**PIMPOM**

**APIリファレンス**

****

©Copyright Yasuyuki Ikeda　2020

All Rights Reserved

**改訂履歴**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver.** | **日付** | **担当** | **内容** | **ページ** |
| 1.0 | 2014.02.15 | Y.Ikeda | 新規作成 | すべて |
| 1.1 | 2015.01.13 | Y.Ikeda | DrawLineOnImage,DrawCircleOnImage,DrawRectOnImageの引数座標をintからdoubleに変更 | p26-27 |
| 1.2 | 2020.09.14 | Y.Ikeda | Mat2Img, Img2Matを追加 | p42 |
| 1.5 | 2020.12.28 | Y.Ikeda |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

[1. 概要 4](#概要)

[1.1. 目的 4](#概要)

[1.2. 用語の定義 4](#概要)

[1.3. 参考資料一覧 4](#概要)

[2. API仕様 5](#API仕様)

[2.1. クラスメンバ一覧 6](#クラスメンバ一覧)

[2.1.1. CPimpomAPI 6](#CPimpomAPI)

[2.1.2. CDataUnit 7](#CDataUnit)

[2.1.3. CConfiemDlg 8](#CConfirmDlg)

[2.2. 仕様詳細 9](#仕様詳細)

[2.2.1. CPimpomAPI 9](#CPimpomAPIのメンバ関数)

[2.2.1. CDataUnit 42](#CDataUnitのメンバ関数)

[2.2.1. CConfirmDlg 45](#CConfirmDlgのメンバ関数)

[2.3. カスタムファンクションのイベントハンドラ 47](#カスタムファンクションのイベントハンドラ)

[2.4. 定数一覧 49](#定数一覧)

# 概要

## 目的

本仕様書はPIMPOMのカスタムファンクションで使用するAPIについて記載する

## 用語の定義

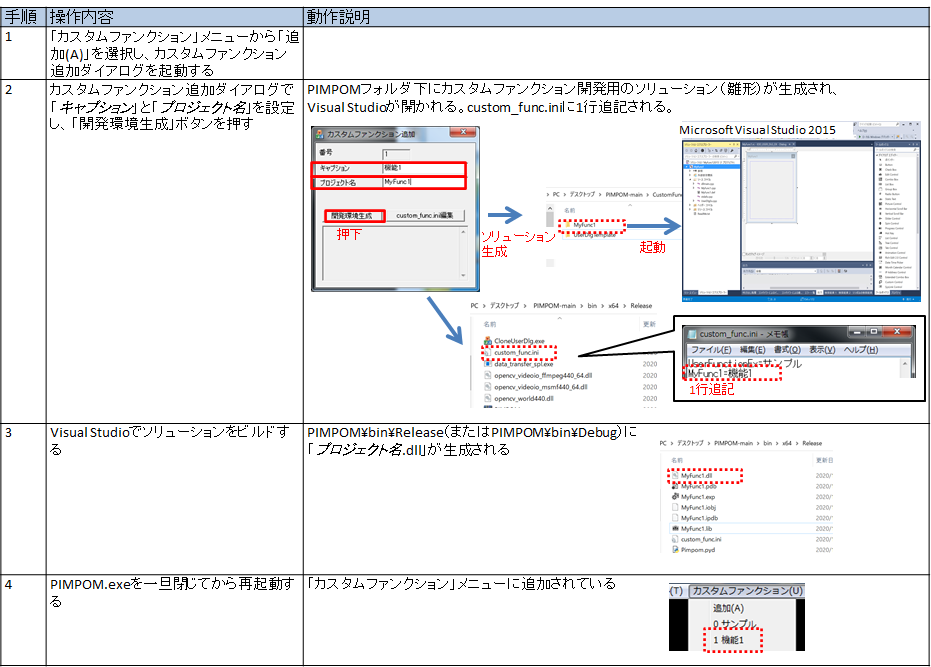
（本仕様書で使用する用語の定義を記載する。既に共通で使用している用語に関しては記載不要とする。）

|  |  |
| --- | --- |
| **用語** | **説明** |
|  |  |
|  |  |
|  |  |

## 参考資料一覧

|  |  |  |
| --- | --- | --- |
| **No.** | **ドキュメント名** | **用途** |
| 1 | PIMPOMソフトウェア仕様書 | PIMPOMの各機能の詳細を記載 |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

## カスタムファンクション追加手順



# API仕様

## クラスメンバ一覧

### CPimpomAPI

概要

PIMPOMの各種機能をラッピングしたクラス。

カスタムファンクションのプロジェクトではグローバル変数として本クラスのオブジェクト「API」が宣言されており、UserDlg.cpp内で各メンバ関数を使用できる

メンバ関数一覧

|  |  |  |
| --- | --- | --- |
| **機能分類** | **関数名** | **説明** |
| 画像メモリアクセス | [GetDataUnitNumber](#GetDataUnitNumber) | 画像メモリの最大数を取得する |
| [GetCurrentImageNumber](#GetCurrentImageNumber) | 現在選択中の画像メモリ番号を取得する |
| [SetCurrentImageNumber](#SetCurrentImageNumber) | 選択画像メモリの番号を設定する |
| [GetDataUnit](#GetDataUnit) | CDataUnitを取得する |
| [GetByteMemory](#GetByteMemory) | BYTE型の画像データの先頭ポインタを取得する |
| [GetWordMemory](#GetWordMemory) | WORD型の画像データの先頭ポインタを取得する |
| [GetDwordMemory](#GetDwordMemory) | DWORD型の画像データの先頭ポインタを取得する |
| [GetShortMemory](#GetShortMemory) | SHORT型の画像データの先頭ポインタを取得する |
| [GetLongMemory](#GetLongMemory) | LONG型の画像データの先頭ポインタを取得する |
| [GetFloatMemory](#GetFloatMemory) | FLOAT型の画像データの先頭ポインタを取得する |
| [GetRGBMemory](#GetRGBMemory) | RGB型の画像データの先頭ポインタを取得する |
| [Get3DMemory](#Get3DMemory) | 3D型の画像データの先頭ポインタを取得する |
| [GetF3DMemory](#GetF3DMemory) | FLOAT3D型の画像データの先頭ポインタを取得する |
| [GetRGB3DMemory](#GetRGB3DMemory) | RGB3D型の画像データの先頭ポインタを取得する |
| [GetXYMemory](#GetXYMemory) | XY型の画像データの先頭ポインタを取得する |
| [GetXYZMemory](#GetXYZMemory) | XYZ型の画像データの先頭ポインタを取得する |
| [GetComplexMemory](#GetComplexMemory) | COMPLEX型の画像データの先頭ポインタを取得する |
| [GetMaskMemory](#GetMaskMemory) | マスクの先頭ポインタを取得する |
| [FormatDataUnit](#FormatDataUnit) | 画像メモリを新しく取得する |
| [CopyDataUnit](#CopyDataUnit) | 画像メモリをコピーする |
| [DeleteDataUnit](#DeleteDataUnit) | 画像メモリを削除する |
| [BackupDataUnit](#BackupDataUnit) | 画像メモリをバックアップ領域に入れる |
| [RecoverDataUnit](#RecoverDataUnit) | バックアップ領域から画像メモリを復元する |
| [SetWorkArea](#SetWorkArea) | 作業領域を設定する |
| [GetWorkArea](#GetWorkArea) | 作業領域を取得する |
| [AddMark](#AddMark) | マークを１点追加する |
| [DeleteMark](#DeleteMark) | マークを１点削除する |
| [GetMarkVal](#GetMarkVal) | マークの値を取得する |
| [GetMarkNumber](#GetMarkNumber) | マークの数を取得する |
| GetSrcFilePath | 読み込み元のファイルのフルパスを返す |
| 画像データの変換 | [ConvertDataUnit](#ConvertDataUnit) | 画像データの型変換する |
| [ConvertToByte](#ConvertToByte) | 画像データの型をBYTEに変換する |
| [ClipDataUnit](#ClipDataUnit) | 画像データを矩形で切り取る |
| [ResizeDataUnit](#ResizeDataUnit) | 画像データのサイズ変換する |
| [PerspectiveTransDataUnit](#PerspectiveTransDataUnit) | 画像データの射影変換する |
| [IsoTarnsDataUnit](#IsoTarnsDataUnit) | 画像データの等方変換(回転、平行移動、拡大)する |
| [FilterSmooth](#FilterSmooth) | 画像データを移動平均で平滑化する |
| FilterGauss | 画像データにガウシアンフィルタをかける |
| FilterGaussUnshade | ガウシアン自己差分によるシェーディング除去 |
| [FilterMedian](#FilterMedian) | 画像データにメディアンフィルタをかける |
| [FilterErosion](#FilterErosion) | 画像データに収縮フィルタをかける |
| [FilterDilation](#FilterDilation) | 画像データに膨張フィルタをかける |
| [FilterEdgeSobel](#FilterEdgeSobel) | 画像データのエッジ抽出する |
| FilterLaplaceGauss | 画像データにラプラシアンオブガウシアンフィルタ |
| [FilterThinLog](#FilterThinLog) | 画像データの細線化エッジ抽出する(LoGゼロクロス) |
| FilterThinCanny | 画像データの細線化エッジ抽出する(Canny法) |
| FilterMirror | 画像データを鏡像反転する |
| FilterNega | 画像データを濃度反転する |
| [Binalize](#Binalize) | 画像データを2値化する |
| BinalizeThresholdOtsu | 大津の手法で2値化閾値を求める |
| BinalizeThresholdPTile | Pタイル法で2値化閾値を求める |
| BinalizeLocalGauss | ガウシアン自己差分によるシェーディング除去後に2値化する |
| BinalizeLocalLaplaceGauss | ラプラシアンオブガウシアンフィルタの後２値化する |
| [BinalizeNick](#BinalizeNick) | 画像をNICKの手法で2値化 |
| [Label](#Label) | 画像データを2値化してラベリングする |
| [HoughLines](#HoughLines) | Hough変換で直線を抽出する |
| [BlendPoisson](#BlendPoisson) | ポワソン画像合成 |
| [CalcDataValue](#CalcDataValue) | 任意の数式で画像データを生成する |
| [CalcMaskValue](#CalcMaskValue) | 任意の数式でマスクを生成する |
| 画像データへ図形描画 | [DrawLineOnImage](#DrawLineOnImage) | 画像データ上に直線を描画する |
| DrawLineOnImageF | 画像データ上に直線(ax+by+c)を描画する |
| [DrawCircleOnImage](#DrawCircleOnImage) | 画像データ上に円を描画する |
| [DrawRectOnImage](#DrawRectOnImage) | 画像データ上に矩形を描画する |
| [DrawTextOnImage](#DrawTextOnImage) | 画像データ上にテキストを描画する |
| [DrawPointOnImage](#DrawPointOnImage) | 画像データ上に点を描画する |
| [FloodFillOnImage](#FloodFillOnImage) | 画像データの値を塗りつぶす |
| [FloodFillOnMask](#FloodFillOnMask) | マスクの値を塗りつぶす |
| [FillWorkAreaOnImage](#FillWorkAreaOnImage) | 作業領域内の画像データの値を塗りつぶす |
| [FillWorkAreaOnMask](#FillWorkAreaOnMask) | 作業領域内のマスクの値を塗りつぶす |
| 画像表示 | [DrawImage](#DrawImage) | 画像表示をおこなう |
| ScaleImage | 表示画像のスケールを変更する |
| [SetDispRange](#SetDispRange) | 画像表示の濃度上限／下限値を設定する |
| [SetDispRangeAuto](#SetDispRangeAuto) | 画像表示の濃度上限／下限値を自動で設定する |
| メインウィンドウ制御 | [DrawLine](#DrawLine) | メインウィンドウ上に直線を描画する |
| [DrawCursor](#DrawCursor) | メインウィンドウ上に十字カーソルを描画する |
| [DrawRect](#DrawRect) | メインウィンドウ上に矩形を描画する |
| [DrawCircle](#DrawCircle) | メインウィンドウ上に円を描画する |
| [DrawEllipse](#DrawEllipse) | メインウィンドウ上に楕円を描画する |
| [DrawText](#DrawText) | メインウィンドウ上にテキストを描画する |
| [ShowMessage](#ShowMessage) | メインウィンドウにメッセージを出す |
| [DialogOpen](#DialogOpen) | 指定されたビルトインウィンドウを開く |
| [UserDialogOpen](#UserDialogOpen) | カスタムダイアログを開く |
| [GetMarkEditMode](#GetMarkEditMode) | マーク編集のOn/Off状態取得 |
| [SetMarkEditMode](#SetMarkEditMode) | マーク編集のOn/Off状態設定 |
| [GetWorkAreaEditMode](#GetWorkAreaEditMode) | 作業領域編集のOn/Off状態取得 |
| [SetWorkAreaEditMode](#SetWorkAreaEditMode) | 作業領域編集のOn/Off状態設定 |
| [GetRulerToolMode](#GetRulerToolMode) | ルーラーのOn/Off状態取得 |
| [SetRulerToolMode](#SetRulerToolMode) | ルーラーのOn/Off状態設定 |
| [GetVideoCaptureThroughMode](#GetVideoCaptureThroughMode) | ビデオキャプチャのライブ表示On/Off状態取得 |
| [SetVideoCaptureThroughMode](#SetVideoCaptureThroughMode) | ビデオキャプチャのライブ表示On/Off状態設定 |
| [CopyToClipbord](#CopyToClipbord) | 表示画像をクリップボードにコピーする |
| PasteFromClipbord | クリップボードにある画像をメモリに入れる |
| カスタムダイアログ制御 | [GetEditValue](#GetEditValue) | エディットボックスから値を取得する |
| [SetEditValue](#SetEditValue) | エディットボックスに値を設定する |
| [SetCheck](#SetCheck) | チェックボックス／ラジオボタンの値を設定する |
| [GetCheck](#GetCheck) | チェックボックス／ラジオボタンの値を取得する |
| [MessageBox](#MessageBox) | メッセージボックスを出す |
| ファイル入出力 | [SelectAndLoadImageFile](#SelectAndLoadImageFile) | 画像ファイルを読み込む |
| [SelectAndSaveImageFile](#SelectAndSaveImageFile) | 画像データをファイルに保存する |
| [SaveViewAsBitmap](#SaveViewAsBitmap) | 表示画像をビットマップとして保存する |
| [SaveArchivedImage](#SaveArchivedImage) | 画像データをアーカイブファイルとして保存する |
| [OpenFileDialog](#OpenFileDialog) | ファイルを開くダイアログを表示する |
| [FolderDialog](#FolderDialog) | フォルダを選択するダイアログを表示する |
| [GetDefaultPath](#GetDefaultPath) | Pimpom.exeのフォルダパスを取得する |
| グラフ表示 | [AddDataToChart](#AddDataToChart) | 1Dグラフにデータを１点追加する |
| [PlotDataToChart](#PlotDataToChart) | 1Dグラフにデータ配列を追加する |
| [AddDataToChart2](#AddDataToChart2) | 2Dグラフにデータを１点追加する |
| [PlotDataToChart2](#PlotDataToChart2) | 2Dグラフにデータ配列を追加する |
| [ClearChart](#ClearChart) | グラフをクリアする |
| OpenCV | [Img2Mat](#Img2Mat) | 画像データをcv::Mat型に変換する |
| [Mat2Img](#Mat2Img) | cv::Mat型を画像データに変換する |
| ビデオキャプチャ | [CaptureInit](#CaptureInit) | ビデオキャプチャを有効にする |
| [CaptureOne](#CaptureOne) | ビデオキャプチャを１回実行 |
| TCP/IP通信 | [SendString](#SendString) | TCP/IP通信でテキストを送信する |
| [GetSocketStatus](#GetSocketStatus) | TCP/IP通信の状態を取得する |
|  |  |  |
|  |  |

