管理文書 No.SD09-0056-001

S-cube API

リファレンスマニュアル

駿河精機株式会社
OST 事業部
システムソリューション BU　光センサ T

改訂履歴

版数	作成日	変更内容
1.0	2009/11/16	新規作成

目次

1. はじめに

「S-cube」API は、弊社波面光学ヘッドを制御して、波面収差測定を行うためのライブラリです。

2. 動作環境

・パソコン：IBM PC/AT 互換機
・OS：Windows XP
・CPU：Intel Core2 Duo1.80GHz 以上
・メモリ：2GB 以上
・開発環境：Microsoft Visual C++9.0 以降

3．システム概要

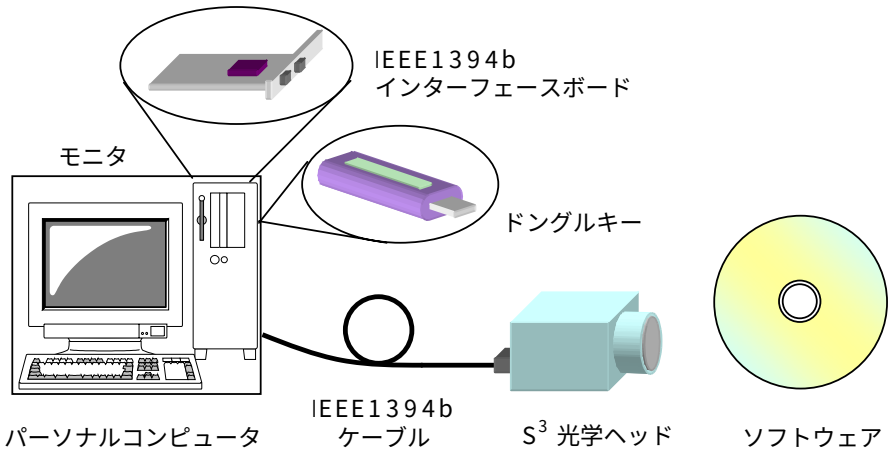
S-cube 高速波面センサのシステム構成は、以下のようになっております。

・波面センサヘッド
・S-cube API
・ドングルキー

周辺機器（別売り）

・IEEE1394ｂケーブル
・IEEE1394ｂインターフェースボード
・パーソナルコンピュータ（IBM PC/AT 互換機）
・モニタ（XGA：1024ｘ768）以上
・小型カメラアダプター（電源装置）※
・12 ピンカメラケーブル※

※PC からの電源供給能力が低い場合必要となります。



4. 開発環境

4.1. 開発環境の設定

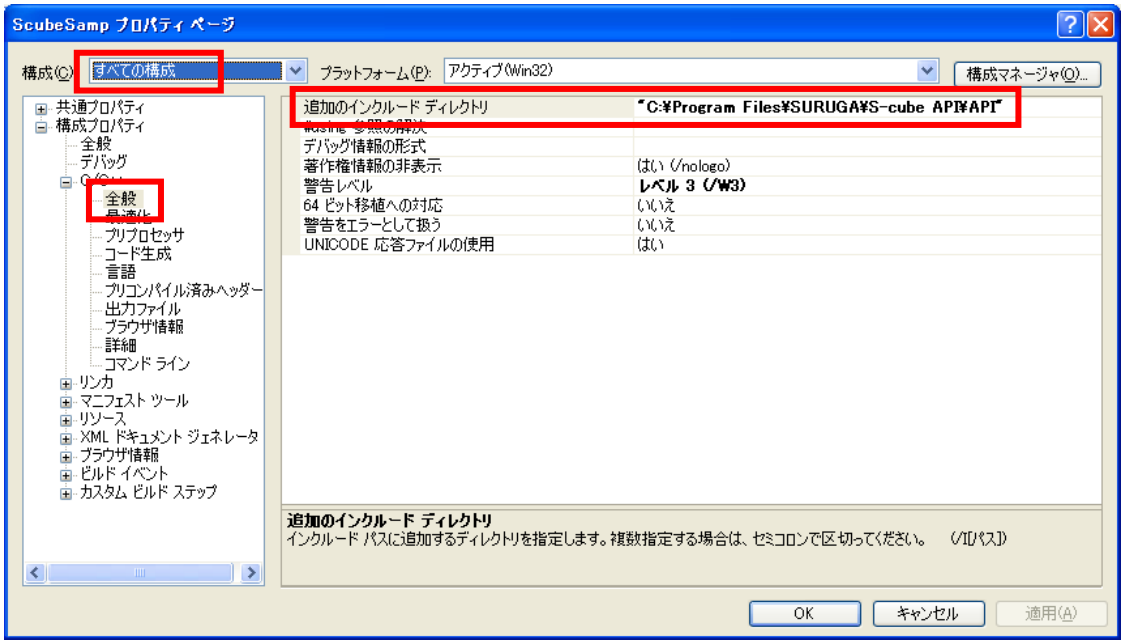
本ライブラリは、以下に示すファイルを含みます。

1. S3Lib.h　…　ヘッダファイル
2. S3API.lib　…　ライブラリファイル
3. S3API.dll　…　ダイナミックリンクライブラリ

本ライブラリを使用する前に、以下の設定を行ってください。

●インクルードファイル(*.h)

①メニューバーの [プロジェクト]→[プロパティ]を選択して以下の画面を開きます。



②構成の欄に「すべての構成」を選択します。

③左のペインから[構成プロパティ]→[C/C++]→[全般]を選択します。

④右のペインから「追加のインクルードディレクトリ」の欄に、インクルードファイルがインストールされているフォルダのパスを指定します。
デフォルトのインストール先の場合、以下のようになります。

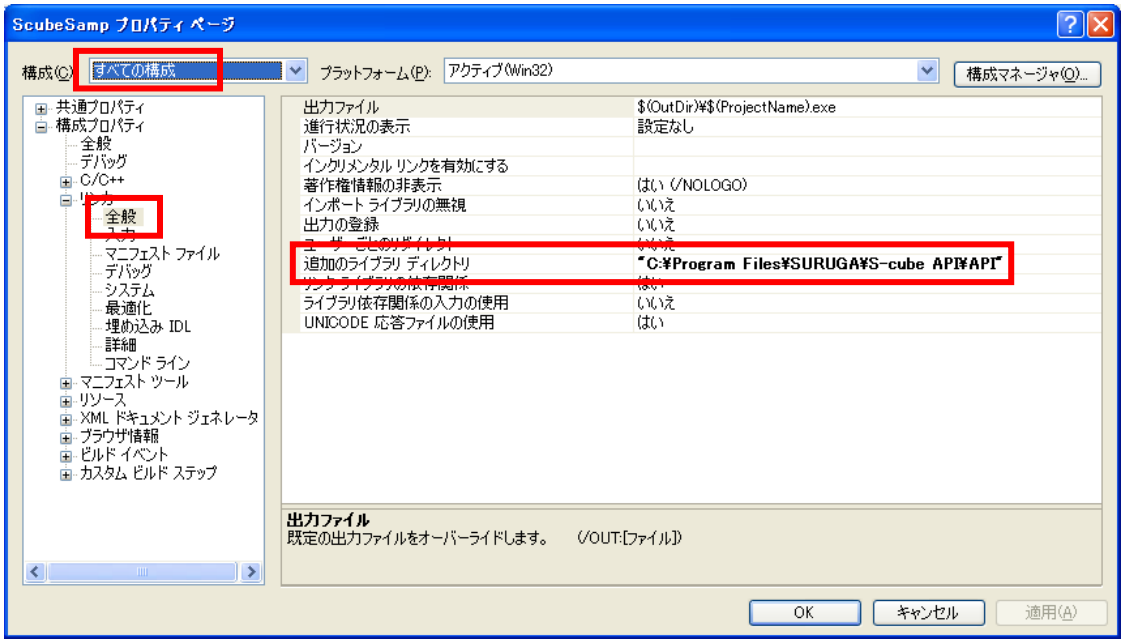
C:¥Program Files¥SURUGA¥S-cube API¥API

⑤作成するアプリケーションの“ STDAFX.h ”ファイルまたは各ソースファイルで、
以下のように “　S3Lib.h ” ファイルをインクルードしてください。

#include “S3Lib.h”

