



X8800
Software Development Kit

目次

はじめに.....	1
放射線の危険に関する注意.....	1
本書の目的.....	1
適用範囲.....	1
はじめに.....	1
クイックスタートガイド.....	2
必要なコンポーネント.....	2
コンピュータのセットアップ.....	2
フレームグラバーのインストール.....	2
X8800 SDKのインストール.....	2
カメラシステムアセンブリ.....	3
デモプログラムの実行.....	3
デモプログラムの機能.....	5
ソースコードサンプル.....	7
アプリケーションプログラミングインターフェース.....	8
トラブルシューティング.....	17
エラーメッセージ.....	17
症状のリスト.....	17
サポート.....	18

はじめに

放射線の危険に関する注意

X8800シリーズのカメラは、エックス線も含めて放射線と共に使用されるように設計されています。したがって、X8800シリーズのカメラを使用する作業者は有害および/または致命的である放射線被曝の危険性があります。X8800シリーズのカメラを使用する作業者は、放射線安全指針と規則に従って訓練を受けなければなりません。

X8800シリーズのカメラを使用する場合は、すべての放射線安全指針と規則に従わなければなりません。いかなる状況でも放射線安全指針と規則を無視してはいけません。放射線安全指針と規則は、本書のほかのすべての指示に優先します。

本書を読んでX8800シリーズのカメラを使用するX8800シリーズのカメラのユーザーは、X8800シリーズのカメラの使用に関して不慮の放射線被曝があっても、X-Scan Imaging Corporation、その所有者、子会社、その従業員は、責任を負わないことを承認するものとします。

本書の目的

本書の目的は、デモプログラムを動作させてX-Scan ImagingのX8800シリーズのカメラからスキャン画像を取り込んだり、X8800シリーズのカメラを使用するソフトウェアアプリケーションを開発したりする場合に、X-Scan ImagingのX8800ソフトウェア開発キット(SDK)の使用を支援することです。

CやC++のようなプログラミング言語でソフトウェアのプログラミングを行うには、SDKを使用してソフトウェアアプリケーションを開発する必要があります。けれども、デモプログラムを動作させるのにプログラミング経験は必要ありません。

適用範囲

本書は、X-Scan ImagingのX8800シリーズのカメラを使用するために提供されているX-Scan ImagingのX8800 SDKのすべてを扱っています。

はじめに

X8800 SDKには、3つのコンポーネントがあります。第1は、ソフトウェアプログラマがX8800シリーズのカメラを設定して画像データを取り込むための簡単で直感的なアプリケーションプログラミングインターフェイス(API)を提供するX8800 Tool Kit(XTK)ライブラリです。XTKライブラリは、Camera Linkフレームワークに連結して、すべての必要な設定を行います。第2は、XTK APIを使用してスキャン画像を取り込むデモプログラムです。第3は、カメラのコンフィギュレーションとスキャンデータの取り込みを示す簡単なサンプルソースコードです。

クイックスタートガイドのセクションは、X8800シリーズのカメラシステムのアセンブリ、SDKのインストール、デモプログラムの動作について説明しています。その次のセクションは、デモプログラムの機能を説明しています。さらに別のセクションにはサンプルソースコードの記述の説明があります。最後のセクションはXTK APIの説明です。

クイックスタートガイド

このガイドは、カメラシステムをすぐに組み立てて画像スキャンを実行できるように支援するものです。

必要なコンポーネント

このクイックスタートガイドを利用するには以下のコンポーネントが必要です。

- Windows XPオペレーティングシステムのあるコンピュータ。
- ベースコンフィギュレーションのCamera Linkフレームグラバー。
- Camera LinkフレームグラバーのインストールCD。
- X8800インターフェースボックス。
- X8800シリーズカメラヘッド。
- Camera Linkケーブル。
- カメラとインターフェースボックスを接続するX8800ケーブル。
- X8800のACからDCへの電源アダプタ。
- X8800 SDKのインストールCD。

コンピュータのセットアップ

クイックスタートガイドのこのセクションは、データ取り込み用コンピュータへのCamera LinkフレームグラバーのインストールとX-Scan ImagingのX8800 SDKのインストールを扱います。コンピュータにCamera LinkフレームグラバーとSDKがすでにインストールされている場合は、このコンピュータセットアップのセクションをスキップして、以下のカメラシステムのインストールのセクションに進んでください。