### CDataUnit

概要

画像メモリを扱うクラス

メンバ変数一覧

|  |  |  |
| --- | --- | --- |
| **データ分類** | **変数名** | **説明** |
| 画像データ | CString 　DataName | 画像データ名 |
| long 　DataType | 画像データのタイプ  BYTE\_FORMAT：8bit符号なしモノクロ静止画  　WORD\_FORMAT：16bit符号なしモノクロ静止画  DWORD\_FORMAT：32bit符号なしモノクロ静止画  SHORT16\_FORMAT：16bit符号つきモノクロ静止画  　LONG32\_FORMAT：32bit符号つきモノクロ静止画  　FLOAT\_FORMAT：32bit浮動小数点モノクロ静止画  RGB\_FORMAT:8bitカラー静止画  　THREE\_D\_FORMAT：8bit符号なしモノクロ動画  　RGB\_3D\_FORMAT：8bit符号なしモノクロ動画  　RGB\_3D\_FORMAT：8bitカラー動画  　XY\_FORMAT：32bit浮動小数点×2チャネル静止画  　XYZ\_FORMAT：32bit浮動小数点×3チャネル静止画  　　　COMPLEX\_FORMAT：複素数静止画 |
| BYTE \*pByteData | BYTE、RGB、3D、RGB3D型の画像データ配列先頭ポインタ |
| WORD \*pWordData | WORD型の画像データ配列先頭ポインタ |
| DWORD \*pDwordData | DWORD型の画像データ配列先頭ポインタ |
| short \*pShortData | SHORT型の画像データ配列先頭ポインタ |
| long \*pLongData | LONG型の画像データ配列先頭ポインタ |
| float \*pFloatData | FLOAT／FLOAT3D／XY／XYZ／COMPLEX型の画像データ配列先頭ポインタ |
| CSize DataSize | 画像データの幅と高さ |
| long PageNumber | 3D／RGB3D型ページ数（他のデータ型の場合は1） |
| long ChannelNumber | チャネル数  (XY型は2、RGB／RGB3D／XYZ型は3、COMPLEX型は4) |
| float Average | 画像データの平均値 |
| float Stdiv | 画像データ値の標準偏差 |
| 表示画像 | BYTE \*pDispBuffer | 表示用のビットマップデータ先頭ポインタ |
| BYTE \*pDispBufferRGB | RGBカラー表示用のビットマップデータ先頭ポインタ |
| long DispPage | 表示ページ番号 |
| long DispChannel | 表示チャネル番号 |
| bool DispRGB | RGBカラー表示するかどうか |
| long DispBufferRow | ビットマップデータの幅（4の倍数） |
| long DispBufferRowRG | RGBカラー表示用のビットマップデータの幅（4の倍数） |
| float BrightMax | 表示データ最大値 |
| float BrightMin | 表示データ最小値 |
| float DispScale | 表示画像の拡大率 |
| CPoint DataDispOffset | 表示位置のオフセット |
| long BrightLevel[256] | 表示明度レベル |
| 作業領域 | CWorkArea　WorkArea | 作業領域 |
| bool DoUseWorkArea | 作業領域使用フラグ |
| マーク | CMark \*pMark | マーク配列 |
| long MarkNum | マーク数 |
| マスク | BYTE \*pMaskData | マスクのデータ列の先頭ポインタ  マスクONの値は1、OFFの値は0 |
| bool DoUseMask | マスク使用フラグ |

メンバ関数一覧

|  |  |  |
| --- | --- | --- |
| **機能分類** | **関数名** | **説明** |
| 画像データを扱う | [Copy](#Copy) | 画像メモリをコピーする |
| [Compare](#Compare) | 画像メモリの型・サイズを比較する |
| [GetDataValue](#GetDataValue) | 指定座標の画像データ値を取得する |
| [GetDisplayBrightness](#GetDisplayBrightness) | 指定座標の表示画像の明度を取得する |
| [GetMaskValue](#GetMaskValue) | 指定座標のマスク値を取得する |
| [SetDataValue](#SetDataValue) | 指定座標の画像データ値を設定する |
| [SetMaskValue](#SetMaskValue) | 指定座標のマスク値を設定する |
| [ClearMask](#ClearMask) | マスクをすべてOFFにする |
| [InitWorkArea](#InitWorkArea) | 作業領域を初期化する  （画面全体の矩形領域が１つだけ設定される） |
| [SetMaskByWorkArea](#SetMaskByWorkArea) | 作業領域外部にマスクをかける |
| [GetRectArea](#GetRectArea) | 作業領域を表す矩形を取得する |
| [GetDataMinMax](#GetDataMinMax) | 画像データの最大値と最小値を計算する |
| [SetDispRange](#SetDispRange) | 画像表示の濃度上限／下限値を設定する |
| [SetDispScale](#SetDispScale) | 表示画像の拡大率を設定する |

### CConfirmDlg

概要

ダイアログボックスを表示して数値／文字データ入力を行うためのクラス

メンバ関数一覧

|  |  |  |
| --- | --- | --- |
| **機能分類** | **関数名** | **説明** |
| 数値／文字データ設定 | [SetWidth](#SetWidth) | ダイアログボックスの幅を設定する |
| [SetTitle](#SetTitle) | ダイアログボックスのタイトルを設定する |
| [RegistVar](#RegistVar) | 入力データの設定をする（ｴﾃﾞｨｯﾄﾎﾞｯｸｽ／ﾁｪｯｸﾎﾞｯｸｽ） |
| [RegistRadioVar](#RegistRadioVar) | 入力データの設定をする（ラジオボタン） |
| [DoModel](#DoModal) | ダイアログボックスを表示する |

## 仕様詳細

### CPimpomAPIのメンバ関数

long GetDataUnitNumber()

|  |  |
| --- | --- |
| 戻り値 | 画像メモリ最大数 |
| 説明 | 画像メモリの最大数を取得する（デフォルトでは30） |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

long GetCurrentImageNumber()

|  |  |
| --- | --- |
| 戻り値 | 現在選択中の画像メモリ番号 |
| 説明 | 現在選択中の画像メモリ番号を取得する |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetCurrentImageNumber(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| 戻り値 |  | |
| 説明 | 選択画像メモリの番号を設定する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

