SVM-03
制御ライブラリ説明書

V1.10

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	2017/02/20	・新規作成	
1.10	2018/09/07	・3.4.3 SVI05API_Open 関数の引数を追加しました ・3.4.12 SVI05API_Open 関数の引数を追加しました ・3.4.13 SVI05API_Open 関数の引数を追加しました ・3.4.14 SVI05API_Open 関数の引数を追加しました ・3.4.15 SVI05API_Open 関数の引数を追加しました	赤字参照

目次

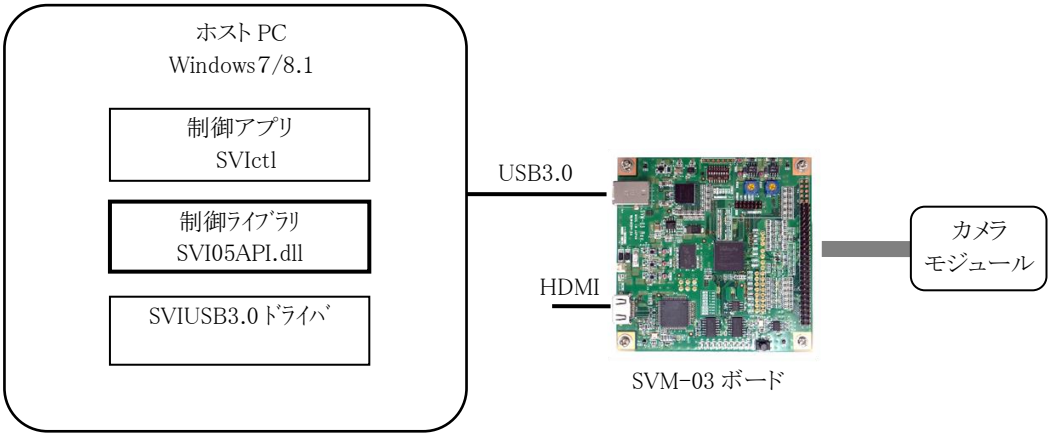
1. 適用

本説明書は SVM-03/SVM-03MIPI に適用します。

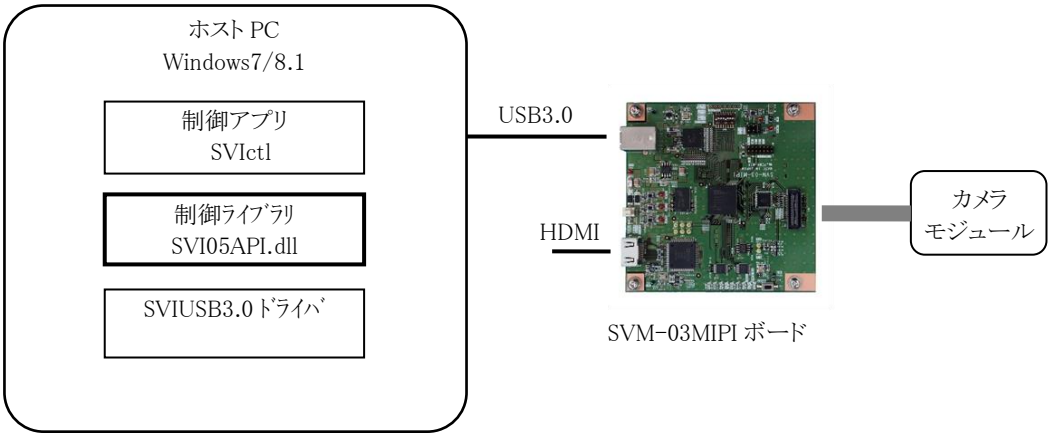
2. 概要

SVM-03/SVM-03MIPI 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 SVM-03/SVM-03MIPI を USB3.0 を介して接続、制御するための制御ライブラリについて記述します。
制御ライブラリを使用することで、SVM-03/SVM-03MIPI に接続されたセンサー、カメラモジュールの制御、SVM-03/SVM-03MIPI ボードの制御を行うことができます。

【図1】 SVM-03 システム構成図



【図2】 SVM-03MIPI システム構成図



3. 仕様

本ライブラリは SVM シリーズ 専用デバイスドライバを呼び出し SVM-03/SVM-03MIPI よりカメラモジュールまたはイメージセンサー、SVM-03/SVM-03MIPI ボードを制御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 SVM-03/SVM-03MIPI のパラメータの設定、ステータスの取得及び I^2C 通信によるカメラモジュールまたはイメージセンサーの制御を実現します。
OS のビット数によって、ライブラリ自体を使い分ける必要あります。
以降の説明では、SVM-03/SVM-03MIPI を SVM-03 のみとして表記します。