フレームグラバーのインストール

X-Scan ImagingのX8800シリーズのカメラシステムは、データ取り込み用コンピュータにCamera Linkフレームグラバーがインストールされている必要があります。使用するフレームグラバーモデルについてフレームグラバーのメーカーのインストールガイドに従ってください。必要な場合は、フレームグラバーをCamera Linkベースコンフィギュレーションに設定してください。

注意: さしあたり、フレームグラバーを初期設定パスにインストールすることが重要です。たとえば、EDTの場合は、C:\¥EDTにインストールします。

X8800 SDKのインストール

フレームグラバーをインストールしたら、X88 SDKソフトウェアをインストールする必要があります。CDIにはインストールする必要のある2つのファイルがあります:

- **lvruntimeeng.msi**: コンピュータにLabviewが入っていない場合は、デモプログラムを動作させるためにこのLabview Runtime Engineをインストールする必要があります。けれども、サンプルプログラムにはこれは不要です。
- **x88_sdk_v*.exe**: これには、サンプルプログラムとデモプログラムの両方のドキュメントがすべて含まれます。インストールが終了したらコンピュータを再起動する必要があります。

カメラシステムアセンブリ



以下の動作を実行する前に、「放射線の危険に関する注意」のセクションを読んでください。

- すべてのシステムコンピュータ、電子回路、エックス線源の電源を切って未接続にします。
- カメラヘッドを適切な位置に取り付けます。エックス線放射シグナルがカメラヘッド表面に垂直に当たって、デテクターラインと直角になるように置かなければなりません。人間と(カメラ電子回路を含む)物体が不慮のエックス線放射被曝を受けるのを避けるために、適切な注意と放射線遮断対策がなされていて、放射線安全指針と規則に従っていることを確認してください。
- 68ピンLVDSケーブルでカメラヘッドとインターフェースボックスを接続します。
- Camera LinkケーブルでインターフェースボックスをCamera Linkフレームグラバーに接続します。Camera LinkケーブルはMDR-26コネクタに接続します。フレームグラバーに複数のMDR-26コネクタがある場合は、ベースコンフィギュレーションに関連するコネクタを使用します。
- ACからDCへの電源アダプタをインターフェースボックスに接続します。
- ACからDCへの電源アダプタを交流電源コンセントに接続します。
- まだ電源を入れないで適切なシステムコンピュータと電子回路を電源に接続します。
- インターフェースボックスの電源スイッチを「ON」の位置へ切り換えます。
- 適切なシステムコンピュータと電子回路の電源を入れます。