CDataUnit \*GetDataUnit(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| 戻り値 | 指定した画像メモリ番号のCDataUnitのポインタ  画像メモリが空の場合はNULLを返す | |
| 説明 | 指定した画像メモリ番号のCDataUnitのポインタを取得する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned char \*GetByteMemory(long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のBYTE型データの先頭ポインタを返す | |
| 説明 | BYTE型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくBYTE型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるBYTE型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned short \* GetWordMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のWORD型データの先頭ポインタを返す | |
| 説明 | WORD型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくWORD型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるWORD型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned long \* GetDwordMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のDWORD型データの先頭ポインタを返す | |
| 説明 | DWORD型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくDWORD型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるDWORD型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

short \* GetShortMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のSHORT型データの先頭ポインタを返す | |
| 説明 | SHORT型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくSHORT型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるSHORT型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

long \* GetLongMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のLONG型データの先頭ポインタを返す | |
| 説明 | LONG型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくLONG型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるLONG型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

float \* GetFloatMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のFLOAT型データの先頭ポインタを返す | |
| 説明 | FLOAT型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくFLOAT型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるFLOAT型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned char \* GetRGBMemory (long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のRGB型データの先頭ポインタを返す | |
| 説明 | RGB型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくRGB型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるRGB型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned char \*Get3DMemory(long number, CSize size, long page, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| long page | (in)画像データのページ数 |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズ・同ページ数の場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号の3D型データの先頭ポインタを返す | |
| 説明 | 3D型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しく3D型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズまたはページ数が異なる3D型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

float \* GetF3DMemory (long number, CSize size, long page, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| long page | (in)画像データのページ数 |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズ・同ページ数の場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のFLOAT3D型データの先頭ポインタを返す | |
| 説明 | FLOAT3D型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくFLOAT3D型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズまたはページ数が異なるFLOAT3D型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned char \*GetRGB3DMemory(long number, CSize size, long page, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| long page | (in)画像データのページ数 |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズ・同ページ数の場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号の3D型データの先頭ポインタを返す | |
| 説明 | RGB3D型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくRGB3D型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズまたはページ数が異なるRGB3D型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

float \* GetXYMemory(long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のXY型データの先頭ポインタを返す | |
| 説明 | XY型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくXY型の画像データを確保しなおし先頭ポインタを返す  ‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるXY型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

float \* GetXYZMemory(long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のXYZ型データの先頭ポインタを返す | |
| 説明 | XYZ型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくXYZ型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるXYZ型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

float \* GetComplexMemory(long number, CSize size, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| CSize size | (in)画像データサイズ（幅、高さ） |
| bool do\_clear\_memory | (in) 指定した画像メモリが同タイプ・サイズの場合、画像メモリを初期化するかどうか。 |
| 戻り値 | 指定した画像メモリ番号のCOMPLEX型データの先頭ポインタを返す | |
| 説明 | ・COMPLEX型の画像データの先頭ポインタを取得する  ・指定した画像メモリが以下の場合は新しくCOMPLEX型の画像データを確保しなおし先頭ポインタを返す  　‐画像メモリが空である  　‐画像メモリに型が異なる画像データが存在する  　‐して画像メモリにサイズが異なるCOMPLEX型データが存在する  ・画像メモリ確保に失敗した場合はNULLを返す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

unsigned char \* GetMaskMemory(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| 戻り値 | 指定した画像メモリ番号のマスクの先頭ポインタを返す | |
| 説明 | マスクの先頭ポインタを取得する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)CDataUnit　\*FormatDataUnit(long number, long type, CSize size, long page, bool do\_clear\_memory)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long type | (in)画像データのタイプ  BYTE\_FORMAT：8bit符号なしモノクロ静止画  　WORD\_FORMAT：16bit符号なしモノクロ静止画  DWORD\_FORMAT：32bit符号なしモノクロ静止画  SHORT16\_FORMAT：16bit符号つきモノクロ静止画  　LONG32\_FORMAT：32bit符号つきモノクロ静止画  　FLOAT\_FORMAT：32bit浮動小数点モノクロ静止画  RGB\_FORMAT:8bitカラー静止画  　THREE\_D\_FORMAT：8bit符号なしモノクロ動画  　RGB\_3D\_FORMAT：8bitカラー動画  　FLOAT\_3D\_FORMAT：32bit浮動小数点モノクロ動画  　XY\_FORMAT：32bit浮動小数点×2チャネル静止画  　XYZ\_FORMAT：32bit浮動小数点×3チャネル静止画  　COMPLEX\_FORMAT：複素数静止画 |
| CSize size | (in)画像データのサイズ（幅／高さ） |
| long page | (in)画像データのページ数  　　THREE\_D\_FORMAT、RGB\_3D\_FORMATの場合はページ数  　　それ以外は1を指定 |
| bool do\_clear\_memory | 既に同タイプ・サイズのメモリが確保されている場合、現在のデータをクリアするかどうか。falseにすると、現在のメモリを返す |
| 戻り値 | DataUnitクラスへのポインタ。失敗した場合はNULL | |
| 説明 | 画像メモリを初期化する  ・numberで指定された番号にDataUnitのメモリ未確保の場合は、新たにメモリ確保する  ・指定さえたタイプ・サイズと異なるメモリがある場合は、内部データのみメモリ取り直す | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CopyDataUnit(long src\_number, long dest\_number, bool show\_confirm=false)

|  |  |  |
| --- | --- | --- |
| 引数 | long src\_number | (in) コピー元画像メモリ番号 |
| long dest\_number | (in) コピー先画像メモリ番号（コピー先は違う画像メモリ番号を指定しなければならない） |
| bool show\_confirm | (in) コピー先画像メモリに既に別の画像データが入っていた場合、上書き実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリをコピーする。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DeleteDataUnit(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 |  | |
| 説明 | 画像メモリを削除する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool BackupDataUnit (long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリをバックアップ領域に入れる | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool RecoverDataUnit (long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)復元対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | バックアップ領域から画像メモリを復元する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SetWorkArea(RECT rect, bool ornot, long number=-1, long figure\_number=0)

bool SetWorkArea(FIGURE\_LINE line, bool ornot, long number=-1, long figure\_number=0)

bool SetWorkArea(FIGURE\_ELLIPSE ellipse, bool ornot, long number=-1, long figure\_number=0)

bool SetWorkArea(FIGURE\_CIRCLE circle, bool ornot, long number=-1, long figure\_number=0)

|  |  |  |
| --- | --- | --- |
| 引数 | RECT rect | (in)作業領域として設定する矩形 |
| FIGURE\_LINE line | (in)作業領域として設定する幅つき直線 |
| FIGURE\_ELLIPSE ellipse | (in)作業領域として設定する楕円 |
| FIGURE\_CIRCLE circle | (in)作業領域として設定する円周 |
| long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long figure\_numbe | (in) 図形番号 |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 作業領域の図形を設定する。  ・図形の種類に応じて、以下の構造体を使う  矩形  typedef struct tagRECT{  LONG left; // 左上Ｘ座標  LONG top; // 左上Ｙ座標  LONG right; // 右下Ｘ座標  LONG bottom; // 右下Ｙ座標  } RECT;  幅つき直線  typedef struct{  int x1; // 始点Ｘ座標  int y1; // 始点Ｙ座標  int x2; // 終点Ｘ座標  int y2; // 終点Ｙ座標  int w; // 幅  }FIGURE\_LINE;  楕円  typedef struct{  int cx; // 楕円の中心点Ｘ座標  int cy; // 楕円の中心点Ｙ座標  int rx; // 楕円のＸ方向半径  int ry; // 楕円のＸ方向半径  }FIGURE\_ELLIPSE;  円  typedef struct {  int cx; // 円の中心点Ｘ座標  int cy; // 円の中心点Ｙ座標  int r; // 円の半径  int w; // 幅  }FIGURE\_CIRCLE; | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

int GetWorkArea(FIGURE\_UNION \*area, bool \*ornot,long number=-1, long figure\_number=0)

|  |  |  |
| --- | --- | --- |
| 引数 | FIGURE\_UNION \*area | (out)作業領域として取得した図形 |
| long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long figure\_numbe | (in) 図形番号 |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 作業領域の図形を取得する。  ・出力引数には以下の共用体を使う  typedef union{  RECT rect;//矩形作業領域  FIGURE\_LINE line;//線状領域  FIGURE\_ELLIPSE　 ellipse;//楕円状領域  FIGURE\_CIRCLE circle;//円周状領域  FIGURE\_ARC arc;//円弧状領域  FIGURE\_POLYGON 　　　　polygon;//多角形状領域  }FIGURE\_UNION;  ※現状で円弧、多角形には対応していない | |
|  |  | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

long AddMark(long number, CPoint src\_point, double \*p\_attrib=NULL)

long AddMark(long number , CMark src\_mark)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CPoint src\_point | (in) マーク座標 |
| double \*p\_attrib | (in) マークの属性（要素3の配列先頭ポインタを指定する。NULLにすると属性は追加されない） |
| CMark src\_mark | (in) マーク  class AFX\_EXT\_CLASS CMark{  public:  long x;//X座標  long y;//Y座標  double attrib[3];//属性  }; |
| 戻り値 | 成功したら追加されたマーク番号を返す。失敗したら-1を返す | |
| 説明 | マークを１つ追加する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool DeleteMark(long number, long data\_number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long data\_number | (in)削除するマーク番号 |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | マークを１つ削除する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetMarkVal(long number, long data\_number, CPoint \*src\_coor, double \*p\_val, double \*p\_attrib=NULL)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long data\_number | (in)マーク番号 |
| CPoint \*src\_coor | (out)マーク座標（ポインタ） |
| double \*p\_val | (out)マーク座標の画像データの値（ポインタ） |
| double \*p\_attri | (out) マークの属性（要素3の配列先頭ポインタを指定する。NULLにすると属性は取得されない） |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | マークの座標、画像データ値、属性を取得する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

long GetMarkNumber(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 | 成功したらマークの数を返す。失敗したら-1を返す。 | |
| 説明 | マークの数を取得する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool ConvertDataUnit(long number, CONVERT\_DATA\_OPTION option, bool show\_confirm=false)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CONVERT\_DATA\_OPTION option | (in)変換オプション |
| bool show\_confirm | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像データの型を変換する  ・変換後も作業領域とマスクは保持される。  ・変換方法の設定には以下の構造体を使う  typedef struct{  int type\_to;//変換後の画像メモリの型  BYTE\_FORMAT：8bit符号なしモノクロ静止画  　WORD\_FORMAT：16bit符号なしモノクロ静止画  DWORD\_FORMAT：32bit符号なしモノクロ静止画  SHORT16\_FORMAT：16bit符号つきモノクロ静止画  　LONG32\_FORMAT：32bit符号つきモノクロ静止画  　FLOAT\_FORMAT：32bit浮動小数点モノクロ静止画  RGB\_FORMAT:8bitカラー静止画  　THREE\_D\_FORMAT：8bit符号なしモノクロ動画  　RGB\_3D\_FORMAT：8bit符号なしモノクロ動画  　RGB\_3D\_FORMAT：8bitカラー動画  　XY\_FORMAT：32bit浮動小数点×2チャネル静止画  　XYZ\_FORMAT：32bit浮動小数点×3チャネル静止画  　COMPLEX\_FORMAT：複素数静止画  int rgb\_to;//RGB型に変換する場合の入力先（1:R成分 2:G成分 3:B成分 0:RGB全成分）  int comp\_to;//複素数型の挿入先（0:実部　1:虚部）  bool satulate;//型式の上下限を超える値を変換するとき飽和させるかどうか  int byte\_from;//表示メモリ,マスクメモリから変換するかどうか  0:データ 1:表示画像 2:マスク  （type\_toがBYTE\_FORMATのときのみ有効）  }CONVERT\_DATA\_OPTION; | |
|  |  | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool ConvertToByte(long number)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像データの型をBYTEに変換する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool ClipDataUnit(long number, CRect clip\_rect, bool show\_message=false)

bool ClipDataUnit(long number, long dst\_data\_number, bool do\_clear\_dst, CRect clip\_rect,

bool show\_message=false)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_data\_number | (in)格納先画像メモリ番号(-1にした場合は現在選択中の画像) |
| bool do\_clear\_dst | 不使用 |
| CRect clip\_rect | (in)切り取る矩形 |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像データから指定した矩形領域内のみ切り出す  ・マスクも同様に切り出される  ・サイズ変換前後で表示画像のコントラストの条件は保持される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool ResizeDataUnit(long number, CSize new\_size, bool do\_stretch, int offset\_x=0, int offset\_y=0,

bool show\_confirm=false)①

bool ResizeDataUnit(long number, long dst\_data\_number, bool do\_clear\_dst, CSize new\_size,

bool do\_stretch, int offset\_x, int offset\_y, bool show\_message=false)②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_data\_number | (in)格納先画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize new\_size | (in)変換後の画像データのサイズ（幅・高さ） |
| bool do\_stretch | (in)変換後のサイズに合わせて画像を拡大/縮小するか |
| int offset\_x | (in)x方向オフセット量 |
| int offset\_y | (in)y方向オフセット量 |
| bool do\_clear\_dst | 不使用 |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像サイズ変換を実行する  ・変換後画像データにおいて元画像からはみ出した画素は黒で塗りつぶされる  ・マスクサイズも同様に変更される  ・サイズ変換前後で表示画像のコントラストの条件は保持される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool PerspectiveTransDataUnit(long number, double p\_param[8], bool show\_message=false)①