3.1. ファイル構成（32bitOS）

本ライブラリは以下のファイルを提供します。

・SVIUSB30.inf （Software-CD の Driver_x86 フォルダに格納）
　SVI USB3.0 ドライバインストール情報ファイル。
・sviusb30.cat （Software-CD の Driver_x86 フォルダに格納）
　SVI USB3.0 ドライバカタログファイル。
・SVIUSB30.sys （Software-CD の Driver_x86 フォルダに格納）
　SVI USB3.0 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。
・SVI05API.h （Software-CD の“source¥制御ライブラリ¥x86”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。
・SVI05API.dll （Software-CD の“Appl_x86”フォルダに格納）
　本ライブラリです。
・SVI05API.lib （Software-CD の“source¥制御ライブラリ¥x86”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. ファイル構成（64bit OS）

本ライブラリは以下のファイルを提供します。

・SviU3drv.inf （Software-CD の Driver_x64 フォルダに格納）
　SVI USB3.0 ドライバインストール情報ファイル。
・sviusb30.cat （Software-CD の Driver_x64 フォルダに格納）
　SVI USB3.0 ドライバカタログファイル。
・SviU3drv.dll （Software-CD の Driver_x64 フォルダに格納）
　SVI USB3.0 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。
・WdfCoInstaller01009.dll （Software-CD の Driver_x64 フォルダに格納）
　ドライバ関連ファイル　Windows ディレクトリ下”System32¥drivers にコピーされ使用されます。
・winusbcoinstaller2.dll （Software-CD の Driver_x64 フォルダに格納）
　ドライバ関連ファイル　Windows ディレクトリ下”System32¥drivers にコピーされ使用されます。
・WUDFUpdate_01009.dll （Software-CD の Driver_x64 フォルダに格納）
　ドライバ関連ファイル　Windows ディレクトリ下”System32¥drivers にコピーされ使用されます。
・SVI05API.h （Software-CD の“source¥制御ライブラリ¥x64”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。
・SVI05API.dll （Software-CD の“Appl_x64”フォルダに格納）
　本ライブラリです。
・SVI05API.lib （Software-CD の“source¥制御ライブラリ¥x64”フォルダに格納）
　本ライブラリリンクモジュールです。

3.3. SVM-03 API 一覧

API名	機能
SVI05API_Init	SVM-03 制御ライブラリを初期化します
SVI05API_End	SVM-03 制御ライブラリの終了処理を行います
SVI05API_Open	SVM シリーズ専用デバイスドライバをオープンします
SVI05API_Close	SVM シリーズ専用デバイスドライバをクローズします
SVI05API_GetVersion	SVM-03 制御ライブラリのバージョン情報を取得します
SVI05API_I2COneBlockWrite	I2C で 1 ブロックを送信します （SVI-03/SVI-06 互換）
SVI05API_I2COneBlockRead	I2C で 1 ブロックを受信します （SVI-03/SVI-06 互換）
SVI05API_I2CBlockWrite	I2C で 1 ブロックを送信します
SVI05API_I2CBlockRead	I2C で 1 ブロックを受信します
SVI05API_SPIFpgaRead	SVM-03 の FPGA レジスタを読み込みます
SVI05API_SPIFpgaWrite	SVM-03 の FPGA レジスタを書き込みます
SVI05API_SVMVersionInfo	SVM ボードの FX3 と FPGA のバージョン情報を取得します
SVI05API_SVMSettingRead	SVMSetting 情報格納先 SPIROM のアドレスへ SPI 受信
SVI05API_SVMSettingWrite	SVMSetting 情報格納先 SPIROM のアドレスへ SPI 送信
SVI05API_SVMSPIBootMemUpdate	SVM の SPIROM に格納されているブートデータを更新します
SVI05API_SVMFX3Update	SVM の SPIROM に格納されている Fx3 のブート情報をアップデートします
SVI05API_SVMFPGAUpdate	SVM の SPIROM に格納されている FPGA のブート情報をアップデートします
SVI05API_SVMSPIBootMemRead	SVM の SPIROM に格納されているブートメモリデータを取得します
SVI05API_SVMFX3BootMemRead	SVM の SPIROM に格納されている FX3 のブートメモリデータを取得します
SVI05API_SVMFPGABootMemRead	SVM の SPIROM に格納されている FPGA のブートメモリデータを取得します
SVI05API_SPIRead	SPI 受信
SVI05API_SPIWrite	SPI 送信
SVI05API_SVM03USettingRead	SVMSetting 情報格納先 SPIROM のアドレスへ SPI 受信
SVI05API_SVM03USettingWrite	SVMSetting 情報格納先 SPIROM のアドレスへ SPI 送信
SVI05API_SVM03UFX3Update	SVM の SPIROM に格納されている Fx3 のブート情報をアップデートします
SVI05API_SVM03UFPGAUpdate	SVM の SPIROM に格納されている FPGA のブート情報をアップデートします
SVI05API_SVM03UFX3BootMemRead	SVM の SPIROM に格納されている FX3 のブートメモリデータを取得します
SVI05API_SVM03UFPGABootMemRead	SVM の SPIROM に格納されている FPGA のブートメモリデータを取得します
SVI05API_Update	SVI-09 のファームウェア、FPGA をアップデートします

