

Sky Vision Image (SVI) プラグイン仕様書

V1.03

承認	承認	照査	作成

株式会社ネットビジョン

※本ソフトウェアは開発中につき、内容に変更を生じることがあります。

改定履歴

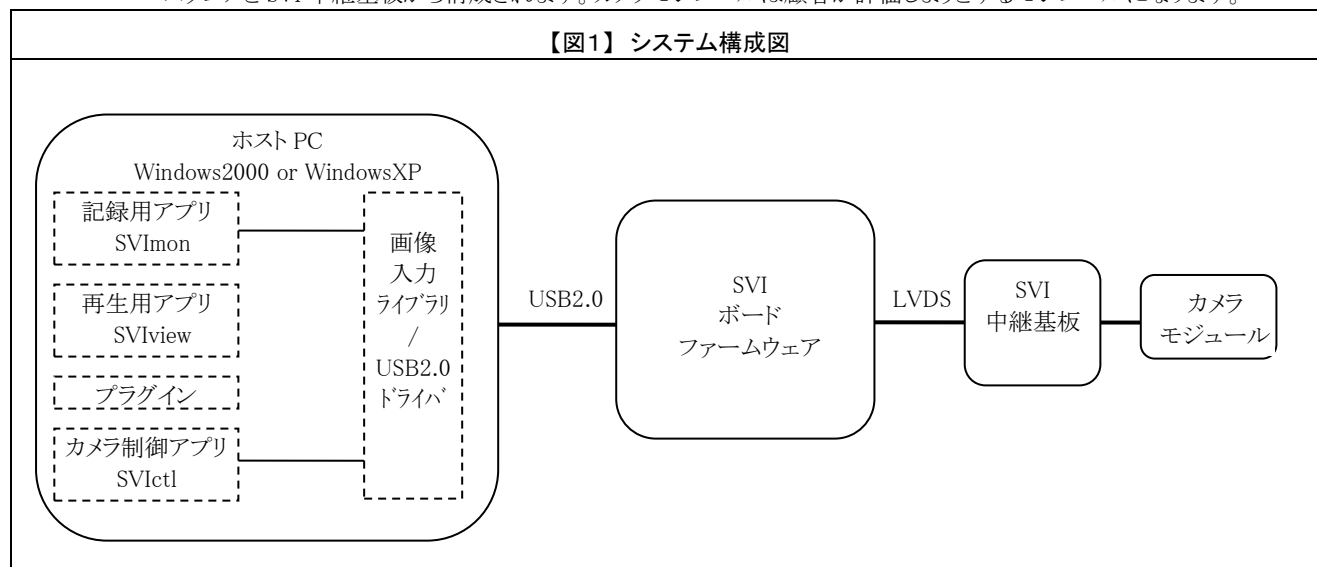
版数	日付	内容	担当	備考
1.00	2004/09/06	・新規作成	柏木	
1.01	2004/10/06	<ul style="list-style-type: none"> ・3.2 Plugin の起動の図 2 を修正 ・plg_sviGetPluginInfo について一部修正 ・plg_sviCreatePlugin について一部修正 ・plg_sviWaitSignal 関数についての説明を新規追加 ・新規にウィンドウメッセージを 4.5.1 通信に用いるウィンドウメッセージの種類に追加 ・付録 1 の参考ソースを修正 赤字の部分が修正箇所です。	木村	
1.02	2005/05/04	<ul style="list-style-type: none"> ・WM_SETPICMODE メッセージについて追加 ・WM_GETSTILLIMAGE メッセージについて追加 ・WM_SETSITLLSIZE メッセージについて追加 	木村	
1.03	2022/4/20	<ul style="list-style-type: none"> ・WM_EASYLOGIC_ANALYZER_POS について追加 ・16bit フォーマット以外に 32bit の説明追加 	赤嶺	
		・		

目次

1. 概要.....	3
2. プラグイン方法	3
3. プラグインモジュールインターフェース.....	4
3.1. 前提	4
3.2. PLUGINの起動.....	4
3.2.1. plg_sviGetPluginInfo	5
3.2.2. plg_sviCreatePlugin	5
3.2.3. plg_sviWaitSignal	6
3.2.4. plg_sviBootThread.....	6
3.3. 呼び出し元識別およびメッセージの送信方法	7
3.4. 共有メモリへのアクセス	7
3.5. POSTMESSAGE()による呼び出し元との通信内容	9
3.5.1. 通信に用いるウィンドウメッセージの種類.....	9
付録1 参考ソース	16

1. 概要

Sky Vision Image (以降:SVI)とはカメラ・モジュールの評価を目的とした Windows 上のソフトウェアと SVI ボード及びファームウェアと SVI 中継基板から構成されます。カメラモジュールは顧客が評価しようとするモジュールになります。