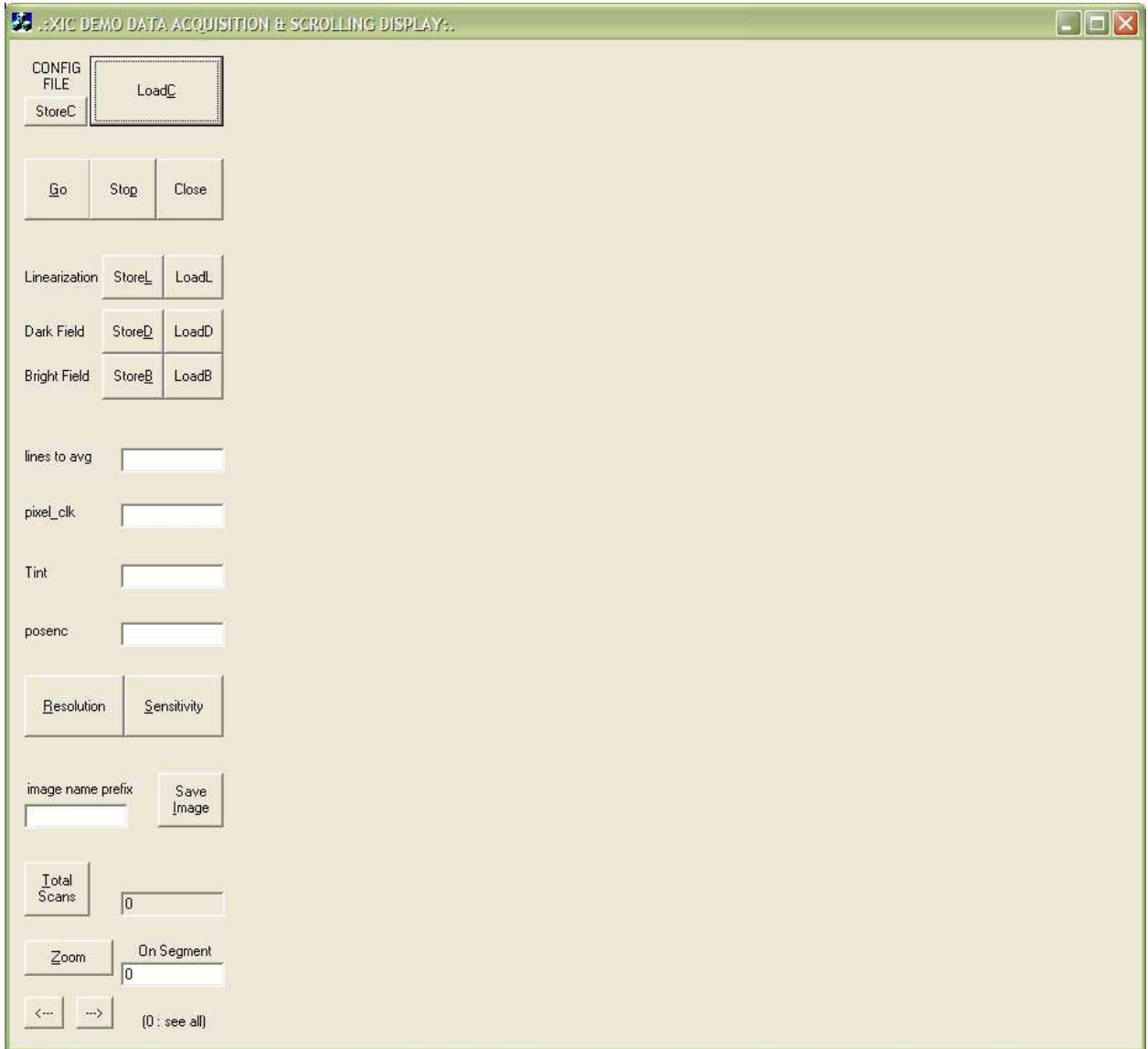
デモプログラムの実行

以下に進む前に、「放射線の危険に関する注意」のセクションを読んでください。以下の動作を実行するとスキャン画像が取り込まれます。

- データ取り込み用コンピュータのオペレーティングシステムが起動したら、スタートメニューをクリックしてXTKデモプログラムを実行します。
- **LoadC**をクリックして、コンフィギュレーションファイルを読み出します。コンフィギュレーションファイルの正しい名称をデータシートでチェックします。
- エックス線源が切断されている状態で、**StoreD**ボタンをクリックします。プログラムはオフセットキャリブレーションを実行します。OKをクリックします。
- **Go**ボタンをクリックしてスキャンを開始します。OKをクリックします。
- 作業者が不慮のエックス線放射被曝を受けることを避けるために適切な注意を払って、放射線安全指針と規則に従いながら、エックス線源の電源を入れます。画像が飽和しないようにエックス線出力を調整します。
- エックス線出力が十分でない場合は、積算時間を増加させる必要があるかもしれません。その場合は、秒の単位で**Tint**の値を変更します。それからステップ3から5までを繰り返します。
- **StoreB**ボタンをクリックします。プログラムはゲインキャリブレーションを実行します。OKをクリックします。
- **Go**ボタンをクリックしてスキャンを開始します。



画像を保存するには、いつでも**Save Image**ボタンをクリックできます。保存される画像は16ビットTIFFフォーマットになり、**Image J** (フリーウェア)のようなプログラムを使用して開くことができます。



(デモプログラムのスクリーンショット)