●ライブラリファイル（*.lib）

①メニューバーの［プロジェクト]→[プロパティ]を選択して以下の画面を開きます。



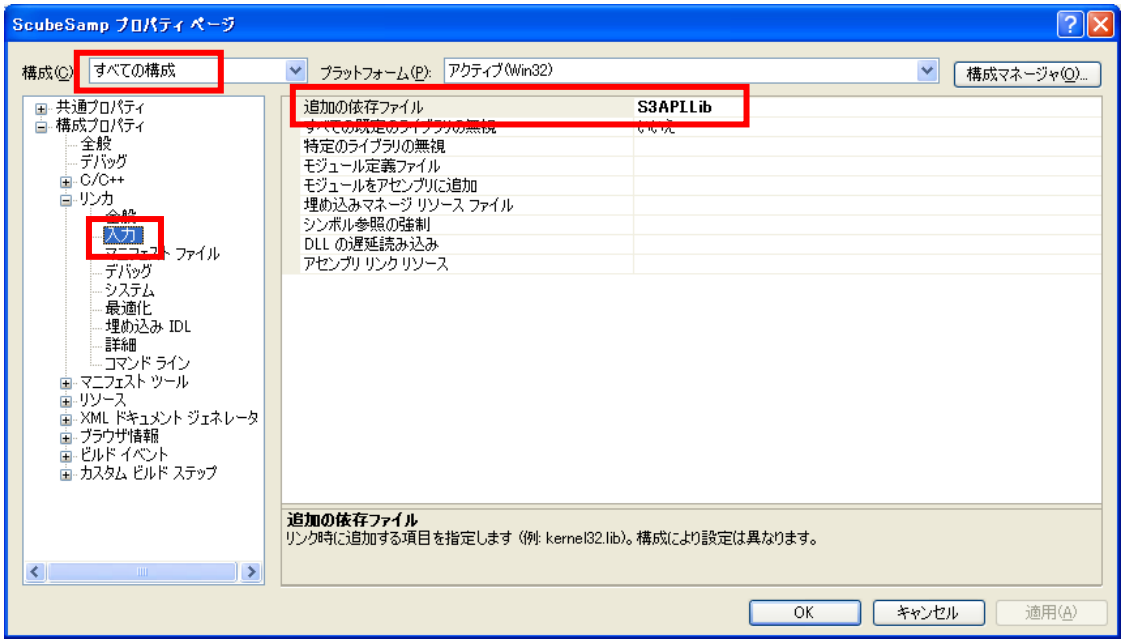
②構成の欄に「すべての構成」を選択します。

③左のペインから[構成プロパティ]→[リンカ]→[全般]を選択します。

④右のペインから「追加のライブラリ ディレクトリ」の欄に、ライブラリファイルがインストールされているフォルダのパスを指定します。
デフォルトのインストール先の場合、以下のようになります。

C:¥Program Files¥SURUGA¥S-cube API¥API

⑤左のペインから[構成プロパティ]→[リンカ]→[入力]を選択します。



⑥右のペインから「追加の依存ファイル」の欄に、以下のライブラリファイルを指定します。

S3Api.lib

5. 基本動作フロー

以下に基本的な「S-cube」ライブラリの動作フローを示します。

5.1. ライブラリ起動時

①ライブラリのオープン

s3_Open()関数を実行して「S-cube」ライブラリの使用を開始します。

②ライブラリパラメータの初期値設定

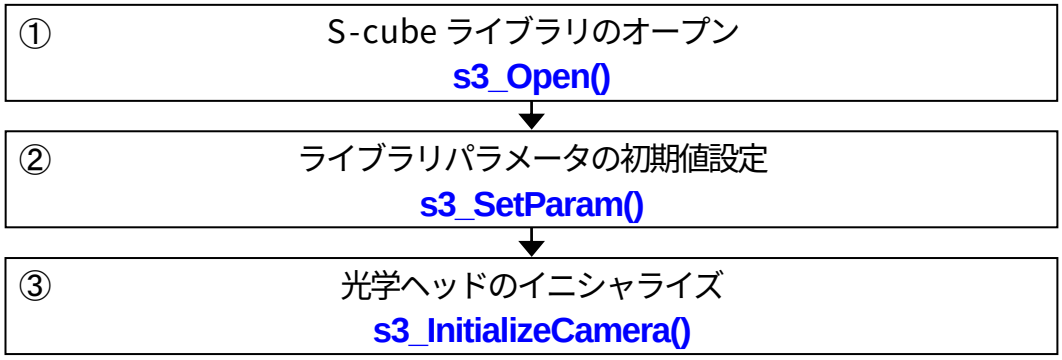
s3_SetParam()関数を実行してライブラリの初期パラメータを設定しておきます。

※ライブラリのパラメータは、上記関数を実行することで随時変更できます。

③光学ヘッドのイニシャライズ

s3_InitializeCamera()関数を実行して光学ヘッドのイニシャライズを行います。

これで、光学ヘッドから測定データを入力できる状態になります。



5.2. 校正データの生成

波面収差測定を行う前に、校正データを作成する必要があります。
校正データの作成には、理想的な光学レイアウトを仮定した「デフォルト校正」と、実際の映像をもとに生成する「ユーザー校正」の２通りの方法があります。

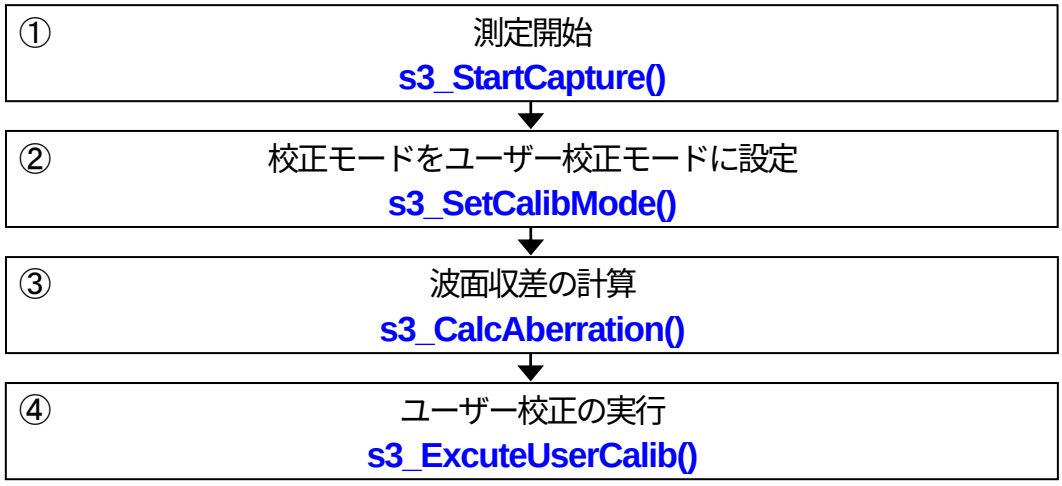
ユーザー校正の場合は、光学ヘッド固有の光学特性を加味して校正を行っているため、入光状態が悪くスポットの点数が少なくなったりする場合には、適正に校正できないことがあります。

※ユーザー校正データについては、出荷時に弊社で作成した校正ファイルを同梱していますので、通常はそちらのデータを使用することをお勧めします。

A）デフォルト校正の場合
デフォルト校正データはライブラリの起動時に自動的に作成されます。

B）ユーザー校正の場合
ユーザー校正データは測定データをもとに生成するので、測定中にs3_ExcuteUserCalib()関数を実行します。

また、ファイルに保存された校正データを読み込む場合は、光学ヘッドのイニシャライズ完了後、任意のタイミングで s3_ReadUserCalibData()関数を実行します。この場合は、測定中である必要はありません。