このシステムは、SVI ボードによって制御されたカメラ・モジュールの画像データを WindowsXP/Windows2000 搭載ホスト PC に USB2.0 インターフェースで取り込むことが可能なシステムです。本仕様書では SVImon、SVIview アプリケーションに機能を動的に追加するための方法を記述します。動的に追加することをプラグインと呼び、プラグインするモジュールをプラグインモジュールと呼びます。

2. プラグイン方法

提供されるプラグインモジュールを SVImon、SVIview のフォルダの plg フォルダにコピーしてください。SVImon、SVIview では、起動時 plg フォルダのプラグインモジュール (DLL:ダイナミックリンクライブラリ)を検索し、存在すればプラグイン名が PLUGIN メニューのプルダウンメニューに表示されます。

該当するプラグインかどうかは 3.2.1 項の `plg_sviGetPluginInfo()`により判断します。

3. プラグインモジュールインターフェース

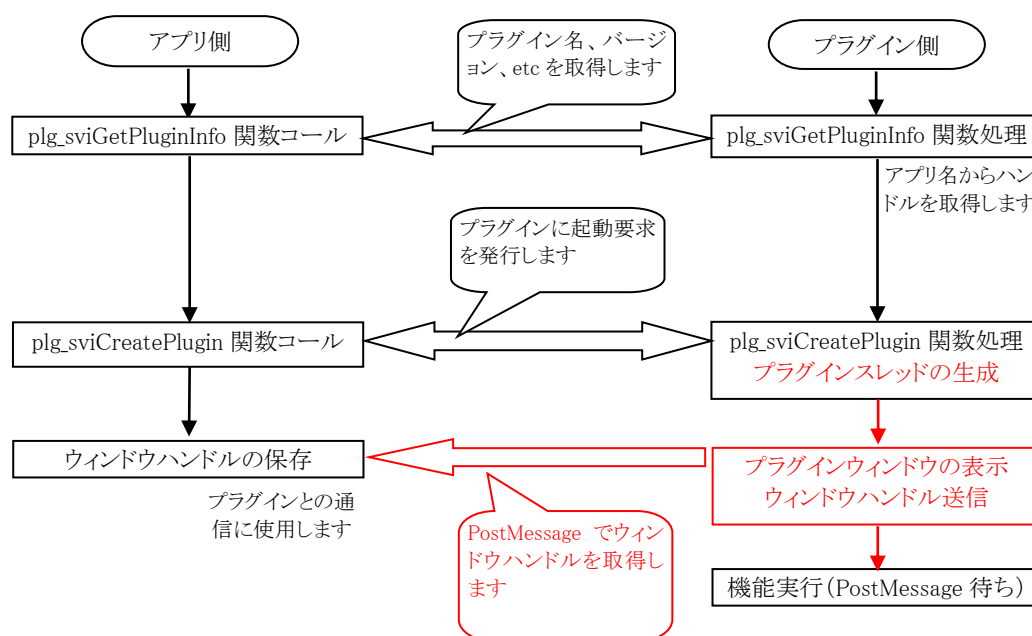
3.1. 前提

プラグインを作成するには SVIPluginComm.h および SVIComm.cpp をインクルードする必要があります。このヘッダーファイルには、通信に必要なウィンドウメッセージ、引数および共有メモリ関連の関数などが定義されています。

3.2. Plugin の起動

プラグインの起動(プラグインの確認、ウィンドウ表示等)に関する説明を記述します。下図に起動シーケンスを示します。

【図 2】 プラグイン起動シーケンス



SVImon、SVIview は LoadLibrary 関数により呼び出し元と同ディレクトリにある plg ディレクトリ内の各 plugin(dll)にアクセスを行い、plg_sviGetPluginInfo 関数へのアクセスが正常だった場合、そのプラグイン名をメニューに追加します。

3.2.1. plg_sviGetPluginInfo

plg_sviGetPluginInfo 関数は必須関数です、その仕様は以下のとおりです。

void CALLBACK plg_sviGetPluginInfo(

 HWND hParentWnd, // [IN] 呼び出し元ウィンドウハンドル
 BOOL bPMode, // [IN] 呼び出し元識別フラグ
 LPSTR strSharedName, // [IN] 呼び出し元共有メモリ名
 LPSTR strPlugInfo, // [OUT] プラグイン情報
 int* iSpec) // [OUT] プラグイン情報

{

 第一引数 hParentWnd (IN) : 呼び出し元ウィンドウハンドル

 第二引数 bPMode (IN) : 呼び出し元識別フラグ

 TRUE: SVImon

 FALSE: SVIview

 第三引数 strSharedName (IN) : 呼び出し元共有メモリ名

 第四引数 strPlugInfo (OUT) : プラグイン情報

 プラグイン名, バージョン, 起動関数名

 ・プラグイン名: プラグインの名前。この名前が SVImon、SVIview の PLUGIN のプルダウンメニューに表示されます。例(”MonHeadMonior”, ”Histogram”等プラグイン名)

