https://www.ximea.com/support/wiki/apis/XiAPI

xiAPI　XIMEA アプリケーション プログラミング インターフェイス

xiAPI は、XIMEA アプリケーション プログラミング インターフェイスの略です。
これはすべての XIMEA カメラ用の新しい共通インターフェースであり、汎用 M3API の簡略化されたバージョンを表します。

ドキュメント

xiAPI マニュアル
カメラ トリガーと同期信号
画像データ フロー
サンプル コード

アーキテクチャ

API は、カメラ システム ドライバーとアプリケーション間のソフトウェア インターフェイスです。
Windows の場合: xiAPI は xiapi32.dll または xiapi64.dll にコンパイルされます。
Linux の場合: xiAPI は /usr/lib/libm3api.so にコンパイルされます。

マルチプロセス

複数のプロセスからすべてのカメラを列挙できます。xiGetParamInfoString を使用して、任意のプロセスまたはスレッドから SN、User_ID、またはモデル名を取得できます。ただし、カメラは xiOpenDevice によって 1 回しか開くことができません。カメラが 1 つのプロセスまたはスレッドによって開かれた場合、前のプロセスが xiCloseDevice によってハンドルを閉じるか、オペレーティング システムによって終了するまで、他のプロセスはこのカメラを開くことができません。

マルチスレッド

1 つのカメラの xiOpenDevice によって返された同じハンドルを、1 つのプロセス内の複数のスレッドから使用できます。
推奨される方法:
1 つの取得スレッド (**xiGetImage** を呼び出す)
別のメイン コントロール スレッド (**xiGetParam**、**xiSetParam** を呼び出す)

メイン コントロール スレッドによって取得が停止される前に取得スレッドを閉じる。

インストール

Windowsの場合: xiAPIサポートを追加するには、XIMEA WindowsソフトウェアパッケージをPCまたはCURRERAにインストールしてください。
Linuxの場合: xiAPIサポートを追加するには、XIMEA LinuxソフトウェアパッケージをPCまたはCURRERAにインストールしてください。

サンプル

以下は、後処理なしでセンサーから1つの生画像データをキャプチャするサンプルコードです。

露光時間は10ミリ秒に設定されています。

```
#include "..¥include¥xiApi.h"
#include <memory.h>
#define HandleResult(res,place) if (res!=XI_OK) {printf("Error after %s (%d)",place,res);goto finish;}

HANDLE xiH = NULL;
XI_RETURN stat = XI_OK;

// image buffer
XI_IMG image;
memset(&image,0,sizeof(image));
image.size = sizeof(XI_IMG);

// Retrieving a handle to the camera device
stat = xiOpenDevice(0, &xiH);
HandleResult(stat,"xiOpenDevice");
```

```
// Setting Exposure Time parameter (10ms)
stat = xiSetParamInt(xiH, XI_PRM_EXPOSURE, 10000); // microseconds
HandleResult(stat,"xiSetParam (exposure time set)");

// Note:
// The default parameters of each camera might be different
// in different API versions, please set all parameters
// expected by your application to required value.

// Start acquisition
stat = xiStartAcquisition(xiH);
HandleResult(stat,"xiStartAcquisition");

// getting image from camera
stat = xiGetImage(xiH, 1000, &image);
HandleResult(stat,"xiGetImage");

printf("OK - Got one image (%dx%d) from camera¥n", image.width, image.height);

finish:
// Close device
if (xiH) xiCloseDevice(xiH);
```

取込みスタート ストップ付き XIMEA API サンプル

インターフェース

xiAPI の中核となるのは次の 6 つの関数で、カメラ機能のほとんどを制御できます。

```
// open interface
XI_RETURN xiOpenDevice(IN DWORD DevId, OUT PHANDLE hDevice);

// set parameter
XI_RETURN xiSetParam(IN HANDLE hDevice, const char* prm, void* val, DWORD size,
PRM_TYPE type);

// get parameter
XI_RETURN xiGetParam(IN HANDLE hDevice, const char* prm, void* val, DWORD *
size, PRM_TYPE * type);

// start acquisition
XI_RETURN xiStartAcquisition(IN HANDLE hDevice);
```

```
// get next image from queue
XI_RETURN xiGetImage(IN HANDLE hDevice, IN DWORD timeout_ms, OUT LPXI_IMG img);

// close interface
XI_RETURN xiCloseDevice(IN HANDLE hDevice);
```

オフライン処理

xiAPI では、オフライン処理を使用して、すでにキャプチャされ保存されている画像を処理できます。詳細については、xiAPI オフライン処理をご覧ください。

GenTL 経由の取込み

xiAPI での画像バッファの処理が簡素化されています。
アプリケーションによるバッファの割り当てや、各画像バッファの個別制御は許可されません。

ただし、アプリケーションでバッファリングを完全に制御する必要がある場合は、標準の **GenTL** 取込みインターフェイスと **xiAPI** パラメータ インターフェイスを組み合わせて使用できます。

利点

GenTL 経由の取込みを備えた xiAPI には、次の利点があります。
GenTL 経由のアプリケーションによるバッファの割り当て
GenTL 経由のカメラで埋める準備ができているバッファの制御
GenTL 経由の処理用にアプリケーションが保持するバッファの制御
xiAPI 経由のカメラ パラメータの制御 (例: xiSetParam)
DSAnnounceBuffer、DSQueueBuffer など、複数の GenTL 関数が xiAPI ライブラリに実装されています。

例

サンプル アプリケーションのソース ファイルは、xiAPI-capture-50-images-gentl-main.cpp です。

多くの例は、Windows SP または Linux SP の Examples/xiAPI/capture-50-images-gentl フォルダーに含まれています。

カメラの為にサポートされています

xiAPI acquisition over GenTL is supported for all PCIe interface models (xiX, xiB, xiT) and the most recent USB3 interface (xiC) camera models.

https://www.ximea.com/support/wiki/apis/XiAPI

xiAPI

xiAPI stands for XIMEA Application Programming Interface.
It is a new common interface for all XIMEA cameras, and represents a simplified version of generic **M3API**.

