sequential Shutter utilizes UserSet memory function for storing and loading camera settings used in Sequential Shutter mode capturing sequence. TeliCamAPI will change camera settings by loading camera settings in UserSet memory specified by SequentialShutterSequenceTable register array value (SequentialShutterEntry node) before the beginning of each exposure process.

The sequence length of Sequential Shutter mode is controlled by SequentialShutterTerminateAt register value. On incrementing Sequential Shutter sequence index, which starts from 1, the index will be cleared to 1 when the index value is equal to SequentialShutterTerminateAt register value.

The following figure shows how Sequential Shutter works.

This example assumes that 3 kind of exposure time settings are stored in UserSet memory (index 1, 2 and 4), and 3 is set to sequence length (SequentialShutterTerminateAt register), beforehand.

Register values	
SequentialShutterEnable	: On
SequentialShutterTerminateAt	: 3
SequentialShutterEntry (index=1)	: 1 (Use UserSet1)
SequentialShutterEntry (index=2)	: 4 (Use UserSet4)
SequentialShutterEntry (index=3)	: 2 (Use UserSet2)
Exposure time in UserSet1	: 10msec
Exposure time in UserSet2	: 20msec
Exposure time in UserSet4	: 2msec



Note that sequence index starts from 1 and it will not be cleared to 1 even if image acquisition is stopped. Sequence index will be cleared to 1 by disabling SequentialShutterEnable and enabling it again during image acquisition is not active.

2. Camera registers relating to Sequential Shutter mode.

The following table shows camera registers (32bit registers) relating to Sequential Shutter mode.
Sequential Shutter mode is often used in combination with BulkTrigger mode.

	Register name	Description
Sequential Shutter	SequentialShutterEnable_Inquiry 0x21F300 RegMapBU::ADR_ VENDOR_SEQ_SHT_ENABLE_ENUM32B	Inquiry information of Sequential Shutter feature. When msb is 1, Sequential Shutter is implemented. This register is read-only register.
	SequentialShutterEnable 0x21F31C RegMapBU::ADR_VENDOR_SEQ_SHT_ENABLE_I	Enables / disables Sequential Shutter mode. 1: On (Enabled) 0: Off (Disabled)
	SequentialShutterTerminateAt 0x21F33C RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_I	Sequence length of Sequential Shutter mode.
	SequentialShutterTerminateAt_Min 0x21F334 RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_INT32B + RegMapBU::OFS_FCSR_INT32_MIN_I	The minimum sequence length value. This register is read-only register.
	SequentialShutterTerminateAt_Max 0x21F338 RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_INT32B + RegMapBU::OFS_FCSR_INT32_MAX_I	The maximum sequence length value. This register is read-only register.
	SequentialShutterSequenceTable_0 0x500040	The first member (sequence index = 1) of SequentialShutterTable register array.
	SequentialShutterSequenceTable_Len 0x50002C	Member count of SequentialShutterTable register array.
	SequentialShutterSequenceTableVMin 0x500038	The minimum value available in SequentialShutterTable (SequentialShutterEntry).
	SequentialShutterSequenceTableVMax 0x50003C	The maximum value available in SequentialShutterTable (SequentialShutterEntry).
UserSet	UserSetSelector 0x20807C RegMapBU::ADR_USET_SEL_I	Specifies index of UserSet memory accessed through UserSetCommand register.
	UserSetCommand 0x20809C RegMapBU::ADR_USET_USERSET_CMD_I	Accesses UserSet memory specified by value of UserSetSelector register. The following values are available. 0 : Done (for checking that access completed.) 8 : Load 9 : Save 120 : QuickSave (Saves in RAM) *1
Trigger	TriggerMode 0x20703C RegMapBU::ADR_TRG_TRG_MODE_I	Enables / disables Random Trigger Shutter mode. 1: On (Enabled) 0: Off (Disabled)
	TriggerSequence 0x20705C RegMapBU::ADR_TRG_TRG_SEQUENCE_I	Sets sequence type of Random Trigger Shutter mode. The following values are available. 0 : Edge mode (TriggerSequence0) 1 : Level mode (TriggerSequence1) 6 : Bulk mode (TriggerSequence6)
	TriggerSource 0x20707C RegMapBU::ADR_TRG_TRG_SOURCE_I	Selects signal source of Random Trigger Shutter. 0 : Line0 (IOLine0) 64 : SoftwareTrigger (SoftwareTrigger)
	TriggerAdditionalParameter 0x20709C RegMapBU:: ADR_TRG_ADDITIONAL_PARAM_RAW_I	Sets the number of frames to exposure in Bulk mode.
	TriggerDelay_Raw 0x2070B0 RegMapBU:: ADR_TRG_TRIGGER_DELAY_RAW_I	Sets delay time from trigger edge to exposure start. Delay in seconds = TriggerDelay_Raw / TriggerDelay_Div

	Register name	Description
	TriggerDelay_Div 0x2070BC RegMapBU:: ADR_TRG_TRIGGER_DELAY_INT32B + RegMapBU::OFS_FCSR_INT32_DIV_I	Division constant for converting TriggerDelay_Raw register value to TriggerDelay value in seconds.
	TriggerSoftware 0x2070DC RegMapBU::ADR_TRG_SOFT_TRG_I	Sends trigger signal. The following values are available. 0: Inactive 1: Active 8: Impulse (register value will be changed to 0 after activating trigger action.)

