

SVI-03
画像入カライブラリ説明書

V1.03

株式会社ネットビジョン

改定履歴

版数	日付	内容	備考
1.00	2006/06/12	・ SVI-01/02 用画像入力ライブラリより引用	
1.01	2006/09/25	・ SVIUSB20_PIIUpdate API を追加	
1.02	2006/11/16	・ 画像入力ライブラリー使用例を変更 ・ 3.3.29. SVI-03 の設定を追加	
1.03	2008/11/13	・ SVIUSB20_RecStart の REC_PARAM 構造体を変更 ・ SVIUSB20_GetVersion API を追加 ・ SVIUSB20_End API を追加 ・ SVIUSB20_DeviceSelect API を追加 ・ SVIUSB20_DeviceRelease API を追加 ・ SVIUSB20_EnumDevice API を追加 ・ SVIUSB20_PIISet API を追加 ・ 画像入力ライブラリー使用例を変更	
1.04	2010/10/20	・ 「3.3.35. SVI-03 の設定」にて設定内容のミスがあったのを修正	

目次

1. 適用.....	4
2. 概要.....	4
3. 仕様.....	4
3.1. ファイル構成	4
3.2. SVI-03 API 一覧	5
3.3. SVI-03 画像入力ライブラリAPI リファレンス	6
3.3.1. SVIUSB20_Init	6
3.3.2. SVIUSB20_Open	6
3.3.3. SVIUSB20_Close	7
3.3.4. SVIUSB20_BusReset	7
3.3.5. SVIUSB20_DeviceReset	8
3.3.6. SVIUSB20_GetStatus	8
3.3.7. SVIUSB20_GetStatus2	9
3.3.8. SVIUSB20_SetParam	10
3.3.9. SVIUSB20_MonStart	11
3.3.10. SVIUSB20_MonStop	11
3.3.11. SVIUSB20_MonGetHeader	12
3.3.12. SVIUSB20_MonGetFrame	13
3.3.13. SVIUSB20_RecStart	13
3.3.14. SVIUSB20_RecStop	14
3.3.15. SVIUSB20_RecGetHeader	14
3.3.16. SVIUSB20_RecGetFrames	15
3.3.17. SVIUSB20_FirmwareUpdate	17
3.3.18. SVIUSB20_FpgaUpdate	17
3.3.19. SVIUSB20_I2CStop	18
3.3.20. SVIUSB20_I2COneBlockWrite	19
3.3.21. SVIUSB20_I2COneBlockRead	20
3.3.22. SVIUSB20_I2CBufferWrite	21
3.3.23. SVIUSB20_GpioAcc	22
3.3.24. SVIUSB20_PllUpdate	22
3.3.25. SVIUSB20_PllSet	23
3.3.26. SVIUSB20_GetVersion	23
3.3.27. SVIUSB20_End	24
3.3.28. SVIUSB20_DeviceSelect	24

3.3.29.	SVIUSB20_DeviceRelease	24
3.3.30.	SVIUSB20_EnumDevice.....	25
3.3.31.	モニタリングモード時の画像入力ライブラリ使用例.....	26
3.3.32.	レコーディングモード時の画像入力ライブラリ使用例	28
3.3.33.	I2C によるコマンド送信時の画像入力ライブラリ使用例	30
3.3.34.	I2C によるコマンド受信時の画像入力ライブラリ使用例	31
3.3.35.	複数台の SVI-03 を使用する場合のオープン/クローズ方法	32
3.3.36.	SVI-03 の設定	33

1. 適用

本説明書は SVI-03 に適用します。

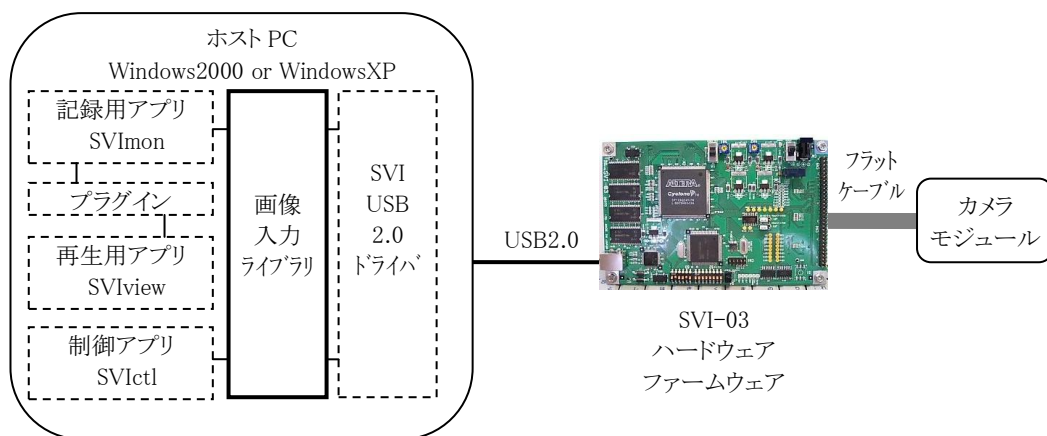
2. 概要

SVI-03 とはカメラ・モジュールを評価する Windows 上のソフトウェアとハードウェア及びファームウェアから構成されます。

本説明書では SVI-03 を USB2.0 を介して接続、制御するための画像入力ライブラリについて記述します。

画像入力ライブラリを使用することで、カメラモジュールからのモニタリング画像、レコーディング画像を取り込むことができます。また、カメラモジュール制御として I²C 通信をサポートします。

【図1】システム構成図



3. 仕様

本ライブラリは SVI USB2.0 ドライバを呼び出し SVI-03 より画像を入力するためのライブラリです。アプリケーションからは本ライブラリを使用して制御します。本ライブラリ内 API をコールすることにより SVI-03 のパラメータの設定、ステータスの取得及び画像データの受信を実現します。

アプリケーションでは、本ライブラリ内 API を呼び出し、I²C 通信でカメラモジュールの制御や汎用ポートの制御を行うことができます。

3.1. ファイル構成

本ライブラリは以下のファイルを提供します。

- SVIUSB20.INF (SDK-CD の Driver フォルダに格納)
SVI USB2.0 ドライバインストール情報ファイル。
- SVIUSB20.SYS (SDK-CD の Driver フォルダに格納)
SVI USB2.0 ドライバ本体 Windows ディレクトリ下”System32¥drivers”にコピーされ使用されます。
- SVIUSB20.H (SDK-CD の“SVI-source¥SVI ライブラリ¥画像入力”フォルダに格納)
本ライブラリを使用する際に必要なインクルードファイルです。
- SVIUSB20.DLL (SDK-CD の“Appl”フォルダに格納)
本ライブラリです。
- SVIUSB20.LIB (SDK-CD の“SVI-source¥SVI ライブラリ¥画像入力”フォルダに格納)
本ライブラリリンクモジュールです。

3.2. SVI-03 API 一覧

A P I 名	機能
SVIUSB20_Init	SVI-03 画像入カライブラリを初期化します
SVIUSB20_Open	SVI ドライバをオープンします
SVIUSB20_Close	SVI ドライバをクローズします
SVIUSB20_BusReset	SVI-03 にバスリセットを発行します
SVIUSB20_Dev_Reset	SVI-03 を初期化します
SVIUSB20_GetStatus	SVI-03 の現在のステータスを取得します
SVIUSB20_GetStatus2	SVI-03 の現在の基本ステータスのみを取得します
SVIUSB20_SetParam	SVI-03 にハードウェア設定情報を通知します
SVIUSB20_MonStart	モニタリングを開始します
SVIUSB20_MonStop	モニタリングを停止します
SVIUSB20_MonGetHeader	モニタリング中、フレームヘッダを取得します
SVIUSB20_MonGetFrame	モニタリング中、フレームデータを受信します
SVIUSB20_RecStart	レコーディングを開始します
SVIUSB20_RecStop	レコーディングを停止します
SVIUSB20_RecGetHeader	レコーディング中、ヘッダを取得します
SVIUSB20_RecGetFrames	レコーディング中、全フレームデータを受信します
SVIUSB20_FirmwareUpdate	ファームウェアのアップデートを行います
SVIUSB20_FpgaUpdate	FPGA コンフィグレーション・データのアップデートを行います
SVIUSB20_I2CStop	I2C ストップコンディションを通知します
SVIUSB20_I2COneBlockWrite	I2C で 1 ブロックを送信します
SVIUSB20_I2COneBlockRead	I2C で 1 ブロックを受信します
SVIUSB20_I2CBufferWrite	I2C の Write コマンドをバッファリングし、コマンド群を一括でカメラモジュールへ送信します
SVIUSB20_GpioAcc	SVI-03 ボードの GPIO ポートをアクセスします
SVIUSB20_PllUpdate	PLL データのアップデートを行います
SVIUSB20_PllSet	PLL データの一時アップデートを行います（電源断まで有効）
SVIUSB20_GetVersion	SVIUSB20.DLL のバージョン情報を取得します
SVIUSB20_End	本ライブラリの終了処理を行います
SVIUSB20_DeviceSelect	使用する SVI-03 ボードを指定します
SVIUSB20_DeviceRelease	使用指定した SVI-03 ボードを開放します
SVIUSB20_EnumDevice	SVI ボードの接続台数及び接続された SVI ボードの番号を接続台数分取得します