デモプログラムの機能

このセクションはXTKデモプログラムのボタンについて説明します。また、ボタン名の次の括弧内にこれらのボタンのホットキーも示しています。ディスプレイからAltキーを押すことによってホットキー(アンダーラインが引かれた文字)を表示することができます。

- **LoadC (C)**: コンフィギュレーションファイルを読み出します。コンフィギュレーションファイルはインストールされたフォルダにあります(通常はC: ¥Program Files¥X88 SDK)。カメラに合った正しいファイルを選択していることを確認します。
- **StoreC**: カメラの現在のコンフィギュレーションを次回に使用できるように保存します。
- **Go (G)**: データの取り込みを開始します。表示のスクロールが開始します。すでに2ポイント補正を実行しているかどうかを示すメッセージを表示します。
- **Stop (P)**: データの取り込みを停止します。表示のスクロールが停止します。
- **Close (Esc)**: プログラムを閉じます。
- **StoreL (L)**: 直線化プロセスを実行します。これはカメラの各ピクセルでの多項式補正のための係数を計算します。通常、このプロセスは約1分かかります。以下は直線化を実行する方法です:
 - GUIから望みのTintを設定して、エックス線の電源を入れてGoをクリックします。エックス線シグナルが飽和状態になっていないかあるいは小さすぎないかを確認するためにシグナルレベルをチェックします(これは、画像を取り込んでプロフィールを調べることによって行います)。良好なシグナルが得られるようにエックス線を調整した後(通常、65Kカウントのうちの約40Kカウントです)、次のステップへ進みます。
 - **StoreL**をクリックして直線化プロセスを実行します。補正係数の場所を指定するように尋ねられます。
 - 終了すると、成功/失敗を示すメッセージボックスが表示されます。「Please adjust your Xray」というエラーメッセージが表示されたら第2ステップに戻って、最適な結果になるようにエックス線を調整します。
 - 直線化を終了すると、プログラムは標準の2ポイント補正を実行するように促します。以前の2ポイント補正はこのとき破棄されます。以前の非線形のダークデータと明データを使用することはできません。
- **LoadL**: 直線化をその都度、実行する代わりに、このボタンで以前の係数データを使用することができます。
- **StoreD (D)**: ダークフィールド取り込み。このボタンをクリックすると、プログラムはオフセットキャリブレーションのためのダークフィールドスキャンを取り込んで平均化します。エックス線源はオフにしなければなりません。
- **LoadD**: 以前のダークデータを使用します。
- **StoreB (B)**: 明フィールド取り込み。このボタンをクリックすると、プログラムはゲインキャリブレーションのための明フィールドスキャンを取り込んで平均化します。エックス線源はオンにしなければなりません。ダークフィールドと明フィールドの両方が取り込まれると、ゲインキャリブレーションは自動的に行われます。

- **LoadB**: 以前の明データを使用します。
- **lines to avg**: この機能は、ある数の連続したスキヤンの平均化を実行します。初期設定は1です。
- **pixel_clk**: 現在のピクセルクロックを変更します。プログラムは可能な最も近い値を使用します。
- **Tint**: 積算時間を選択します。積算時間は秒の単位で入力できます。リセット時間と積算時間の合計で、各ラインスキヤンの期間、したがってラインレートが決まります。
- **posenc**: 位置エンコーダカウントを選択します。カメラは位置エンコーダ入力に同期することができます。この位置エンコーダカウント値は、カメラがラインスキヤン単位で待つ位置エンコーダの繰返数を決定します。たとえば、値が1の場合、カメラは位置エンコーダ入力の1サイクルごとに1ラインのスキヤンを実行します。値が2の場合、カメラは位置エンコーダ入力の2サイクルごとに1ラインのスキヤンを実行します。値が0の場合、カメラはリセット時間と積算時間で決まるラインレートに同期しないでフリーランで動作します。
- **Resolution (R)**: 高解像度と低解像度を選択します。このボタンをクリックすると、プログラムは高解像度と低解像度を切り換えます(XB8804とXB8850のみ)。初期設定では高解像度になっています。
- **Sensitivity (S)**: 高感度と低感度を選択します。このボタンをクリックすると、プログラムは高感度と低感度を切り換えます。初期設定では高感度になっています。
- **image name prefix**: 保存される画像の名前のプレフィックスです。
- **Save Image (I)**: 画像保存ファンクションです。このボタンをクリックすると、プログラムは16ビット画像ファイルを保存する場所を尋ねます。
- **Total Scans (T)**: 取り込まれるラインの数です。このボタンをクリックすると、プログラムは現在の取り込みのスキヤンの総数を表示します。
- **Zoom (Z)**: 拡大縮小を切り換えます。このボタンをクリックすると、プログラムはフルカメラビュー(縮小)とシングルセグメントビュー(拡大)を切り換えます。
 - ←: 次のセグメントを左へ移動します。
 - →: 次のセグメントを右へ移動します。
 - **On Segment**: 現在表示されているセグメントの値です。0の値は、プログラムがフルカメラビューモード(縮小)になっていることを示します。