bool PerspectiveTransDataUnit(long number, long dst\_data\_number, bool do\_clear\_dst, double p\_param[8],

bool show\_message=false)②

bool PerspectiveTransDataUnit(long number,

double ti, double tj, double tX,

double tY, double theta, double phai, double psi,

double fl\_phisical, double ccd\_pix\_size, bool show\_message=false)③

bool PerspectiveTransDataUnit(long number,long dst\_data\_number, bool do\_clear\_dst,

double ti, double tj, double tX, double tY,

double theta, double phai, double psi,

double fl\_phisical, double ccd\_pix\_size, bool show\_message=false)④

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_data\_number | (in)格納先画像メモリ番号(-1にした場合は現在選択中の画像) |
| bool do\_clear\_dst | 不使用 |
| double p\_param[8] | (in)射影変換行列 |
| double ti,tj | (in) 仮想カメラの撮像素子ズレ[pix] |
| double tX,tY | (in) 元画像データの撮像面内移動[mm] |
| double theta | (in)仮想カメラのZ軸周り回転角度[deg] |
| double phai | (in)仮想カメラのY軸周り回転角度[deg] |
| double psi | (in)仮想カメラのZ軸周り回転角度[deg] |
| double fl\_phisical | (in)仮想カメラの焦点距離[mm] |
| Double ccd\_pix\_size | (in) 仮想カメラの撮像素子サイズ[μm] |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像サイズ変更を実行する  ・変換後画像データにおいて元画像からはみ出した画素は黒で塗りつぶされる  ・ページ／チャネルを複数持つ画像データ型の場合は、すべてのページ／チャネルに対して幾何変換が行われる  ・マスクサイズも同様に変換される  ・サイズ変換前後で表示画像のコントラストの条件は保持される  【射影変換】  　XYZ座標からなる3次元の仮想空間において、XY面上に元画像データが1pix=1mmサイズ  　で張り付いていると想定し、元画像データを仮想カメラで撮影したときの見え方に変換する。  　仮想カメラの位置は、視野サイズが元画像データの平均高さとなり、光軸上にXYZ原点が  　来るように設定する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool IsoTarnsDataUnit(long number, double ti, double tj, double theta, double magnify,

bool show\_message=false)　①

bool IsoTarnsDataUnit(long number, long dst\_data\_number, bool do\_clear\_dst, double ti, double tj, double theta, double magnify, bool show\_message=false)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_data\_number | (in)格納先画像メモリ番号(-1にした場合は現在選択中の画像) |
| bool do\_clear\_dst | 不使用 |
| double ti | (in) X方向平行移動量 |
| double tj | (in) Y方向平行移動量 |
| double theta | (in) 回転移動[deg] |
| double magnify | (in)拡大率 |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像データの等方変換(回転、平行移動、拡大)を実行する  ・変換後画像データにおいて元画像からはみ出した画素は黒で塗りつぶされる  ・ページ／チャネルを複数持つ画像データ型の場合は、すべてのページ／チャネルに対して幾何変換が行われる  ・マスクサイズも同様に変換される  ・サイズ変換前後で表示画像のコントラストの条件は保持される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CalcDataValue(long number, long type, CSize size, CString com\_str, bool satulate,

String \*pErr\_str=NULL, CProgressCtrl \*pProg=NULL)　①

bool CalcDataValue(long number, CString com\_str, bool satulate,

CString \*pErr\_str=NULL, CProgressCtrl \*pProg=NULL)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long type | (in) 画像メモリのタイプ  BYTE\_FORMAT：8bit符号なしモノクロ静止画  　WORD\_FORMAT：16bit符号なしモノクロ静止画  DWORD\_FORMAT：32bit符号なしモノクロ静止画  SHORT16\_FORMAT：16bit符号つきモノクロ静止画  　LONG32\_FORMAT：32bit符号つきモノクロ静止画  　FLOAT\_FORMAT：32bit浮動小数点モノクロ静止画  RGB\_FORMAT:8bitカラー静止画  　　（RGB\_FORMATを選択すると、RGB各成分に同じ値が入る） |
| CSize size | (in)画像メモリのサイズ |
| CString com\_str | (in)演算式 |
| bool satulate | (in)上下限を超えた場合に飽和させる(falseにするとオーバーフローする) |
| CString \*pErr\_str | (out)エラーメッセージへのポインタ（NULLにするとエラーメッセージを出力しない） |
| CProgressCtrl \*pProg | (in)プログレスバーへのポインタ(プログレスバーを使わない場合はNULL) |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 演算式を用いて画像データを生成する  ・作業領域使用時は、作業領域内部に対してのみ実行する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CalcMaskValue(long number, CString com\_str, double mask\_max, double mask\_min,

CString \*pErr\_str=NULL, CProgressCtrl \*pProg=NULL)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize size | (in)画像メモリのサイズ |
| CString com\_str | (in)演算式 |
| double mask\_max | (in)マスクOFFとする演算結果の上限 |
| double mask\_min | (in)マスクOFFとする演算結果の下限 |
| CString \*pErr\_str | (out)エラーメッセージへのポインタ（NULLにするとエラーメッセージを出力しない） |
| CProgressCtrl \*pProg | (in)プログレスバーへのポインタ(プログレスバーを使わない場合はNULL) |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 演算式を用いてマスクを生成する  ・作業領域使用時は、作業領域内部に対してのみ実行する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterSmooth(long number, CSize filter\_size, bool rem\_min\_max)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| bool rem\_min\_max | (in)平滑化時に最大／最小値を除くかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリに平滑化フィルタをかける | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterMedian(long number, CSize filter\_size)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリにメディアンフィルタをかける | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterErosion (long number, CSize filter\_size)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリに収縮フィルタをかける | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterDilation (long number, CSize filter\_size)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリに膨張フィルタをかける | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterEdgeSobel(long number, CSize filter\_size, double min\_thresh)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| double min\_thresh | (in)抽出するエッジ強度最低値 |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリにエッジ抽出をおこなう  ・アルゴリズムはSobel | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FilterThinLog(long number, CSize filter\_size, double min\_thresh)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| CSize filter\_size | (in)フィルタサイズ |
| double min\_thresh | (in)抽出するエッジ強度最低値 |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリにエッジ細線化をおこなう  ・アルゴリズムはラプラシアンによるゼロ交差判別 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool Binalize(long src\_number, long dst\_number, double min\_thresh, double max\_thresh, bool inverse,

BYTE on\_label=255, BYTE off\_label=0, bool show\_message=false)　①

bool Binalize(long src\_number, double min\_thresh, double max\_thresh, bool inverse)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long src\_number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_number | (in)格納先画像メモリ番号  (src\_numberとは異なる番号にしなければならない) |
| double min\_thresh | (in)最小閾値 |
| double max\_thresh | (in)最大閾値 |
| bool inverse | (in)２値反転 |
| BYTE on\_label | (in)2値化後の白画素値 |
| BYTE off\_label | (in)2値化後の黒画素値 |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリを2値化する。  ・結果はBYTE型として格納される  ・作業領域使用フラグON状態では、作業領域内部のみ2値化する  ・結果格納先が画像メモリで、マスク使用フラグON状態のときは、マスクOFFの画素のみ2値化する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

double BinalizeThresholdOtsu(long src\_number)

|  |  |  |
| --- | --- | --- |
| 引数 | long src\_number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| 戻り値 | ２値化の最小閾値 | |
| 説明 | 大津の手法で２値化閾値を求める | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool BinalizeNick(long src\_number, long dst\_number, int windowSize, float k, bool inverse,

BYTE fg\_val = 255, BYTE bg\_val = 0)

|  |  |  |
| --- | --- | --- |
| 引数 | long src\_number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_number | (in)2値化後の画像を入れるメモリ |
| int windowSize | (in)ウィンドウサイズ |
| float k | (in)パラメータk |
| bool inverse | (in)２値反転 |
| BYTE fg\_val | (in)2値化後の白画素値 |
| BYTE bg\_val | (in)2値化後の黒画素値 |
|  |  |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | NICKの手法で画像メモリを2値化する。  ・結果はBYTE型として格納される  ・作業領域使用フラグON状態では、作業領域内部のみ2値化する  ・結果格納先が画像メモリで、マスク使用フラグON状態のときは、マスクOFFの画素のみ2値化する  NICKの手法：  以下の式でウィンドウごとにの2値化閾値Tを決める  T = ave + k \* sqrt{ (∑I^2 + ave^2)/np }  　　　I: pixel value  　　　ave: averaged pixel value within the window  　　　np: number of pixels within the window  Khurshid, K., Siddiqi, I., Faure, C., Vincent, N.: Comparison of Niblack inspired Binarization methods for ancient documents. In: 16th International conference on Document Recognition and Retrieval. SPIE, USA 2010 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

int Label(long src\_number, long dst\_number, double min\_thresh, double max\_thresh, bool inverse,

bool sort\_label, int min\_area, bool show\_message=false

|  |  |  |
| --- | --- | --- |
| 引数 | long src\_number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| long dst\_number | (in)格納先画像メモリ番号  (src\_numberとは異なる番号にしなければならない) |
| double min\_thresh | (in)最小閾値 |
| double max\_thresh | (in)最大閾値 |
| bool inverse | (in)２値反転 |
| bool sort\_label | (in)ラベルを面積でソートするかどうか |
| bool show\_message | (in) 実行前に確認メッセージボックスを出すかどうか |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 画像メモリを2値化してからラベリングをする。  ・結果はWORD型となり、ラベル番号が格納される。  ・作業領域使用フラグON状態では、作業領域内部のみ2値化する  ・結果格納先が画像メモリで、マスク使用フラグON状態のときは、マスクOFFの画素のみ2値化する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

int HoughLines(long image\_num, int pixSkip, int dirSkip, int voteThresh,

double minLineLength, double maxLineGap )

|  |  |  |
| --- | --- | --- |
| 引数 | long image\_num, | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int pixSkip | (in)投票空間の距離分解能[pix] |
| int dirSkip | (in)投票空間の角度分解能[deg] |
| int voteThresh | (in)投票の閾値パラメータ．十分な票を得た直線のみが出力される |
| double minLineLength | (in)最小の線分長．これより短い線分は棄却される |
| double maxLineGap | (in)2点が同一線分上にあると見なす場合に許容される最大距離． |
| 戻り値 | 抽出した直線数 | |
| 説明 | エッジ画像に対し、確率的Hough変換を行う  https://docs.opencv.org/3.4/dd/d1a/group\_\_imgproc\_\_feature.html#ga8618180a5948286384e3b7ca02f6feeb | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool　BlendPoisson(long image\_num, long blend\_data\_number, long dst\_data\_number, int offsetx, int offsety,

int mode=2)

|  |  |  |
| --- | --- | --- |
| 引数 | long image\_num, | (in)背景画像メモリ番号(-1にした場合は現在選択中の画像) |
| long blend\_data\_number | (in)合成対象画像メモリ番号 |
| long dst\_data\_number | (in)結果格納先画像メモリ番号(-1にした場合は現在選択中の画像) |
| int offsetx | (in)合成開始位置x |
| int offsety | (in)合成開始位置y |
| int mode | (in)合成方法 0:NORMAL\_CLONE, 1:MIXED\_CLONE 2:MONOCHROME\_TRANSFER |
| 戻り値 | 成功したらtrueを返す。失敗したらfalseを返す | |
| 説明 | 濃度勾配を配慮した継ぎ目の少ない画像合成  Patrick P?rez, Michel Gangnet, and Andrew Blake. Poisson image editing. In ACM Transactions on Graphics (TOG), volume 22, pages 313-318. ACM, 2003. | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawLineOnImage(long number, double x0, double y0, double x1, double y1, int pen\_size, COLORREF col, int copy\_image\_num)　①