3.3. SVI-03 画像入力ライブラリーAPI リファレンス

3.3.1. SVIUSB20_Init

API SVIUSB20_Init

機能 SVI-03 画像入力ライブラリーサポート・ライブラリの内部変数を初期化します

プロトタイプ

```
void SVIUSB20_Init( void );
```

戻り値

なし

備考

・必ず最初に呼び出して下さい。(SVIUSB20_Open API よりも！)

3.3.2. SVIUSB20_Open

API SVIUSB20_Open

機能 SVI シリーズ専用 USB2.0 デバイスドライバをオープンします

プロトタイプ

```
DWORD SVIUSB20_Open (
    ULONG          ulAppWho,          // オープン元のアプリケーションを指定
                                     // SVIUSB20_APP_WHO_REC (表示アプリ用)
                                     // SVIUSB20_APP_WHO_CTL (制御アプリ用)
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_DEVOPEN	デバイスドライバをオープンできません
SVIUSB20_RET_ERROR_MULTIOOPEN	同じアプリケーションからは2重にオープンできません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

3.3.3. SVIUSB20_Close

API SVIUSB20_Close

機能 SVI シリーズ専用 USB2.0 デバイスドライバをクローズします

プロトタイプ

```

DWORD SVIUSB20_Close (
    ULONG ulAppWho                // オープン元のアプリケーションを指定
                                // SVIUSB20_APP_WHO_REC (表示アプリ用)
                                // SVIUSB20_APP_WHO_CTL (制御アプリ用)
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります

備考

表示アプリ用でオープンした場合は、表示アプリ用でクローズしてください

3.3.4. SVIUSB20_BusReset

API SVIUSB20_BusReset

機能 SVI-03 に USB リセットを発行します

プロトタイプ

```

DWORD SVIUSB20_BusReset ( void );

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

・SVI-03 ボードとの通信異常時、SVIUSB20_DeviveReset API を発行しても正常な状態にならない場合、本 API を発行して下さい。

3.3.5. SVIUSB20_DeviceReset

API SVIUSB20_DeviceReset

機能 SVI-03 を初期化し、アイドル状態にします

プロトタイプ

DWORD SVIUSB20_DeviceReset (void);

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

SVI-03 との通信異常時、本 API を発行することで通信が回復することがあります。

3.3.6. SVIUSB20_GetStatus

API SVIUSB20_GetStatus

機能 SVI-03 の現在のステータス情報を取得します

プロトタイプ

```

DWORD SVIUSB20_GetStatus (
    PGET_STATUS    pStatus    // ステータス情報構造体のポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

SVIUSB20_RET_ERROR_PARAMETER

引数に間違いがあります

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```

typedef    _GET_STATUS {
    ULONG    ulBasicStatus;        // SVI ハードウェアの基本ステータス(詳細は備考参照)
    ULONG    ulHWVersion;         // ハードウェアのバージョン番号
    ULONG    ulFWVersion;         // ファームウェアのバージョン番号
    ULONG    ulOPStatus;          // 現在の動作モード
                                    // 0:アイドル、1:モニタリング中
                                    // 2:レコーディング中、8:アップデート中
    ULONG    ulStartupStatus;     // ファームウェア起動ステータス(0:ノーマル、1:リカバリ)
    ULONG    ulOrgSizeW;          // カメラが出力している画像サイズの幅
    ULONG    ulOrgSizeH;          // カメラが出力している画像サイズの高さ
    ULONG    ulCutout_x;          // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG    ulCutout_y;          // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG    ulCutSizeW;          // 切り出し後の画像サイズの幅
    ULONG    ulCutSizeH;          // 切り出し後の画像サイズの高さ
    ULONG    ulCamStatus;         // カメラステータス
                                    // D0-7:信号ステータス
                                    // D8-15:デジタル映像データ
                                    // D16-23:汎用ポートデータ
} GET_STATUS, *PGET_STATUS;

```

備考

実行結果として SVI-03 の基本ステータスのみを取得したい場合は SVIUSB20_GetStatus2 API を使用して下さい。
SVI ハードウェアの基本ステータスの一覧を以下に示します。

SVL_STS_SUCCESS	: 正常終了
SVL_STS_FRAMEPENDING	: 正常終了 (保留フレーム/データあり)
SVL_STS_CAPCANCELED	: 正常終了 (キャプチャが中止された)
SVL_STS_BUSY	: ビジーでコマンドが実行できない
SVL_STS_RECOVERYMODE	: リカバリーモードのためコマンドが実行できない
SVL_STS_I2CACTIVE	: I2C コントローラアクティブ
SVL_STS_CMD_INVALID	: コマンドが不正である
SVL_STS_PRM_INVALID	: パラメータが不正である
SVL_STS_SEQ_INVALID	: パケットの発行シーケンスが不正である
SVL_STS_I2C_ACKTIMEOUT	: I2C でスレーブからの ACK を受信できずタイムアウトが発生した
SVL_STS_I2C_PRETIMEOUT	: I2C でプリタイムアウトが発生した
SVL_STS_I2C_POSTTIMEOUT	: I2C でポストタイムアウトが発生した
SVL_STS_USB_ERROR	: USB 通信時にエラーが発生した
SVL_STS_UPDATE_INVALID	: モジュールデータが不正である
SVL_STS_FROMERS_ERROR	: フラッシュメモリーの消去に失敗した
SVL_STS_FROMWT_ERROR	: フラッシュメモリーの書き込みに失敗した
SVL_STS_INTERNAL_ERROR	: 内部エラーが発生した
SVL_STS_RESOURCE_ERROR	: 内部リソースが不足して処理が実行できない

3.3.7. SVIUSB20_GetStatus2

API SVIUSB20_GetStatus2

機能 SVI-03 の現在の基本ステータスのみを取得します

プロトタイプ

```
DWORD SVIUSB20_GetStatus2 (
    PULONG    pStatus    // 基本ステータス格納ポインタ (詳細は SVIUSB20_GetStatus と同様です)
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

3.3.8. SVIUSB20_SetParam

API SVIUSB20_SetParam

機能 SVI-03 のハードウェア設定を行います

プロトタイプ

```
DWORD SVIUSB20_SetParam (
    PSET_PARAM    pParam           // ハードウェア設定構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _SET_PARAM {
    ULONG    ulParamBitFlag;           // 以下のパラメータの有効(1)/無効(0)を指定する
                                         // SVIUSB20_SETPARAM_BIT0 : 画像情報通知
                                         // SVIUSB20_SETPARAM_BIT1 : カメラ電源スイッチ
                                         // SVIUSB20_SETPARAM_BIT2 : I2C スピード
                                         // SVIUSB20_SETPARAM_BIT3 : 予約
                                         // SVIUSB20_SETPARAM_BIT4 : 予約
                                         // SVIUSB20_SETPARAM_BIT5 : VSYNC デアサートフラグ
                                         // SVIUSB20_SETPARAM_BIT6 : モニタリングモード
                                         // 複数指定は OR 演算子を使用
    ULONG    ulVsyncFlag;             // VSYNC デアサートフラグ ~ 初期値:0
                                         // 0 : デアサートを待つ
                                         // 1 : デアサートを待たない
    ULONG    ulCamPower;              // カメラ電源スイッチ(0:OFF、1:ON) ~ 初期値:1
    ULONG    ulI2CSpeed;              // I2C スピード(0:400Kbps、1:200Kbps、2:100Kbps) ~ 初期値:0
    ULONG    ulMonMode;               // モニタリングモード(0 : ダブルバッファ、1 : リングバッファ) ~ 初期値:0
    ULONG    ulPixellInfo;            // 画像情報通知bit0-1 : 色成分数(1or2or3)
                                         // bit2 : 1 色分のビット幅(0:8bit,1:16bit)
} SET_PARAM, *PSET_PARAM;
```

備考

VSYNC デアサートフラグの説明:

0 の場合、VSYNC(VD) がアクティブになったらフレームを認識します。

1 の場合、VSYNC(VD) がアクティブになる前に指定のサイズまで取り込んだらフレームを認識します。

モニタリングモードの説明:

ダブルバッファモードでは SDRAM を 2 区画に分けて交互にフレームを格納し、交互に PC へ転送します。バッファが空いていなければ、そのフレームは廃棄されます。フレーム格納領域が2つしかないのでリアルタイム表示が可能となります。