 (区切られた文字の合計の総数は最大で 128 バイトとし、終端は NULL とします。)

 ・バージョン: 各プラグインのバージョン情報。例”1.0.0.1”

 ・起動関数名: 各プラグインが起動される関数名。この関数名が違くと、プラグインは起動されません。

 現在”plg_sviCreatePlugin”で固定です。

 第五引数 iSpec(OUT): プラグイン情報

 15 - 8 ビット プラグインオート起動 ON/OFF

 PLG_AUTO_ON : オート起動 ON

 PLG_AUTO_OFF : オート起動 OFF

 7 - 0 ビット プラグイン対応種別

 SPEC_MON : SVImov 専用プラグイン

 SPEC_VIEW : SVIview 専用プラグイン

 SPEC_COMMON : 共通プラグイン

 呼び出し元でメニューボタンが押されたら plugin の起動関数名をコールし plugin を起動表示します。

3.2.2. plg_sviCreatePlugin

plg_sviCreatePlugin 関数は起動関数で必須関数です、その仕様は以下のとおりです。

UINT plg_sviCreatePlugin(int nID)

 第一引数 nID (IN) : プラグイン識別番号

戻り値

UINT:エラーコード。

この関数はプラグインスレッドを生成します。

3.2.3. plg_sviWaitSignal

plg_sviWaitSignal 関数は終了関数で必須関数です、その仕様は以下のとおりです。

void plg_sviWaitSignal ()

戻り値

なし。

この関数は、アプリケーション終了時にプラグインスレッドの終了を待ちます。

3.2.4. plg_sviBootThread

plg_sviBootThread 関数はワーカースレッドの制御関数です、その仕様は以下の通りです。

static UINT plg_sviBootThread(LPVOID pParam)

戻り値

UINT:エラーコード。

この関数は、アプリケーション終了時にプラグインスレッドの終了を待ちます。

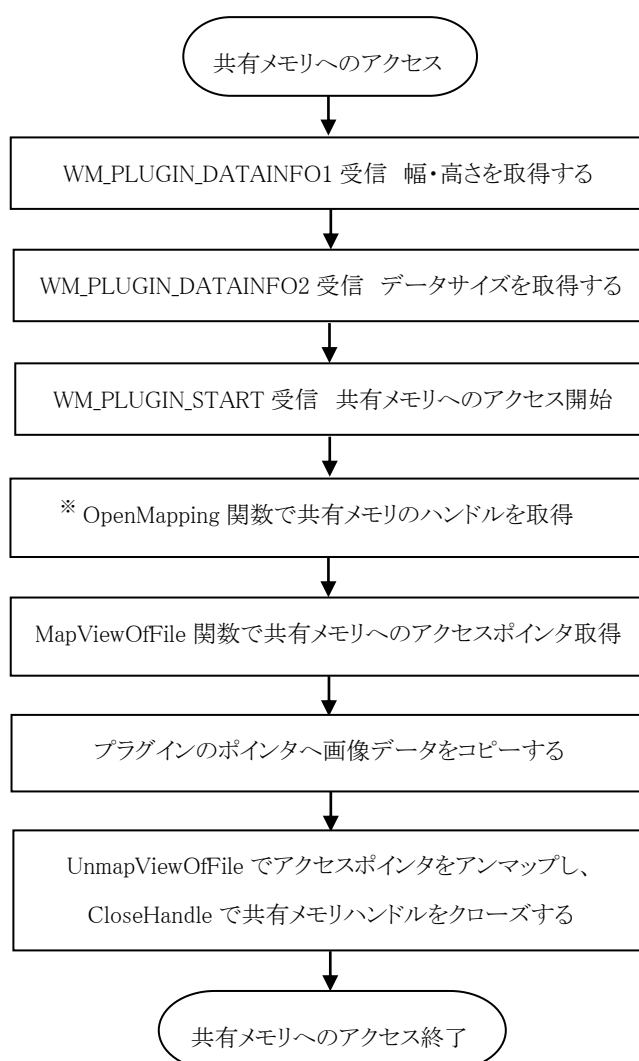
3.3. 呼び出し元識別およびメッセージの送信方法

プラグイン側で呼び出し元を識別するためには `plg_sviGetPluginInfo` 関数の第一引数である `strExeName` を用いる必要があります。詳細については付録 1 参考ソースをご参照下さい。