※グレーで網かかっている API はお客様ではご使用になれませんので、以降の API 説明を省略させていただきます。

3.4. SVM-03 制御ライブラリーAPI リファレンス

3.4.1. SVI05API_Init

API　　SVI05API_Init

機能　　SVM-03 制御ライブラリの内部変数を初期化します

プロトタイプ
void SVI05API_Init(void);

戻り値
なし

備考
・必ず最初に呼び出して下さい。（SVI05API_Open API よりも！）

3.4.2. SVI05API_End

API　　SVI05API_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 SVI05API_End(void);

戻り値
なし

備考
・必ず最後に呼び出して下さい。

3.4.3. SVI05API_Open

API　　SVI05API_Open

機能　　SVM シリーズ専用デバイスドライバをオープンします

プロトタイプ

```
DWORD SVI05API_Open (
        ULONG       ulAppWho,       // オープン元のアプリケーションを指定
                                    //   SVI05API_APP_WHO_REC（表示アプリ用）※使用不可
                                    //   SVI05API_APP_WHO_CTL（制御アプリ用）
        int         deviceIndex     // 予約 0 を代入してください
);
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_DEVOPEN	デバイスドライバをオープンできません
SVI05API_RET_ERROR_MULTIOPEN	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
SVI05API_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

引数 ulAppWho には必ず SVI05API_APP_WHO_CTL を指定してください。SVI05API_APP_WHO_REC を指定した場合は SVI05API_RET_ERROR_PARAMETERが返りエラーとなります。

3.4.4. SVI05API_Close

API　　SVI05API_Close

機能　　SVM 専用デバイスドライバをクローズします

プロトタイプ

```
DWORD SVI05API_Close (
        ULONG ulAppWho              // オープン元のアプリケーションを指定
                                    //   SVI05API_APP_WHO_REC（表示アプリ用）※使用不可
                                    //   SVI05API_APP_WHO_CTL（制御アプリ用）
);
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_NOOPEN	オープンされていません
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります

備考

引数 ulAppWho には必ず SVI05API_APP_WHO_CTL を指定してください。SVI05API_APP_WHO_REC を指定した場合は SVI05API_RET_ERROR_PARAMETERが返りエラーとなります。

3.4.5. SVI05API_GetVersion

API　　SVI05API_GetVersion
機能　　SVI05API.DLL のバージョン情報を取得します
プロトタイプ

```
DWORD SVI05API_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

SVI05API_RET_NORMAL	正常終了
SVI05API_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
```

3.4.6. SVI05API_I2COneBlockWrite

API　　SVI05API_I2COneBlockWrite

機能　　SVM-03 に I²C で 1 ブロック送信します

プロトタイプ

```
DWORD   SVI05API_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,         // bit0-30   :  予約(常に 0)
                                       // bit31     :  再送オン(StopCondition 発行せず)
        PUCHAR    pucSendBuf           //  送信コマンドデータバッファのポインタ
) ;
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_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:以下参照)
SVI_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
SVI_STS_I2C_PRETIMEOUT	I2C でプリタイムアウトが発生した
SVI_STS_I2C_POSTTIMEOUT	I2C でポストタイムアウトが発生した

備考

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。

3.4.7. SVI05API_I2COneBlockRead

API　　SVI05API_I2COneBlockRead

機能　　SVM-03 より I²C で 1 ブロック受信します

プロトタイプ