リングバッファモードでは SDRAM を最大 32 区画に分けてフレームを順次格納し、先頭から順次 PC へ転送します。バッファが空いていなければ、そのフレームは廃棄されます。USB 転送レートよりフレーム格納レートが速ければバッファは常にフル状態となり、PC へ転送されるフレーム最大32フレーム前のフレームが転送されることになります。このモードは先頭十数フレームを取りこぼしたくない時、使用することをおすすめします。

画像情報通知の説明:

YUV-8 ビット取り込みの場合、色成分数は 2、ビット幅は 0 を指定します。

YUV-16 ビット取り込みの場合、色成分数は 2、ビット幅は 1 を指定します。

RAW-8 ビット取り込みの場合、色成分数は 1、ビット幅は 0 を指定します。

RAW-10 ビット取り込みの場合、色成分数は 1、ビット幅は 1 を指定します。

3.3.9. SVIUSB20_MonStart

API SVIUSB20_MonStart

機能 SVI-03 にモニタリング開始を通知します

プロトタイプ

```
DWORD SVIUSB20_MonStart (
    PMONITOR_PARAM pMonParam    // モニタリング情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _MONITOR_PARAM {
    ULONG ulCutout_x;    // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG ulCutout_y;    // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG ulCutout_w;    // 切り出し範囲の幅(偶数)
    ULONG ulCutout_h;    // 切り出し範囲の高さ
} MONITOR_PARAM, *PMONITOR_PARAM;
```

備考

・本 API の発行前に、SVIUSB20_GetStatus API を発行し、SVI-03 がアイドル状態であることを確認して下さい。

3.3.10. SVIUSB20_MonStop

API SVIUSB20_MonStop

機能 SVI-03 にモニタリング停止を通知します

プロトタイプ

```
DWORD SVIUSB20_MonStop ( void );
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

・本 API の発行前に、SVIUSB20_GetStatus2 API を発行し、SVI-03 の基本ステータスを取得して下さい。基本ステータスが SVLSTS_FRAMEPENDING(正常終了(保留フレーム/データあり))の場合、保留フレームが SVI-03 上に残っているのでヘッダ、画像データを全部取り込んで下さい。

3.3.11. SVIUSB20_MonGetHeader

API SVIUSB20_MonGetHeader

機能 SVI-03 よりモニタリングヘッダ情報を取得します

プロトタイプ

```
DWORD SVIUSB20_MonGetHeader (
    PMON_HEADER    pHeader    // ヘッダ情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _MON_HEADER {
    ULONG    ulStatus;                // 実行結果のステータス
                                           // (詳細は SVIUSB20_GetStatus の基本ステータスと同様)
    ULONG    ulPendingInfo;           // 保留フレーム数を返す
    ULONG    ulOrgSizeW;               // カメラが出力している画像サイズの幅
    ULONG    ulOrgSizeH;               // カメラが出力している画像サイズの高さ
    ULONG    ulCutout_x;               // オリジナル画像の左上端からの切り出し範囲の左上端の X 座標
    ULONG    ulCutout_y;               // オリジナル画像の左上端からの切り出し範囲の左上端の Y 座標
    ULONG    ulMonSizeW;               // 切り出し後の画像サイズの幅
    ULONG    ulMonSizeH;               // 切り出し後の画像サイズの高さ
    ULONG    ulFrameRate;              // カメラが出力している画像のフレームレート
    ULONG    ulLoseFrame;              // 前フレームから本フレーム間に破棄したフレーム数
    ULONG    ulOpticalAxisInfo;        // 予約
    USHORT    usCursorX;               // 予約
    USHORT    usCursorY;               // 予約
    USHORT    usWakuLeftX;             // 予約
    USHORT    usWakuLeftY;             // 予約
    USHORT    usWakuRightX;            // 枠の右下の X 座標
    USHORT    usWakuRightY;            // 枠の右下の Y 座標
    ULONG    ulReserve[1];             // 予約
    ULONG    ulVsyncReadLen;           // V 同期読み出しのデータ長 (0~448)
    ULONG    ulVsyncReadVal[448/4];    // V 同期読み出しのデータ
} MON_HEADER, *PMON_HEADER;
```

備考

- ・構造体メンバ ulPendingInfo (保留フレーム数) が 0 の時、既にモニタリング停止通知済みであれば全フレームを受信したことを表します。モニタリング停止未通知であれば何らかのエラーが発生しています。戻り値を確認して下さい。
- ・構造体メンバ ulPendingInfo (保留フレーム数) が 1 または 2 の時、引き続き画像データの受信をして下さい。
- ・画像データが YUV-8bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulCutSizeW} \times 2) \times \text{ulCutSizeH}$$
- ・画像データが YUV-16bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulCutSizeW} \times 2) \times \text{ulCutSizeH}$$
- ・画像データが RAW-8bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulCutSizeW} \times 1) \times \text{ulCutSizeH}$$
- ・画像データが RAW-10bit のバイト数は以下の式で求められます。

$$\text{画像データのバイト数} = (\text{ulCutSizeW} \times 21) \times \text{ulCutSizeH}$$

3.3.12. SVIUSB20_MonGetFrame

API SVIUSB20_MonGetFrame

機能 SVI-03 よりモニタリング・フレームデータを受信します

プロトタイプ

```

DWORD SVIUSB20_MonGetFrame (
    LPVOID    lpFrameBuf,    // フレームデータ格納バッファのポインタ
    ULONG     ulRcvLen,      // 画像データのバイト数(64 の倍数)
    PULONG    pulRetRcvLen   // 実際に読み込んだバイト数を格納するポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります(ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

ulRcvLen と*pulRetRcvLen が異なるときは何らかのエラーが発生しています。

3.3.13. SVIUSB20_RecStart

API SVIUSB20_RecStart

機能 SVI-03 にレコーディング開始を通知します

プロトタイプ

```

DWORD SVIUSB20_RecStart (
    PRECORD_PARAM    pRecParam    // レコーディング情報構造体のポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```

typedef struct _RECORD_PARAM {
    ULONG    ulFrameCount;    // レコーディングフレーム数
    ULONG    ulBufferCount;   // レコーディングサイズ(バイト)
    ULONG    ulRecMode;       // レコーディングモード
                                // bit31 : 0=Recording, 1=Monitoring&Save Mem
                                // bit30-01 : Reserve
                                // bit00 : 0=フレーム数, 1=サイズ
} RECORD_PARAM, *PRECORD_PARAM;

```

備考

・本 API の発行前に、SVIUSB20_GetStatus API を発行し、SVI-03 がアイドル状態であることを確認して下さい。

3.3.14. SVIUSB20_RecStop

API SVIUSB20_RecStop

機能 SVI-03 にレコーディング停止を通知します

プロトタイプ

DWORD SVIUSB20_RecStop (void);

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

・本 API の発行前に、SVIUSB20_GetStatus2 API を発行し、SVI-03 の基本ステータスを取得して下さい。基本ステータスが SVI_STS_FRAMEPENDING(正常終了(保留フレーム/データあり))の場合、保留レコーディング・データが SVI-03 上に残っているのでヘッダ、画像データを全部取り込んで下さい。

3.3.15. SVIUSB20_RecGetHeader

API SVIUSB20_RecGetHeader

機能 SVI-03 よりレコーディング・ヘッダ情報を取得します

プロトタイプ

```
DWORD SVIUSB20_RecGetHeader (
    PREC_HEADER pHeader // ヘッダ情報構造体のポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL

正常終了

SVIUSB20_RET_ERROR_PARAMETER

引数に間違いがあります

SVIUSB20_RET_ERROR_NOOPEN

オープンされていません

その他

Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

構造体

```
typedef struct _REC_HEADER {
    ULONG    ulStatus;                // 実行結果のステータス
                                           // (詳細は SVIUSB20_GetStatus の基本ステータスと同様)
    ULONG    ulPendingInfo;          // 保留レコーディング・データサイズを返す
    UBYTE    ubFirmVersion;          // ファームウェアバージョン番号
    UBYTE    ubHardVersion;          // ハードウェアバージョン番号
    USHORT   usNumChannel;           // チャンネル数(1h 固定)
    UBYTE    ubCompFlag;             // 圧縮フラグ(0:非圧縮、1:圧縮)
    ULONG    ulNumScan;              // データサイズ
    USHORT   usDataWidth;            // データ幅(0:8bits、1:16bits)
    UBYTE    ubReserve[1];           // 予約
    ULONG    ulRecHeaderSize;        // レコーディングモードが Monitoring&Save Mem 時の
                                           // レコーディングフレームヘッダーサイズ
} REC_HEADER, *PREC_HEADER;
```

備考

・構造体メンバ ulPendingInfo(保留レコーディング・データサイズ)が実際に SVI-03 が保持しているレコーディング・データサイズです。0 の場合は何らかのエラーが発生しています。ulStatus を確認して下さい。

3.3.16. SVIUSB20_RecGetFrames

API SVIUSB20_RecGetFrames

機能 SVI ハードウェアよりレコーディング・データを受信します

プロトタイプ