*1: When “QuickSave” command is written, the camera will save current camera settings in RAM instead of writing in flash memory. The written camera parameters will be lost on power down but user application can save parameters quickly. It will take less than 100 microseconds to write in RAM, on the other hand, it will take about 700 milliseconds to write in flash memory. This command value was appended in the recent generation models.

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
Sequential Shutter	GetCamSequentialShutterEnable ()	SequentialShutterEnable
	SetCamSequentialShutterEnable ()	
	GetCamSequentialShutterTerminateAtMinMax()	SequentialShutterTerminateAt_Min and Max
	GetCamSequentialShutterTerminateAt()	SeauentialShutterTerminateAt
	SetCamSequentialShutterTerminateAt()	
	GetCamSequentialShutterIndexMinMax()	SequentialShutterSequenceTable_Len
	GetCamSequentialShutterEntryMinMax()	SequentialShutterSequenceTableVMin and VMax
	GetCamSequentialShutterEntry()	SequentialShutterSequenceTable
	SetCamSequentialShutterEntry()	
	ExecuteCamUserSetLoad()	UserSetSelector
	ExecuteCamUserSetSave() *2	UsersetCommand
Trigger	GetCamTriggerMode()	TriggerMode
	SetCamTriggerMode()	
	GetCamTriggerSequence()	TriggerSequence
	SetCamTriggerSequence()	
	GetCamTriggerSource()	TriggerSource
	SetCamTriggerSource()	
	GetCamTriggerAdditionalParameterMinMax()	
	GetCamTriggerAdditionalParameter()	TriggerAdditionalParameter
	SetCamTriggerAdditionalParameter()	
	GetCamTriggerDelayMinMax()	
	GetCamTriggerDelay()	TriggerDelay_Raw
	SetCamTriggerDelay()	TriggerDelay_Div
	ExecuteCamSoftwareTrigger()	TriggerSoftware

*2: If writing “QuickSave” command is required, write 120 to UserSetCommand register after writing UserSet memory index to UserSetSelector register, using Cam_WriteReg() function.

FAQ How to use Sequential Shutter.

<CameraControl class> (.NET framework)

	Methods	Target register
Sequential Shutter	*CameraControl*.GetSequentialShutterEnable()	SequentialShutterEnable
	CameraControl.SetSequentialShutterEnable()	
	CameraControl.GetSequentialShutterTerminateAtMinMax()	SequentialShutterTerminateAt_Min and Max
	CameraControl.GetSequentialShutterTerminateAt()	SeauentialShutterTerminateAt
	CameraControl.SetSequentialShutterTerminateAt()	
	CameraControl.GetSequentialShutterIndexMinMax	SequentialShutterSequenceTable_Len
	CameraControl.GetSequentialShutterEntryMinMax()	SequentialShutterSequenceTableVMin, VMax
	CameraControl.GetSequentialShutterEntry()	SequentialShutterSequenceTable
	CameraControl.SetSequentialShutterEntry()	
	CameraControl.ExecuteUserSetLoad()	UserSetSelector
	CameraControl.ExecuteUserSetSave() *3	UserSetCommand
Trigger	*CameraControl*.GetTriggerMode()	TriggerMode
	CameraControl.SetTriggerMode()	
	CameraControl.GetTriggerSequence()	TriggerSequence
	CameraControl.SetTriggerSequence()	
	CameraControl.GetTriggerSource()	TriggerSource
	CameraControl.SetTriggerSource()	
	CameraControl.GetTriggerAdditionalParameterMinMax()	
	CameraControl.GetTriggerAdditionalParameter()	TriggerAdditionalParameter
	CameraControl.SetTriggerAdditionalParameter()	
	CameraControl.GetTriggerDelayMinMax()	
	CameraControl.GetTriggerDelay()	TriggerDelay
	CameraControl.SetTriggerDelay()	
	CameraControl.ExecuteSoftwareTrigger()	TriggerSoftware

*3: If writing “QuickSave” command is required, write 120 to UserSetCommand register after writing UserSet memory index to UserSetSelector register, using *CameraDevice*.WriteRegister() method.