5.3. 測定

①測定開始

s3_StartCapture()関数を実行して測定を開始します。光学ヘッドの映像を取得できる状態になります。

②校正モードの選択

計算を開始する前に、s3_SetCalibMode()関数を実行して校正モードを指定します。

③波面収差の計算

s3_CalcAberration()関数を実行して、入力映像から波面収差の各測定値を計算します。

④測定結果の取得

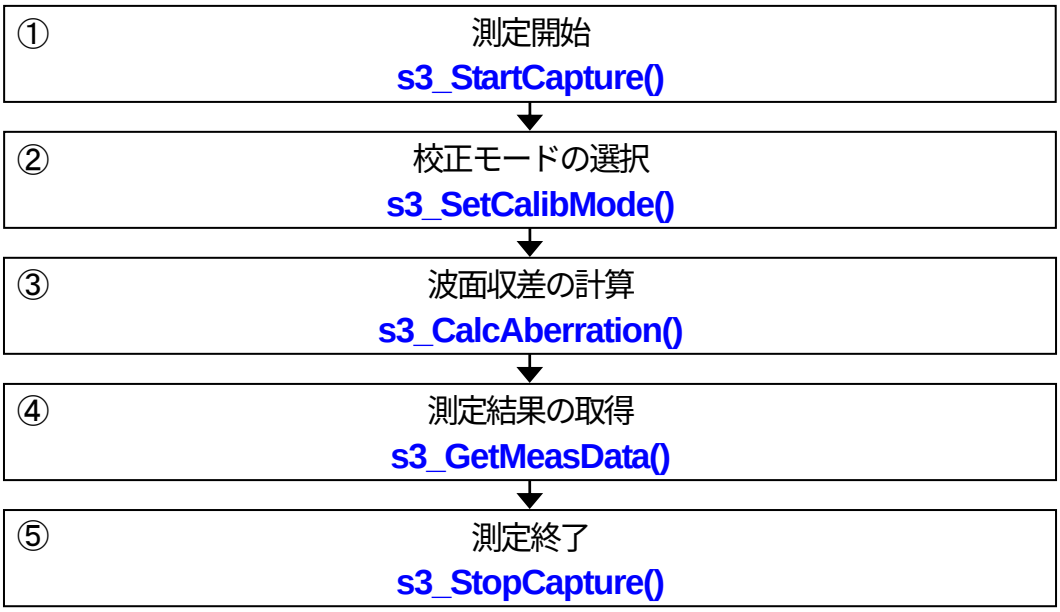
s3_GetMeasData()関数を実行して③で計算した測定結果を取得します。

※測定結果の更新

測定結果を最新の値に更新したい場合は、③④の処理を繰り返し実行します。

⑤測定終了

測定を終了する場合は、s3_StopCapture()関数を実行します。再開する場合は再度、s3_StartCapture()関数を実行します。



5.4. マスクデータ

本ライブラリでは、指定した計測エリア内の校正データがゼルニケ係数の算出に使用されますが、このデータにマスク処理を行うことで、計測エリア内の一部の校正データを使用しなかったり、逆に計測エリア外の校正データを使用したりすることができます。この校正データの選択情報を格納したものが、マスクデータになります。

マスクデータの作成は必須ではありませんが、計測光の状態が良くない場合には、マスク機能を使用することで、計測結果に悪影響を与える測定点を除外して、より安定した測定を行うことができます。（デフォルトの設定では、マスク処理は無効となっています）

①最初に s3_CalcMask()関数を実行して、マスクの初期状態を計算します。
初期状態では、各マスク要素は以下のように分類されます。

・計測エリアの内側の校正スポット点 … マスク無効
・計測エリアの外側の校正スポット点 … マスク有効
・計算不能な校正スポット点 … 対応点なし

②マスクデータの編集
SetMaskAt()関数, s3_SetMaskList()関数を使用して、現在のマスクデータの構成要素の状態値および座標を取得することができます。取得したマスク情報には、そのマスク要素の位置情報が含まれているので、どのスポット点を使用して、どのスポット点を使用しないかという選択を行うことができます。変更したマスク情報を指定するには、s3_GetMaskAt()関数, s3_GetMaskList()関数を使用します。

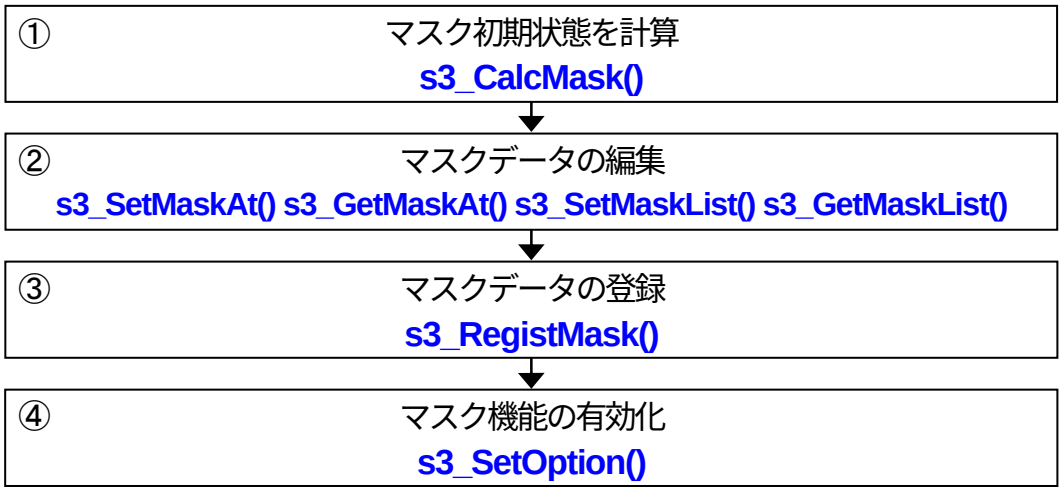
③マスクデータの登録
作成したマスクデータを測定に適用する前に、マスクデータを登録する必要があります。登録には、s3_RegistMask()関数を使用します。

④マスク機能の有効化
③で登録したマスクデータを使用する場合には、S3_T_OPTION 構造体の UseMask メンバを１に設定してマスク機能を有効化します。

⑤マスクデータの保存

作成したマスクデータを保存する場合には、s3_SaveMask()関数を使用します。保存したマスクデータは、s3_ReadMask()関数を使用することで、再度読み込んで利用することができます。

※マスクデータは現在の校正データを元に作成されるので、校正データが変更された場合には、マスクデータをもう一度作成し直す必要があります。



5.5. スナップショット測定

通常の測定では、光学ヘッドからの映像を連続して取り込みながらリアルタイムに解析処理を行いますが、本ライブラリでは、そのような測定の他に、測定中にフレーム画像をファイルに保存しておくことで、ある瞬間の入力映像を、オフラインで解析を行うことができます。

5.5.1. スナップショット画像の保存

①測定開始

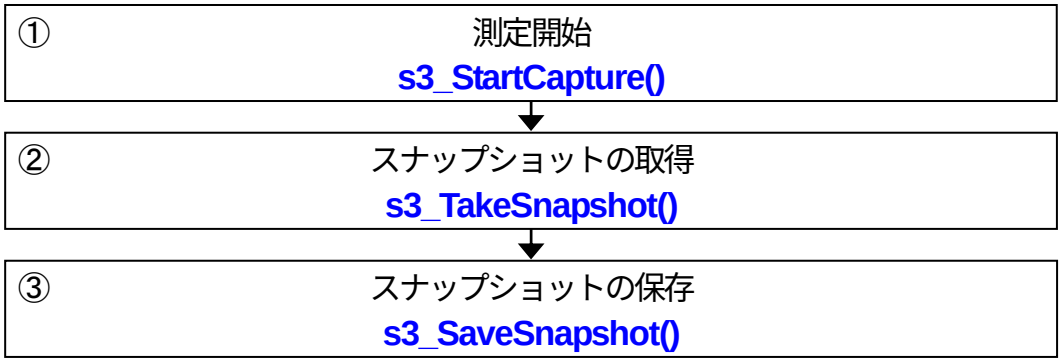
s3_StartCapture()関数を実行して、測定を開始します。

②スナップショットの取得

s3_TakeSnapshot()関数を実行して、現在のスナップショット画像を取得します。

③スナップショットの保存

s3_SaveSnapshot()関数を実行して、②で取得したスナップショット画像を保存します。



5.5.2. スナップショット画像の測定

①測定開始

s3_StartCapture()関数を実行して、測定を開始します。

②スナップショットの読み込み

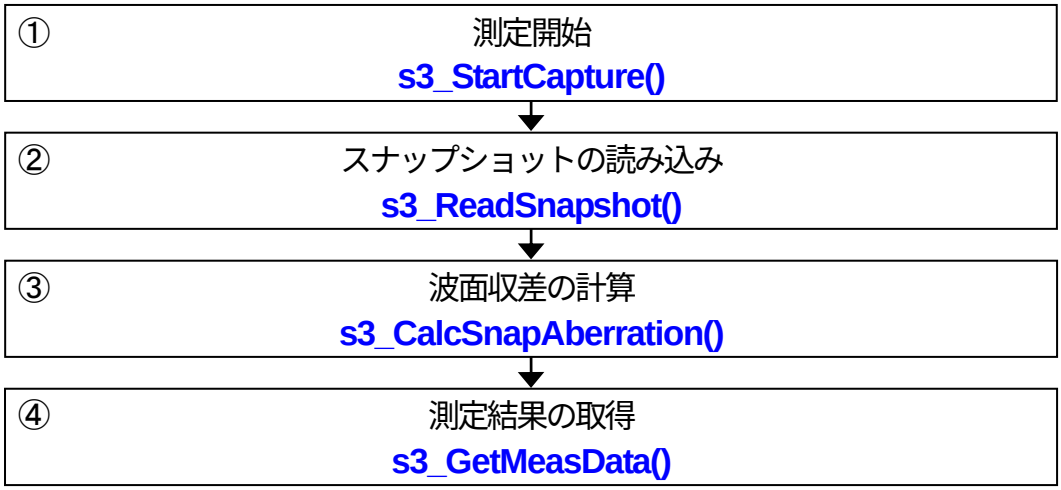
s3_ReadSnapshot()関数を実行して、5.5.1.章で作成したスナップショット画像を読み込みます。