```

DWORD SVIUSB20_RecGetFrames (
LPVOID    lpFrameBuf    // レコーディング・データ格納バッファのポインタ
ULONG     ulRcvLen,      // 画像データのバイト数(64 の倍数)
PULONG    pulRetRcvLen   // 実際に読み込んだバイト数を格納するポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります(ポインタが NULL)
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)

備考

ulRcvLen と*pulRetRcvLen が異なるときは何らかのエラーが発生しています。

データ・フォーマット(レコーディングモードが Recording の場合)

本APIにより取得したレコーディング・データは、1画素16ビットで上位8ビットに6ビットの汎用ポート情報とVSYNCビット、HSYNCビットが格納され、下位8ビットに画像データが格納されます。

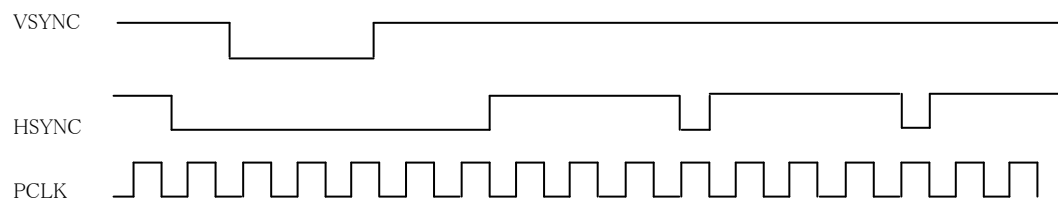
ビット番号															
15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)
PDATA5	PDATA4	PDATA3	PDATA2	PDATA1	PDATA0	HSYNC	VSYNC	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

データの記録開始は、VSYNCの立ち上がりをトリガとしてデータの保存を開始します。フレームの区切りも同じポイントになります。

8ビット入力の場合、PDATA0-PDATA5は不定データとなります。

16ビット入力の場合、PDATA0-PDATA5はD8-D13になり、D14、D15はレコーディングされません。

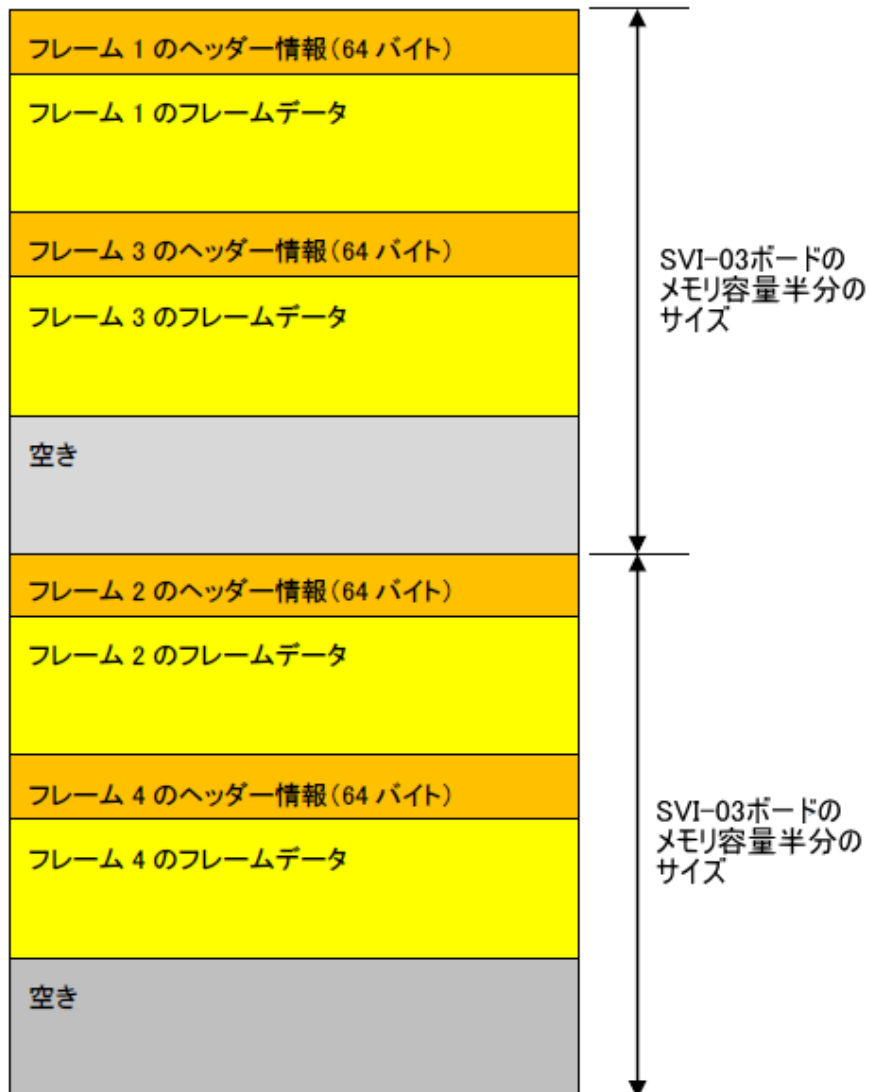
以下に概要例を示します。



bit15-bit12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bit11-bit08	3	1	0	0	0	1	1	3	3	3	1	3	3	1	3
bit07-bit04	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
bit03-bit00	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

データ・フォーマット(レコーディングモードが Monitoring&Save Mem の場合)

レコーディング・データは、下図の構造になっています。



レコーディングモード (Monitoring&Save Mem) は、SVI-03ボードのメモリにモニタリングフレームデータを蓄積します。蓄積方法は、フレーム毎に64バイトのヘッダーとフレームデータのペアでメモリを半分にして、上半分に奇数番目、下半分に偶数番目を蓄積していきます。(前ページの図を参照)
フレーム毎に付属する 64 バイトのヘッダー構造は下表のようになっています。

オフセット	項目	備考
0	フレーム番号(最上位ビットが1になっている)	ビッグエンディアン
4	画像幅	ビッグエンディアン
8	画像高さ	ビッグエンディアン
12	フレームサイズ	ビッグエンディアン
16-63	未使用	

オフセット0のフレーム番号が0の場合は、フレームエンドを示しますので、直前のフレームが最終フレームとなります。フレーム番号は0から始まりますが、フレームエンドと識別するために最上位ビットを1にしていますので、正確なフレーム番号を抽出するには最上位ビットを0にしてください。

各項目は 4 バイト整数ですが、ビッグエンディアンになっていますので、必要に応じてエンディアン変換をして下さい。

3.3.17. SVIUSB20_FirmwareUpdate

API SVIUSB20_FirmwareUpdate

機能 SVI-03 のファームウェア・アップデートを行います

プロトタイプ

```

DWORD SVIUSB20_FirmwareUpdate (
    LPVOID lpFirmData    // ファームウェア・アップデート格納ポインタ
    ULONG ulDataSize;    // アップデータバイトサイズ (64 の倍数)
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります (ポインタがNULL)
SVIUSB20_RET_ERROR_NOTIDLE	SVI-03 がアイドル状態ではありません
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_FMUPDATE_T	書き込み終了待ちでタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

本 API 実行中は、他の API は使用しないで下さい

本 API 実行前に SVI-03 がアイドル状態であることを確認して下さい。

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

3.3.18. SVIUSB20_FpgaUpdate

API SVIUSB20_FpgaUpdate

機能 SVI-03 の FPGA コンフィグレーション・データのアップデートを行います

プロトタイプ

```

DWORD SVIUSB20_FpgaUpdate (
    LPVOID lpFpgaData    // FPGA コンフィグレーション・アップデート格納ポインタ
    ULONG ulDataSize;    // アップデータバイトサイズ (64 の倍数)
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります (ポインタがNULL)
SVIUSB20_RET_ERROR_NOTIDLE	SVI-03 がアイドル状態ではありません
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_FPUUPDATE_T	書き込み終了待ちでタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

本 API 実行中は、他の API は使用しないで下さい

本 API 実行前に SVI-03 がアイドル状態であることを確認して下さい。

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

3.3.19. SVIUSB20_I2CStop

API SVIUSB20_I2CStop

機能 I²C 通信にて I²C 信号ラインをストップコンディションにします

プロトタイプ

DWORD SVIUSB20_I2CStop (void);

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H、bit23-0:基本ステータス)

備考

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

SVIUSB20_I2COneBlockWrite API、SVIUSB20_I2COneBlockRead API が異常終了したときなど、本 API を実行することにより、I²C 信号ラインをアイドルに戻すことができます