3.4. 共有メモリへのアクセス

呼び出し元が表示している画像データをプラグイン側で取得する際は共有メモリを介して画像データの取り込みを行います。現在呼び出し元との共有メモリにマッピングされている画像データの幅・高さおよびデータサイズは `PostMessage` の `WM_PLUGIN_DATAINFO1` および `WM_PLUGIN_DATAINFO2` で呼び出し元から情報を受け取ります。共有メモリからデータを取り込むタイミングはプラグイン側で `WM_PLUGIN_START` を受信したときです。

【図 4.2】 共有メモリへのアクセス



※ OpenMapping 関数:SVIComm で定義されている共有メモリハンドル取得関数。

```
int OpenMapping(  
    DWORD    dwDesiredAccess,          // [IN] 共有メモリアクセスモード  
    HANDLE*   hMapping,                 // [IO] 共有メモリマッピングハンドル  
    CString   strSharedMemoryName );   // [IN] 共有メモリ名
```

アクセスモード一覧表

FILE_MAP_WRITE	読み書きアクセスです。対象となるファイルマッピングオブジェクトは、PAGE_READWRITE で作成されていなければなりません。
FILE_MAP_READ	読み取り専用アクセスです。対象となるファイルマッピングオブジェクトは、PAGE_READWRITE か PAGE_READ で作成されていなければなりません。
FILE_MAP_ALL_ACCESS	FILE_MAP_WRITE と同じです。
FILE_MAP_COPY	読み書きアクセスです。ただし、元のファイルは変更されず、元のファイルの複製が使われます。対象となるファイルマッピングオブジェクトは、PAGE_WRITECOPY で作成されていなければなりません。

なお、Recorder、Viewer は共有メモリを FILE_MAP_ALL_ACCESS モードで作成しています。

戻り値

DEF_ERR:共有メモリオープン失敗

DEF_OK:成功

3.5. PostMessage()による呼び出し元との通信内容

送信側は PostMessage()関数を使用しメッセージを送信します。PostMessage のフォーマットを以下に示します。

```
PostMessage(  
    送信先ウィンドウハンドル,  
    ウィンドウメッセージ,  
    第一引数(WPARAM),  
    第二引数(LPARAM)  
);
```

受信側はメッセージマップに受信用関数を宣言しておく必要があります。受信用関数については付録 1 参考ソースをご参照下さい。

3.5.1. 通信に用いるウィンドウメッセージの種類

ウィンドウメッセージ及び引数の対応を以下に示します。

3.5.1.1. 呼び出し元からプラグインへのメッセージ

(1) WM_CLOSEPARENTS

呼び出し元が閉じられたこと知らせるメッセージです。このメッセージを受信した場合はプラグインは速やかにクローズ処理を行わなければなりません。

第一引数: なし

第二引数: なし

(2) WM_PLUGIN_DATAINFO1

現在呼び出し元で表示されている画像のXYサイズをプラグイン側で受信するためのメッセージです。メッセージが送信されるタイミングは親側で画像が表示された直後です。画像表示が無効の場合でもタイミングは同じです。(画像表示が無効の場合とは描画停止などの状態です)

第一引数: 画像の X サイズ(DWORD 型にキャストして使用)

第二引数: 画像の Y サイズ(DWORD 型にキャストして使用)

(3) WM_PLUGIN_DATAINFO2

現在呼び出し元で表示されている画像のサイズ(パディングは考慮せず)をプラグイン側で受信するためのメッセージです。メッセージが送信されるタイミングは WM_PLUGIN_DATAINFO1 の直後です。

第一引数: 画像のサイズをバイトで現します(DWORD 型にキャストして使用)

第二引数: なし

(4) WM_PLUGIN_START

プラグインの計算開始タイミングメッセージ。この時点で、呼び出し元で現在表示している画像の生データが共有メモリにマッピングされているのでそのデータを読み込むタイミングにも使用するです。メッセージが送信されるタイミングは WM_PLUGIN_DATAINFO2 の直後です。

第一引数: 現在共有メモリにマッピングされているデータフォーマット

TRUE: YUV データ FALSE: 16 ビット RGB データ

第二引数: 絶対有効フラグ (使用しなくてもよい)

SURELYREAD: 必ず処理を行ってほしい場合 (最初や最後の画像等)

UNSURELYREAD: 必しも処理を行わなくてもよい場合

(5) WM_PLUGIN_XYPOINT

現在表示されている画像の Y 信号 (生データ) の一番高い位置を受信するためのメッセージです。メッセージが送信されるタイミングは読み込んだ画像を表示データへ変換した直後です。

第一引数: 現在表示されている画像で Y 信号が一番高い X 座標 (DWORD 型にキャスト)

第二引数: 現在表示されている画像で Y 信号が一番高い Y 座標 (DWORD 型にキャスト)