③波面収差の計算

s3_CalcSnapAberration()関数を実行して、スナップショット画像から波面収差の各測定値を計算します。

④測定結果の取得

s3_GetMeasData()関数を実行して③で計算した測定結果を取得します。



5.6. ライブラリ終了時

①ライブラリのクローズ

s3_Close()関数を実行して、「S-cube」ライブラリの使用を終了します。

6. 関数一覧

1	s3_Open()	S-cube ライブラリの開始
2	s3_Close()	S-cube ライブラリの終了
3	s3_GetLibVersion()	S-cube ライブラリのバージョン取得
4	s3_GetLastError()	エラーコードの取得
5	s3_SetParam()	S-cube ライブラリのパラメータ設定
6	s3_GetParam()	S-cube ライブラリのパラメータ取得
7	s3_GetDeviceInfo()	デバイス情報の取得
8	s3_SetCalibMode()	校正モードの設定
9	s3_GetCalibMode()	校正モードの取得
10	s3_SetUnitType()	測定値の出力単位の設定
11	s3_GetUnitType()	測定値の出力単位の取得
12	s3_SetFormatType()	測定値の出力形式の設定
13	s3_GetFormatType()	測定値の出力形式の取得
14	s3_SetAngUnitType()	測定値の出力角度単位の設定
15	s3_GetAngUnitType()	測定値の出力角度単位の取得
16	s3_InitializeCamera()	光学ヘッドの初期化
17	s3_UninitializeCamera()	光学ヘッドのリソース解放
18	s3_StartCapture()	測定開始
19	s3_StopCapture()	測定停止
20	s3_SetCameraShutterSpeed()	光学ヘッドシャッタースピード値の設定
21	s3_GetCameraShutterSpeed()	光学ヘッドシャッタースピード値の取得
22	s3_SetCameraGain()	光学ヘッドゲイン値の設定
23	s3_GetCameraGain()	光学ヘッドゲイン値の取得
24	s3_ReadUserCalibData()	ユーザー校正データ読み込み
25	s3_SaveUserCalibData()	ユーザー校正データ保存
26	s3_ExcuteUserCalib()	ユーザー校正の実行
27	s3_GetMaskHandle()	マスクオブジェクトのハンドル取得
28	s3_isMaskEmpty()	空のマスクオブジェクトかどうかの確認
29	s3_GetMaskCount()	マスク要素数の取得
30	s3_GetMaskInfo()	マスクのパラメータ情報の取得
31	s3_CalcMask()	マスクの計算
32	s3_UpdateMask()	マスクの更新

33	s3_ReadMask()	マスクデータ読み込み
34	s3_SaveMask()	マスクデータ保存
35	s3_ClearMask()	マスクのクリア
36	s3_RegistMask()	マスクの登録
37	s3_SetMaskAt()	マスク要素の設定
38	s3_GetMaskAt()	マスク要素の取得
39	s3_SetMaskList()	マスク要素列の設定
40	s3_GetMaskList()	マスク要素列の取得
41	s3_CreateMask()	マスクオブジェクトの生成
42	s3_CopyMask()	マスクのコピー
43	s3_ReleaseMask()	マスクオブジェクトの解放
44	s3_SetZernikeTermUse()	ゼルニケ有効係数の設定
45	s3_SetZernikeOffsetGain()	ゼルニケ係数オフセット・ゲイン補正値の設定
46	s3_GetZernikeOffsetGain()	ゼルニケ係数オフセット・ゲイン補正値の取得
47	s3_CalcAberrationMap()	波面収差マップデータの計算
48	s3_CalcIntensityMap()	強度マップデータの計算
49	s3_GetSpotPeakValue()	ビームスポットの最大ピーク階調値の取得
50	s3_CountSaturationPixels()	飽和ピクセル数の取得
51	s3_GetSpotNum()	ビームスポット数の取得
52	s3_GetBeamProfile()	ビームプロファイルの取得
53	s3_GetAutoAreaSize()	計測エリアサイズの取得
54	s3_CalcAberration()	波面収差の計算実行
55	s3_GetMeasData()	測定データの取得
56	s3_GetZernikeAberration()	ゼルニケ収差測定データの取得
57	s3_GetSeidelAberration()	ザイデル収差測定データの取得
58	s3_GetTotalAberration()	総合波面収差測定データの取得
59	s3_GetCapturedFrameHandle()	フレームオブジェクトのハンドル取得
60	s3_GetFrameBmpInfo()	フレームのビットマップ情報取得
61	s3_GetFrameSize()	フレームのサイズ取得
62	s3_GetFramePixel()	フレームのピクセルデータ取得
63	s3_GetFramePixelBufferSize()	フレームのピクセルデータのバッファサイズ取得
64	s3_isMeasuring()	測定中かどうかの確認
65	s3_isCalibDone()	校正完了の確認
66	s3_isUserCalibDone()	ユーザー校正完了の確認
67	s3_isMaskDone()	マスク登録完了の確認

68	s3_isSnapshotEmpty()	スナップショットデータが存在するどうかの確認
69	s3_TakeSnapshot()	スナップショット画像の取得
70	s3_GetSnapshotHandle()	スナップショットのハンドル取得
71	s3_CalcSnapAberration()	スナップショットの波面収差の計算実行
72	s3_ReadSnapshot()	スナップショット画像の読み込み
73	s3_SaveSnapshot()	スナップショット画像の保存

7. 関数リファレンス

7.1. s3_Open()

形式

BOOL s3_Open(void)

機能

S-cube ライブラリのオープンを行います。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

本関数は S-cube ライブラリの関数を使用する前に、必ず一度実行しておく必要があります。

7.2. s3_Close()

形式

BOOL s3_Close(void)

機能

S-cube ライブラリを終了します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

s3_Open()関数にてオープンしたライブラリをクローズします。
アプリケーションは S-cube ライブラリの使用を終了するときに、必ず本関数を実行しなければなりません。

7.3. s3_GetLibVersion()

形式

BOOL s3_GetLibVersion(unsigned long **pMajor*, unsigned long **pMinor*)

機能

ライブラリのバージョン番号を取得します。

引数

pMajor	[OUT]	ライブラリのメジャー番号を受け取る変数へのポインタです。
pMinor	[OUT]	ライブラリのマイナー番号を受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

例えば、バージョン“ 1.23 ”の場合は、メジャー番号は 1、マイナー番号は 23 を返します。

7.4. s3_GetLastError()

形式

S3_T_ERR_CORD s3_GetLastError(void)

機能

最後に発生したエラーコードを取得します。
エラーコードの詳細については、10 章のエラーコード一覧をご覧ください。

引数

なし。

戻り値

取得するエラーコード。

例

```
S3_T_ERR_CORD err;

if(!s3_InitializeCamera()) {
    err=s3_GetLastError();          //失敗した場合にエラーコードを取得します
}
```

7.5. s3_SetParam()

形式

BOOL s3_SetParam(S3_T_OPTION **pParam*)

機能

ライブラリパラメータを設定します。

引数

pParam　　[IN] パラメータ情報を指定する構造体へのポインタです。

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

測定や光学ヘッドの制御に必要なパラメータを、引数の S3_T_OPTION 構造体により設定します。

例

```
S3_T_OPTION Option;

if(!s3_GetParam(&Option)) {                    //パラメータの取得
    //パラメータの取得に失敗した場合の処理
}
Option.CAMERA.Gain=0.5;                        //カメラゲイン値の変更
Option.IMAGE.UseFrameAve=0;                    //フレーム平均処理有無の変更
Option.OUTPUT.UnitType=1;                      //出力単位モードの変更
if(!s3_SetParam(&Option)) {                    //パラメータの設定
    //パラメータの設定に失敗した場合の処理
}
```

7.6. s3_GetParam()

形式

BOOL s3_GetParam(S3_T_OPTION **pParam*)

機能

現在設定されているライブラリパラメータを取得します。

引数

pParam　　[OUT] パラメータ情報を受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

7.7. s3_GetDeviceInfo()

形式

BOOL s3_GetDeviceInfo(S3_T_DEV_INFO **pInfo*)