3.3.20. SVIUSB20_I2COneBlockWrite

API SVIUSB20_I2COneBlockWrite

機能 SVI-03 に I²C で 1 ブロック送信します

プロトタイプ

```

DWORD SVIUSB20_I2COneBlockWrite (
    ULONG    ulSlaveAdr,        // I2C スレーブアドレス(7bit)
    ULONG    ulLen,             // 送信するコマンドデータ数
    ULONG    ulWriteMode,       // bit0-13 : デレイ時間(0-10000, 単位 10usec で10の倍数で指定)
                                // bit14 : V 同期極性フラグ(0=立上がり, 1=立下り)
                                // bit15 : V 同期送信オンフラグ(0=OFF, 1=ON)
                                // bit16-30 : 予約(常に 0)
                                // bit31 : 再送オン(StopCondition 発行せず)
    PCHAR    pucSendBuf         // 送信コマンドデータバッファのポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

本 API を発行することにより、スタートコンディション、データ送信、ストップコンディションを一回の転送で行うことができます。

ulWriteMode のデレイ時間は V 同期送信オン状態で V 同期信号がアクティブになってからの時間を指定します。指定時間ウェイト後、I²C ブロックを送信します。再送オンは最後のストップコンディションを発行しません。

3.3.21. SVIUSB20_I2COneBlockRead

API SVIUSB20_I2COneBlockRead

機能 SVI-03 に I²C で 1 ブロック送信します

プロトタイプ

```

DWORD SVIUSB20_I2COneBlockRead (
    ULONG    ulSlaveAdr,      // I2C スレーブアドレス(7bit)
    ULONG    ulLen,          // 読み出し要求バイト数
    ULONG    ulReadMode,     // bit0-13 : デレイ時間 (0-10000, 単位 10usec で10の倍数で指定)
                                // bit14 : V 同期極性フラグ(0=立上がり, 1=立下り)
                                // bit15 : V 同期送信オンフラグ(0=OFF, 1=ON)
                                // bit16-30 : 予約(常に 0)
                                // bit31 : 再送オン (Re-StartCondition 発行)
    PCHAR    pucRcvBuf       // 読み出しデータバッファのポインタ
);

```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

本 API を発行することにより、スタートコンディション、データ受信、ストップコンディションを一回の転送で行うことができます。

最後の1バイトのデータ受信時は ACK コードをスレーブデバイスに送信しません。

ulReadMode のデレイ時間は V 同期送信オン状態で V 同期信号がアクティブになってからの時間を指定します。指定時間ウェイト後、I²C ブロックを送信します。再送オンは最後のストップコンディションを発行しません。

3.3.22. SVIUSB20_I2CBufferWrite

API SVIUSB20_I2CBufferWrite
 機能 SVI-03 ボードにて I²C の Write コマンドをバッファリングし、コマンド群を一括でカメラモジュールへ送信します

プロトタイプ

```
DWORD SVIUSB20_I2CBufferWrite (
  ULONG    ulLen,           // 送信バッファバイトサイズ(最大 4096 バイト)
  ULONG    ulWriteMode,     // bit0-13 : デレイ時間(0-10000, 単位 10usec で10の倍数で指定)
                                // bit14 : V 同期極性フラグ(0=立上がり, 1=立下り)
                                // bit15 : V 同期送信オンフラグ(0=OFF, 1=ON)
                                // bit16-31 : 予約(常に 0)
  PCHAR    pucBuffer       // 送信バッファのポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_GSTS2_T	SVIUSB20_GetStatus2 実行でタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

送信バッファに以下のコマンドをセットして、コマンドバッファを作成し、そのバッファを一回の転送でカメラへ I²C で送信することができます。コマンドは2バイト構成になっており、第1バイトでコマンド、第2バイトでデータを指定します。

	第 1,	第 2
①スタートコンディションコマンド	8 0 H,	0 0 H
②ストップコンディションコマンド	6 0 H,	0 0 H
③スレーブアドレス指定コマンド	4 0 H,	スレーブアドレス値 (左 1 ビットシフトしません)
④データ指定コマンド	1 0 H,	データ値

例) 以下のコマンドを送信したい場合:

(START), 3CH, 0AH, 08H, (STOP)

80H, 00H, 40H, 3CH, 10H, 0AH, 10H, 08H, 60H, 00H

ulLen には 0AH を代入します

3.3.23. SVIUSB20_GpioAcc

API SVIUSB20_GpioAcc

機能 SVI-03 ボードの GPIO ポート(入力 8 ポート、出力 8 ポート)をアクセスします

プロトタイプ

```
DWORD SVIUSB20_GpioAcc (
    ULONG    ulMode,          // 入出力フラグ(0:入力、1:出力)
    ULONG    ulBitMask,       // bit7 から bit0 でアクセスしたいポートのみ1をセット、0で無効
    PULONG   pulData          // 入出力データ(bit7 から bit0 が有効)
    ULONG    ulRsv;           // 予約
);
```

戻り値

SVIUSB20_RET_NORMAL 正常終了

SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります

SVIUSB20_RET_ERROR_NOOPEN オープンされていません

その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

その他 デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

3.3.24. SVIUSB20_PllUpdate

API SVIUSB20_PllUpdate

機能 SVI-03 の PLL データのアップデートを行います

プロトタイプ

```
DWORD SVIUSB20_PllUpdate (
    LPVOID    lpPllData        // PLL データ格納ポインタ
    ULONG     ulDataSize;      // アップデータバイトサイズ(64 の倍数)
);
```

戻り値

SVIUSB20_RET_NORMAL 正常終了

SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります(ポインタがNULL)

SVIUSB20_RET_ERROR_NOTIDLE SVI-03 がアイドル状態ではありません

SVIUSB20_RET_ERROR_NOOPEN オープンされていません

SVIUSB20_RET_ERROR_FPUPDATE_T 書き込み終了待ちでタイムアウトが発生しました

その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

その他 デバイスエラー (bit31-24:F1H, bit23-0:基本ステータス)

備考

本 API 実行中は、他の API は使用しないで下さい

本 API 実行前に SVI-03 がアイドル状態であることを確認して下さい。

デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。

本 API は、SVI-03 用ファームウェア V1.01 から有効な API です。また PLL データを変更するために SVI-03 用ブート ROM が PLL 変更を対応している必要があります。

3.3.25. SVIUSB20_PllSet

API SVIUSB20_PllSet

機能 SVI-03 ボードの PLL 設定を一時的に変更します

プロトタイプ

```
DWORD SVIUSB20_PllSet(
    LPVOID lpPllData          // PLL データ格納ポインタ
    ULONG ulDataSize;         // アップデータバイトサイズ(64 の倍数)
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります(ポインタがNULL)
SVIUSB20_RET_ERROR_NOTIDLE	SVI-03 がアイドル状態ではありません
SVIUSB20_RET_ERROR_NOOPEN	オープンされていません
SVIUSB20_RET_ERROR_FPUUPDATE_T	書き込み終了待ちでタイムアウトが発生しました
その他	Win32API エラー (bit31-28:EH、bit27-0:GetLastError 戻り値)
その他	デバイスエラー (bit31-24:F1H、bit23-0:基本ステータス)

備考

本 API 実行中は、他の API は使用しないで下さい
 本 API 実行前に SVI-03 がアイドル状態であることを確認して下さい。
 デバイスエラーの基本ステータスは SVIUSB20_GetStatus の基本ステータスと同様です。
 本 API での PLL 設定の変更は一時的なもので SVI-03 ボードの電源を切るまで有効です。

3.3.26. SVIUSB20_GetVersion

API SVIUSB20_GetVersion

機能 SVIUSB20.DLL のバージョン情報を取得します

プロトタイプ

```
DWORD SVIUSB20_GetVersion(
    char *pcVerBuf          // バージョン番号文字列を格納するポインタ
);
```

戻り値

SVIUSB20_RET_NORMAL	正常終了
SVIUSB20_RET_ERROR_PARAMETER	引数に間違いがあります(ポインタがNULL)

備考

・以下のように文字列ポインタにバージョン番号が格納されます。

例)バージョン番号が 1.0.0.0 の場合

