NDC00055-V2.10

SVM-03/06/SVP-01-U
制御ライブラリ説明書

V2.10

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
2.00	2023/05/16	・SVM-03, SVM-06, SVP-01-U の共通ライブラリの説明書に変更 ・3.4.4 SVI05API_OpenEx 関数を追加しました ・3.4.8 SVI05API_ GetBoardInfo 関数を追加しました ・3.5.3. ボード名取得時の制御ライブラリ使用例 を追加しました ・3.5.4. ボード ID を指定してボードを開く時の制御ライブラリ使用例を追加しました	
2.01	2023/08/31	・3.4.6. SVI05API_RestartSVM 関数を追加しました	
2.10	2023/11/28	・3.3 SVI05API_GetRevision 関数を追加しました（内部向け） ・3.4.11 SVI05API_I2CBlockWrite 関数の引数を変更しています。 bWord (bool) -> subAddrType (int) (0-2 まで) ・3.4.10 SVI05API_I2CblockRead 関数の引数を変更しています。 bWord (bool) -> subAddrType (int) (0-2 まで)	

目次

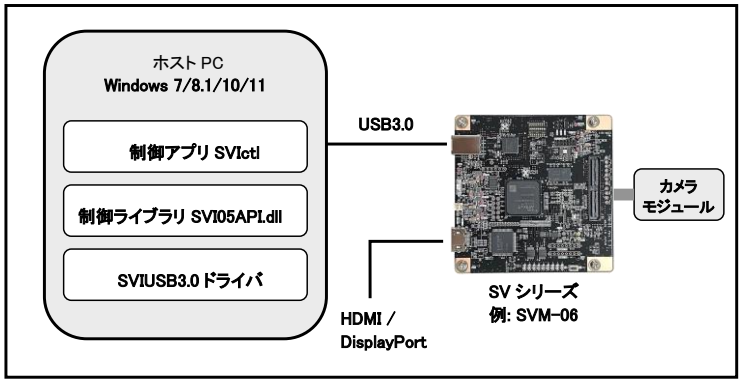
1. 適用

本説明書は SVM-03/SVM-03MIPI, SVM-06, SVP-01-U に適用します。
以降の説明では、SVM-03/SVM-03MIPI, SVM-06, SVP-01-U を SV シリーズ（ベンダー版を除く）として表記します。

2. 概要

SV シリーズはカメラ･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 SV シリーズを 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、SV シリーズに接続されたセンサー、カメラモジュールの制御、SV シリーズの制御を行うことができます。

【図1】 SV シリーズシステム構成図



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 SV シリーズよりカメラモジュールまたはイメージセンサー、SV シリーズ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 SV シリーズ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。

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。

・SVI05API.h （Software-CD の“Library¥Library_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“Library¥Library_x86”フォルダに格納）
　本ライブラリです。
・SVI05API.lib （Software-CD の“Library¥Library_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。

・SVI05API.h （Software-CD の“Library¥Library_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“Library¥Library_x64”フォルダに格納）
　本ライブラリです。
・SVI05API.lib （Software-CD の“Library¥Library_x64”フォルダに格納）
　本ライブラリリンクモジュールです。

- ライブラリは弊社 Web ページからダウンロードできる「SVMCtl_I2C」プロジェクトにも格納されています。

3.3. SV シリーズ API 一覧

API名	機能
SVI05API_Init	SV シリーズ制御ライブラリを初期化します
SVI05API_End	SV シリーズ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_OpenEx	SV シリーズ(ベンダー版を除く)デバイスドライバをオープンします。 OpenEx()では SVM シリーズに加えて、SVO シリーズも開くことができます。
SVI05API_Close	Open()または OpenEx()で開いたボードをクローズします。
SVI05API_RestartSVM	SV シリーズボードの再起動を行います。
SVI05API_GetVersion	SV シリーズ制御ライブラリのバージョン情報を取得します
SVI05API_GetBoardInfo	Board 名, Board ID 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	SV シリーズの FPGA レジスタを読み込みます
SVI05API_SPIFpgaWrite	SV シリーズの 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_GetRevision	NV ソフトウェアで使用する SV ボードの分類番号を返却します
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

コメントの追加 [YK1]: GetRevision()の追加

※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. SV シリーズ制御ライブラリ API リファレンス

3.4.1. SVI05API_Init

API　　SVI05API_Init

機能　　SV シリーズ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 SVI05API_Open ()よりも先に、最初に呼び出して下さい。