機能

現在設定されているデバイス情報を取得します。

引数

pInfo	[OUT] デバイス情報を受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

9.13.章 S3_T_DEV_INFO 構造体

7.8. s3_SetCalibMode()

形式

BOOL s3_SetCalibMode(S3_T_CALIB_MODE *Mode*)

機能

校正モードを設定します。

引数

Mode　　[IN] 校正モードを指定する変数です。
（S3_C_STANDARD_CALIB ：デフォルト校正
/ S3_C_USER_CALIB　　：ユーザー校正）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・２種類の校正モード（デフォルト校正/ユーザー校正）のうち、どちらのモードで測定を行うのかを指定します。
・校正モードは S3_T_OPTION 構造体の CalibMode メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

例

```
S3_T_CALIB_MODE Mode=S3_C_USER_CALIB;
s3_SetCalibMode(Mode);                    //ユーザー校正モードに設定
```

7.9. s3_GetCalibMode()

形式

BOOL s3_GetCalibMode(S3_T_CALIB_MODE **pMode*)

機能

現在の校正モードを取得します。

引数

Mode　　[OUT] 校正モードを受け取る変数へのポインタです。
（S3_C_STANDARD_CALIB ：デフォルト校正
/ S3_C_USER_CALIB　　：ユーザー校正）

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
S3_T_CALIB_MODE Mode;

if(!s3_GetCalibMode(&Mode)) {          //校正モードの取得
    //校正モードの取得に失敗した場合の処理
}
if(Mode==S3_C_STANDERD_CALIB) {
    //デフォルト校正モードの場合の処理
}else if(Mode==S3_C_USER_CALIB) {
    //ユーザー校正モードの場合の処理
}
```

7.10. s3_SetUnitType()

形式

BOOL s3_SetUnitType(S3_T_UNIT_TYPE *Type*)

機能

出力単位モードを設定します。

引数

Type	[IN] 出力単位モードを指定する変数です。 （S3_C_UNIT_LAMBDA：λ / S3_C_UNIT_MICRO：μm）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・ゼルニケ係数、ザイデル係数、その他の測定値の測定単位を設定します。
・出力単位モードは、S3_T_OPTION 構造体の UnitType メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

7.11. s3_GetUnitType()

形式

BOOL s3_GetUnitType(S3_T_UNIT_TYPE *pType)

機能

現在の出力単位モードを取得します。

引数

pType	[OUT] 出力単位モードを受け取る変数へのポインタです。 （S3_C_UNIT_LAMBDA：λ / S3_C_UNIT_MICRO：μm）

戻り値

TRUE	正常終了
FALSE	異常終了

7.12. s3_SetFormatType()

形式

BOOL s3_SetFormatType(S3_T_FORMAT_TYPE *Type*)

機能

出力形式モードを設定します。

引数

Type	[IN] 出力形式モードを指定する変数です。 （S3_C_FMT_PV：PV / S3_C_FMT_RMS：RMS）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・ゼルニケ係数、ザイデル係数、その他の測定値の出力形式を設定します。

・出力形式モードは、S3_T_OPTION 構造体の UnitFormat メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

7.13. s3_GetFormatType()

形式

BOOL s3_GetFormatType(S3_T_FORMAT_TYPE **pType*)

機能

現在の出力形式モードを取得します。

引数

pType	[OUT] 出力形式モードを受け取る変数へのポインタです。 （S3_C_FMT_PV：PV / S3_C_FMT_RMS：RMS）

戻り値

TRUE	正常終了
FALSE	異常終了

7.14. s3_SetAngUnitType()

形式

BOOL s3_SetAngUnitType(S3_T_ANG_UNIT_TYPE *Type*)

機能

出力角度単位モードを設定します。

引数

Type	[IN] 出力角度単位モードを指定する変数です。 （S3_C_ANG_DEG ：度 / S3_C_ANG_MRAD：ミリラジアン / S3_C_ANG_MIN ：分）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・総合波面収差 入射角度 x / y / D 項の測定値の出力角度単位を設定します。

・出力角度単位モードは、S3_T_OPTION 構造体の AngUnitType メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

7.15. s3_GetAngUnitType()

形式

BOOL s3_GetAngUnitType(S3_T_ANG_UNIT_TYPE *pType)

機能

現在の出力角度単位モードを取得します。

引数

pType	[OUT] 出力角度単位モードを受け取る変数へのポインタです。 （S3_C_ANG_DEG ：度 / S3_C_ANG_MRAD：ミリラジアン / S3_C_ANG_MIN ：分）

戻り値

TRUE	正常終了
FALSE	異常終了

7.16. s3_InitializeCamera()

形式

BOOL s3_InitializeCamera(void)

機能

光学ヘッドの初期化を行います。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

測定を開始する前に、この関数を実行して、光学ヘッドを初期化しておく必要があります。

7.17. s3_UninitializeCamera()

形式

BOOL s3_UninitializeCamera(void)

機能

光学ヘッドの解放を行います。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

7.18. s3_StartCapture()

形式

BOOL s3_StartCapture(void)

機能

カメラ映像の入力を開始します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

この関数を実行する前に、s3_InitializeCamera()関数を実行して、光学ヘッドの初期化を行っておく必要があります。

7.19. s3_StopCapture()

形式

BOOL s3_StopCapture(void)

機能

カメラ映像の入力を停止します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

7.20. s3_SetCameraShutterSpeed()

形式

BOOL s3_SetCameraShutterSpeed(double *Speed*)

機能

光学ヘッドのシャッタースピード値の設定を行います。

引数

Speed　　[IN] シャッタースピードを指定する変数です。
0．01～16000．00（0．01ms 単位）

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

シャッタースピード値は、S3_T_OPTION 構造体の Shutter メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

例

```
if(!s3_SetCameraShutterSpeed(10.5)) {       //シャッタースピード値を 10.5ms に設定
    //シャッタースピード値の設定に失敗した場合の処理
}
```

7.2 1. s3_GetCameraShutterSpeed()

形式

BOOL s3_GetCameraShutterSpeed(double **pSpeed*)

機能

現在の光学ヘッドのシャッタースピード設定値を取得します。

引数

pSpeed　　[OUT] シャッタースピードを受け取る変数へのポインタです。
　　0．01～16000．00（0．01ms 単位）

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
double Speed;
if(!s3_GetCameraShutterSpeed(&Speed)) { //シャッタースピード値の取得
    //シャッタースピード値の取得に失敗した場合の処理
}
```

7.22. s3_SetCameraGain()

形式

BOOL s3_SetCameraGain(double *Gain*)

機能

光学ヘッドのゲイン値を設定します。

引数

Gain　　[IN] ゲイン値を指定する変数です。
　　0．00～24．00（0．05dB 単位）

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

ゲイン値は、S3_T_OPTION 構造体の Gain メンバによっても設定/取得できます。
（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

例

```
if(!s3_SetCameraGain(0.0)) {          //ゲイン値を 0dB に設定
    //ゲイン値の設定に失敗した場合の処理
}
```

7.2 3. s3_GetCameraGain()

形式

BOOL s3_GetCameraGain(double **pGain*)

機能

現在の光学ヘッドのゲイン設定値を取得します。

引数

pGain　　[OUT] ゲイン値を受け取る変数へのポインタです。
0．00～24．00（0．05dB 単位）

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
double Gain;
if(!s3_GetCameraGain(&Gain)) {          //ゲイン値の取得
    //ゲイン値の取得に失敗した場合の処理
}
```

7.2 4. s3_ReadUserCalibData()

形式

BOOL s3_ReadUserCalibData(char **pCalibFileName*)

機能

指定したユーザー校正ファイルからユーザー校正データを読み込みます。

引数

pCalibFileName	[IN]	読み込み元のユーザー校正ファイルのパス名を指定する文字列へのポインタです。（拡張子 " .CLB " ）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

ファイルから校正データを読み込み、ユーザー校正データとしてセットします。実際に、ここで読み込んだ校正データを測定に反映させるには、校正モードを " ユーザー校正 " モードに設定しておかなければなりません。

7.2 5. s3_SaveUserCalibData()

形式

BOOL s3_SaveUserCalibData(char **pCalibFileName*)

機能

指定したファイル名でユーザー校正データを保存します。

引数

pCalibFileName	[IN]	保存元のユーザー校正ファイルのパス名を指定する文字列へのポインタです。（拡張子“.CLB”）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

現在、ユーザー校正データが作成されている場合、そのデータをファイルに保存します。ユーザー校正データが作成されていない場合は、この関数は失敗します。

7.2 6. s3_ExcuteUserCalib()

形式

BOOL s3_ExcuteUserCalib(S3_T_USER_CALIB_MODE *Mode*)

機能

指定したユーザー校正モードで校正処理を行います。

引数

Mode　　[IN] ユーザー校正モードを指定する変数です。
（S3_C_PLANE_WAVE_CALIB　　：平面波校正
/ S3_C_SPHERICAL_WAVE_CALIB ：球面波校正）

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

・S3_C_PLANE_WAVE_CALIB を指定した場合は、平面波校正を実行します。
・S3_C_SPHERICAL_WAVE_CALIB を指定した場合は、球面波校正を実行します。
※この関数は測定中でないと正常に実行できません。

例