```
*(pcVerBuf+0) = '1' // 0x31
*(pcVerBuf+1) = '.' // 0x2e
*(pcVerBuf+2) = '0' // 0x30
*(pcVerBuf+3) = '.' // 0x2e
*(pcVerBuf+4) = '0' // 0x30
*(pcVerBuf+5) = '.' // 0x2e
*(pcVerBuf+6) = '0' // 0x30
*(pcVerBuf+7) = '\0' // 0x00
```


3.3.27. SVIUSB20_End

API SVIUSB20_End
機能 本ライブラリの終了処理を行います
プロトタイプ
void SVIUSB20_End(void);

戻り値
なし

備考
・必ず最後に呼び出して下さい。

3.3.28. SVIUSB20_DeviceSelect

API SVIUSB20_DeviceSelect
機能 使用するSVI-03 ボードを指定します
プロトタイプ
void SVIUSB20_DeviceSelect (
 ULONG ulBoardNo; // SVI-03 ボード番号(0～3)
);

戻り値
SVIUSB20_RET_NORMAL 正常終了
SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります(ポインタがNULL)

備考
SVI-03 ボード番号はSVIUSB20_EnumDevice API で取得可能です。
SVI-03 画像入力ライブラリ説明書

3.3.29. SVIUSB20_DeviceRelease

API SVIUSB20_DeviceRelease
機能 使用指定したSVI-03 ボードを開放します
プロトタイプ
void SVIUSB20_DeviceRelease ();

戻り値
なし

備考

3.3.30. SVIUSB20_EnumDevice

API SVIUSB20_EnumDevice
 機能 SVI ボードの接続台数及び接続されたSVI ボードの番号を接続台数分取得します
 プロトタイプ

```
DWORD SVIUSB20_EnumDevice (
    ULONG    ulAppWho,          // オープン元のアプリケーションを指定
                                // SVIUSB20_APP_WHO_REC (表示アプリ用)
                                // SVIUSB20_APP_WHO_CTL (制御アプリ用)