void DrawLineOnImage(long number, double x0, double y0, double x1, double y1, int pen\_size, double val)②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x0,y0 | (in)直線描画始点座標 |
| int x1,y1 | (in)直線描画終点座標 |
| int pen\_size, | (in)直線太さ |
| COLORREF col | (in)カラーで描画する場合の直線の色 |
| double val | (in) モノクロで描画する場合の直線の濃度 |
| int copy\_image\_num | (in) カラーで描画する場合のコピー元画像メモリ番号  （copy\_image\_numの画像データをnumberの画像メモリにコピーして、RGB型に変換してから、その上にカラー直線を描画する） |
| 戻り値 |  | |
| 説明 | 画像データ上に直線を描画する。  ・モノクロ画像上に描画する場合は、以下の2つの方法が使える   1. RGB型に変換してからカラーで描画する（①）   （copy\_image\_numに変換元画像メモリ番号を設定する。対象画像がカラーの場合に-1を設定する  　と対象画像番号として扱われる）   1. 元のデータ型で描画する（②） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawCircleOnImage(long number, double cx, double cy, double rad, COLORREF col, bool fill,

int copy\_image\_num)　①

void DrawCircleOnImage(long number, double cx, double cy, double rad, double val, bool fill)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int cx,cy | (in)円中心座標 |
| double rad | (in)円半径 |
| bool fill | (in)円の中を塗りつぶすかどうか |
| COLORREF col | (in)カラーで描画する場合の円の色 |
| double val | (in) モノクロで描画する場合の円の濃度 |
| int copy\_image\_num | (in) カラーで描画する場合のコピー元画像メモリ番号  （copy\_image\_numの画像データをnumberの画像メモリにコピーして、RGB型に変換してから、その上にカラー直線を描画する。） |
| 戻り値 |  | |
| 説明 | 画像データ上に円を描画する。  ・モノクロ画像上に描画する場合は、以下の2つの方法が使える   1. RGB型に変換してからカラーで描画する（①）   （copy\_image\_numに変換元画像メモリ番号を設定する。対象画像がカラーの場合に-1を設定する  　と対象画像番号として扱われる）  　（２）元の型のままで描画する（②） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawRectOnImage(long number, double x0, double y0, double x1, double y1, COLORREF col, bool fill=false, COLORREF fill\_col=0, int copy\_image\_num=-1)　①

void DrawRectOnImage(long number, double x0, double y0, double x1, double y1, double val, bool fill=false, double fill\_val=0)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x0,y0 | (in)矩形左上座標 |
| Int x1,y1 | (in)矩形右下座標 |
| double rad | (in)円半径 |
| COLORREF col | (in)カラーで描画する場合の矩形の色 |
| double val | (in)モノクロで描画する場合の矩形の濃度 |
| bool fill | (in)矩形の中を塗りつぶすかどうか |
| COLORREF fill\_col | (in)カラーで描画する場合の塗りつぶし色 |
| double fill\_val | (in)モノクロで描画する場合の塗りつぶし濃度 |
| int copy\_image\_num | (in) カラーで描画する場合のコピー元画像メモリ番号  （copy\_image\_numの画像データをnumberの画像メモリにコピーして、RGB型に変換してから、その上にカラー直線を描画する。） |
| 戻り値 |  | |
| 説明 | 画像データ上に矩形を描画する。  ・モノクロ画像上に描画する場合は、以下の2つの方法が使える   1. RGB型に変換してからカラーで描画する（①）   （copy\_image\_numに変換元画像メモリ番号を設定する。対象画像がカラーの場合に-1を設定する  　と対象画像番号として扱われる）  　（２）元の型のままで描画する（②） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawTextOnImage(long number, int x, int y, char str[], COLORREF col, int src\_image\_num=-1)　①

void DrawTextOnImage(long number, int x, int y, char str[], double val)　②

void DrawTextOnImage(long number, int x, int y, COLORREF col, char \*format, [, argument]...)　③

void DrawTextOnImage(long number, int x, int y, double val, char \*format, [, argument]...)　④

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x,y | (in)テキストの左上座標 |
| char str[] | (in)描画するASCIIテキスト |
| char \*format | (in) 描画するASCIIテキストのフォーマット |
| COLORREF col | (in)カラーで描画する場合のテキストの色 |
| double val | (in) モノクロで描画する場合のテキストの濃度 |
| int copy\_image\_num | (in) カラーで描画する場合のコピー元画像メモリ番号  （copy\_image\_numの画像データをnumberの画像メモリにコピーして、RGB型に変換してから、その上にカラー直線を描画する。） |
| 戻り値 |  | |
| 説明 | 画像データ上にテキストを描画する。  ・描画できるのはASCIIテキストのみである。  ・モノクロ画像上に描画する場合は、以下の2つの方法が使える   1. RGB型に変換してからカラーで描画する（①③）   （copy\_image\_numに変換元画像メモリ番号を設定する。対象画像がカラーの場合に-1を設定する  　と対象画像番号として扱われる）  　（２）元の型のままで描画する（②④） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawPointOnImage(long number, int x, int y, int dot\_size, COLORREF col, int copy\_image\_num)①

void DrawPointOnImage(long number, int x, int y, int dot\_size, double val)②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x,y | (in)ドット中心位置 |
| int dot\_size | (in)ドットの直径 |
| COLORREF col | (in)カラーで描画する場合の色 |
| double val | (in) モノクロで描画する場合の濃度 |
| int copy\_image\_num | (in) カラーで描画する場合のコピー元画像メモリ番号  （copy\_image\_numの画像データをnumberの画像メモリにコピーして、RGB型に変換してから、その上にカラー直線を描画する。） |
| 戻り値 |  | |
| 説明 | 画像データ上にドットを描画する。  ・モノクロ画像上に描画する場合は、以下の2つの方法が使える   1. RGB型に変換してからカラーで描画する（①）   （copy\_image\_numに変換元画像メモリ番号を設定する。対象画像がカラーの場合に-1を設定する  　と対象画像番号として扱われる）  　（２）元の型のままで描画する（②） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawPointOnMask(long number, int x, int y, int dot\_size, BYTE val)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x,y | (in)ドット中心位置 |
| int dot\_size | (in)ドットの直径 |
| BYTE val | (in)マスク値（1:マスクON 0:マスクOFF） |
| 戻り値 |  | |
| 説明 | マスク上にドットを描画する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void FloodFillOnImage(long number, int x, int y, COLORREF col)　 ①

void FloodFillOnImage(long number, int x, int y, double val) 　 ②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x,y | (in)塗りつぶし開始座標 |
| COLORREF col | (in)カラー画像を塗りつぶす場合の色 |
| double val | (in) モノクロ画像を塗りつぶす場合の濃度 |
| 戻り値 |  | |
| 説明 | 画像データ上を塗りつぶす。開始座標から同色が続いている領域をぬりつぶす  ・①はカラー画像データ用、②はモノクロ画像データ用である | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void FloodFillOnMask(long number, int x, int y, BYTE val)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int x,y | (in)塗りつぶし開始座標 |
| BYTE val | (in)マスク値（1:マスクON 0:マスクOFF） |
| 戻り値 |  | |
| 説明 | マスク上を塗りつぶす。開始座標から同値が続いている領域をぬりつぶす | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void FillWorkAreaOnImage(long number, COLORREF col)　①

void FillWorkAreaOnImage(long number, double val)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| COLORREF col | (in)カラー画像を塗りつぶす場合の色 |
| double val | (in) モノクロ画像を塗りつぶす場合の濃度 |
| 戻り値 |  | |
| 説明 | 作業領域内部の画像データをすべて塗りつぶす  ・①はカラー画像データ用、②はモノクロ画像データ用である | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void FillWorkAreaOnMask(long number, BYTE val)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| BYTE val | (in)マスク値（1:マスクON 0:マスクOFF） |
| 戻り値 |  | |
| 説明 | 作業領域内部のマスクをすべて塗りつぶす | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawImage(long number = CURRENT\_IMAGE, bool do\_update = true, bool do\_adjust = false)　①

void DrawImage(long number, CString name, bool do\_update, bool do\_adjust)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| bool do\_update | (in)表示画像を作り直すかどうか |
| bool do\_adjust | (in)サイズをメインウィンドウの表示サイズにあわせるかどうか |
| CString name | (in)変更する画像データ名 |
| 戻り値 |  | |
| 説明 | 画像表示する  ・メインウィンドウのサイズに合わせて、拡大率を変更することもできる  ・画像データ名を変更することもできる（②） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetDispRange(long number = CURRENT\_IMAGE, float min=0 , float max=256)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| float min | (in) 画像表示の濃度上限 |
| float max | (in) 画像表示の濃度下限 |
| 戻り値 |  | |
| 説明 | 画像表示の濃度上限／下限を設定したのち、画像表示する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetDispRangeAuto(long number = CURRENT\_IMAGE, int type=0)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)対象画像メモリ番号(-1にした場合は現在選択中の画像) |
| int type | (in) 自動設定タイプ　（0:最大/最小 1:平均値±標準偏差×2） |
| 戻り値 |  | |
| 説明 | 画像表示の濃度上限／下限を自動で設定したのち、画像表示する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawLine(double x0, double y0, double x1, double y1, int size, COLORREF col, int width=1)　①

void DrawLine(double x0, double y0, double x1, double y1, int width=1)　②

|  |  |  |
| --- | --- | --- |
| 引数 | double x0,y0 | (in)直線描画始点座標 |
| double x1,y1 | (in)直線描画終点座標 |
| int pen\_size, | (in)線太さ |
| COLORREF col | (in)線の色 |
| int width | (in)線の幅 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上に直線を描画する  ・この関数で描画された図形は画像表示すると消える。  ・widthに1より大きい値を指定すると、幅つき直線領域になる  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawCursor(double x, double y, int pen\_size, COLORREF col)　①

void DrawCursor(double x, double y)　②

|  |  |  |
| --- | --- | --- |
| 引数 | double x,y | (in)十字交点の座標 |
| int pen\_size, | (in)線の太さ |
| COLORREF col | (in)色 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上に十字カーソルを描画する  ・この関数で描画された図形は画像表示すると消える。  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawRect(double left, double top, double right, double bottom, int pen\_size, COLORREF col)　①

void DrawRect(double left, double top, double right, double bottom)　②

|  |  |  |
| --- | --- | --- |
| 引数 | double left | (in)左辺のx座標 |
| double top | (in)上辺のy座標 |
| double right | (in)右辺のx座標 |
| double bottom | (in)下辺のy座標 |
| int pen\_size | (in)線の太さ |
| COLORREF col | (in)線の色 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上に矩形を描画する  ・この関数で描画された図形は画像表示すると消える。  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawCircle(double x, double y, double r, int pen\_size, COLORREF col, int width=1)　①

void DrawCircle(double x, double y, double r, int width=1)　②

|  |  |  |
| --- | --- | --- |
| 引数 | double x,y | (in)円中心座標 |
| double r | (in)円半径 |
| int pen\_size | (in)線の太さ |
| COLORREF col | (in)線の色 |
| int width | (in)線の幅 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上に円を描画する。  ・この関数で描画された図形は画像表示すると消える。  ・widthに1より大きい値を指定すると、幅つき円弧領域になる  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawEllipse(double x, double y, double rx, double ry, int pen\_size, COLORREF col, int width=1)　①

void DrawEllipse(double x, double y, double rx, double ry)　②

|  |  |  |
| --- | --- | --- |
| 引数 | double x,y | (in)楕円中心座標 |
| double rx | (in)楕円のx方向半径 |
| double ry | (in)楕円のy方向半径 |
| int pen\_size | (in)線の太さ |
| COLORREF col | (in)線の色 |
| int width | (in)線の幅 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上に円を描画する。  ・この関数で描画された図形は画像表示すると消える。  ・widthに1より大きい値を指定すると、幅つき楕円弧領域になる  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void DrawText(double x, double y, CString str, COLORREF col, int size=-1)　①