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   // 選択したいボードの認識された順番（0 から）
```

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_OpenEx

API　　SVI05API_OpenEx

機能　　SV シリーズ(ベンダー版を除く)デバイスドライバをオープンします

プロトタイプ

```
DWORD SVI05API_OpenEx (
    ULONG    ulAppWho,     //   オープン元のアプリケーションを指定
                           //   SVI05API_APP_WHO_REC
                           //   SVI05API_APP_WHO_CTL
    int      deviceIndex,  //   選択したいボードの認識された順番（0 から）
    int      boardIndex    //   SVM 系: 0, SVO 系: 1
)
```

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

boardType は 0 or 1 から選択します。

0: SVM 系(SVM シリーズ HDMI モード、UVC モード、SVO シリーズ HDMI モード、SVP-01U/SVS-01U DP モード、UVC モード、SVP-01G/SVS-01G DP モード、UVC モード

1: SVO 系(SVO シリーズ USB モード、SVP-01G/SVS-01G USB モード、USB モード)

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_Close

API　　SVI05API_Close

機能　　SVM シリーズ専用デバイスドライバ，または SV シリーズ(ベンダー版を除く)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。

3.4.6. SVI05API_RestartSVM

API　　SVI05API_RestartSVM
機能　　SVシリーズボードの再起動を行います。
プロトタイプ
```
void SVI05API_RestartSVM( void );
```

戻り値
なし

備考
・本APIを呼び出した後、SVI05API_Close()、SVI05API_End()を
呼び出して、本ライブラリの使用を終了してください。
本APIを呼び出した後、他にAPIは使用できません。

3.4.7. 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
) ;s
```

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
```

3.4.8. SVI05API_GetBoardInfo

API　　SVI05API_GetBoardInfo
機能　　Board名, Board IDを取得します
プロトタイプ

```
DWORD SVI05API_GetBoardInfo(
    int       boardType,           //  SVM系: 0, SVO系: 1
    int       choosedDeviceIndex,  //  選択したいボードの認識された順番, 0から
    char*     boardInfo,           //  ボード文字列を格納するためのポインタ
    size_t    boardInfoSize,       //  ボード文字列のサイズ
    int*      pBoardID             //  SVMCtl等で設定できるボード番号を格納するためのポインタ
);
```

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（ポインタがNULL）
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
0: SVM 系(SVM シリーズ HDMI モード、UVC モード、SVO シリーズ HDMI モード、SVP-01U/SVS-01U DP モード、UVC モード、SVP-01G/SVS-01G DP モード、UVC モード
1: SVO 系(SVO シリーズ USB モード、SVP-01G/SVS-01G USB モード、USB モード)

・デバイスエラーの基本ステータスは3.4.8.SVI05API_GetStatusの基本ステータスと同様です。
また、boardInfo, pBoardIDはNULLを許容し、NULLではない方の結果のみを返すこともできます。

3.4.9. SVI05API_I2COneBlockWrite

API　　SVI05API_I2COneBlockWrite

機能　　SV シリーズに I^2C 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。

3.4.10. SVI05API_I2COneBlockRead

API　　SVI05API_I2COneBlockRead

機能　　SV シリーズより 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。

3.4.11. SVI05API_I2CBlockWrite

API　　SVI05API_I2CBlockWrite

機能　　SV シリーズに 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
        int       sAddrType            // サブアドレスビット幅フラグ(0:8bit, 1:16bit, 2: 32bit)
```