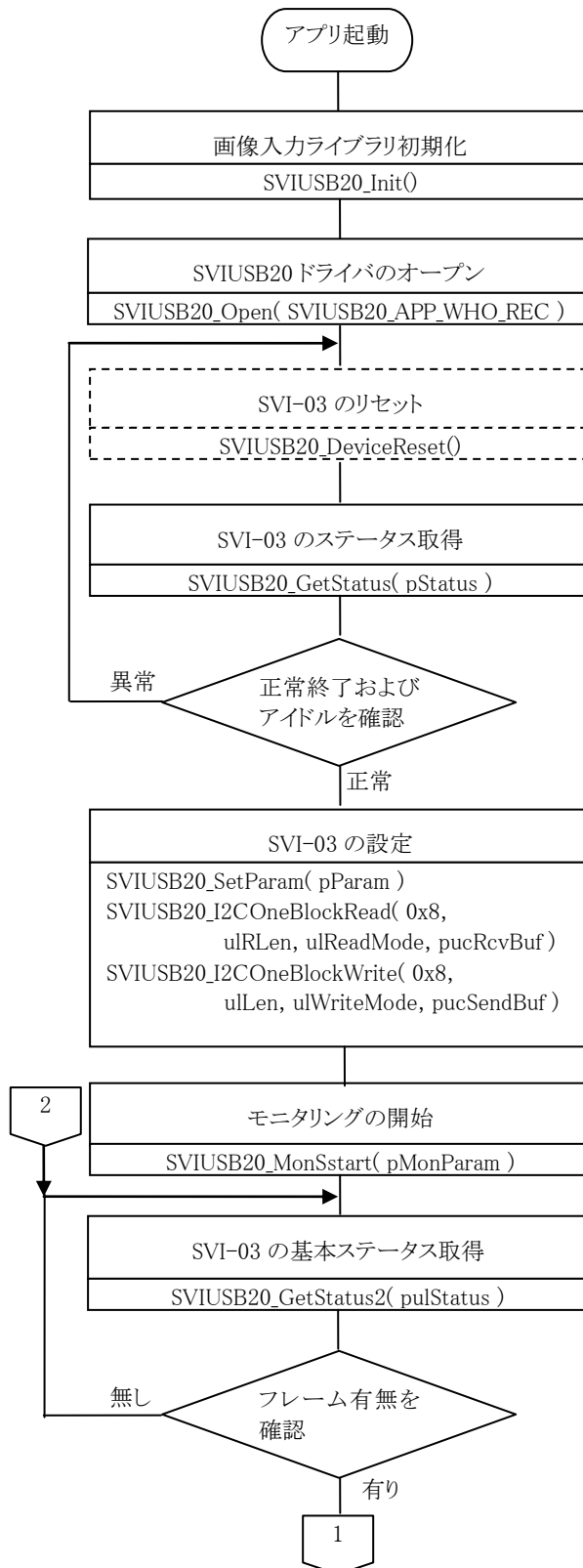
    PULONG    *pulSVI_Num,      // 接続台数を格納するポインタ
    PULONG    *pulSVI_NumTable // 接続台数分のボード番号を格納する配列のポインタ
                                // このポインタがNULL の場合、接続台数のみ格納します
);
```

戻り値
 SVIUSB20_RET_NORMAL 正常終了
 SVIUSB20_RET_ERROR_PARAMETER 引数に間違いがあります
 その他 Win32API エラー (bit31-28:EH, bit27-0:GetLastError 戻り値)

備考

- ・必ずSVIUSB20_Init API の後に呼び出して下さい。
- ・本API は最初に接続台数を検知するために呼び出し、接続台数分の配列(メモリ)を確保した後、再度呼び出しで各SVI ボードのボード番号を取得できます。

3.3.31. モニタリングモード時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要であれば行って下さい。また調子が悪い時にも有効です。(3秒かかります)

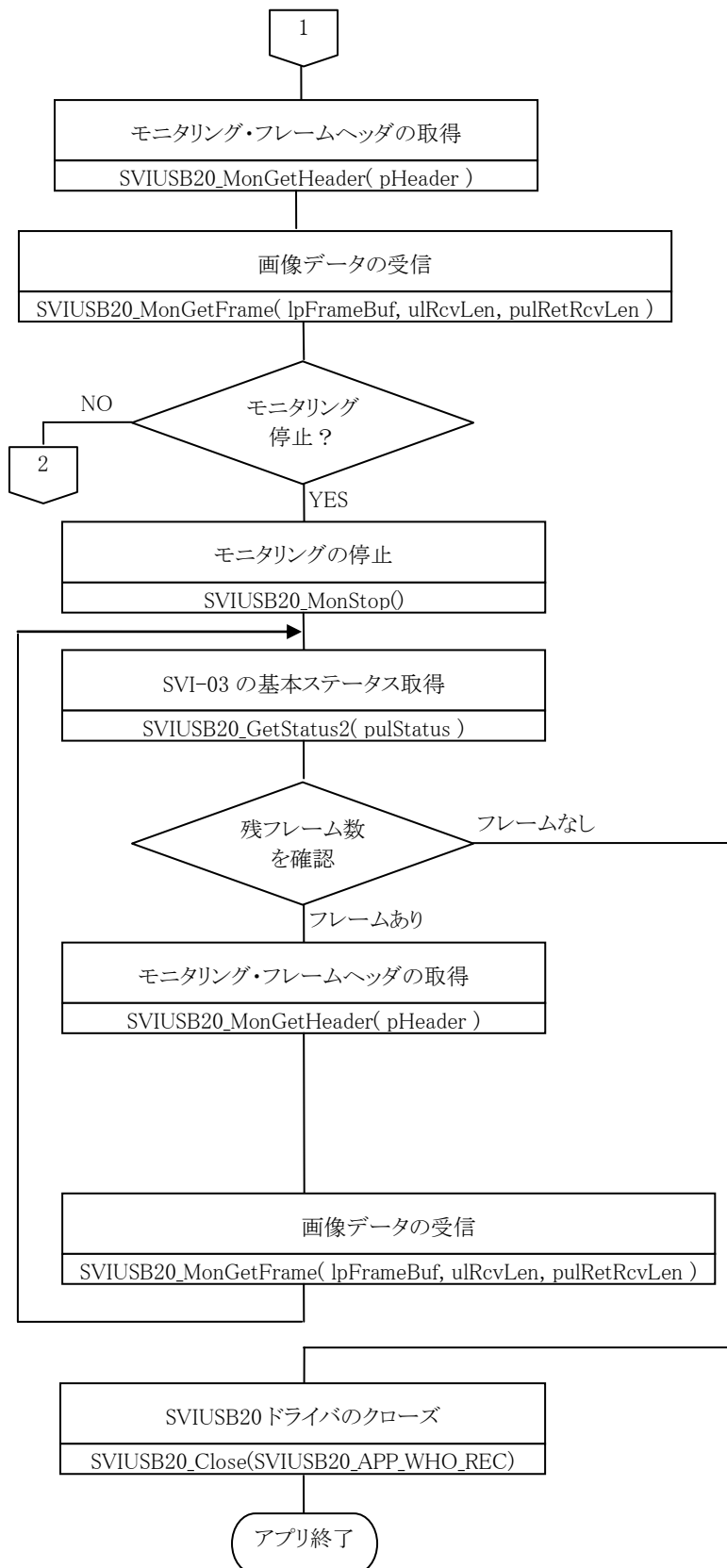
※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

詳細は「3.3.29.SVI-03 設定」をご覧ください

※ SVIUSB20_MonSstart コール後、SVIUSB20_GetStatus2 で、SVI-03 にキャプチャ画像フレームの有無を確認します。

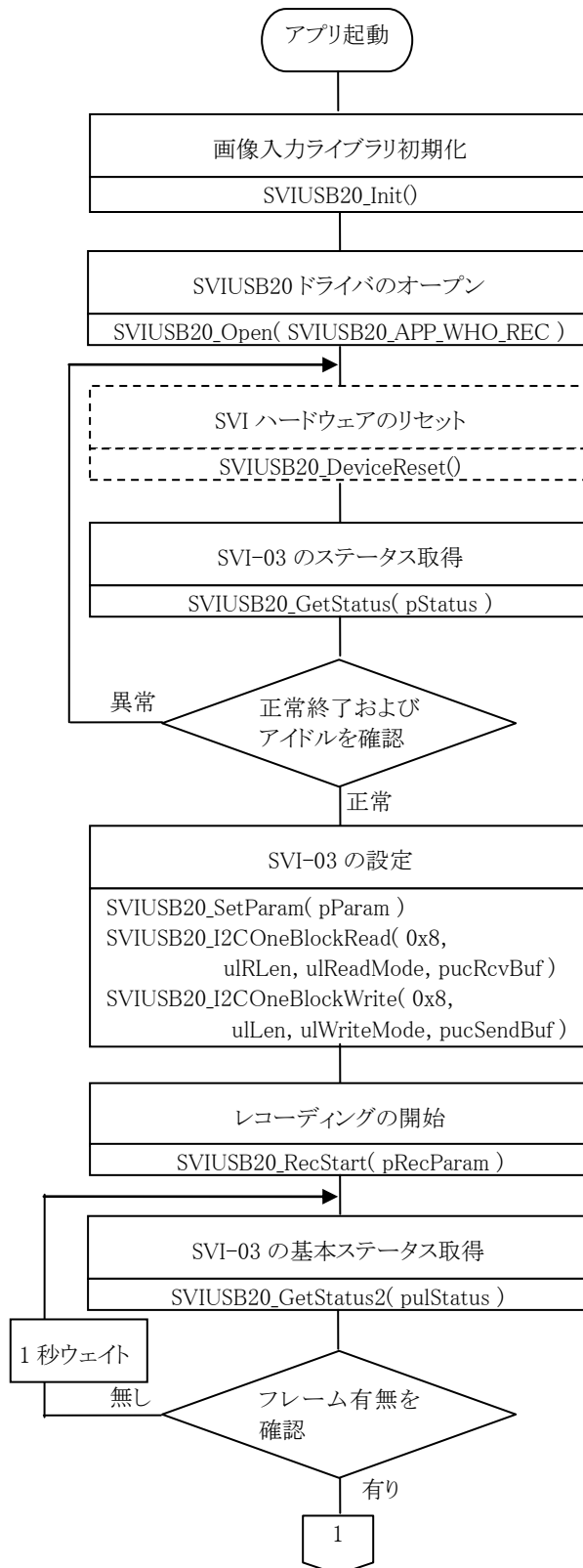


※取得したフレームヘッダを参照して
ulRcvDataSize を求めて下さい。

※モニタリング停止でなければ連続して受信
して下さい。

※停止後、基本ステータスを取得し、残フレ
ームの有無を確認します。残フレームがなく
なるまで受信動作を繰り返します。

3.3.32. レコーディングモード時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

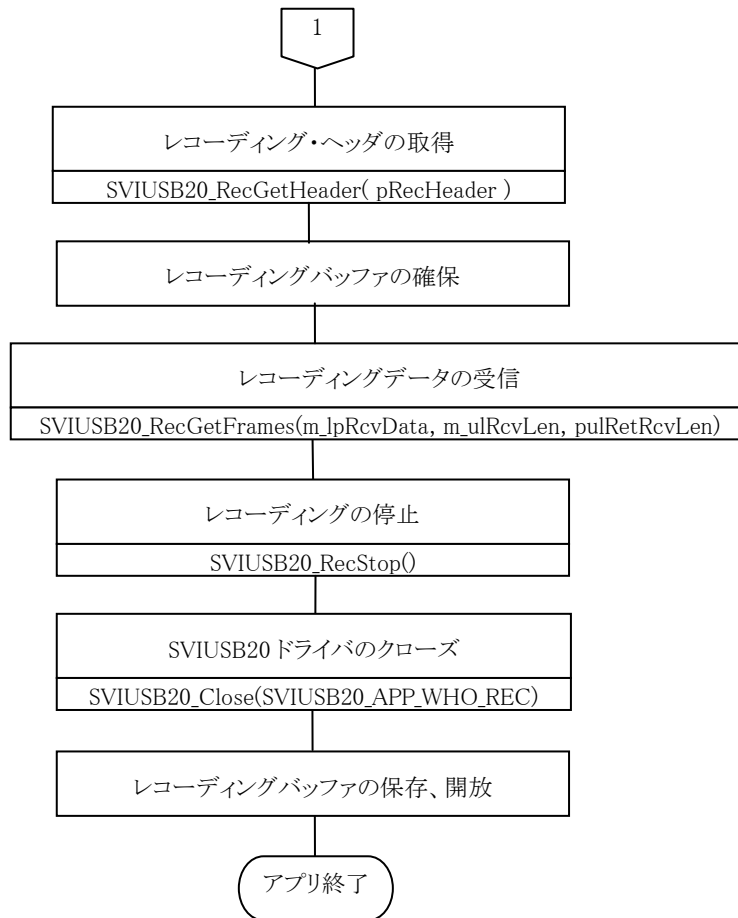
※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

詳細は「3.3.29.SVI-03 設定」をご覧ください

※ SVIUSB20_RecStart コール後、SVIUSB20_GetStatus2 で、SVI-03 にレコーディングデータの有無を確認します。

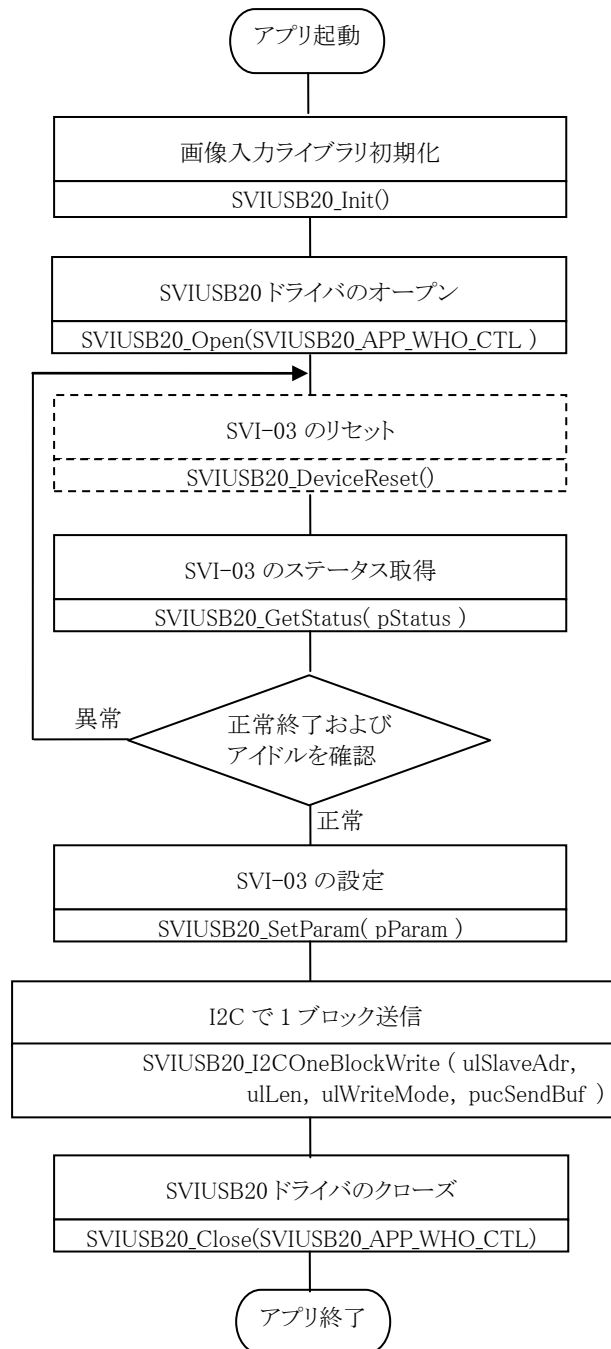


※ 取得したヘッダから RecHeader.ulNumScan がレコーディングした実バイト数です。この値を 64 で割り切れるよう桁上げた値が転送バイト数になります。

64の倍数に切り上げた値でメモリを確保してください。

m_lpRcvData は予め m_ulRcvLen 分確保しておいて下さい。

3.3.33. I2C によるコマンド送信時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

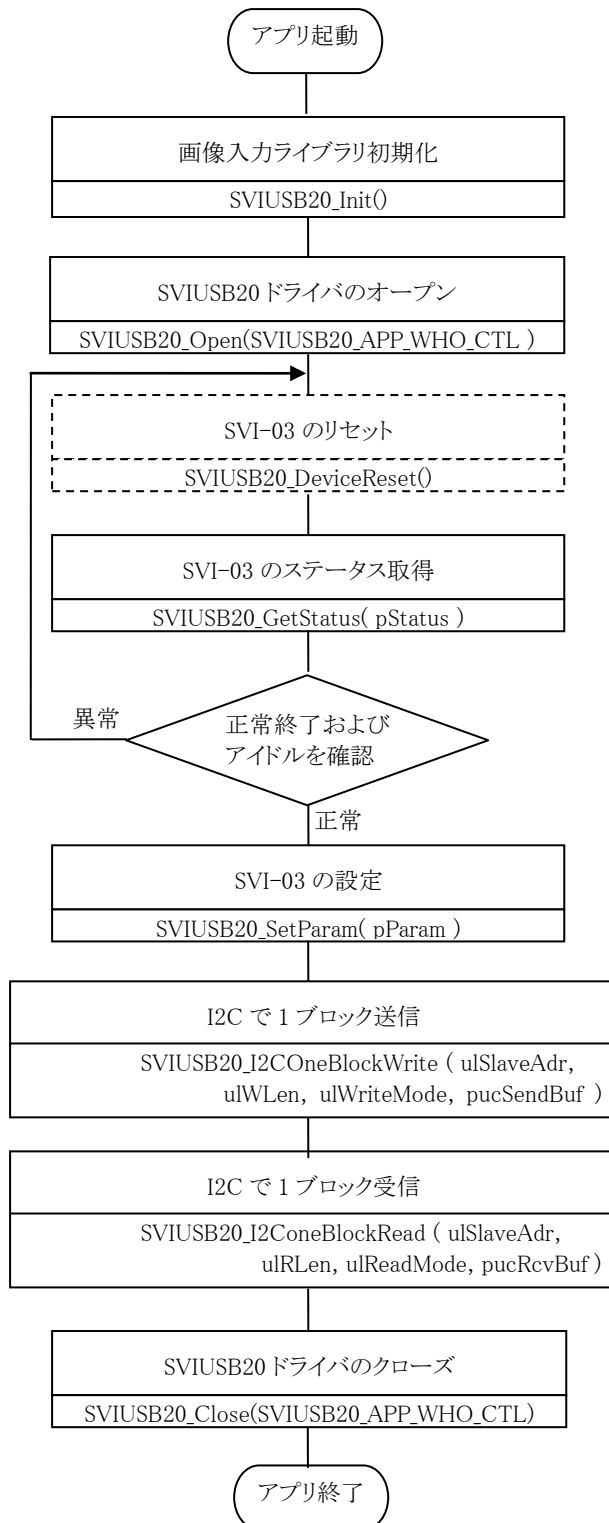
※必ず行って下さい。

※ulSlaveAdr にはスレーブ ID を代入する。

(API の中で左に 1 ビットシフトしている)

※ulLen にはスレーブ ID を含まないサブアドレスからのバイト数を指定する。

3.3.34. I2C によるコマンド受信時の画像入力ライブラリ使用例



※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※必要あれば行って下さい。また調子が悪い時にも有効です。(3 秒かかります)

※基本ステータスだけであれば SVIUSB20_GetStatus2(pulStatus)でも可能です。

※3 回リトライしても異常であればエラー終了の手続きをして下さい。

※必ず行って下さい。

※ulSlaveAdr にはスレーブ ID を代入する。

(API の中で左に 1 ビットシフトしている)

※ulWLen にはサブアドレスのみなので1を代入する。

※ulRLen には受信するバイト数を指定する。

3.3.35. 複数台の SVI-03 を使用する場合のオープン/クローズ方法

【オープン時】

①SVIUSB20_EnumDevice API を呼び出し、ライブラリーを初期化します。

②SVIUSB20_EnumDevice API を呼び出し、SVI-03ボードの接続台数を取得します。