<Node for GenICam functions and methods>

	Node name	Node type	Enumeration value	Target register
	“SequentialShutterEnable” (XmlFeatures::XF_ID_SEQ_SHUTTER_ENABLE)	Enumeration	“On” “Off”	SequentialShutterEnable
	“SequentialShutterTerminateAtMin” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT_MIN)	Integer		SequentialShutterTerminateAt_Min
	“SequentialShutterTerminateAtMax” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT_MAX)	Integer		SequentialShutterTerminateAt_Max
	“SequentialShutterTerminateAt” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT)	Integer		SequentialShutterTerminateAt
	“SequentialShutterIndex” (XmlFeatures::XF_ID_SEQ_SHUTTER_SHUTER_INDEX)	Integer		
	“SequentialShutterEntry” (XmlFeatures::XF_ID_SEQ_SHUTTER_SHUTTER_ENTRY)	Integer		SequentialShutterSequenceTable
Trigger	“UserSetSelector” (XmlFeatures::XF_ID_USERSET_SELECTOR)	Enumeration	“Default” “UserSet1” ….. “UserSet15”	
	“UserSetLoad” (XmlFeatures::XF_ID_USERSET_LOAD)	Command		
	“UserSetSave” (XmlFeatures::XF_ID_USERSET_SAVE)	Command		
	“UserSetQuickSave” *4	Command		
	“TriggerMode” (XmlFeatures::XF_ID_TRIGGER_MODE)	Enumeration	“On” “Off”	TriggerMode
	“TriggerSequence” (XmlFeatures::XF_ID_TRIGGER_SEQUENCE)	Enumeration	“TriggerSequence0” “TriggerSequence1” “TriggerSequence6”	TriggerSequence
	“TriggerSource” (XmlFeatures::XF_ID_TRIGGER_SOURCE)	Enumeration	“Line0” “Software”	TriggerSource
	“TriggerAdditionalParameter”	Integer		TriggerAdditionalParameter
	“TriggerDelay” (XmlFeatures::XF_ID_TRIGGER_DELAY)	Float		TriggerDelayRaw TriggerDelayDiv
	“TriggerSiftware” (XmlFeatures::XF_ID_TRIGGER_SOFTWARE)	Command		TriggerSoftware

***4:** This node is available in the recent generation model.

3. Image acquisition in SequentialShutter mode.

3.1. Saving camera settings in UserSet memory

Camera settings used in Sequential Shutter sequence should be saved in UserSet memory before starting image acquisition.

It will take about 700 milliseconds to write data in UserSet because UserSet memory is composed of flash memory.

Camera controlling function `ExecuteUserSetSave()` will send UserSet saving command to a camera and wait until writing to the UserSet completes. You can write the program without paying attention to the completion of writing to the flash memory when you use `ExecuteUserSetSave()`.

GenICam functions `GenAPI_CmdExecute()` or `Nd_CmdExecute()` will also send UserSet saving command to a camera, but finish without waiting the completion of writing to the UserSet. You should write the program paying attention to the completion of writing to the flash memory when you use GenICam functions for saving in UserSet. `GenAPI_GetCmdIsDone()` or `Nd_GetCmdIsDone()` is available for checking the state of UserSet saving.
(`GenAPI_CmdExecute()` and `GenAPI_GetCmdIsDone()` are available in TeliCamSDK PkgVer3.0.0.1 or later.)

Writing 'Load' command value to UserSetCommand register is also available for saving parameters in UserSet memory. UserSetCommand register value will return to 0 ('Done') when writing to flash memory completes.

Note that if you start writing register values for the next UserSet save without waiting completion of current UserSet saving operation, the parameter values saved in the UserSet memory might be different from the intended values.

In recent generation models, "QuickSave" command (0x0078) is available for saving camera settings in UserSet memory on RAM.

Note that image size setting in UserSet memory will not be loaded in SequentialShutter sequence while image offset will be loaded. Image size setting at the timing that image acquisition is started will be used in every SequentialShutter sequence. Never set the image size that will cause ROI inconsistency in SequentialShutter mode image acquisition.

The example code in the next page is a code for writing camera parameters in UserSet memory. This example does not contain codes for handling errors.

========== example code ==========

```
#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

#define QuickSave 120

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   int32_t          iTmp;
   . . . . . .

   // Saving UserSet1 settings.
   // Sets exposure time 10msec.
   uiSts = SetCamExposureTime(hCam, 0.01);
   . . . . . .

   // Saves current settings in UserSet1.
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET1);

   //// QuickSave command.
   //// Selects UserSet1.
   // iTmp = 1;
   // uiSts = CAM_WriteReg(hCam, RegMapBU::ADR_USET_SEL_I, 1, &iTmp);
   //// Sets QuickSave command.
   // iTmp = QuickSave;
   //// Sends QuickSave command.
   // uiSts = CAM_WriteReg(hCam, RegMapBU::ADR_USET_USERSET_CMD_I, 1, &iTmp);

   // Saving UserSet2 settings.
   // Sets exposure time 20msec.
   uiSts = SetCamExposureTime(hCam, 0.02);
   . . . . . .
   // Saves current settings in UserSet2.
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET2);

   // Saving UserSet4 settings.
   // Sets exposure time 2msec.
   uiSts = SetCamExposureTime(hCam, 0.002);
   . . . . . .
   // Saves current settings in UserSet4.
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET4);
```

========== end of example code ==========

3.2. Starting image acquisition (setting Sequential Shutter mode)

User application should sets Sequential Shutter parameters before starting image acquisition.

The following example is a code for setting Sequential Shutter parameters. This example also sets software Bulk trigger mode.

This example does not contain codes for handling errors.