void DrawText(double x, double y, CString str)　②

void DrawText(double x, double y, COLORREF col, int size, char \*format, [, argument]...)　③

|  |  |  |
| --- | --- | --- |
| 引数 | double x,y | (in)テキスト描画座標 |
| CString str | (in)テキスト |
| char \*format | (in)テキストのフォーマット |
| int size | (in)文字サイズ |
| COLORREF col | (in)線の色 |
| int width | (in)線の幅 |
| 戻り値 |  | |
| 説明 | メインウィンドウの表示画像上にテキストを描画する。  ・この関数で描画されたテキストは画像表示すると消える。  ・サイズと色を指定しない場合（②）は、サイズ1、色は緑で描画される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void ShowMessage(bool do\_clear, int num, bool force, CString str)　①

void ShowMessage(bool do\_clear, int num, bool force, char \*format, [, argument]...)　②

void ShowMessage(bool do\_clear, char \*format, [, argument]...)　③

|  |  |  |
| --- | --- | --- |
| 引数 | bool do\_clear | (in)過去のメッセージをクリアしてから今回のメッセージを表示するかどうか |
| int num | (in)メインウィンドウのメッセージ領域の番号（0 or 1） |
| bool force | (in)メッセージ表示番号を切りかえて表示するかどうか |
| CString str | (in)メッセージテキスト |
| char \*format | (in)テキストのフォーマット |
| 戻り値 |  | |
| 説明 | メインウィンドウのメッセージ領域にテキストを表示する。  ・メッセージ表示番号を指定しない場合（③）は、メッセージ領域1に表示される。メッセージ領域0が表示中の場合は切りかえて表示する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool DialogOpen(int id, bool show)

|  |  |  |
| --- | --- | --- |
| 引数 | int id | (in)ウィンドウのリソースID  　IDD\_DATA\_DLG：データウィンドウ  　IDD\_HISTGRAM\_DLG：ヒストグラムウィンドウ  　IDD\_MARK\_DLG：マークウィンドウ  　IDD\_WORKAREA\_DLG：作業領域ウィンドウ  　IDD\_CALC\_DLG：任意数式ウィンドウ  　IDD\_CHART\_DLG：チャートウィンドウ  　IDD\_BATCH\_DLG：バッチ実行ウィンドウ  　IDD\_BINALIZE\_DLG：2値化ウィンドウ |
| bool show | (in)true:可視状態にする　false:不可視状態にする |
| 戻り値 | 成功するとtrueが返る | |
| 説明 | ビルトインされているウィンドウ（モードレスダイアログとして実装されているもの）の状態を変更する。各ウィンドウの詳細は「PIMPOMソフトウェア仕様書」を参照。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool UserDialogOpen(int n)

|  |  |  |
| --- | --- | --- |
| 引数 | int n | (in)カスタムファンクション番号 |
| 戻り値 | 成功するとtrueが返る | |
| 説明 | カスタムファンクションを起動する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetMarkEditMode()

|  |  |
| --- | --- |
| 戻り値 | マーク編集のOn/Off状態 |
| 説明 | マーク編集のOn/Off状態（下記ボタンの押下状態）を取得する  　｝button_mark_edit |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetMarkEditMode(int mode)

|  |  |  |
| --- | --- | --- |
| 引数 | int mode | (in)マーク編集のOn/Off状態 |
| 説明 | マーク編集のOn/Off状態（下記ボタンの押下状態）を設定する。  button_mark_edit | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetWorkAreaEditMode ()

|  |  |
| --- | --- |
| 戻り値 | 作業領域編集のOn/Off状態 |
| 説明 | 作業領域編集のOn/Off状態（下記ボタンの押下状態）を取得する  button_work_area_edit |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetWorkAreaEditMode (int mode)

|  |  |  |
| --- | --- | --- |
| 引数 | int mode | (in) 作業領域編集のOn/Off状態 |
| 説明 | 作業領域編集のOn/Off状態（下記ボタンの押下状態）を設定する。  button_work_area_edit | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetRulerToolMode ()

|  |  |
| --- | --- |
| 戻り値 | ルーラーのOn/Off状態 |
| 説明 | ルーラーのOn/Off状態（下記ボタンの押下状態）を取得する  button_calc_distance |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetRulerToolMode (int mode)

|  |  |  |
| --- | --- | --- |
| 引数 | int mode | (in) ルーラーのOn/Off状態 |
| 説明 | ルーラーのOn/Off状態（下記ボタンの押下状態）を設定する。  button_calc_distance | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetVideoCaptureThroughMode()

|  |  |
| --- | --- |
| 戻り値 | ビデオキャプチャのライブ表示On/Off状態 |
| 説明 | ビデオキャプチャのライブ表示On/Off状態（下記ボタンの押下状態）を取得する  button_video |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void SetVideoCaptureThroughMode(int mode)

|  |  |  |
| --- | --- | --- |
| 引数 | int mode | (in) ビデオキャプチャのライブ表示On/Off状態 |
| 説明 | ビデオキャプチャのライブ表示On/Off状態（下記ボタンの押下状態）を設定する。  button_video | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CopyToClipbord ()

|  |  |
| --- | --- |
| 戻り値 | 成功したらtrueを返す |
| 説明 | 表示画像をクリップボードにコピーする |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetEditValue(HWND h\_wnd, UINT id , int \*value)　①

bool GetEditValue(HWND h\_wnd, UINT id , short \*value)　②

bool GetEditValue(HWND h\_wnd, UINT id , float \*value)　③

bool GetEditValue(HWND h\_wnd, UINT id , double \*value)　④

bool GetEditValue(HWND h\_wnd, UINT id , CString \*str) ⑤

|  |  |  |
| --- | --- | --- |
| 引数 | HWND h\_wnd | (in)ウィンドウハンドル |
| UINT id | (in)エディットボックスのリソースID |
| int \*value  short \*value  float \*value  double \*value  CString \*str | (out)エディットボックスから取り出した値 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | エディットボックスから値を取得する。  ・カスタムファンクションの場合、ウィンドウハンドルには必ず「m\_hWnd」を指定する。  ・エディットボックスに少数点以下含む実数が記載されていた場合、①②で取り出した値は小数点以下を切りすてた値となる。  ・エディットボックスに数字以外の文字が含まれていた場合、①②③④では取り出せない（戻り値としてfalseが返る） | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SetEditValue(HWND h\_wnd, UINT id , double value)　①

bool SetEditValue(HWND h\_wnd, UINT id , CString str)　②

|  |  |  |
| --- | --- | --- |
| 引数 | HWND h\_wnd | (in)ウィンドウハンドル |
| UINT id | (in)エディットボックスのリソースID |
| double value | (in)エディットボックスに設定する数値 |
| CString str | (in)エディットボックスに設定する文字列 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | エディットボックスに値を設定する。  ・カスタムファンクションの場合、ウィンドウハンドルには必ず「m\_hWnd」を指定する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool GetCheck(HWND h\_wnd, UINT id )

|  |  |  |
| --- | --- | --- |
| 引数 | HWND h\_wnd | (in)ウィンドウハンドル |
| UINT id | (in)チェックボックスのリソースID |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | チェックボックスのチェック状態を取得する  ・カスタムファンクションの場合、ウィンドウハンドルには必ず「m\_hWnd」を指定する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SetCheck(HWND h\_wnd, UINT id ,bool value)

|  |  |  |
| --- | --- | --- |
| 引数 | HWND h\_wnd | (in)ウィンドウハンドル |
| UINT id | (in)チェックボックスのリソースID |
| bool value | (in)チェック状態 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | チェックボックスのチェック状態を設定する  ・カスタムファンクションの場合、ウィンドウハンドルには必ず「m\_hWnd」を指定する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void MessageBox(char \*format, [, argument]...)

|  |  |  |
| --- | --- | --- |
| 引数 | char \*format | (in)テキストのフォーマット |
| 戻り値 |  | |
| 説明 | メッセージボックスを表示する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SelectAndLoadImageFile(long number) ①

bool SelectAndLoadImageFile(long number, CString filename, short width, short height, short page)②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)ファイルを読み込む画像メモリ番号 |
| CString filename | (in)画像ファイル名 |
| short width | 不使用 |
| short height | 不使用 |
| short page | 不使用 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | ファイルから画像を読み込む。  ・読み込めるファイル形式は以下の通り   |  |  |  |  | | --- | --- | --- | --- | | ファイルの種類 | 拡張子 | 画像データの型 | 備考 | | Windows Bitmap | \*.bmp | 8bit →　BYTE  24bit/32bit → RGB |  | | JPEG | \*.jpg | RGB |  | |  |  |  |  | | 8bot符号なし整数 | \*.raw | BYTE |  | | 16bit符号なし整数 | \*.word | WORD |  | | 16bit符号つき整数 | \*.short | SHORT |  | | 32bit符号なし整数 | \*.dword | DWORD |  | | 32bit符号つき整数 | \*.long | LONG |  | | 32bit浮動小数点 | \*.float | FLOAT |  | | 2次元分布 | \*.xy | XY |  | | 3次元分布 | \*.xyz | XYZ |  | | 複素数 | \*.comp | COMPLEX |  | | 8bit動画 | \*.3d | 3D |  | | 32bit浮動小数動画 | \*.f3d | FLOAT3D |  | | PIMPOMアーカイブ | \*.pim | すべて | このファイル形式は画像メモリの全データを持っており、保存時の状態が再現される |   ・ファイル名を指定しない場合（①）はファイルを指定するためにファイルダイアログを開く。  　ファイルダイアログでは複数ファイルを指定することができる。複数ファイルを指定した場合は、  　numberで示す画像メモリから次の番号の画像メモリに順次展開される | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SelectAndSaveImageFile (long number)　①

bool SelectAndSaveImageFile(long number, CString filename)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)ファイルとして保存する画像メモリ番号 |
| CString filename | (in)画像ファイル名 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 画像データをファイルとして保存する。  ・保存できるファイル形式は以下の通り。   |  |  |  | | --- | --- | --- | | 画像データの型 | ファイル形式 | 拡張子 | | BYTE | 8bot符号なし整数 | \*.raw | | WORD | 16bit符号なし整数 | \*.word | | SHORT | 16bit符号つき整数 | \*.short | | DWORD | 32bit符号なし整数 | \*.dword | | LONG | 32bit符号つき整数 | \*.long | | FLOAT | 32bit浮動小数点 | \*.float | | XY | 2次元分布 | \*.xy | | XYZ | 3次元分布 | \*.xyz | | COMPLEX | 複素数 | \*.comp | | 3D | 8bit動画 | \*.3d | | FLOAT3D | 32bit浮動小数動画 | \*.f3d |   ・ファイル名を指定しない場合（①）はファイルを指定するためにファイルダイアログを開く。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SaveViewAsBitmap(long number)　①

bool SaveViewAsBitmap(long number, CString filename)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)表示画像を保存する画像メモリ番号 |
| CString filename | (in)画像ファイル名 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 表示画像をビットマップファイルとして保存する  ・画像データがRGB/RGB3Dで、RGBカラー表示ONの場合は24bitカラービットマップとして保存する。  ・ファイル名を指定しない場合（①）はファイルを指定するためにファイルダイアログを開く。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool SaveArchivedImage (long number)　①

bool SaveArchivedImage (long number, CString filename)　②

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)表示画像を保存する画像メモリ番号 |
| CString filename | (in)画像ファイル名 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 画像メモリの内容をPIMPOMアーカイブファイル（\*.pim）として保存する  ・ファイル名を指定しない場合（①）はファイルを指定するためにファイルダイアログを開く。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool OpenFileDialog(int file\_number\_max, CString file\_kind\_ext, CString file\_kind\_disp,