ソースコードサンプル

xtk_simpleは、データ取り込みのステップを示す簡単なプログラムです。コードの詳細については「xtk_simple.cpp」ファイルを参照してください。

「xtk_simple.cpp」ファイルの先頭に、このプリプロセッサマクロ定義があります:

```
#define CONFIGURATION_FILE "XR8804-06.txt"
```

簡単にするために、ここではXR8804-06カメラを使用することにします。このコンフィギュレーションファイル名を、ユーザーのカメラコンフィギュレーションファイル名の位置にいつでも変更することができます。

xtk_simpleプロジェクトをコンパイルして動作させると、ユーザーの入力/出力のためのDOSターミナルが呼び出されます。

以下はこのプログラムの6ステップのリストです(プログラムの開発の場合もほとんどはこのようになるはずです):

- 初期化。このステップで、ユーザーはパラメーターを提供します: `xtk_init`ファンクションのための`char *cfg_file` (コンフィギュレーションファイル名)です。
- コンフィギュレーション。このステップでは`xtk_config`ファンクションのために事前に初期化された`xtk_p`ストラクトを使用します。これは単に、コンフィギュレーションファイルに従ってすべての初期設定値を設定することです。このステップの後で、プログラムはユーザーがこれらの設定値を手動で変更したいかどうか尋ねます。
1を入力してマニュアルコンフィグモードにすると、クロック分割、積算時間、リセット時間などを変更するためのオプションがある選択テーブルが表示されます。
変更を実行するための様々なパラメーターについてはドキュメントを参照してください。
- PCインターフェースの開始。名前で見えるように、このステップは`xtk_start`ファンクションによるEDTのようなpcインターフェースを開始するのに必要なすべてのステップを実行します。
- データ取り込みの実行。このステップでは、ユーザーは`get_line`ファンクションを呼び出す前にデータ配列のためのメモリを割り当てる必要があります。この配列は無符号短精度整数型でなければならず、長さは1配列当たりのピクセル数です(`xtk_get_pixels_per_array`ファンクションを使用して取得することができます)。
スキャンラインを得るには2つのファンクションが利用可能です: `xtk_get_line`と`xtk_get_line_timestamp`です。この2つの違いは、2番目のファンクションが、欠落したスキャンラインをチェックする目的で無符号短精度整数のタイムスタンプも返すことです。
- PCインターフェースの停止。データ取り込みが終了したら、PCインターフェースを正しく終了するためにこの`xtk_stop`ファンクションを使用しなければなりません。これに失敗すると次の取り込みで問題が生じる場合があります。
- 割り当てたメモリすべての解放。プログラムの終了時に`xtk_close`ファンクションを使用して割り当てたメモリを解放するのは常に望ましいことです。

xtk_simpleが終了すると、取り込まれたデータは初期設定の「data.txt」ファイルに保存されます。

ファンクションの記述について詳しくは、以下の「アプリケーションプログラミングインターフェース」のセクションを参照してください。

アプリケーションプログラミングインターフェース

```
XtkDev * xtk_init(char *cfg_file)
```

コンフィギュレーションファイルに基づいてXtkDevストラクトのためのメモリを割り当てます。

パラメーター:

cfg_file コンフィギュレーションファイルの名前/位置

返り値:

成功の場合は初期化されたXtkDevストラクトのアドレス
失敗の場合はNULLストラクト

```
char xtk_config(XtkDev *xtk_p)
```

XtkDevストラクトパラメーターからの情報を使用してすべてのレジスタを設定します。リセット、積算時間、読み出し可能ディレイ、読み出し可能パルス間の関係をチェックします。問題が起こった場合は、エラーコードを返す前にメッセージバッファ`xtk_p->message`にエラーメッセージを書き込みます。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクトへのポインター

返り値:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)

```
char xtk_start(XtkDev *xtk_p)
```

PCインターフェースの処理を開始します。
現在のところXICは、USB、EDT CameraLink、MTX CameraLinkに対応しています。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクトへのポインター

返り値:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)

```
char xtk_get_line(XtkDev *xtk_p,
                 unsigned short *PixelsData)
```

ユーザーが提供した**PixelsData*にデコードされた1スキャンラインを返します。CameraLinkインターフェースの場合、このファンクションは取り込まれた1フレームの中の各ラインを返します。フレーム内のすべてのラインを返した後に、別のフレームのデータを取得します。USBの場合、どれほどのデータが収集されたかを突き止めて、次のスキャンのために余分なものを(存在する場合は)保存します。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクへのポインター
**PixelsData* デコードされたラインを格納するためにユーザーが提供するポインターアレイ

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)

```
char xtk_get_line_timestamp(XtkDev *xtk_p,
                            unsigned short *PixelsData,
                            unsigned short *ln_cnt)
```

ユーザーが提供した**PixelsData*にデコードされた1スキャンラインを返して、*ln_cnt*にラインカウントを返します。CameraLinkインターフェースの場合、このファンクションは取り込まれた1フレームの中の各ラインを返します。フレーム内のすべてのラインを返した後に、別のフレームのデータを取得します。USBの場合、どれほどのデータが収集されたかを突き止めて、次のスキャンのために余分なものを(存在する場合は)保存します。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクへのポインター
**PixelsData* デコードされたラインを格納するためにユーザーが提供するポインターアレイ
**ln_cnt* ラインカウントを格納するためにユーザーが提供する変数

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)

```
void xtk_stop(XtkDev *xtk_p)
```

PCインターフェースの処理を停止します。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクトへのポインター

```
void xtk_close(XtkDev *xtk_p)
```

PCインターフェースを閉じます。割り当てたメモリをすべて解放します。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクトへのポインター

```
char xtk_set_single_reg(XtkDev *xtk_p,  
                        unsigned char Address,  
                        unsigned char Reg,  
                        unsigned char Value)
```

シングルレジスタを設定します (チェックのために書き込んで読み返されます)。

備考: このプログラムはレジスタを設定するだけで、XtkDevストラクト内の関連パラメーターは設定しません。ストラクトを本当に理解していて、どうしたいのかということがわかっていない場合は、これを使用しないでください。これらのファンクションの代わりに、以下のもののほうがユーザーに適している場合があります。

パラメーター:

**xtk_p* 事前に初期化されたXtkDevストラクトへのポインター
Addr 書き込むアドレス、unsigned char型の値
Reg 書き込むレジスタ、unsigned char型の値
Val 書き込む値、unsigned char型の値

返り値:

成功の場合はSUCCESS (0)

失敗の場合はFAILURE (1)

Xtk_pストラクトがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_rst_time(XtkDev *xtk_p,  
                     unsigned char mantissa,  
                     unsigned char exponent)
```

仮数および指数フォーマットを使用してリセット時間を設定します。

リセット時間を計算する公式は以下のようになります:

$$\text{Trst} = \text{mantissa} * (2 \wedge \text{exponent}) / (4 * \text{pixel_clk})$$

仮数と指数が小さすぎる場合：

```
mantissa * (2^exponent) < RST_TIME_MIN = 10
```

仮数は3に変更され、指数は0に変更されます。

指数が大きすぎる場合：