Documentation

xiAPI Manual
Camera Trigger and Synchronization Signals
Image Data Flow
Sample code

Architecture

API is a software interface between the camera system driver and application.
On Windows: xiAPI is compiled into xiapi32.dll or xiapi64.dll
On Linux: xiAPI is compiled into /usr/lib/libm3api.so

Multi-process

It is possible to enumerate all cameras from multiple processes. It is possible to use xiGetParamInfoString to obtain SN, User_ID or Model name from any process or thread. However the camera can be opened only once by xiOpenDevice. If camera is opened by one process or thread - other processes can't open this camera until previous process closes the handle by xiCloseDevice or is finished by operating system.

Multi-thread

It is possible to use the same handle returned by xiOpenDevice of one camera from multiple threads within one process.
It is suggested:
to have one Acquisition thread (calling **xiGetImage**)
to have other Main Control Thread (calling **xiGetParam**, **xiSetParam**)

to close acquisition thread before the acquisition is stopped by Main Control Thread.

Installation

On Windows: To add xiAPI support please install **XIMEA Windows Software Package** to PC or CURRERA.
On Linux: To add xiAPI support please install **XIMEA Linux Software Package** to PC or CURRERA.

Sample

Here follows a sample code which captures one raw image data from the sensor without any post-processing.
Exposure time is set 10 ms.

```
#include "..¥include¥xiApi.h"
#include <memory.h>
#define HandleResult(res,place) if (res!=XI_OK) {printf("Error after %s (%d)",place,res);goto finish;}

HANDLE xiH = NULL;
XI_RETURN stat = XI_OK;

// image buffer
XI_IMG image;
memset(&image,0,sizeof(image));
image.size = sizeof(XI_IMG);

// Retrieving a handle to the camera device
stat = xiOpenDevice(0, &xiH);
HandleResult(stat,"xiOpenDevice");
```

```
// Setting Exposure Time parameter (10ms)
stat = xiSetParamInt(xiH, XI_PRM_EXPOSURE, 10000); // microseconds
HandleResult(stat,"xiSetParam (exposure time set)");

// Note:
// The default parameters of each camera might be different
// in different API versions, please set all parameters
// expected by your application to required value.

// Start acquisition
stat = xiStartAcquisition(xiH);
HandleResult(stat,"xiStartAcquisition");

// getting image from camera
stat = xiGetImage(xiH, 1000, &image);
HandleResult(stat,"xiGetImage");

printf("OK - Got one image (%dx%d) from camera¥n", image.width, image.height);

finish:
// Close device
if (xiH) xiCloseDevice(xiH);
```

XIMEA API Sample With Start Stop Acquisition

Interface

The core of xiAPI are the following 6 functions, which allow to control most of the camera functionality.

```
// open interface
XI_RETURN xiOpenDevice(IN DWORD DevId, OUT PHANDLE hDevice);

// set parameter
XI_RETURN xiSetParam(IN HANDLE hDevice, const char* prm, void* val, DWORD size,
PRM_TYPE type);

// get parameter
XI_RETURN xiGetParam(IN HANDLE hDevice, const char* prm, void* val, DWORD *
size, PRM_TYPE * type);

// start acquisition
XI_RETURN xiStartAcquisition(IN HANDLE hDevice);
```

```
// get next image from queue
XI_RETURN xiGetImage(IN HANDLE hDevice, IN DWORD timeout_ms, OUT LPXI_IMG img);

// close interface
XI_RETURN xiCloseDevice(IN HANDLE hDevice);
```

Offline Processing

xiAPI allows to process already captured and stored images using Offline Processing. Please read more at **xiAPI Offline Processing**.

Acquisition over GenTL

Handling of image buffers in xiAPI is simplified.
It does not allow allocating of buffers by application or controlling each image buffer independently.

However, if your application requires full control over buffering, the standard **GenTL** acquisition interface combined with **xiAPI** parameters interface can be used.

Benefits

xiAPI with acquisition over GenTL offers:
Allocation of buffers by application over **GenTL**
Control which buffer is ready to be filled by the camera over **GenTL**
Control which buffer should be kept by application for processing over **GenTL**
Control of camera parameters over **xiAPI** (e.g. xiSetParam)
Multiple GenTL functions are implemented under xiAPI library, such as DSAnnounceBuffer, DSQueueBuffer.

Examples

You can find the source file for sample application - **xiAPI-capture-50-images-gentl-main.cpp**.
Full example is included in Windows SP or in Linux SP - within the folder Examples/xiAPI/capture-50-images-gentl

Supported for Cameras

xiAPI acquisition over GenTL is supported for all PCIe interface models (xiX, xiB, xiT) and the most recent USB3 interface (xiC) camera models.