========== example code ==========

```
#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   uint32_t         uiIndex;
   uint32_t         uiUserSetIndex;
   . . . . . .

   // Setting Sequential Shutter mode.
   // Enables Sequential Shutter.
   uiSts = SetCamSequentialShutterEnable(hCam, TRUE);
   // Sets sequence length 3.
   uiSts = SetCamSequentialShutterTerminateAt(hCam, 3);
   // Sets SequentialShutterEntries.
   //  Uses UserSet1 in Sequence index 1. (Sequence index starts from 1.)
   uiIndex          = 1;
   uiUserSetIndex   = 1;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);
   //  Uses UserSet4 in Sequence index 2.
   uiIndex          = 2;
   uiUserSetIndex   = 4;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);
   //  Uses UserSet2 in Sequence index 3.
   uiIndex          = 3;
   uiUserSetIndex   = 2;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);

   // Sets Software Bulk trigger mode.
   //  Enables Randam Trigger Shutter.
   uiSts = SetCamTriggerMode(hCam, TRUE);
   //  Sets TrigggerSequence6 (Bulk mode).
   uiSts = SetCamTriggerSequence(hCam, CAM_TRIGGER_SEQUENCE6);
   //  Sets TriggerSource SoftwareTriggger.
   uiSts = SetCamTriggerSource(hCam, CAM_TRIGGER_SOFTWARE);

   // Starts image acquisition.
   . . . . . .
```

========== end of example code ==========

3.3. Acquiring image

Image acquiring sequence is exactly same as ordinary image acquisition case. User application can use ordinary image acquisition code.

Note that user application has no way to detect Sequential Shutter sequence index of a received image. When each sequence has different mage offset, user application can detect sequence index using image offset data in CAM_IMAGE_INFO structure,

In the future model camera, camera will have a function to send sequence index with image data as a chunk data.

3.4. Clearing Sequential Shutter mode

The following example is a code for disabling Sequential Shutter mode. Never fail to stop image acquisition before editing Sequential shutter related registers.

This example does not contain codes for handling errors.

========== example code ==========

```
#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   uint32_t         uiIndex;
   uint32_t         uiUserSetIndex;
   . . . . . .

   // Stops image acquisition.
   . . . . . .

   // Disabling Sequential Shutter mode.
   //  Disables Sequential Shutter.
   uiSts = SetCamSequentialShutterEnable(hCam, FALSE);
```

========== end of example code ==========

1. シーケンシャルシャッタとは

シーケンシャルシャッタは複数の撮像条件下で撮影した被写体画像を取得する機能です。例えば露光時間・ゲイン・画像オフセットなどを変えた複数の画像を取得するときなどに使用します。

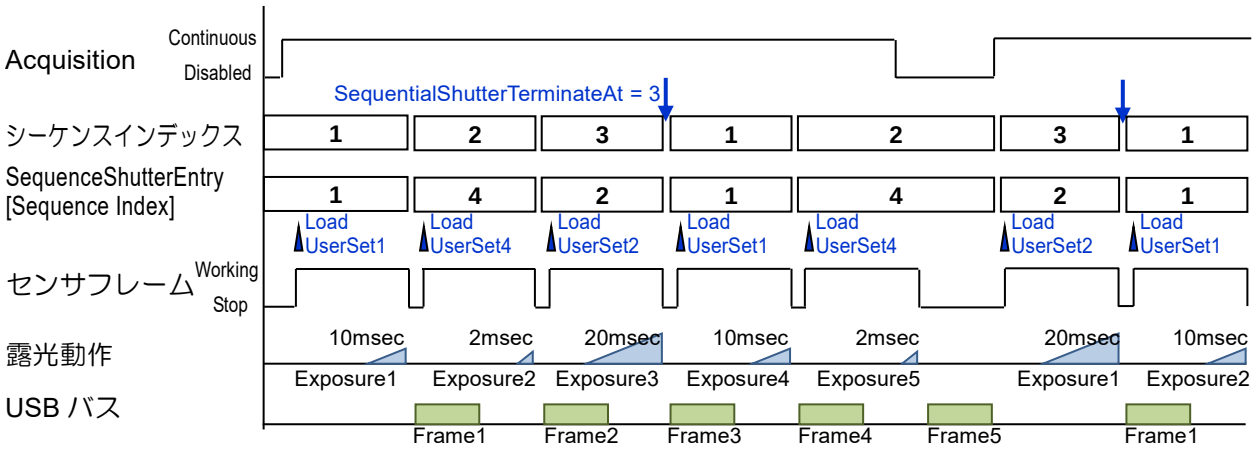
シーケンシャルシャッタは UserSet メモリの機能を使用してシーケンシャルシャッタモードの撮像に使用するカメラパラメータの保存・呼び出しを行なっています。TeliCamAPI はセンサの露光プロセスを開始する前に、SequentialShutterSequenceTable レジスタ配列（SequentialShutterEntry ノード）のレジスタ値で指定されたインデックスの UserSet メモリの内容をロードすることにより、各種カメラ設定を切り替えます。

シーケンシャルシャッタモードのシーケンス長は SequentialShutterTerminateAt レジスタの値で制御しています。シーケンシャルシャッタのシーケンスインデックス（１から加算を開始する値）を加算するときに、更新前のシーケンスインデックスの値が SequentialShutterTerminateAt レジスタの値と等しいときはシーケンスインデックスインデックスを１にクリアしています。

以下の図にシーケンシャルシャッタの動作を示します。

この例は、予め３種類の露光時間が１番、２番、４番の UserSet メモリに保存されており、シーケンシャルシャッタのシーケンス長（SequentiaShutterTerminateAt レジスタ値）が３に設定されている前提で動作を示しています。