```
DWORD   SVI05API_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,          // bit0-30   :  予約(常に 0)
                                       // bit31     :  再送オン(Re-StartCondition 発行)
        PUCHAR    pucRcvBuf            //  読み出しデータバッファのポインタ
) ;
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_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:以下参照)
SVI_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
SVI_STS_I2C_PRETIMEOUT	I2C でプリタイムアウトが発生した
SVI_STS_I2C_POSTTIMEOUT	I2C でポストタイムアウトが発生した

備考

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 の再送オンは ReStartCondtion を指定します。

3.4.8. SVI05API_I2CBlockWrite

API　　SVI05API_I2CBlockWrite

機能　　SVM-03 に I²C で 1 ブロック送信します

プロトタイプ

```
DWORD   SVI05API_I2CBlockWrite (
        ULONG     ulSlaveAdr,       // I²C スレーブアドレス(7bit)
        ULONG     ulSubdr,          // サブアドレス
        ULONG     ulLen,            // 送信するコマンドデータ数
        PUCHAR    pucSendBuf,       // 送信コマンドデータバッファのポインタ
        BOOL      bWord             // サブアドレスビット幅フラグ(false:8bit, true:16bit)
) ;
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_ PARAMETER	引数に間違いがあります
SVI05API_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:以下参照)
SVI_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
SVI_STS_I2C_PRETIMEOUT	I2C でプリタイムアウトが発生した
SVI_STS_I2C_POSTTIMEOUT	I2C でポストタイムアウトが発生した

備考

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。

本 API では必ずサブアドレス指定が必要になります。

3.4.9. SVI05API_I2CBlockRead

API　　SVI05API_I2CBlockRead

機能　　SVM-03 より I²C で 1 ブロック受信します

プロトタイプ

```
DWORD   SVI05API_I2CBlockRead (
        ULONG    ulSlaveAdr,        // I²C スレーブアドレス(7bit)
        ULONG    ulSubdr,           // サブアドレス
        ULONG    ulLen,             // 読み出し要求バイト数
        PUCHAR   pucRcvBuf,         // 読み出しデータバッファのポインタ
        BOOL     bWord              // サブアドレスビット幅フラグ（false:8bit, true:16bit）
) ;
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_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:以下参照）
SVI_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
SVI_STS_I2C_PRETIMEOUT	I2C でプリタイムアウトが発生した
SVI_STS_I2C_POSTTIMEOUT	I2C でポストタイムアウトが発生した

備考

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。
本 API では必ずサブアドレス指定が必要になります。

3.4.10. SVI05API_SPIFpgaRead

API　　SVI05API_SPIFpgaRead

機能　　SVM-03 の FPGA レジスタを読み込みます

プロトタイプ

```
DWORD   SVI05API_SPIFpgaRead (
        UCHAR     ucCommdId,        // コマンド番号(7 固定)
        UCHAR     ucSSId,           // ID(0x99 固定)
        ULONG     ulAddress,        // FPGA 空間レジスタアドレス
        ULONG     ulLen,            // 読み出しバイト数
        PUCHAR    pucRcvBuf         // 読み出しデータの格納バッファのポインタ
) ;
```

戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_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

FPGA空間レジスタアドレスは別紙SVM-03レジスタ表をご覧ください。

3.4.11. SVI05API_SPIFpgaWrite

API　　SVI05API_SPIFpgaWrite

機能　　SVM-03 の FPGA レジスタを書き込みます

プロトタイプ

```
DWORD   SVI05API_SPIFpgaWrite (
        UCHAR     ucCommdId,        // コマンド番号(8 固定)
        UCHAR     ucSSId,           // ID(0x99 固定)
        ULONG     ulAddress,        // FPGA 空間レジスタアドレス
        ULONG     ulLen,            // 書き出しバイト数
        PUCHAR    pucSendBuf        // 書き出しデータの格納バッファのポインタ
) ;
```