　　　　CString \*pfile\_path, CString pfile\_names[], int \*pfile\_number);

|  |  |  |
| --- | --- | --- |
| 引数 | int file\_number\_max | (in)選択するファイルの最大数 |
| CString file\_kind\_ext | (in)ファイルの拡張子 |
| CString file\_kind\_disp | (in)ファイルの種類説明 |
| CString \*pfile\_path | (out)選択されたファイルのあるフォルダのパス |
| CString pfile\_names[] | (out)選択された全ファイル名の配列 |
| int \*pfile\_number | (out)選択されたファイル数 |
| 戻り値 | OKボタンを押されて終了したらtrueを返す | |
| 説明 | ファイルダイアログを開いて複数のファイルを選択する。  ・ファイルの拡張子は1種類のみ可能  **使用例**  **＜プログラム＞**  CString filepath;//ファイルのあるフォルダのパス  CString filename[10];//ファイル名  int fileNum=0;//選択したファイル数  API.OpenFileDialog(10,"txt","テキストデータ", &filepath, filename, &fileNum);  **<実行時の様子>** | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool FolderDialog(CString \*pfolder)

|  |  |  |
| --- | --- | --- |
| 引数 | CString \*pfolder | (out)選択されたフォルダパス |
| 戻り値 | OKボタンを押されて終了したらtrueを返す | |
| 説明 | フォルダダイアログを開いてフォルダを選択する。 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

static CString GetDefaultPath()

|  |  |  |
| --- | --- | --- |
| 引数 |  |  |
| 戻り値 | Pimpom.exeのフォルダパス | |
| 説明 | Pimpom.exeのフォルダパスを取得する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool AddDataToChart(int chart\_id, float value)

|  |  |  |
| --- | --- | --- |
| 引数 | int chart\_id | (in)グラフ番号 |
| float value | (in)1次元分布グラフに追加するデータ値 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 1次元グラフにデータを１点追加する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool PlotDataToChart(int chart\_id, int data\_num, float \*pValueArr, bool doClear=true)①

bool PlotDataToChart(int chart\_id, int data\_num, double \*pValueArr, bool doClear=true)②

bool PlotDataToChart(int chart\_id, int data\_num, int \*pValueArr, bool doClear=true)③

bool PlotDataToChart(int chart\_id, int data\_num, BYTE \*pValueArr, bool doClear=true)④

|  |  |  |
| --- | --- | --- |
| 引数 | int chart\_id | (in)グラフ番号 |
| float \*pValueArr  double \*pValueArr  int \*pValueArr  BYTE \*pValueArr | (in)1次元分布グラフに追加するデータ配列の先頭ポインタ |
| bool doClear | (in)既に入っているデータをすべて消してから追加するかどうか |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 1次元グラフグラフにデータ配列を追加する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool AddDataToChart2(int chart\_id, float valueX, float valueY)

|  |  |  |
| --- | --- | --- |
| 引数 | int chart\_id | (in)グラフ番号 |
| float valueX, valueY | (in)2次元分布グラフに追加するデータ値 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 2次元グラフにデータを１点追加する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool PlotDataToChart2(int chart\_id, int data\_num, float \*pValueArrX, float \*pValueArrY,

bool doClear=true)①

bool PlotDataToChart2(int chart\_id, int data\_num, double \*pValueArrX, double \*pValueArrY,

bool doClear=true)②

bool PlotDataToChart2(int chart\_id, int data\_num, int \*pValueArrX, int \*pValueArrY,

bool doClear=true)③

bool PlotDataToChart2(int chart\_id, int data\_num, BYTE \*pValueArrX, BYTE \*pValueArrY,

bool doClear=true)④

|  |  |  |
| --- | --- | --- |
| 引数 | int chart\_id | (in)グラフ番号 |
| float \*pValueArrX, \*pValueArrY  double \*pValueArrX, \*pValueArrY  int \*pValueArrX, \*pValueArrY  BYTE \*pValueArrX, \*pValueArrY | (in)2次元分布グラフに追加するデータ配列の先頭ポインタ |
| bool doClear | (in)既に入っているデータをすべて消してから追加するかどうか |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 2次元グラフグラフにデータ配列を追加する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

void ClearChart(int chart\_id)

|  |  |  |
| --- | --- | --- |
| 引数 | int chart\_id | (in)グラフ番号 |
| 戻り値 |  | |
| 説明 | グラフのデータをすべてクリアする | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool Img2Mat(long image\_num, void \*pSrcMat, void \*pClippedMat)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| void \*pSrcMat | (out)cv::MATのアドレス |
| Void \*pClippedMat | (out)計測画像でクリップしたcv::MATのアドレス |
| 戻り値 | 変換に失敗した場合はfalse。 | |
| 説明 | 画像データをOpenCVのcv::Mat型に変換する  ・変換できるフォーマットの関係は以下のとおり   |  |  | | --- | --- | | 画像データの型 | Cv::Matの型 | | BYTE | CV\_8UC1 | | WORD | CV\_16UC1 | | SHORT | CV\_16SC1 | | DWORD | 変換不可 | | LONG | 変換不可 | | FLOAT | CV\_32FC1 | | RGB | CV\_8UC3 | | XY | CV\_32FC2 | | XYZ | CV\_32FC3 | | COMPLEX | CV\_32FC4 | | 3D | CV\_8UC1（最初のページを変換） | | FLOAT3D | CV\_32FC1（最初のページを変換） | | RGB3D | CV\_8UC3（最初のページを変換） | | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool Mat2Img(long image\_num, void \*pMat, long src\_image\_num=-1)

|  |  |  |
| --- | --- | --- |
| 引数 | long number | (in)画像メモリ番号 |
| void \*pMat | (in)cv::MATのアドレス |
| long src\_image\_num |  |
| 戻り値 | 変換に失敗した場合はfalse。 | |
| 説明 | OpenCVのcv::Mat型から画像データに変換する  ・変換できるフォーマットの関係は以下のとおり   |  |  | | --- | --- | | Cv::Matの型 | 画像データの型 | | CV\_8UC1 | BYTE | | CV\_8UC3 | RGB | | CV\_16UC1 | WORD | | CV\_16SC1 | SHORT | | CV\_32FC1 | FLOAT | | CV\_32FC2 | XY | | CV\_32FC3 | XYZ | | CV\_32FC4 | COMPLEX | | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CaptureInit(CSize size)

|  |  |  |
| --- | --- | --- |
| 引数 | CSize size | (in)撮像画像のサイズ |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | ビデオキャプチャを有効にする | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

bool CaptureOne(bool doDrawImage=true)

|  |  |  |
| --- | --- | --- |
| 引数 | bool doDrawImage | (in)撮像後に画像表示するかどうか |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | ビデオキャプチャを１回実行 | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

int SendString(CString str)

|  |  |  |
| --- | --- | --- |
| 引数 | CString str | (in)送信文字列 |
| 戻り値 | 送信されたデータ数[byte] | |
| 説明 | TCP/IP通信でテキストを送信する | |

[関数一覧へ](#CPimpomAPIメンバ関数一覧)

int GetSocketStatus(CString\* pHostName, UINT\* pPortNo)

|  |  |  |
| --- | --- | --- |
| 引数 | CString str | (out)開かれている場合はホスト名またはIPアドレスが入る |
| UINT\* pPortNo | (out)開かれている場合はポート番号が入る |
| 戻り値 | -1:開かれていない  0:サーバとして開かれている  1:クライアントとして開かれている | |
| 説明 | TCP/IP通信の状態を取得する | |

### CDataUnitのメンバ関数

bool Copy(CDataUnit \*p\_src\_du) ①

bool Copy(CDataUnit \*p\_src\_du, bool do\_copy\_data, bool do\_copy\_mask, bool do\_copy\_workarea,

bool do\_copy\_range, bool do\_copy\_mark)　②

|  |  |  |
| --- | --- | --- |
| 引数 | CDataUnit \*p\_src\_du | (in)コピー元の画像メモリ |
| bool do\_copy\_data | (in)画像データをコピーするかどうか |
| bool do\_copy\_mask | (in)マスクをコピーするかどうか |
| bool do\_copy\_workarea | (in)作業領域をコピーするかどうか |
| bool do\_copy\_range | (in)画像データの表示最大／最小値をコピーするかどうか |
| bool do\_copy\_mark | (in)マークをコピーするかどうか |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 他の画像メモリの内容をコピーする  ・コピーする項目を選択することもできる（②）  　項目選択をしない場合（①）は、画像メモリのすべての内容をコピーする | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

bool Compare( CDataUnit \*p\_du )①

bool Compare(long type, CSize size, long page)②

|  |  |  |
| --- | --- | --- |
| 引数 | CDataUnit \*p\_src\_du | (in)比較するの画像メモリ |
| long type | (in)画像データの型 |
| long page | (in)画像データのページ |
| CSize size | (in)画像データの高さと幅 |
| 戻り値 | すべて同じならtrueを返す | |
| 説明 | 画像データを比較する  ・①は他の画像データと型／幅／高さ／ページ数を比較する  ・②は型／幅／高さ／ページ数を数値比較する | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

double GetDataValue(CPoint point, long page=-1, int channel=-1) ①

double GetDataValue(double x, double y, long page=-1, int channel=-1)②

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint point | (in)データ値を取得する座標 |
| double x, y | (in)データ値を取得する座標 |
| long page | (in)画像データのページ（-1の場合は現在表示中のもの） |
| int channel | (in)画像データのチャネル（-1の場合は現在表示中のもの） |
| 戻り値 | 画像データ値 | |
| 説明 | 指定した座標の画像データ値を取得する  ・座標を実数値で指定する場合（②）は、バイリニア補間で求めたデータ値が取得される  ・RGB/RGB3D型でカラー表示の場合、channel=-1とすると以下の式で計算された明度値が取得される  　　　　　　　 I = 2.99\*R + 5.87\*G+1.14\*B | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

unsigned char GetDisplayBrightness(CPoint point)

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint point | (in)表示画像明度を取得する座標 |
| 戻り値 | 表示画像明度 | |
| 説明 | 指定した座標の表示画像の明度を取得する | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

unsigned char GetMaskValue(CPoint point)

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint point | (in)表示画像明度を取得する座標 |
| 戻り値 | マスク値（0:マスクOFF 1:マスクON） | |
| 説明 | 指定した座標のマスク値を取得する | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

bool SetDataValue(int x, int y, double data, long page, int channel, bool satulate=true)

|  |  |  |
| --- | --- | --- |
| 引数 | int x,y | (in)画像データ値を設定する座標 |
| double data | (in)設定するデータ値 |
| long page | (in)画像データのページ（-1の場合は現在表示中のもの） |
| int channel | (in)画像データのチャネル（-1の場合は現在表示中のもの） |
| bool satulate | (in)データ上下限で飽和させるかどうか  　　(falseの場合はオーバーフローする) |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 指定した座標の画像データ値を設定する  ・RGB/RGB3D型でカラー表示の場合、channel=-1とすると全チャネルに同じ値が設定される | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

bool SetMaskValue(int x, int y, BYTE mask)

|  |  |  |
| --- | --- | --- |
| 引数 | int x,y | (in)画像データ値を設定する座標 |
| BYTE mask | (in)設定するマスク値（0:マスクOFF 1:マスクON） |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 指定した座標のマスク値を設定する  ・RGB/RGB3D型でカラー表示の場合、channel=-1とすると全チャネルに同じ値が設定される | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

void ClearMask()

|  |  |
| --- | --- |
| 戻り値 |  |
| 説明 | マスクをすべてOFFにする |

[関数一覧へ](#CDataUnitメンバ関数一覧)

void InitWorkArea ()

|  |  |
| --- | --- |
| 戻り値 |  |
| 説明 | 作業領域を初期化する  ・画面全体の矩形領域が１つだけ設定される |

[関数一覧へ](#CDataUnitメンバ関数一覧)

void SetMaskByWorkArea ()

|  |  |
| --- | --- |
| 戻り値 | 成功したらtrueを返す |
| 説明 | 作業領域外にすべてマスクをかける |

[関数一覧へ](#CDataUnitメンバ関数一覧)