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。

32bit 幅サブアドレスは一部のボード・FW のみ対応しています。詳しくはお問い合わせください。

3.4.12. SVI05API_I2CBlockRead

API　　SVI05API_I2CBlockRead

機能　　SV シリーズより 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
        int       sAddrType            // サブアドレスビット幅フラグ(0:8bit, 1:16bit, 2: 32bit)
);
```

コメントの追加 [YK2]: int 型，0-2 まで受けることを追記

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。
32bit 幅サブアドレスは一部のボード・FW のみ対応しています。詳しくはお問い合わせください。

3.4.13. SVI05API_SPIFpgaRead

API　　SVI05API_SPIFpgaRead

機能　　SV シリーズの 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。
SVM-03 以外のボードについては別途お問い合わせください。

3.4.14. SVI05API_SPIFpgaWrite

API　　SVI05API_SPIFpgaWrite

機能　　SV シリーズの 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

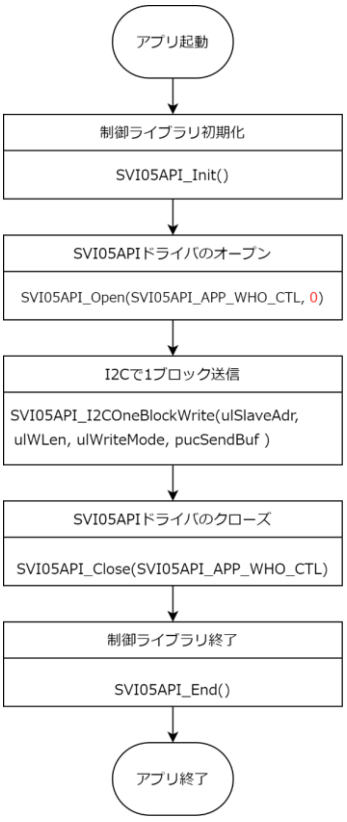
SVI05API_RET_NORMAL	正常終了
SVI05API_RET_ERROR_PARAMETER	引数に間違いがあります
SVI05API_RET_ERROR_NOOPEN	オープンされていません
その他	Win32API エラー(bit31-28:EH、bit27-0:GetLastError 戻り値)

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。
SVM-03 以外のボードについては別途お問い合わせください。

3.5. ライブラリの使用例

3.5.1. 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。

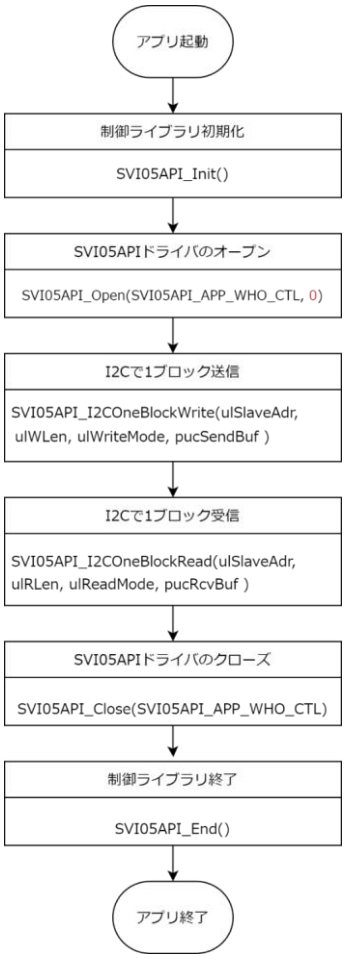