(6) WM_PLUGIN_YVALUE

現在表示されている画像の Y 信号 (生データ) の一番高い値を受信するためのメッセージです。メッセージが送信されるタイミングは WM_PLUGIN_XYPOINT を送信した直後です。

第一引数: 現在表示されている画像で一番高い Y の値

第二引数: なし

(7) WM_PLUGINCHANGEAUTORESIZE

呼び出し元の AutoResize 機能が有効／無効をチェックするメッセージです。メッセージが送信されるタイミングは WM_PLUGIN_YVALUE を送信した直後または Option で OK ボタンが押された直後です。

第一引数: AutoResize の有効／無効

有効: DEF_AUTORESIZE 無効: DEF_UNAUTORESIZE

第二引数: なし

(8) WM_MEMORYRELOCK

SVImon でモニタリングバッファサイズ及び共有メモリサイズが新しく設定されたときに送信されるメッセージです。メッセージが送信されるタイミングは Option で OK ボタンが押された直後です。

第一引数: モニタリングバッファサイズ (DWORD 型にキャスト)

第二引数: 共有メモリサイズ (DWORD 型にキャスト)

(9) WM_PLUGIN_SHAREDMEMORY

SVImon で Recoding が終了したときに送信されるメッセージです。

第一引数: レコーディングデータサイズ (DWORD 型にキャスト)

第二引数:なし

(10) **WM_SETMONSIZE**

SVImon でモニタリングの切り出しサイズが変更されたときに送信されるメッセージです。

第一引数:切り出し位置及びサイズ(CRect 型へのアドレス)

第二引数:なし

(11) **WM_STARTMONITORING**

SVImon でモニタリングを開始した時に送信されるメッセージです。

第一引数:なし

第二引数:なし

(12) **WM_STOPMONITORING**

SVImon でモニタリングを停止した時に送信されるメッセージです。

第一引数:なし

第二引数:なし

(13) **WM_SETPICMODE**

SVImon のカラー識別が送信されます。プラグイン側から WM_SETPICMODE を設定した際の戻り値としても送信されます。

第一引数:カラー識別 (PicType)

第二引数:なし

(14) **WM_GETSTILLIMAGE**

SVImon でスチル画像モードの結果を送信します。このメッセージは SVImon でスチル画像を描画している間に再描画等が行われると再送されます。

第一引数:スチル取得結果 (StillResult 型値が代入)

第二引数:なし

(15) **WM_EASYLOGIC_ANALYZER_POS**

SVIView で画像をクリックするとその位置の波形を表示するメッセージです。

第一引数:なし

第二引数:表示位置(16 or 32bit でもポジション値は1でカウント)

3.5.1.2. プラグインから呼び出し元へのメッセージ

(1) **WM_CLOSEPLUGIN**

プラグインが閉じられたときに呼び出し元へ送信するメッセージです。送信するタイミングは、プラグイン側で WM_DESTROY または WM_CLOSE メッセージ受信後が理想です。

第一引数: 閉じられたプラグインのウィンドウハンドル (DWORD 型にキャスト)

第二引数: NULL

(2) WM_WAVE_DRAWSTARTPOS

呼び出し元に描画する赤い線に対して表示／非表示を切り替えるためのメッセージです。送信するタイミングはとくに指定はありません。

第一引数: 引く線の向き

UNDRAWLINE: 線を非表示にします

DRAW_HORIZON: 平行線を描画します

DRAW_VERTICAL: 垂直線を描画します

第二引数: 線を引く位置 (DWORD 型)

(3) WM_WAVE_DRAWENDPOS

呼び出し元に描画する青い線に対して表示／非表示を切り替えるためのメッセージです。送信するタイミングはとくに指定はありません。

第一引数: 引く線の向き

UNDRAWLINE: 線を非表示にします

DRAW_HORIZON: 平行線を描画します

DRAW_VERTICAL: 垂直線を描画します

第二引数: 線を引く位置 (DWORD 型)

(4) WM_PLUGINSETSTEPCLIP

呼び出し元にある Y の値以上の部分を赤く描画させる場合に用いる値をセットするメッセージ。

第一引数: 0 から設定されたステップで明るくなるように描画するための値

第二引数: この値で設定された Y レベル以上のピクセルを赤に変換します

(5) WM_DRAWREDOFFSETSTART

呼び出し元である Y レベル以上の値のピクセルを赤く描画する処理を開始するメッセージです。この設定は WM_DRAWREDOFFSETEND メッセージで赤くする描画を停止するまで有効です。