```
exponent > MAX_EXP = 8
```

指数は8に変更されます。

備考：リセット、積算時間、読み取り可能ディレイ、読み取り可能パルスの中のタイミング関係がうまく合わない場合にリセットを減少させると問題が起こることがあります。

パラメーター：

<i>*xtk_p</i>	事前に初期化されたXtkDevストラクツへのポインター
<i>mantissa</i>	仮数、unsigned char型の値
<i>exponent</i>	指数、unsigned char型の値

返り値：

成功の場合はSUCCESS (0)

失敗の場合はFAILURE (1)

Xtk_pストラクツがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_int_time(XtkDev *xtk_p,  
                    unsigned char mantissa,  
                    unsigned char exponent)
```

仮数および指数フォーマットを使用して積算時間を設定します。

積算時間を計算する公式は以下のようになります：

```
Tint = mantissa * (2 ^ exponent) / (4*pixel_clk)
```

指数が大きすぎる場合：

```
exponent > MAX_EXP = 8
```

指数は8に変更されます。

このファンクションには、組み込みのチェックタイミングがあります。リセット、積算時間、読み取り可能ディレイ、読み取り可能パルスの中のタイミング関係がうまく合わない場合はFAILURE (1)を返します。

パラメーター：

<i>*xtk_p</i>	事前に初期化されたXtkDevストラクツへのポインター
<i>mantissa</i>	仮数、unsigned char型の値
<i>exponent</i>	指数、unsigned char型の値

返り値：

成功の場合はSUCCESS (0)

失敗の場合はFAILURE (1)

Xtk_pストラクツがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_pixel_clk(XtkDev *xtk_p, float value)
```

ピクセルクロックを設定します。

備考：ユーザーが提供する値は望みの値にすぎないので、プログラムは可能な最も近い値を計算して設定します。実際の値を読み返すには関数`xtk_get_pixel_clk`を使用してください。

パラメーター:

<code>*xtk_p</code>	事前に初期化されたXtkDevストラクへのポインター
<code>value</code>	望みのピクセルクロック、float型の値

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)
Xtk_pストラクがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_posenc(XtkDev *xtk_p, unsigned char value)
```

位置エンコーダを設定します。

この値は、スキャンラインあたりに要求される位置エンコーダクロック周期を決定します。

パラメーター:

<code>*xtk_p</code>	事前に初期化されたXtkDevストラクへのポインター
<code>value</code>	新たな位置エンコーダ、unsigned char型の値

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)
Xtk_pストラクがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_gain(XtkDev *xtk_p, char gns)
```

ゲインを設定します。1は「High」を、0は「Low」を意味します。

パラメーター:

<code>*xtk_p</code>	事前に初期化されたXtkDevストラクへのポインター
<code>gns</code>	Hi/Loの設定、0または1

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)

Xtk_pストラクトがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_sensitivity(XtkDev *xtk_p, char sns)
```

感度を設定します。1は「High」を、0は「Low」を意味します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクトへのポインター
sns Hi/Loの設定、0または1

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)
Xtk_pストラクトがNULLの場合はNULL_XTK_STRUCT (-1)

```
char xtk_set_resolution(XtkDev *xtk_p, bool res)
```

解像度の設定をします。1は「High」を、0は「Low」を意味します。
このファンクションは、このモードに応じてピクセルの数の設定もします。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクトへのポインター
res Hi/Loの設定、0または1

返回值:

成功の場合はSUCCESS (0)
失敗の場合はFAILURE (1)
Xtk_pストラクトがNULLの場合はNULL_XTK_STRUCT (-1)

```
float xtk_get_rst_time(XtkDev *xtk_p)
```

リセット時間をfloat型で返します。
xtk_pストラクトがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクトへのポインター

```
float xtk_get_int_time(XtkDev *xtk_p)
```

積算時間をfloat型で返します。

xtk_pストラクツがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

***xtk_p** 事前に初期化されたXtkDevストラクツへのポインター

```
float xtk_get_rden_delay(XtkDev *xtk_p)
```

読み出し可能ディレイをfloat型で返します。

xtk_pストラクツがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

***xtk_p** 事前に初期化されたXtkDevストラクツへのポインター

```
float xtk_get_rden_pulse(XtkDev *xtk_p)
```

読み出し可能パルスをfloat型で返します。

xtk_pストラクツがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

***xtk_p** 事前に初期化されたXtkDevストラクツへのポインター

```
char xtk_get_resolution(XtkDev *xtk_p)
```

解像度の設定を返します。1は高解像度、0は低解像度です。

xtk_pストラクツがすでに初期化されている場合にのみこのファンクションを使用します。

パラメーター:

***xtk_p** 事前に初期化されたXtkDevストラクツへのポインター

```
char xtk_get_sensitivity(XtkDev *xtk_p)
```

感度の設定を返します。1は高感度、0は低感度です。

Xtk_pストラクツがすでに初期化されている場合にのみこのファンクションを使用します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクツへのポインター

```
char xtk_get_gain(XtkDev *xtk_p)
```

ゲインの設定を返します。1は高ゲイン、0は低ゲインです。

Xtk_pストラクツがすでに初期化されている場合にのみこのファンクションを使用します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクツへのポインター

```
long xtk_get_pixels_per_array(XtkDev *xtk_p)
```

デテクターアレイ当たりのピクセル数をlong型で返します。

xtk_pストラクツがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクツへのポインター

```
float xtk_get_pixel_clk(XtkDev *xtk_p)
```

ピクセルクロック周波数をfloat型で返します。

xtk_pストラクツがNULLの場合、負の数NULL_XTK_STRUCT (-1)を返します。

パラメーター:

*xtk_p 事前に初期化されたXtkDevストラクツへのポインター

```
char * xtk_get_info(XtkDev *xtk_p)
```

スキャナのメーカー、スキャナのモデル、ソフトウェアのバージョンに関する情報を示すポインターを3行文字列で返します。

xtk_pストラクツがNULLの場合、負の数を返します。

例:

```
ScannerManufacturer: XIC  
ScannerModel: XR8804  
SoftwareVersion: XIC_DEMO v1.0
```

パラメーター:

**xtk_p*

事前に初期化されたXtkDevストラクツへのポインター

トラブルシューティング

エラーメッセージ

以下は一般的なエラーメッセージと考えられる原因のリストです。

エラーメッセージ	考えられる原因
FPGA CONFIG FAILURE	DC電源が供給されていない/電源スイッチがオンになっていない。 インターフェースボックスがCamera LinkケーブルによってCamera Linkフレームグラバに接続されていない。
	Camera Linkフレームグラバがコンピュータに正しくインストールされていない。
ADC CONFIG FAILURE	カメラヘッドがインターフェースボックスに接続されていない。 カメラモデルが正しく選択されていない。
Fail to perform xtk_init	EDT CameraLinkソフトウェアがコンピュータに適切にインストールされていない。 EDT Camera Linkコンフィギュレーションファイルが適切なディレクトリにインストールされていない。 この問題を解決するために、環境変数を変更してコンフィギュレーションファイルのパスを修正することができます。 初期設定: EDT_INITCAM: C:¥EDT¥pdv¥initcam.exe EDT_CONFIG: C:¥EDT¥pdv¥camera_config
	コンピュータ内のフレームグラバモデルに合わないソフトウェアをインストールした。ソフトウェアがフレームグラバモデルに合っているかどうかチェックしてください。
	XTKライブラリがコンピュータに適切にインストールされていない。
Unable to get display and acquisition resources	ユーザーがGoボタンをクリックしたときに取り込みと表示がすでに開始している。 Stopボタンをクリックして、やり直してください。
Unable to open image	デモプログラムで取り込まれた16ビットTIFF画像を開くプログラムが必要です。Image Jは、こうした画像を開くことができるフリーのソフトウェアです。

症状のリスト

以下は一般的な症状と考えられる原因のリストです。

症状	考えられる原因
デモプログラムのビデオ表示が灰色で変化がない。	取り込みが開始していない。Goボタンをクリックしてください。
デモプログラムのビデオ表示が黒い。	カメラヘッドに適切な角度で十分な量のエクス線放射が当たっていない。 積算時間が短すぎる。

サポート

X-Scan Imaging Corporationの技術支援が必要な場合は、support@x-scanimaging.comへ電子メールを送ってください。問題の説明、デテクターの型番、シリアルナンバー、インストールしたソフトウェア開発キットのバージョンナンバー、コンピュータのオペレーティングシステムのバージョンをメールに含めてください。