※必ず最初に行ってください。

※2 重オープンはできないので気を付けて下さい。

※ulSlaveAdr にはスレーブアドレスを代入する。
（API の中で左に 1 ビットシフトしている）
※ulLen にはスレーブアドレスを含まないサブアドレスからのバイト数を指定する。

s

3.5.2. I2C によるコマンド受信時の制御ライブラリ使用例



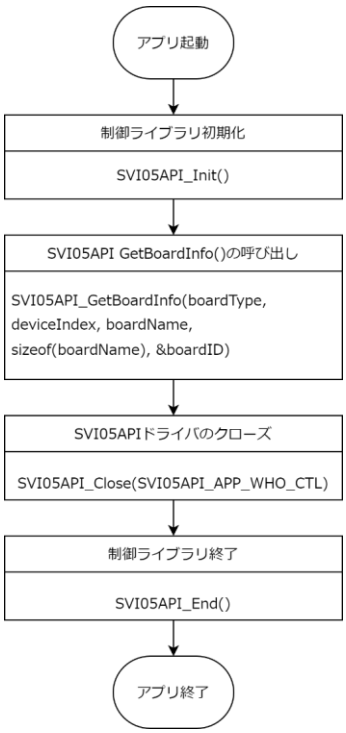
※各 API コール後エラー処理を行って下さい。

※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 にはスレーブアドレスを代入する。
（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。

3.5.3. ボード名取得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。

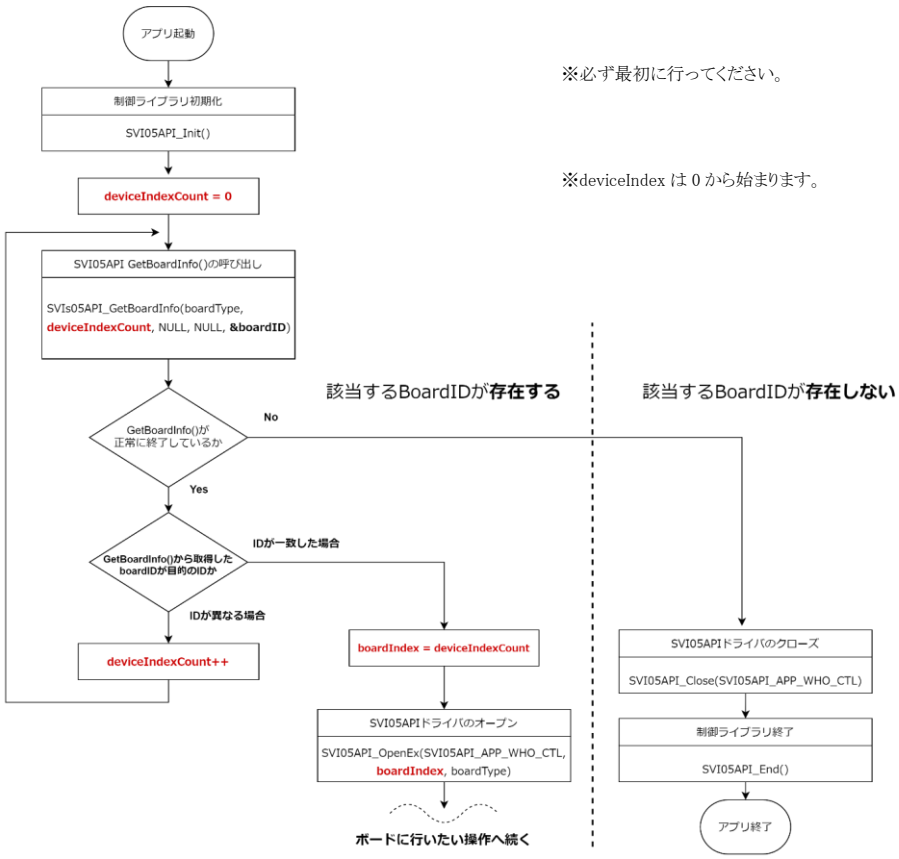
※1 deviceIndex は 0 から始まります。

3.5.4. ボード ID を用いてボードを開く時の制御ライブラリ使用例

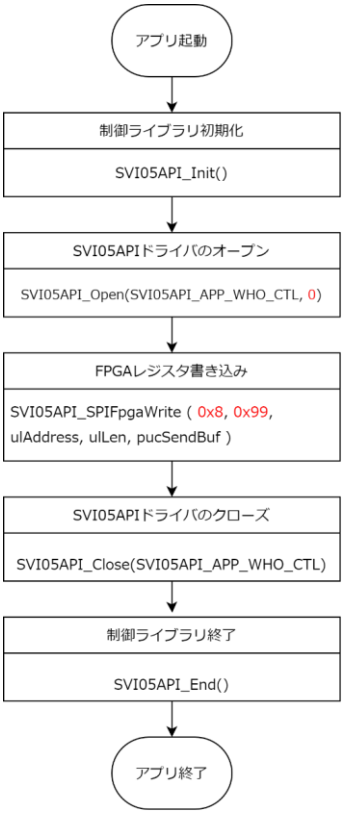
※各 API コール後エラー処理を行って下さい。

※必ず最初に行ってください。

※deviceIndex は 0 から始まります。



3.5.5. FPGA レジスタの書き込み時の制御ライブラリ使用例



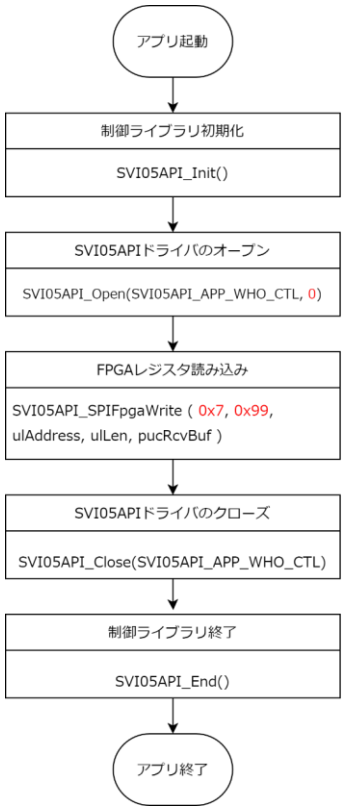
※各 API コール後エラー処理を行って下さい。

※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。

※1 ulAddress は SVM-03 レジスタ表を参照してください。SVM-03 以外のボードについては別途お問い合わせください。
※2 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。

3.5.6. 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。

※1 ulAddress は SVM-03 レジスタ表を参照してください。SVM-03 以外のボードについては別途お問い合わせください。
※2 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。

s