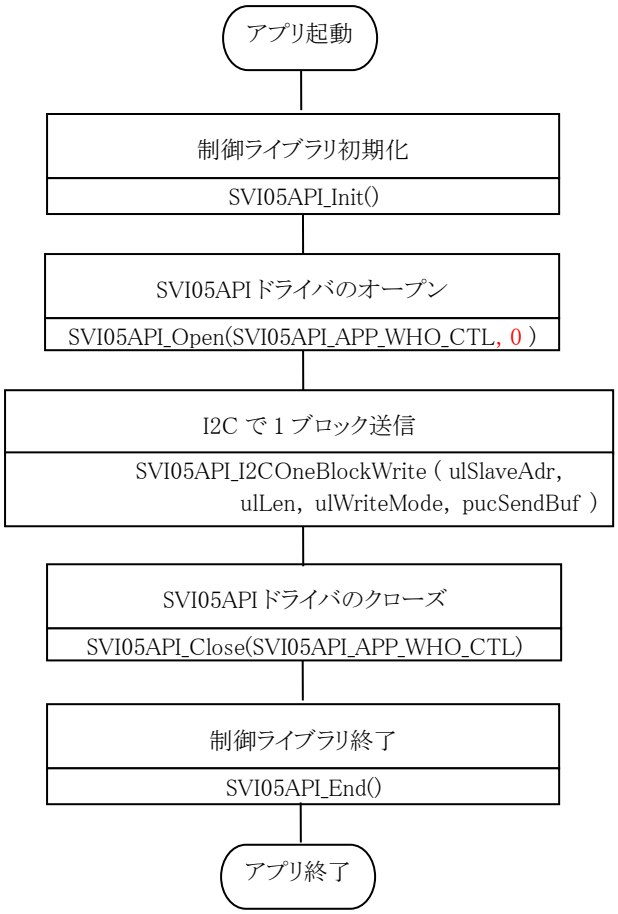
戻り値

SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_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