第一引数: NULL

第二引数: NULL

(6) WM_DRAWREDOFFSETEND

呼び出し元である Y レベル以上の値のピクセルを赤く描画する処理を停止するメッセージです。送信するタイミングは特に指定はありません。

第一引数: NULL

第二引数: NULL

(7) WM_STARTDRAWRECT

呼び出し元で緑の四角形を描画する処理を開始するためのメッセージです。この設定は WM_STOPDRAWRECT メッセージで描画を停止するまで有効です。

第一引数: NULL

第二引数: NULL

(8) WM_STOPDRAWRECT

呼び出し元で緑の四角形を描画する処理を停止するためのメッセージです。送信するタイミングは特に指定はありません。

第一引数: NULL

第二引数: NULL

(9) WM_SETRECTPOS

呼び出し元で緑の四角形を描画する際に描画する四角形の左上位置を指定するためのメッセージです。送信は WM_STARTDRAWRECT メッセージで描画を有効にする直前に送信するのが理想です。

第一引数: 描画する四角形の X 位置 (int または DWORD 型)

第二引数: 描画する四角形の Y 位置 (int または DWORD 型)

(10) WM_SETRECTWH_AND_DRAW

呼び出し元で緑の四角形を描画する際に描画する四角形の幅と高さを指定するためのメッセージです。送信は WM_SETRECTPOS メッセージを送信した直後が理想です。

第一引数: 描画する四角形の幅 (int または DWORD 型)

第二引数: 描画する四角形の高さ (int または DWORD 型)

(11) WM_PAINTCONTROL

呼び出し元の絵の描画を停止させるためのメッセージです。呼び出し元では絵が表示されないだけで、モニタリングを停止したり、プラグインへのメッセージの送信を停止したり等はしません。メッセージを送信するタイミングは特に指定はありません。

第一引数: NULL

第二引数: NULL

(12) WM_UNPAINTCONTROL

呼び出し元の絵の描画を再開させるためのメッセージです。

第一引数: NULL

第二引数: NULL

(13) WM_PLUGINYGAINOFFSET

呼び出し元の Y レベルに対して行うオフセットの値をセットするためのメッセージです。詳細については付録 1

参考ソースを参照してください。

第一引数:ゲイン値(BYTE または int 型)

ゲイン値に DEF_GETDOUBLELENGTH を乗じた値

第二引数:オフセット値(BYTE または int 型)

(14) WM_PLUGINYGAINOFFSETENABLE

呼び出し元のYレベルに対して行うオフセットの有効／無効を指定します。

第一引数:有効／無効

TRUE:有効 FALSE:無効

第二引数:NULL

(15) WM_PLUGINBMPSNAPSHOT

呼び出し元(SVImon)に現在表示されている画像のビットマップスナップショットを要求します。このメッセージ受信後、SVImon は SVImon のいるディレクトリ内に BmpSnapshot ディレクトリを作成し、その中にビットマップファイルを作成し保存します。

第一引数:NULL

第二引数:NULL

(16) WM_PLUGINCHANGEAUTORESIZE

呼び出し元に AutoResize 機能が有効／無効を送信するメッセージです。メッセージが送信されるタイミングはプラグインモジュールで AutoResizeWindow のチェックが変更された直後です。

第一引数:AutoResize の有効／無効

有効: DEF_AUTORESIZE 無効: DEF_UNAUTORESIZE

第二引数:なし

(17) WM_CREATEWINDOW

プラグインウィンドウが表示される時に送信するメッセージです。

第一引数:呼び出し元からの識別番号

第二引数:プラグインのウィンドウハンドル

(18) WM_SETMONSIZE

呼び出し元(SVImon)にモニタリング時の切り出しサイズの変更を要求するメッセージです。

呼び出し元(SVImon)がモニタリング中の場合、一度モニタリングが停止されます。設定終了後モニタリングを行うかは第二引数の設定で行います。

第一引数:切り出し位置及びサイズ(CRect 型へのアドレス)

第二引数:モニタリング自動開始フラグ

TRUE:モニタリング自動開始を行わない

FALSE:モニタリング自動開始を行う

(19) WM_STARTMONITORING

呼び出し元(SVImon)にモニタリング開始を要求するメッセージです。呼び出し元(SVImon)がモニタリング中であれば、モニタリング開始処理は行わずモニタリング開始メッセージのみを送信します。

第一引数:なし

第二引数:なし

(20) WM_STOPMONITORING

呼び出し元(SVImon)にモニタリング停止を要求するメッセージです。呼び出し元(SVImon)がモニタリングを停止していれば、モニタリング停止処理は行わずモニタリング停止メッセージのみを送信します。