```
DWORD dwRet;
ULONG ulSVI_Num;
dwRet = SVIUSB20_EnumDevice (
    SVIUSB20_APP_WHO_REC, // オープン元のアプリケーションを指定
                          // SVIUSB20_APP_WHO_REC (表示アプリ用)
                          // SVIUSB20_APP_WHO_CTL (制御アプリ用)
    &ulSVI_Num,           // 接続台数を格納するポインタ
    NULL                  // 接続台数分のボード番号を格納する配列のポインタ
                          // このポインタがNULL の場合、接続台数のみ格納します
);
```

③SVIUSB20_EnumDevice API を呼び出し、SVI-03ボードの接続台数分のボード番号を取得します。

```
PULONG pulSVI_NumTable;
pulSVI_NumTable = (PULONG) new ULONG[ulSVI_Num];
dwRet = SVIUSB20_EnumDevice (
    SVIUSB20_APP_WHO_REC, // オープン元のアプリケーションを指定
                          // SVIUSB20_APP_WHO_REC (表示アプリ用)
                          // SVIUSB20_APP_WHO_CTL (制御アプリ用)
    &ulSVI_Num,           // 接続台数を格納するポインタ
    pulSVI_NumTable       // 接続台数分のボード番号を格納する配列のポインタ
                          // このポインタがNULL の場合、接続台数のみ格納します
);
```

④ボード番号配列 (pulSVI_NumTable) から、SVI-03ボードのボード番号を抽出し、使用するSVI-03ボードを選択します。選択はSVIUSB20_DeviceSelect API を呼び出します。

ボード番号配列pulSVI_NumTableのひとつの要素は上位16ビットにボード番号、会16ビットにWindows管理番号を格納していますので、使用するボード番号が格納されている要素をSVIUSB20_DeviceSelect APIにて指定します。

```
SVIUSB20_DeviceSelect (
    pulSVI_NumTable[0]; // SVI-03ボード番号を格納した (0~3)
);
```

⑤以降はSVIUSB20_Open APIにて使用できます。

【クローズ時】

①SVOUSB20_Close APIにてデバイスドライバをクローズした後、SVIUSB20_DeviceRelease API を呼び出し、選択したSVI-03ボードを開放します。

```
SVIUSB20_DeviceRelease ( );
```

②SVIUSB20_End API を呼び出し、ライブラリを終了します。

```
SVIUSB20_End ( );
```

3.3.36. SVI-03 の設定

モニタリング、レコーディングする際に、SVI-03 ボードの設定をしなければなりません。

設定は SVIUSB20_SetParam API、SVIUSB20_I2COneBlockWrite API、SVIUSB20_I2COneBlockRead API にて行います。

設定できる内容は、SVIUSB20_SetParam API の場合は、

- ・画像情報通知 ～ 取り込みフレームの情報を通知します
- ・カメラ電源スイッチ ～ モニタリング、レコーディングには関係ありません
- ・I2C スピード ～ モニタリング、レコーディングには関係ありません
- ・VSYNC デアサートフラグ ～ VSYNC アサートを待たずにフレームを取り込むか通知します
- ・モニタリングモード ～ SVI-03 ボード上の SDRAM の使用方法を通知します

になります。

SVIUSB20_SetParam API については、「3.3.8. SVIUSB20_SetParam」をご覧ください。

SVIUSB20_I2COneBlockWrite API、SVIUSB20_I2COneBlockRead API にて設定できる内容は、別紙「SVI-03 I2C レジスタ表」に記載されているレジスタで

- ・取り込みデータビット幅の選択(アドレス 0h) ～ 8 ビットが初期値
- ・同期信号極性(アドレス 20h) ～ Low Enable が初期値
- ・データ取り込みタイミング設定(アドレス 2Eh) ～ DCK の立ち上がりでデータを取り込むが初期値

となります。

例1) 取り込みデータビット幅を 16 ビットに変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x0; // サブアドレス 0h を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[0] );
ucBuf[1] |= 0x40;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
```

例 2) 同期信号極性を High Enable に変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x20; // サブアドレス 20h を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[0] );
ucBuf[1] |= 0x2;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
```

例 3) データ取り込みタイミングを立ち下がりに変更する場合

```
DWORD dwRet;
UCHAR ucBuf[8];
ucBuf[0] = 0x2E; // サブアドレス 2Eh を指定
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
dwRet = SVIUSB20_I2COneBlockRead ( 0x8, 1, 0, &ucBuf[0] );
ucBuf[1] |= 0x1;
dwRet = SVIUSB20_I2COneBlockWrite ( 0x8, 1, 0, &ucBuf[0] );
```