レジスタ値	
SequentialShutterEnable	: On
SequentialShutterTerminateAt	: 3
SequentialShutterEntry (index=1)	: 1 (Use UserSet1)
SequentialShutterEntry (index=2)	: 4 (Use UserSet4)
SequentialShutterEntry (index=3)	: 2 (Use UserSet2)
UserSet1 の Exposure time	: 10msec
UserSet2 の Exposure time	: 20msec
UserSet4 の Exposure time	: 2msec



シーケンスインデックスは１から加算されていきますが、Aquisition を停止状態にしても１にクリアされません。Aquisition 停止中に SequenceShutterEnable レジスタを一旦ディス栄ぶるにしてから再度イネーブルにすることによりシーケンスインデックスが１にクリアされます。

2. シーケンシャルシャッタ関連レジスタ

以下の表にシーケンシャルシャッタ関連のレジスタを示します。各レジスタは 32bit レジスタです。
シーケンシャルシャッタモードはしばしば BulkTrigger モードと同時に使用されます。

	レジスタ名	内　　容
シーケンシャルシャッタ	SequentialShutterEnable_Inquiry 0x21F300 RegMapBU::ADR_ VENDOR_SEQ_SHT_ENABLE_ENUM32B	シーケンシャルシャッタ機能に関する情報。 msb が 1 のときシーケンシャルシャッタ機能が実装されています。このレジシタは読み出し専用です。
	SequentialShutterEnable 0x21F31C RegMapBU::ADR_VENDOR_SEQ_SHT_ENABLE_I	シーケンシャルシャッタ機能の有効無効設定。 1: On (有効) 0: Off (無効).
	SequentialShutterTerminateAt 0x21F33C RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_I	シーケンシャルシャッタモードのシーケンス長。
	SequentialShutterTerminateAt_Min 0x21F334 RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_INT32B + RegMapBU::OFS_FCSR_INT32_MIN_I	シーケンシャルシャッタモードの最小シーケンス長。 このレジシタは読み出し専用です。
	SequentialShutterTerminateAt_Max 0x21F338 RegMapBU::ADR_VENDOR_SEQ_SHT_TRMNAT_AT_INT32B + RegMapBU::OFS_FCSR_INT32_MAX_I	シーケンシャルシャッタモードの最大シーケンス長。 このレジシタは読み出し専用です。
	SequentialShutterSequenceTable_0 0x500040	SequentialShutterTable レジスタ配列の先頭レジスタ r (シーケンスインデックス=1)。
	SequentialShutterSequenceTable_Len 0x50002C	SequentialShutterTable レジスタ配列のメンバ数
	SequentialShutterSequenceTableVMin 0x500038	SequentialShutterTable（SequentialShutterEntry ノード）に設定可能な値の最小値。
	SequentialShutterSequenceTableVMax 0x50003C	SequentialShutterTable（SequentialShutterEntry ノード）に設定可能な値の最大値。
UserSet	UserSetSelector 0x20807C RegMapBU::ADR_USET_SEL_I	UserSetCommand レジスタでアクセスする UserSet メモリのインデックス
	UserSetCommand 0x20809C RegMapBU::ADR_USET_USERSET_CMD_I	UserSetSelector で指定された UserSet メモリにアクセスします。 以下の値が指定可能です。 0 ：Done (アクセス完了確認用。) 8 ：Load 9 ：Save 120 ：QuickSave (RAM に保存します。) *1
トリガ	TriggerMode 0x20703C RegMapBU::ADR_TRG_TRG_MODE_I	ランダムトリガシャッタモードの有効無効設定。 1: On (有効) 0: Off (無効).
	TriggerSequence 0x20705C RegMapBU::ADR_TRG_TRG_SEQUENCE_I	ランダムトリガシャッタモードの動作シーケンス。 以下の値が指定可能です。 0 ： エッジモード　(TriggerSequence0) 1 ： レベルモード　(TriggerSequence1) 6 ： バルクモード　(TriggerSequence6)
	TriggerSource 0x20707C RegMapBU::ADR_TRG_TRG_SOURCE_I	ランダムトリガシャッタの信号源指定。 0： ライン 0　(IOLine0) 64： ソフトトリが　(SoftwareTrigger)
	TriggerAdditionalParameter 0x20709C RegMapBU:: ADR_TRG_ADDITIONAL_PARAM_RAW_I	Bulk モードで 1 回のトリガー入力に対応して露光するフレーム数の設定。

	レジスタ名	内　　容
	TriggerDelay_Raw 0x2070B0 RegMapBU:: ADR_TRG_TRIGGER_DELAY_RAW_I	トリガ入力から露光開始までの遅延時間設定内部値。 遅延時間(秒) =) TriggerDelay_Raw / TriggerDelay_Div
	TriggerDelay_Div 0x2070BC RegMapBU:: ADR_TRG_TRIGGER_DELAY_INT32B + RegMapBU::OFS_FCSR_INT32_DIV_I	TriggerDelay_Raw レジスタ値を秒単位の遅延時間値に変換するための定数。
	TriggerSoftware 0x2070DC RegMapBU::ADR_TRG_SOFT_TRG_I	トリガ信号を送出します。 0: Inactive 1: Active 8: Impulse (トリガ後にレジスタ値は0に戻ります、)