FPGA 空間レジスタアドレスは別紙 SVM-03 レジスタ表をご覧ください。

3.4.12. I2C によるコマンド送信時の制御ライブラリ使用例



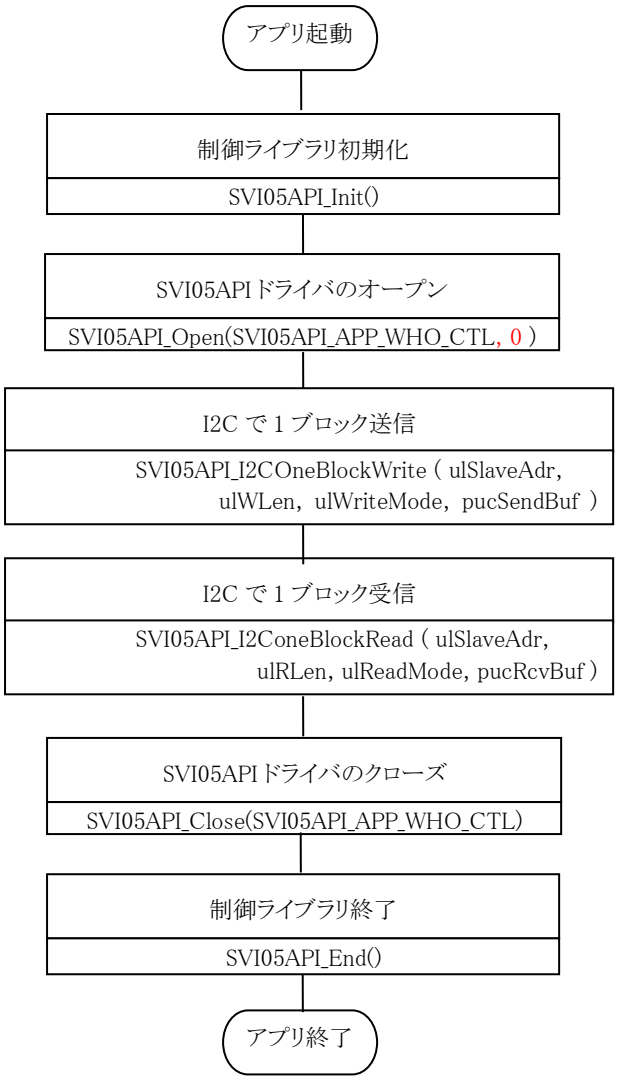
※各 API コール後エラー処理を行って下さい。

※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。

※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。

3.4.13. I2C によるコマンド受信時の制御ライブラリ使用例



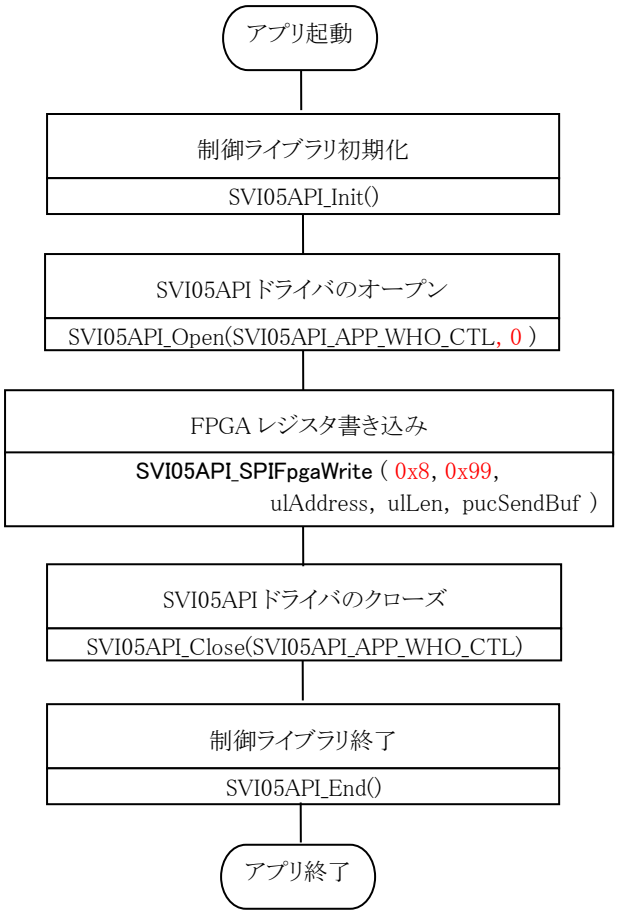
※各 API コール後エラー処理を行って下さい。

※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。

※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。

3.4.14. FPGA レジスタの書き込み時の制御ライブラリ使用例



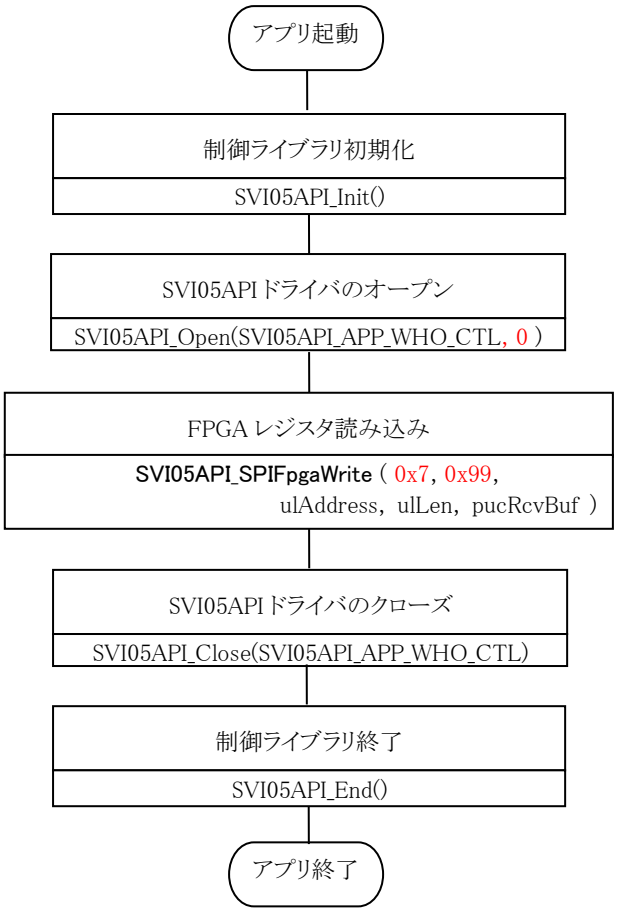
※各 API コール後エラー処理を行って下さい。

※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。

※ulAddress は SVM-03 レジスタ表を参照してください。
※FPGA のレジスタは 32bit なので ulLen はレジスタ数 x 4 (バイト数)を指定してください。

3.4.15. FPGA レジスタの読み込み時の制御ライブラリ使用例



※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。

※ulAddress は SVM-03 レジスタ表を参照してください。
※FPGA のレジスタは 32bit なので ulLen はレジスタ数 x 4 (バイト数)を指定してください。