```
s3_StartCapture();              //測定開始
s3_CalcAberration();            //計算実行
if(!s3_ExcuteUserCalib(S3_C_PLANE_WAVE_CALIB)) {
    //平面波モードでのユーザー校正に失敗した場合の処理
}
```

7.27. s3_GetMaskHandle()

形式

BOOL s3_GetMaskHandle(S3_T_hMASK **hMask*);

機能

現在のマスクオブジェクトへのハンドル値を取得します。

引数

hMask　　[OUT] マスクオブジェクトのハンドルを受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hMASK hMask;
if(!s3_GetMaskHandle(&hMask)) {
    //マスクハンドルの取得に失敗した場合の処理
}
```

7.2 8. s3_isMaskEmpty()

形式

BOOL s3_isMaskEmpty(S3_T_hMASK *hMask*)

機能

指定したマスクオブジェクトが要素をもつかどうかの確認を行います。

引数

なし。

戻り値

TRUE	要素をもたない
FALSE	要素をもつ

例

```
S3_T_hMASK hMask;
s3_GetMaskHandle(&hMask);
if(s3_isMaskEmpty(hMask)) {
    //マスク要素が存在しない場合の処理
}
```

7.29. s3_GetMaskCount()

形式

BOOL s3_GetMaskCount(S3_T_hMASK *hMask*, int **pNum*)

機能

指定したマスクオブジェクトの要素数を取得します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
pNum	[OUT]	マスクオブジェクトの要素数を受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hMASK hMask;
int Num;

s3_GetMaskHandle(&hMask);
if(!s3_GetMaskCount(hMask, &Num)) {     //マスク要素数の取得
    //マスク要素数の取得に失敗した場合の処理
}
```

7.30. s3_GetMaskInfo()

形式

BOOL s3_GetMaskInfo(S3_T_hMASK *hMask*, S3_T_MASK_INFO **pInfo*)

機能

指定したマスクオブジェクトのパラメータ情報を取得します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
pInfo	[OUT]	マスクパラメータ情報を受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hMASK hMask;
S3_T_MASK_INFO Info;

s3_GetMaskHandle(&hMask);
if(!s3_GetMaskInfo(hMask, &Info)) {     //マスク付加情報の取得
    //マスク付加情報の取得に失敗した場合の処理
}
```

7.3 1. s3_CalcMask()

形式

BOOL s3_CalcMask(void)

機能

現在の校正データに対して、マスクデータの初期状態を計算します。
初期状態では、マスクの各要素が以下のように分類されます。

・計測エリアの内側の校正スポット点 … マスク無効（S3_C_MSK_OFF）
・計測エリアの外側の校正スポット点 … マスク有効（S3_C_MSK_ON）
・計算不能な校正スポット点 … 対応点なし（S3_C_MSK_NO_PAIR）

（5.4.章参照）

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

現在の校正データに対して、マスクデータを作成します。デフォルト校正データ、ユーザー校正データどちらに対しても、マスクデータを計算することができます。

7.3 2. s3_UpdateMask()

形式

BOOL s3_UpdateMask(void)

機能

現在のマスクデータを更新します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・本関数を使用する前に、s3_CalcMask()関数によりマスクデータを作成しておく必要があります。

・現在、設定されているマスクデータの状態値を、測定データに従って更新します。マスク有効の状態値（S3_C_MSK_ON）を持つ要素はそのまま保持され、その他の状態値を持つ要素のみ更新されます。

7.3 3. s3_ReadMask()

形式

BOOL s3_ReadMask(char ***pMaskFileName*)

機能

指定したマスクファイルからデータを読み込み、現在設定されている校正データのマスクデータとして登録します。

引数

pMaskFileName	[IN]	マスクファイル名を指定する文字列へのポインタです。（拡張子“.ＭＳＫ”）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

マスクデータ保存時の校正データとマスクデータ読み込み時の校正データが異なる場合、上記の関数は失敗します。

7.3 4. s3_SaveMask()

形式

BOOL s3_SaveMask(char ***pMaskFileName*)

機能

指定したマスクファイル名でマスクデータを保存します。

引数

pMaskFileName　　[IN]　マスクファイル名を指定する文字列へのポインタです。
（拡張子“.ＭＳＫ”）

戻り値

TRUE　　正常終了
FALSE　　異常終了

7.3 5. s3_ClearMask()

形式

BOOL s3_ClearMask(void)

機能

現在のマスクデータから、ソフトアパーチャ内のマスク要素の状態値をすべて無効（S3_C_MSK_OFF）にします。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・本関数を使用する前に、s3_CalcMask()関数により、マスクデータを作成しておく必要があります。

7.3 6. s3_RegistMask()

形式

BOOL s3_RegistMask(void)

機能

現在の校正データにマスクデータを登録します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
s3_SetCalibMode(S3_C_STANDERD_CALIB);          //校正モードの選択
if(!s3_CalcMask()) {                           //マスクの計算
    //マスクデータの計算に失敗した場合の処理
}
if(!s3_RegistMask()) {                         //マスクデータの登録
    //マスクデータの登録に失敗した場合の処理
}
```

備考

・S3_T_OPTION 構造体の CenterType メンバが 1（追従）に設定されているとき、もしくは、AutoAreaSize メンバが 1（自動）に設定されている場合は、上記の関数は失敗します。

・マスクデータ保存時の校正データとマスクデータ読み込み時の校正データが異なる場合は、上記の関数は失敗します。

7.3 7. s3_SetMaskAt()

形式

BOOL s3_SetMaskAt(S3_T_hMASK *hMask*, int *Index*, S3_T_MASK_ELEMENT **pElement*)

機能

指定したマスクオブジェクトにマスク要素を設定します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
Index	[IN]	マスク要素のインデックスを指定する変数です。 （０～マスク要素数－１）
pElement	[IN]	マスク要素を指定する変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

・先頭から１１番目のマスク要素を設定します

```
S3_T_hMASK hMask;
S3_T_MASK_ELEMENT Element;

s3_GetMaskHandle(&hMask);          //マスクハンドル値の取得
if(!s3_GetMaskAt(hMask, 10, &Element)) {
    //マスク要素取得失敗した場合の処理
}
if(Element.State!=S3_C_MSK_ON) {
    Element.State=S3_C_MSK_ON;
}
if(!s3_SetMaskAt(hMask, 10, &Element)) {
    //マスク要素設定失敗した場合の処理
}
```

7.3 8. s3_GetMaskAt()

形式

BOOL s3_GetMaskAt(S3_T_hMASK *hMask*, int *Index*, S3_T_MASK_ELEMENT **pElement*)

機能

指定したマスクオブジェクトからマスク要素を取得します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
Index	[IN]	マスク要素のインデックスを指定する変数です。 （0～マスク要素数－1）
pElement	[OUT]	マスク要素を受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

7.3 9. s3_SetMaskList()

形式

BOOL s3_SetMaskList(S3_T_hMASK *hMask*, S3_T_MASK_ELEMENT **pList*, int *ListNum*)

機能

指定したマスクオブジェクトの要素列を設定します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
pList	[IN]	マスク要素列を指定するバッファへのポインタです。
ListNum	[IN]	マスク要素列の要素数を指定する変数です。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hMASK hMask;
int num;
S3_T_MASK_ELEMENT *pList;

if(!s3_GetMaskHandle(&hMask)) {
   //マスクハンドル値の取得に失敗した場合の処理
}
s3_GetMaskCount(hMask, &num);                    //マスク要素数の取得
pList=new S3_T_MASK_ELEMENT[num];
if(!s3_GetMaskList(hMask, pList, num)) {         //マスク要素列の取得
   //マスク要素列の取得に失敗した場合の処理
}
for(int i=0;i<num;i++) {
    pList[i].State=S3_C_MSK_OFF;                 //マスク要素の変更
}
if(!s3_SetMaskList(hMask, pList)) {              //マスク要素列の設定
   //マスク要素列の設定に失敗した場合の処理
```

```
}
delete [] pList;
```