*1: QuickSave コマンドを書き込むと、カメラは現状のカメラ設定をフラッシュメモリではなく RAM に書き込みます。RAM に保存されたカメラ設定はカメラの電源を切ると失われますが、パラメータの書き込み時間を短縮することができます。フラッシュメモリ書き込み時は書き込み時間が約 700msec かかりますが、RAM に書く場合は 100μsec で書き込み完了します。QuickSave コマンドはこの最近の世代のカメラで使用可能です。

シーケンシャルシャッタ関連レジスタにアクセスする関数、メソッドを以下の表に示します。

【カメラ制御関数】(ネイティブ C++)

	カ メ ラ 制 御 関 数	対 象 カ メ ラ レ ジ ス タ
シーケンシャルシャッタ	GetCamSequentialShutterEnable ()	SequentialShutterEnable
	SetCamSequentialShutterEnable ()	
	GetCamSequentialShutterTerminateAtMinMax()	SequentialShutterTerminateAt_Min and Max
	GetCamSequentialShutterTerminateAt()	SeauentialShutterTerminateAt
	SetCamSequentialShutterTerminateAt()	
	GetCamSequentialShutterIndexMinMax()	SequentialShutterSequenceTable_Len
	GetCamSequentialShutterEntryMinMax()	SequentialShutterSequenceTableVMin and VMax
	GetCamSequentialShutterEntry()	SequentialShutterSequenceTable
	SetCamSequentialShutterEntry()	
	ExecuteCamUserSetLoad()	UserSetSelector UsersetCommand
	ExecuteCamUserSetSave()	
	ExecuteCamUserSetQuickSave() *2	
トリガ	GetCamTriggerMode()	TriggerMode
	SetCamTriggerMode()	
	GetCamTriggerSequence()	TriggerSequence
	SetCamTriggerSequence()	
	GetCamTriggerSource()	TriggerSource
	SetCamTriggerSource()	
	GetCamTriggerAdditionalParameterMinMax()	
	GetCamTriggerAdditionalParameter()	TriggerAdditionalParameter
	SetCamTriggerAdditionalParameter()	
	GetCamTriggerDelayMinMax()	
	GetCamTriggerDelay()	TriggerDelay_Raw TriggerDelay_Div
	SetCamTriggerDelay()	
	ExecuteCamSoftwareTrigger()	TriggerSoftware

*2: QuickSave 用関数は PkgVer2.1.0.1 以降の TeliiCamSDK で追加されました。PkgVer2.0.1.1 以前で QuickSave コマンドを使用する場合は、UserSetSelector レジスタに書き込み先 UserSet メモリインデックスを書いてから UserSetCommand レジスタに 120 を書いてください。レジスタへの値書き込みには Cam_WriteReg()関数を使用してください。
QuickSave コマンドはこの最近の世代のカメラで使用可能です。

【CameraControl クラス】 (.NET framework)

	メソッド	対象カメラレジスタ
シーケンシャルシャッタ	*CameraControl*.GetSequentialShutterEnable()	SequentialShutterEnable
	CameraControl.SetSequentialShutterEnable()	
	CameraControl.GetSequentialShutterTerminateAtMinMax()	SequentialShutterTerminateAt_Min and Max
	CameraControl.GetSequentialShutterTerminateAt()	SeauentialShutterTerminateAt
	CameraControl.SetSequentialShutterTerminateAt()	
	CameraControl.GetSequentialShutterIndexMinMax	SequentialShutterSequenceTable_Len
	CameraControl.GetSequentialShutterEntryMinMax()	SequentialShutterSequenceTableVMin, VMax
	CameraControl.GetSequentialShutterEntry()	SequentialShutterSequenceTable
	CameraControl.SetSequentialShutterEntry()	
	CameraControl.ExecuteUserSetLoad()	UserSetSelector UserSetCommand
	CameraControl.ExecuteUserSetSave()	
	CameraControl.ExecuteUserSetQuickSave() *3	
トリガ	*CameraControl*.GetTriggerMode()	TriggerMode
	CameraControl.SetTriggerMode()	
	CameraControl.GetTriggerSequence()	TriggerSequence
	CameraControl.SetTriggerSequence()	
	CameraControl.GetTriggerSource()	TriggerSource
	CameraControl.SetTriggerSource()	
	CameraControl.GetTriggerAdditionalParameterMinMax()	
	CameraControl.GetTriggerAdditionalParameter()	TriggerAdditionalParameter
	CameraControl.SetTriggerAdditionalParameter()	
	CameraControl.GetTriggerDelayMinMax()	
	CameraControl.GetTriggerDelay()	TriggerDelay
	CameraControl.SetTriggerDelay()	
	CameraControl.ExecuteSoftwareTrigger()	TriggerSoftware