第一引数:なし

第二引数:なし

(21) WM_SETPICMODE

呼び出し元(SVImon)の画像カラーの種別を変更します。また現在設定されたカラー種別を取得することもできます。

第一引数:動作識別フラグ

TRUE:現在のカラー識別を取得する

FALSE:呼び出し元のカラー識別を第二引数に設定されているカラー識別に変更する

第二引数:カラー識別

共通ヘッダーに定義されている PictureType 型を設定する

(22) WM_GETSTILLIMAGE

呼び出し元(SVImon)にスチル画像の取得モードに移行するよう要求するメッセージです。このメッセージの前にモニタリングを停止させるとスチル取得率が上昇します。

第一引数:スチル動作モード

TRUE:スチル画像取得

FALSE:スチル画像取得キャンセル

第二引数:モニタリング自動開始

TRUE:自動開始

FALSE:現状のモード

(23) WM_SETSTILLSIZE

呼び出し元(SVImon)にスチル画像判別のためのサイズを設定します。

第一引数:スチル画像の幅

第二引数:スチル画像の高さ

付録 1 参考ソース

各用途の簡略化したソースを以下に示します。

① plg_sviGetPluginInfo 関数

```
void CALLBACK plg_sviGetPluginInfo(
                                HWND    hParentWnd,    // [IN] 呼び出し元ウィンドウハンドル
                                BOOL    bPMode,        // [IN] 呼び出し元識別フラグ
                                LPSTR   strSharedName,  // [IN] 呼び出し元共有メモリ名
                                LPSTR   strPlugInfo,    // [OUT] プラグイン情報
                                int*    iSpec )         // [OUT] プラグイン対応種別
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    g_hParentWnd    = hParentWnd;    // ハンドル保存
    g_bParentMode   = bPMode;        // 呼び出し元識別フラグ保存
    g_strSharedName = strSharedName; // 呼び出し元共有メモリ名

    // オート起動 ON, SVImon 専用
    *iSpec          = PLG_AUTO_ON|SPEC_MON;

    sprintf( strPlugInfo, "%s,%s,%s",
                                DEF_NAME_LENSSHADING,    // プラグイン名
                                DEF_VER_LENSSHADING,      // プラグインバージョン
                                DEF_PROCNAME_PLUGIN );     // 起動関数名
}
```

② plg_sviCreatePlugin 関数

```
static CWinThread*    ggpsviBootThread;    // プラグインスレッド
UINT CALLBACK plg_sviCreatePlugin(int nNo)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    // スレッド起動
    ggpsviBootThread = NULL;
    ggpsviBootThread = AfxBeginThread( plg_sviBootThread, NULL );
    theApp.m_nPluginNo = nNo;
}
```

③ plg_sviBootThread 関数

```
static CWinThread*      ggpsviBootThread;    // プラグインスレッド
static UINT plg_sviBootThread(LPVOID pParam)
{
    AFX_MANAGE_STATE( AfxGetStaticModuleState() );
    // プラグインダイアログ起動
    theApp.m_Dlg5cu.DoModal();
    return 0;
}
```

※ theApp:WinApp クラスのグローバル変数。MFC によって自動的に生成されます。

④ plg_sviWaitSignal 関数

```
static CWinThread*      ggpsviBootThread;    // プラグインスレッド
void CALLBACK plg_sviWaitSignal()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    // ggpsviBootThread の解放
    if (ggpsviBootThread != NULL) {
        ::WaitForSingleObject( ggpsviBootThread->m_hThread, INFINITE );
        ggpsviBootThread = NULL;
    }
}
```

⑤ 共有メモリ読み込み

共有メモリをオープン後はロックがかかってしまうため、他のアプリからのアクセスができなくなります。そのため共有メモリからのデータの読み込みは素早く行うようにしてください。

```
// 共有メモリマッピング
if( OpenMapping( FILE_MAP_READ, &hBuffMapping, theApp.m_strSharedName ) == DEF_ERR ) {
    return;
}

m_pYUVBuff = (LPBYTE)::MapViewOfFile( hBuffMapping, FILE_MAP_READ, 0, 0, 0 );
if( m_pYUVBuff == NULL ) { // 失敗
    // 共有メモリも削除
    ::CloseHandle( hBuffMapping );
    hBuffMapping = NULL;
    return;
}
else
{
    // データのコピー処理
    if(m_pDataBuff!=NULL) // ヘッダー一部読み飛ばし
        memcpy( m_pDataBuff, m_pYUVBuff + sizeof( MON_HEADER ), m_DataSize);

    ::UnmapViewOfFile( m_pYUVBuff );
    ::CloseHandle( hBuffMapping );
}
}
```

※ OpenMapping 関数:iCAMPluginComm.h に定義されています。

※ theApp:WinApp クラスのグローバル変数。MFC によって自動的に生成されます。

⑥ 呼び出し元へのメッセージ送信

```
if( theApp.m_hParentWnd != NULL ){ // ウィンドウハンドルが NULL 以外の場合
    ::PostMessage( theApp.m_hParentWnd, WM_STARTDRAWRECT, NULL, NULL );
} // 呼び出し元ハンドル取得
```