7.40. s3_GetMaskList()

形式

BOOL s3_GetMaskList(S3_T_hMASK *hMask*, S3_T_MASK_ELEMENT **pList*)

機能

指定したマスクオブジェクトの要素列を取得します。

引数

hMask	[IN]	マスクオブジェクトのハンドルを指定します。
pList	[OUT]	マスク要素列を受け取るバッファへのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

7.4 1. s3_CreateMask()

形式

BOOL s3_CreateMask(S3_T_hMASK **hMask*)

機能

空のマスクオブジェクトを生成し、そのハンドルを取得します。

引数

hMask　　[OUT] マスクオブジェクトのハンドルを受け取る変数へのポインタです。

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
S3_T_hMASK hSrc,hDest;

s3_GetMaskHandle(&hSrc);
if(s3_CreateMask(&hDest)) {          //マスクの生成
    s3_CopyMask(hSrc, hDest);        //マスクのコピー
    //処理
    s3_ReleaseMask(&hDest);          //マスクの解放
}
```

7.4 2. s3_CopyMask()

形式

BOOL s3_CopyMask(S3_T_hMASK *hSrc*, S3_T_hMASK *hDest*)

機能

マスクオブジェクトのコピーを生成します。

引数

hSrc	[IN]	コピー元のマスクオブジェクトのハンドルを指定します。
hDest	[OUT]	コピー先のマスクオブジェクトのハンドルを指定します。

戻り値

TRUE	正常終了
FALSE	異常終了

7.4 3. s3_ReleaseMask()

形式

void s3_ReleaseMask(S3_T_hMASK **hMask*)

機能

マスクオブジェクトに割り当てられたリソースを解放します。

引数

hMask　　[IN/OUT] 解放するマスクオブジェクトのハンドルへのポインタ

戻り値

なし。

備考

s3_CreateMask()関数によりマスクオブジェクトを生成した場合、本関数によりオブジェクトを解放する必要があります。

7.44. s3_SetZernikeTermUse()

形式

BOOL s3_SetZernikeTermUse(int *Index*, BOOL *Use*)

機能

総合波面収差計算および収差マップの計算において、使用するゼルニケ係数の項を設定します。

引数

Index　[IN] ゼルニケ係数の項に対応するインデックスを指定する変数です。
（Z_2：2，Z_3：3，…，Z_{36}：36）

Use　[IN] Index で指定した項を使用するかどうかを指定する変数です。
（0：使用しない / 1：使用する）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

本パラメータは、S3_T_OPTION 構造体の UseZernikeTerm メンバによっても設定/取得できます。（7.5.章 s3_SetParam 関数/7.6.章 s3_GetParam 関数 参照）

例

・ゼルニケ係数のZ_2～Z_9までを使用する場合

```
for(int i=2;i<S3_C_MAX_ZERNIKE_NUM;i++) {
    if(i<10) {
        //Z2～Z9までの場合は有効にします。
        s3_SetZernikeTermUse(i, TRUE);
    }else{
        //Z10～Z36までの場合は無効にします。
        s3_SetZernikeTermUse(i, FALSE);
    }
}
```

7.45. s3_SetZernikeOffsetGain()

形式

BOOL s3_SetZernikeOffsetGain(S3_T_UNIT_TYPE *Unit*,
S3_T_FORMAT_TYPE *Format*, S3_T_OFS_GAIN *pOfsGain*)

機能

ゼルニケ係数Z_2～Z_{36}の各項に、オフセット・ゲイン補正処理を適用するかどうかを選択し、補正パラメータを設定します。
オフセット・ゲイン補正を適用すると、各ゼルニケ係数は次のように変換されます。

Z_i＝pOfsGain[i].Gain×Z_i＋pOfsGain[i].Offset（ｉ＝2，3，…，36）

引数

Unit	[IN]	オフセットパラメータの単位を指定する変数です。 （S3_C_UNIT_LAMBDA：λ/ S3_C_UNIT_MICRO：μm）
Format	[IN]	オフセットパラメータの形式を指定する変数です。 （S3_C_FMT_PV：λ/ S3_C_FMT_RMS：μm）
pOfsGain	[IN]	オフセット・ゲインパラメータを指定するバッファへの ポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・引数 OfsGain は、（S3_T_OFS_GAIN 構造体のメモリサイズ）
×S3_C_MAX_ZERNIKE_NUM のサイズをもつバッファへのポインタとなります。
・格納順序は、OfsGain[i]：ゼルニケ係数Z_i項に対する補正値となります。
（ｉ＝0，1，…，36）先頭のZ_0，Z_1の補正値は無視されます。
・オフセット・ゲイン補正機能は、デフォルトで無効に設定されています。
本機能を有効にする場合は、pOfsGain 引数の Use メンバを有効に設定してください。

例

・Z_2～Z_{36}にオフセット・ゲイン補正を適用する場合

```
S3_T_OFS_GAIN OfsGain[S3_C_MAX_ZERNIKE_NUM];
```

```
for(int i=2;i<S3_C_MAX_ZERNIKE_NUM;i++) {
    OfsGain[i].Use=TRUE;            //Z項のオフセット・ゲイン補正機能を有効にします
    OfsGain[i].Gain=1.0;            //Z項のゲインパラメータの設定
    OfsGain[i].Offset=-0.12;        //Z項のオフセットパラメータの設定
}
if(!s3_SetZernikeOffsetGain(
    S3_C_UNIT_LAMBDA, S3_C_FMT_PV, OfsGain)) {
    //オフセット・ゲイン補正パラメータの設定に失敗した場合の処理
}
```

7.4 6. s3_GetZernikeOffsetGain()

形式

BOOL s3_GetZernikeOffsetGain(S3_T_UNIT_TYPE *Unit*,
S3_T_FORMAT_TYPE *Format*, S3_T_OFS_GAIN **pOfsGain*)

機能

ゼルニケ係数Z_2～Z_{36}の各項のオフセット・ゲイン補正パラメータを取得します。

引数

Unit	[IN]	オフセットパラメータの単位を指定する変数です。 （S3_C_UNIT_LAMBDA：λ/ S3_C_UNIT_MICRO：μm）
Format	[IN]	オフセットパラメータの形式を指定する変数です。 （S3_C_FMT_PV：λ/ S3_C_FMT_RMS：μm）
pOfsGain	[OUT]	オフセット・ゲインパラメータを受け取るバッファへのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・引数 OfsGain は、（S3_T_OFS_GAIN 構造体のメモリサイズ）
×S3_C_MAX_ZERNIKE_NUM のサイズをもつバッファへのポインタとなります。
・格納順序は、OfsGain[i]：ゼルニケ係数Z_i項に対する補正値となります。
（i＝0，1，…，36）

例

```
S3_T_OFS_GAIN OfsGain[S3_C_MAX_ZERNIKE_NUM];

if(!s3_GetZernikeOffsetGain(
    S3_C_UNIT_LAMBDA, S3_C_FMT_RMS, OfsGain)) {
    //オフセット・ゲイン補正パラメータの取得に失敗した場合の処理
}
```

7.47. s3_CalcAberrationMap()

形式

BOOL s3_CalcAberrationMap(double **pZernikeCoefficient*,
int *Size*, double **pAberration*, double **pMin*, double **pMax*);

機能

２次元波面収差マップを計算します。

引数

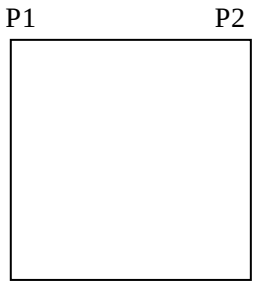
pZernikeCoefficient	[IN]	収差マップの計算に用いるゼルニケ係数を指定するバッファへのポインタです。（Z_0～Z_{36}）
Size	[IN]	波面収差マップの一つの次元のサイズを指定する変数です。（2～1000）
pAberration	[OUT]	２次元波面収差マップデータを受け取るバッファへのポインタです。
pMin	[OUT]	pAberration の最小値を受け取る変数へのポインタです。（求める必要のない場合はNULLを指定します）
pMax	[OUT]	pAberration の最大値を受け取る変数へのポインタです。（求める必要のない場合はNULLを指定します）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・pZernikeCoefficientの先頭のZ_0，Z_1の値については無視されます。
・pAberration の格納データは、収差マップの 1 行目から順に N 行目までのデータが格納されます。
・pAberration の要素数は Size×Size となります。
・N行 M 列上の要素の収差マップデータ値は、pAberration[(M−1)+(N−1)*Size]となります。



P1 … pAberration[0]，座標値(-1,1)
P2 … pAberration[Size−1]，座標値(1,1)
P3 … pAberration[Size×(Size−1)]，座標値(-1,-1)
P4 … pAberration[Size×Size−1]，座標値(1,-1)

例