*3: QuickSave 用関数は PkgVer2.1.0.1 以降の TeliiCamSDK で追加されました。PkgVer2.0.1.1 以前で QuickSave コマンドを使用する場合は、UserSetSelector レジスタに書き込み先 UserSet メモリインデックスを書いてから UserSetCommand レジスタに 120 を書いてください。レジスタへの値書き込みには *CameraDevice.*WriteRegister()メソッドを使用してください。
QuickSave コマンドはこの最近の世代のカメラで使用可能です。

【GenICam 関数および GenApiNode クラスで使用するノード】

	ノード名	ノード型	列挙型値	対象カメラレジスタ
シーケンシャルシャッタ	“SequentialShutterEnable” (XmlFeatures::XF_ID_SEQ_SHUTTER_ENABLE)	Enumeration	“On” “Off”	SequentialShutterEnable
	“SequentialShutterTerminateAtMin” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT_MIN)	Integer		SequentialShutterTerminateAt_Min
	“SequentialShutterTerminateAtMax” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT_MAX)	Integer		SequentialShutterTerminateAt_Max
	“SequentialShutterTerminateAt” (XmlFeatures::XF_ID_SEQ_SHUTTER_TRMNT_AT)	Integer		SequentialShutterTerminateAt
	“SequentialShutterIndex” (XmlFeatures::XF_ID_SEQ_SHUTTER_SHUTER_INDEX)	Integer		
	“SequentialShutterEntry” (XmlFeatures::XF_ID_SEQ_SHUTTER_SHUTTER_ENTRY)	Integer		SequentialShutterSequenceTable
UserSet	“UserSetSelector” (XmlFeatures::XF_ID_USERSET_SELECTOR)	Enumeration	“Default” “UserSet1” ….. “UserSet15”	
	“UserSetLoad” (XmlFeatures::XF_ID_USERSET_LOAD)	Command		
	“UserSetSave” (XmlFeatures::XF_ID_USERSET_SAVE)	Command		
	“UserSetQuickSave” *4	Command		
トリガ	“TriggerMode” (XmlFeatures::XF_ID_TRIGGER_MODE)	Enumeration	“On” “Off”	TriggerMode
	“TriggerSequence” (XmlFeatures::XF_ID_TRIGGER_SEQUENCE)	Enumeration	“TriggerSequence0” “TriggerSequence1” “TriggerSequence6”	TriggerSequence
	“TriggerSource” (XmlFeatures::XF_ID_TRIGGER_SOURCE)	Enumeration	“Line0” “Software”	TriggerSource
	“TriggerAdditionalParameter”	Integer		TriggerAdditionalParameter
	“TriggerDelay” (XmlFeatures::XF_ID_TRIGGER_DELAY)	Float		TriggerDelayRaw TriggerDelayDiv
	“TriggerSiftware” (XmlFeatures::XF_ID_TRIGGER_SOFTWARE)	Command		TriggerSoftware

***4:** このノードは最近の世代のカメラで使用可能です。

3. シーケンシャルシャッタを使用した撮像処理

3.1. UserSet メモリへのカメラ設定の保存

シーケンシャルシャッタのシーケンスで使用するカメラ設定は、撮像を開始する前に UserSet メモリに書き込んでおく必要があります。

UserSet メモリはフラッシュメモリで構成されているため、データの書き込みに 700msec 近く時間がかかる場合があります。

カメラ制御関数 `ExecuteCamUserSetSave()`は、カメラに `UserSet` の書き込みを指示した後に `UserSet` の書き込みが完了したことを確認してから処理をアプリケーションに戻します。フラッシュメモリの書き込み完了を意識することなくプログラムを記述いただいてかまいません。

`GenICam` 関数の `GenAPI_CmdExecute()`または `Nd_CmdExecute()`を使用して `UserSet` の書き込みを指示した場合は、フラッシュメモリの書き込み完了を待つことなく処理がアプリケーションに戻ります。フラッシュメモリの書き込みが完了したことを確認するには `GenAPI_GetCmdIsDone()`または `Nd_GetCmdIsDone()`をお使いください。(`GenAPI_CmdExecute()`と `GenAPI_GetCmdIsDone()`は `PkgVer3.0.0.1` 以降の `TeliCamSDK` で使用できる関数です。)

`UserSetCommand` レジスタに `Load` コマンド値を直接書き込むことにより `UserSet` の書き込みを指示することもできます。この場合、フラッシュメモリへの書き込みが完了するとレジスタ値が `Done(0)`になります。

フラッシュメモリの書き込み完了を待たずに次の `UserSet` 用のパラメータの変更作業を開始し、`UserSet` の書き込みを指示した場合には `UserSet` に保存されたパラメータが意図したものとは異なる値になる場合がありますので注意してください。

最近の世代のカメラでは カメラ設定を RAM 上に保存する QuickSave コマンド(0x0078)が使用可能になっています。

シーケンシャルシャッタ動作では UserSet メモリ内の画像オフセット設定はロードされますが、画像サイズ設定はロードされないことに注意してください。シーケンシャルシャッタ動作では撮像開始時点の画像サイズ設定が各シーケンスの画像オフセット設定との組み合わせで使用されます。各シーケンスの画像オフセット設定との組み合わせで不整合が発生するような画像サイズは設定しないようにしてください。