CRect GetRectArea()

|  |  |
| --- | --- |
| 戻り値 | 矩形 |
| 説明 | 作業領域を表す矩形を取得する  ・作業領域が矩形以外の図形である場合、はたは複数の領域を持つ場合は、すべての作業領域に外接する矩形を取得する |

[関数一覧へ](#CDataUnitメンバ関数一覧)

void GetDataMinMax(float \*min, float \*max)

|  |  |  |
| --- | --- | --- |
| 引数 | float \*min | (out)画像データの最大値 |
| float \*max | (out)画像データの最小値 |
| 戻り値 |  | |
| 説明 | 画像データの最大／最小値を取得する  ・ページ／チャネルについては現在表示中のものを対象とする  ・RGB/RGB3D型でカラー表示の場合、以下の式で計算された明度値の最大／最小を取得する  I = 2.99\*R + 5.87\*G+1.14\*B | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

bool SetDispRange(float min, float max)

|  |  |  |
| --- | --- | --- |
| 引数 | float min | (in)画像データの表示最大値 |
| float max | (in)画像データの表示最小値 |
| 戻り値 | 成功したらtrueを返す | |
| 説明 | 画像データの表示最大／最小値を設定し、再表示する | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

void SetDispScale(float scale)

|  |  |  |
| --- | --- | --- |
| 引数 | float scale | (in)表示拡大率 |
| 戻り値 |  | |
| 説明 | 画像表示拡大率を設定し、再表示する | |

[関数一覧へ](#CDataUnitメンバ関数一覧)

### CConfirmDlgのメンバ関数

void SetWidth(int titlesize, int editsize)

|  |  |  |
| --- | --- | --- |
| 引数 | int titlesize | (in) 変数名を表示する領域の幅 |
| int editsize | (in) 変数の値を入力するエディットボックスの幅 |
| 戻り値 |  | |
| 説明 | 汎用データ入力用ダイアログボックスの幅を設定する | |

[関数一覧へ](#CConfirmDlgメンバ関数一覧)

void SetTitle(CString title)

|  |  |  |
| --- | --- | --- |
| 引数 | CString title | (in) ダイアログボックスのタイトル |
| 戻り値 |  | |
| 説明 | 汎用データ入力用ダイアログボックスのタイトルを設定する | |

[関数一覧へ](#CConfirmDlgメンバ関数一覧)

bool RegistRadioVar(CString title, bool \*pval)

|  |  |  |
| --- | --- | --- |
| 引数 | CString title | (in)入力するデータ名 |
| bool \*pval  CString \*pval  double \*pval  float \*pval  short \*pval  int \*pval | (in)入力するデータを格納する変数のポインタ |
| 戻り値 | 成功するとtrueが返る | |
| 説明 | 入力データの設定をする  ・入力するデータを格納する変数のポインタを登録しておく。DoModel時に登録された変数型に準じたダイアログアイテムを表示する。このアイテムで設定された値は、OKボタン押下時に変数に反映される。  ・①の場合はダイアログアイテムがチェックボックスとなる。それ以外の場合はエディットボックスになる  ・これらの関数を複数回使うことで、複数のデータを入力させることができる。 | |

[関数一覧へ](#CConfirmDlgメンバ関数一覧)

bool RegistVar(CString title, int \*pval)

|  |  |  |
| --- | --- | --- |
| 引数 | CString title | (in)入力するデータ名 |
| bool \*pval | (in)入力するデータを格納する変数のポインタ |
| 戻り値 | 成功するとtrueが返る | |
| 説明 | ラジオボタンで選択するデータの設定をする  ・入力するデータを格納する変数のポインタを登録しておく。DoModel時にラジオボタンを表示する。このアイテムで設定された値は、OKボタン押下時に変数に反映される。  ・これらの関数を複数回使うことで、複数のデータを選択させることができる | |

[関数一覧へ](#CConfirmDlgメンバ関数一覧)

int DoModal()

|  |  |
| --- | --- |
| 戻り値 | OKボタンで終了：IDOK キャンセルボタンで終了：IDCANCEL |
| 説明 | 汎用データ入力用ダイアログボックスを表示する  ・ダイアログ上でOKまたはキャンセルボタンを押すと終了し、押されたボタンに準ずる戻り値を返す |

[関数一覧へ](#CConfirmDlgメンバ関数一覧)

**使用例**

**＜プログラム＞**

//変数宣言

CConfirmDlg dlg;

double param0;

int param1;

bool param2;

bool select[3];

//パラメタ初期値を設定する

param0 = 10.5;

param1 = 5;

param2 = true;

select[0]=true;

select[1]=false;

select[2]=false;

//ダイアログの設定をする

dlg.SetTitle("パラメタ入力");

dlg.RegistVar("パラメタ", &param0);

dlg.RegistVar("パラメタ", &param1);

dlg.RegistVar("パラメタ", &param2);

dlg.RegistRadioVar("選択A", &select[0]);

dlg.RegistRadioVar("選択B", &select[1]);

dlg.RegistRadioVar("選択C", &select[2]);

if( dlg.DoModal() == IDOK ){

　　//OKボタン押下時の処理

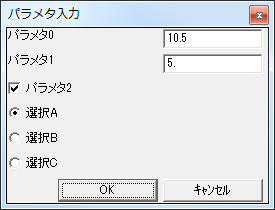
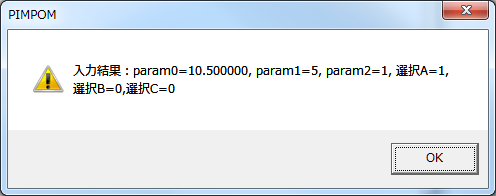
　　API.MessageBox("入力結果：param0=%lf, param1=%d, param2=%d, 選択A=%d,選択B=%d,選択C=%d ",

param0, param1, param2, select[0], select[1], select[2]);

}

**<実行時の様子>**

実行すると以下のダイアログが表示される

OKを押下するとメッセージボックスが表示される

## カスタムファンクションのイベントハンドラ

以下に記載する関数はすべてカスタムファンクション外のイベントを受けるためのイベントハンドラである

カスタムファンクション作成時には、以下の関数内部に処理を追記して使用する。

void OnMouseDownOnImage(CPoint point)

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint point | (in)マウスボタン押下された位置の画像座標 |
| 戻り値 |  | |
| 説明 | メイン画面上でのマウスボタン押下時に呼び出される | |

void OnMouseDrugOnImage(CPoint start\_point, CPoint currnet\_point)

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint start\_point | (in)マウスドラッグ開始した位置の画像座標 |
| CPoint currnet\_point | (in)現在のマウス位置の画像座標 |
| 戻り値 |  | |
| 説明 | メイン画面上でのマウスドラッグ中に呼び出される | |

void OnFinishMouseDrugOnImage(CPoint start\_point, CPoint currnet\_point)

|  |  |  |
| --- | --- | --- |
| 引数 | CPoint start\_point | (in)マウスドラッグ開始した位置の画像座標 |
| CPoint currnet\_point | (in)マウスドラッグ終了した位置の画像座標 |
| 戻り値 |  | |
| 説明 | メイン画面上でのマウスドラッグ終了時に呼び出される | |

void OnBatchExecute(BATCH\_INPUT \*in\_param, BATCH\_OUTPUT \*out\_param)

|  |  |  |
| --- | --- | --- |
| 引数 | BATCH\_INPUT \*in\_param | (in)入力パラメタ  typedef struct{  char param[BATCH\_IN\_PARAM\_NUM][1024];//入力パラメタ  char imae\_file\_path[1024];//入力画像のファイル名  int image\_number;//入力画像が入っているメモリ番号  }BATCH\_INPUT; |
| BATCH\_OUTPUT \*out\_param | (in)出力パラメタ  typedef struct{  char result[BATCH\_OUT\_PARAM\_NUM][1024];//出力パラメタ  int skip\_flg;//処理対象外を示すフラグ  }BATCH\_OUTPUT; |
| 戻り値 |  | |
| 説明 | バッチ実行もしくはライブ実行時に呼び出される  ・バッチ実行時には、「バッチ実行ウィンドウ」で設定された入力パラメタが渡される。また出力パラメタを設定することで、バッチ実行ウィンドウに値を反映することができる | |

void OnReceive(BYTE \*data, int \*ndata)

|  |  |  |
| --- | --- | --- |
| 引数 | BYTE \*data | (in)受信したデータ列の先頭ポインタ |
| int \*ndata | (in)データサイズ[byte] |
| 戻り値 |  | |
| 説明 | TCP/IPからのデータ受信時に呼び出される | |

## 定数一覧

APIで使用する定数一覧を示す。

以下に記載する定数は「global\_define.h」で定義されている

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分類 | 定数名 | | 値 | 内容 |
| 画像メモリ | | CURRENT\_IMAGE | -1 | 現在選択中の画像メモリを表す |
| 画像データ型 | | BYTE\_FORMAT | 0 | 8bit符号なしモノクロ静止画 |
|  | | FLOAT\_FORMAT | 1 | 32bit浮動小数点モノクロ静止画 |
|  | | COMPLEX\_FORMAT | 2 | 複素数静止画 |
|  | | THREE\_D\_FORMAT | 3 | 8bit符号なしモノクロ動画 |
|  | | FLOAT\_3D\_FORMAT | 4 | 32bit浮動小数点モノクロ動画 |
|  | | XY\_FORMAT | 5 | 32bit浮動小数点×2チャネル静止画 |
|  | | XYZ\_FORMAT | 6 | 32bit浮動小数点×3チャネル静止画 |
|  | | RGB\_FORMAT | 7 | 8bitカラー静止画 |
|  | | WORD\_FORMAT | 8 | 16bit符号なしモノクロ静止画 |
|  | | LONG32\_FORMAT | 9 | 32bit符号つきモノクロ静止画 |
|  | | RGB\_3D\_FORMAT | 10 | 8bitカラー動画 |
|  | | SHORT16\_FORMAT | 11 | 16bit符号つきモノクロ静止画 |
|  | | DWORD\_FORMAT | 12 | 32bit符号なしモノクロ静止画 |
| マスク | | MASKED\_PIXEL | 0 |  |
|  | | UNMASKED\_PIXEL | 1 |  |
| 作業領域 | | WORK\_AREA\_TYPE\_NOTHING | -1 | なし |
|  | | WORK\_AREA\_TYPE\_RECT | 0 | 矩形 |
|  | | WORK\_AREA\_TYPE\_LINE | 1 | 幅つき直線 |
|  | | WORK\_AREA\_TYPE\_ELLIPSE | 2 | 楕円 |
|  | | WORK\_AREA\_TYPE\_CIRCLE | 3 | 幅つき円周 |
|  | | WORK\_AREA\_TYPE\_ARC | 4 | 円弧 |
|  | | WORK\_AREA\_TYPE\_POLYGON | 5 | 多角形 |
| バッチ実行 | | BATCH\_IN\_PARAM\_NUM | 5 | バッチ実行の入力パラメタ数 |
|  | | BATCH\_OUT\_PARAM\_NUM | 15 | バッチ実行の出力パラメタ数 |
| その他 | | PI | 3.141592.. | 円周率 |
|  | | CR | 0x0d | キャリッジリターン |
|  | | LF | 0x0a | ラインフィード |
|  | |  |  |  |