※ theApp:WinApp クラスのグローバル変数。MFC によって自動的に生成されます。

⑦ 呼び出し元からのメッセージ受信

CDialog からの派生クラスの場合

CXXXDlg.cpp 内

```
BEGIN_MESSAGE_MAP(CWaveMonitorDlg, CDialog)
   //{{AFX_MSG_MAP(CWaveMonitorDlg)
        ON_MESSAGE( WM_XXXX, OnPostXXX )    // ← ウィンドウメッセージと関数を結びつける
    // 追加終わり
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

Void CXXXDlg::OnPostXXX(WPARAM wParam, LPARAM lParam) // 受信関数

```
{
    処理
}
```

CXXXDlg.h 内

protected:

```
HICON m_hIcon;

// 生成されたメッセージ マップ関数
//{{AFX_MSG(CWaveMonitorDlg)
afx_msg void OnPostXXX( WPARAM wParam, LPARAM lParam );    // ← メッセージマップ定義
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
```

theApp※ theApp:WinApp クラスのグローバル変数。MFC によって自動的に生成されます。

⑥ ゲイン、オフセット値の送信

ゲイン値:1.0 オフセット値:30 の場合

```
int dGainValue      = 1.0;
int iOffsetValue;    = 30;
int iGainValue       = (int)( dGainValue * (double)DEF_GETDOUBLELENGTH );
```

If(theApp.m_hParentWnd != NULL)

```
::PostMessage( theApp.m_hParentWnd, WM_PLUGINYGAINOFFSET, iGainValue, iOffsetValue );
```

⑦ レコーディングデータへのアクセス

ON_MESSAGE(WM_PLUGIN_SHAREDMEMORY, OnPostReadRecordingData)

•

```

UINT OnPostReadRecordingData(
    WPARAM wParam,          // [IN] データサイズ
    LPARAM lParam )         // なし
{
    DWORD* pRecordingData = new DWORD[wParam];
    DWORD* pBuff;
    HANDLE hHandle;

    if( OpenMapping( FILE_MAP_WRITE, &hHandle, SHARED_RECORDER_NAME ) == DEF_ERR ) {
        AfxMessageBox( "共有メモリエラー" );
        return 0;
    }

    pBuff = (DWORD*)::MapViewOfFile( hHandle, FILE_MAP_WRITE, 0, 0, 0 );
    if( pBuff == NULL ) {
        AfxMessageBox( "共有メモリエラー" );
        // 共有メモリも削除
        ::CloseHandle( hBuffMapping );
        hHandle = NULL;
        return 0;
    }

    // SVImon メモリから SVIview メモリへデータコピー
    // レコーディングデータは最大 128M バイトになります、メモリ管理に注意が必要です
    memcpy( pRecordingData, pBuff, wParam );

    // マッピングを解除
    ::UnmapViewOfFile( pBuff );
    ::CloseHandle( hHandle );

    delete pRecordingData;

    return 0;
}

```

⑧ 共有メモリデータ構成

・モニタリングデータ、SVIview で表示されている画像データ

* ヘッダ部 (512バイト)

ULONG	ulStatus;	// 実行結果のステータス(1H:成功、その他:異常)
ULONG	ulPendingInfo;	// 保留フレーム数
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;	// 前フレームから本フレーム間に破棄したフレーム数
USHORT	usCursorX;	// 十字カーソルの X 座標
USHORT	usCursorY;	// 十字カーソルの Y 座標
USHORT	usWakuLeftX;	// 枠の左上の X 座標
USHORT	usWakuLeftY;	// 枠の左上の Y 座標
USHORT	usWakuRightX;	// 枠の右下の X 座標
USHORT	usWakuRightY;	// 枠の右下の Y 座標
ULONG	ulReserve1;	// 予約
ULONG	ulVsyncReadLen;	// V 同期読み出しのデータ長 (0-448)
ULONG	ulVsyncVal[448/4];	// V 同期読み出しのデータ

注: Viewer データの時はカメラが出力している画像サイズ等不定値となります

* データ部

モニタリングデータ、Viewer で表示されている画像データ

※ YUV 形式: y0 u0 v0 y1 u1 v1 y2 u2 v2

※ RGB 形式: RG0 GB0 RG1 GB1 RG2 GB2

•レコーディングデータ

* ヘッダ部 (60 バイト)

ULONG	ulStatus;	// 実行結果のステータス(1H:成功、その他:異常)
ULONG	ulPendingInfo;	// 保留レコーディングデータサイズ
UCHAR	ubFirmVersion;	// ファームウェアバージョン番号
UCHAR	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)
UCHAR	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 2:32bits)
UCHAR	ulReserve[5];	// 予約

* データ部

レコーディングデータ(16Bit or 32bit データ)