カメラパラメータを UserSet メモリに書き込むコード例を次ページに示します。この例ではエラー処理は省略しています。

========== コード例 ==========

```
#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

#define QuickSave 120

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   int32_t          iTmp;
   . . . . . .

   // UserSet1 へのカメラ設定保存
   //   露光時間を 10msec に設定
   uiSts = SetCamExposureTime(hCam, 0.01);
   . . . . . .

   // 現状のカメラ設定を UserSet1 に保存。
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET1);

   //// QuickSave コマンドを使用する場合
   //// UserSet1 の選択.
   // iTmp = 1;
   // uiSts = CAM_WriteReg(hCam, RegMapBU::ADR_USET_SEL_I, 1, &iTmp);
   //// QuickSave コマンドの書き込み
   // iTmp = QuickSave;
   // uiSts = CAM_WriteReg(hCam, RegMapBU::ADR_USET_USERSET_CMD_I, 1, &iTmp);

   // UserSet2 へのカメラ設定保存
   //   露光時間を 20msec に設定
   uiSts = SetCamExposureTime(hCam, 0.02);
   . . . . . .

   // 現状のカメラ設定を UserSet2 に保存。
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET2);

   // UserSet4 へのカメラ設定保存
   //   露光時間を 2msec に設定
   uiSts = SetCamExposureTime(hCam, 0.002);
   . . . . . .

   // 現状のカメラ設定を UserSet4 に保存。
   uiSts = ExecuteCamUserSetSave(hCam, CAM_USER_SET_SELECTOR_USER_SET4);
```

========== コード例終了 ==========

3.2. 画像取得開始（シーケンシャルシャッタパラメータの設定）

画像取得開始に先立ち、シーケンシャルシャッタの設定を行う必要があります。

シーケンシャルシャッタとソフトウェアバルクトリガモードの設定を行うコード例を以下に示します。この例ではエラー処理は省略しています。

```
======================== コード例 ====================================

#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   uint32_t         uiIndex;
   uint32_t         uiUserSetIndex;
   . . . . . .

   // シーケンシャルシャッタモードの設定
   //   シーケンシャルシャッタの有効化
   uiSts = SetCamSequentialShutterEnable(hCam, TRUE);
   //   シーケンシャルシャッタのシーケンス長を 3 に設定
   uiSts = SetCamSequentialShutterTerminateAt(hCam, 3);
   //   シーケンシャルシャッタテーブルの設定.
   //    シーケンスインデックス 1 では UserSet1 を使用。(シーケンスインデックスは 1 から加算開始)
   uiIndex          = 1;
   uiUserSetIndex   = 1;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);
   //    シーケンスインデックス 2 では UserSet4 を使用。
   uiIndex          = 2;
   uiUserSetIndex   = 4;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);
   //    シーケンスインデックス 3 では UserSet2 を使用。
   uiIndex          = 3;
   uiUserSetIndex   = 2;
   uiSts = SetCamSequentialShutterEntry(hCam, uiIndex, uiUserSetIndex);

   // ソフトウェアバルクトリガモードの設定
   //  ランダムトリガシャッタの有効化
   uiSts = SetCamTriggerMode(hCam, TRUE);
   //  TrigggerSequence6 (バルクモード)の設定
   uiSts = SetCamTriggerSequence(hCam, CAM_TRIGGER_SEQUENCE6);
   //  ソフトウェアトリガをトリガ信号源に設定.
   uiSts = SetCamTriggerSource(hCam, CAM_TRIGGER_SOFTWARE);

   // 画像取得開始
   . . . . . .
======================== コード例終了 ====================================
```

3.3. 画像取得

シーケンシャルシャッタモードでの画像取得の手順は通常の画像取得シーケンスとまったく同じです。通常の画像取得コードがシーケンシャルシャッタモードでそのまま使用できます。

受信した画像データからその画像のシーケンシャルシャッタシーケンスインデックスを得ることはできません。各シーケンスで異なる画像オフセットを使用しているときは CAM_IMAGE_INFO 構造体内の画像オフセットからシーケンスインデックスを得ることができます。

将来機種ではシーケンスインデックスをチャンクデータとして画像と一緒に送信することも可能になる予定です。

3.4. シーケンシャルシャッタモードの解除

シーケンシャルシャッタモードを解除するコード例を以下に示します。シーケンシャルシャッタ関連レジスタの値を編集する前に必ず画像取得を停止させておいてください。

この例ではエラー処理は省略しています。

```
======================== コード例 ======================================

#include “TeliCamAPI.h”
#include “RegisterMap_BU.h”

   CAM_API_STATUS   uiSts;
   CAM_HANDLE       hCam;
   uint32_t         uiIndex;
   uint32_t         uiUserSetIndex;
   . . . . . .

   // 画像取得の停止
   . . . . . .

   // シーケンシャルシャッターモードの解除。
   uiSts = SetCamSequentialShutterEnable(hCam, FALSE);

======================== コード例終了 ==================================
```

End of document in Japanese

4. Others

4.1. Revision History

Date	Version	Description
2016/07/18	1.0.0	Created the initial version
2018/11/14	1.0.1	✓ Corrected 'Done' value of UserSetCCommand register. ✓ Added description about waiting completion of flash memory writing action in section 3.1.

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