```
S3_T_MEAS_INFO Info;
int MapSize=100;
double *pMap;
double Min,Max;

s3_CalcAberration();                          //収差計算実行
if(s3_GetMeasData(&Data)) {                   //測定データの取得
    pMap=new double[MapSize*MapSize];         //収差マップ格納バッファの取得
    if(!s3_CalcAberrationMap(Info.Aberration.Zernike,
        MapSize, pMap, &Min, &Max)) {         //収差マップの計算実行
        //2D 収差マップの計算に失敗した場合の処理
    }
    //処理
    delete [] pMap;                           //収差マップ格納バッファの解放
}
```

7.48. s3_CalcIntensityMap()

形式

BOOL s3_CalcIntensityMap(int *Size*, int **pIntensity*, int **pMin*, int **pMax*);

機能

２次元強度マップを計算します。

引数

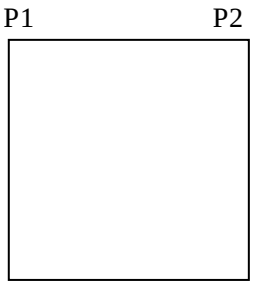
Size	[IN]	強度マップの一つの次元のサイズを指定する変数です。（2～1000）
pIntensity	[OUT]	２次元強度マップデータを受け取るバッファへのポインタです。
pMin	[OUT]	pIntensity の最小値を受け取る変数へのポインタです。（求める必要のない場合はNULLを指定します）
pMax	[OUT]	pIntensity の最大値を受け取る変数へのポインタです。（求める必要のない場合はNULLを指定します）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・pIntensity の格納データは強度マップの 1 行目から順に N 行目までのデータが格納されます。
・pIntensity の要素数は Size×Size となります。
・N行 M 列上の要素の強度マップデータ値は、pIntensity[(M−1)+(N−1)*Size]となります。
・pIntensity の各要素のとりえる値の範囲は、0～255 になります。



P1 … pIntensity[0]，座標値(-1.8, 1.8)
P2 … pIntensity[Size−1]，座標値(1.8, 1.8)
P3 … pIntensity[Size×(Size−1)]，座標値(-1.8, -1.8)
P4 … pIntensity[Size×Size−1]，座標値(1.8, -1.8)

7.49. s3_GetSpotPeakValue()

形式

BOOL s3_GetSpotPeakValue(int **pPeak*)

機能

現在のビームスポットの最大階調値を取得します。

引数

pPeak　　[OUT] 最大階調値を受け取る変数へのポインタです。（0～255）

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
int Peak;

s3_StartCapture();                          //測定開始
s3_CalcAberration();                        //収差計算実行
if(!s3_GetSpotPeakValue(&Peak)) {           //階調値の取得
    //最大階調値の取得に失敗した場合の処理
}
```

7.50. s3_CountSaturationPixels()

形式

BOOL s3_CountSaturationPixels(int **pNum*)

機能

現在のビームスポットの飽和ピクセル数を取得します。

引数

pNum　　[OUT] 飽和ピクセル数を受け取る変数へのポインタです。
（0～921600）

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
int Num;

s3_StartCapture();                          //測定開始
s3_CalcAberration();                        //収差計算実行
if(!s3_CountSaturationPixels(&Num)) {       //飽和ピクセル数の取得
    //飽和ピクセル数の取得に失敗した場合の処理
}
```

7.5 1. s3_GetSpotNum()

形式

BOOL s3_GetSpotNum(int **pNum1*, int **pNum2*)

機能

現在のビームスポットの数を取得します。

引数

pNum1　[OUT] ソフトアパーチャ内のスポット数を受け取る変数へのポインタです。

pNum2　[OUT] 全スポット数を受け取る変数へのポインタです。

戻り値

TRUE　正常終了

FALSE　異常終了

例

```
int Num1,Num2;

s3_StartCapture();                          //測定開始
s3_CalcAberration();                        //収差計算実行
if(!s3_GetSpotNum(&Num1, &Num2)) {          //ビームスポット数を取得
    //ビームスポット数の取得に失敗した場合の処理
}
```

7.5 2. s3_GetBeamProfile()

形式

BOOL s3_GetBeamProfile(S3_T_RECT **pRect*, int **pProfX*, int **pProfY*)

機能

現在の垂直方向および水平方向のビームプロファイルを取得します。

引数

pRect	[IN]	計算対象となる矩形エリアを指定する変数へのポインタです。
pProfX	[OUT]	垂直方向に投射したビームプロファイルを受け取るバッファへのポインタです。 （求める必要のない場合はNULLを指定します）
pProfY	[OUT]	水平方向に投射したビームプロファイルを受け取るバッファへのポインタ （求める必要のない場合はNULLを指定します）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・pProfX，pProfY のバッファの要素数は、それぞれ、pRect->width、pRect->height で指定されたサイズになります。ただし、pRect に NULL が指定された場合は、フレームの水平ピクセル数と垂直ピクセル数がそれぞれのサイズになります。

例

```
S3_T_hFRAME hFrame;
S3_T_SIZE Size;
int *pProfX,*pProfY;

s3_GetCapturedFrameHandle(&hFrame);          //フレームのハンドル値取得
s3_GetFrameSize(&Size);                      //フレームサイズの取得
pProfX=new int[Size.cx];                     //プロファイル格納バッファ生成
pProfY=new int[Size.cy];
```

```
if(!s3_GetBeamProfile(hFrame, NULL, pProfX, pProfY)) {
    //ビームプロファイルの取得に失敗した場合の処理
}
//処理
delete [] pProfX;                                    //プロファイル格納バッファ解放
delete [] pProfY;
```

7.5 3. s3_GetAutoAreaSize()

形式

BOOL s3_GetAutoAreaSize(double **pSize*, S3_T_dPOINT **pCenter*)

機能

計測エリアサイズの自動決定モード時の計測エリアサイズを取得します。

引数

pSize	[OUT]	計測エリアサイズ[mm]を受け取る変数へのポインタです。
pCenter	[OUT]	計測エリアの中心位置[mm]を受け取る構造体へのポインタです。（求める必要のない場合は NULL を指定します。）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・S3_T_OPTION 構造体の AutoAreaSize メンバを１（自動）に設定した場合、計測エリアサイズは、実際のビームサイズに合わせて調整されますが、本関数でこのときの計測エリアサイズを取得できます。

・本関数は S3_T_OPTION 構造体の AreaType メンバが 0（円形）のときは、引数 pSize に計測エリアの直径を返します。AreaType メンバが 1（矩形）のときは、引数 pSize に計測エリアの幅および高さを返します。

例

```
double Size;
S3_T_dPOINT Center;

if(!s3_GetAutoAreaSize(&Size, &Center)) {
    //計測エリアサイズの取得に失敗した場合の処理
}
```

7.5 4. s3_CalcAberration()

形式

BOOL s3_CalcAberration(void)

機能

波面収差を計算します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・この関数を実行する前に、StartCapture()関数を実行して測定を開始しておく必要があります。また、本関数を実行して得た測定結果を取得するには、s3_GetMeasData()関数を使用します。

例

```
s3_StartCapture();                    //測定開始
if(!s3_CalcAberration()) {            //収差計算実行
    //収差計算に失敗した場合の処理
}
```

7.5 5. s3_GetMeasData()

形式

BOOL s3_GetMeasData(S3_T_MEAS_INFO **pData*)

機能

波面収差の測定結果を取得します。

引数

pData　　[OUT] 測定情報を受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・本関数を実行して測定値を取得する前に、予め s3_CalcAberration()関数を実行して測定値を計算しておく必要があります。
・本関数では、ゼルニケ収差測定値、ザイデル収差測定値、総合波面収差測定値の各測定値を一度に取得します。それぞれを個別に取得したい場合は、次項の s3_GetZernikeAberration(), s3_GetSeidelAberration(), s3_GetTotalAberration()関数を使用してください。

例

```
S3_T_MEAS_INFO Info;

s3_StartCapture();                          //測定開始
if(s3_CalcAberration()) {                   //収差計算の実行
    if(!s3_GetMeasData(&Info)) {            //測定結果の取得
        //計算結果の取得に失敗した場合の処理
    }
}
```

7.5 6. s3_GetZernikeAberration()

形式

BOOL s3_GetZernikeAberration(double **pData*)

機能

ゼルニケ収差の測定結果を取得します。

引数

pData　　[OUT] ゼルニケ収差測定値を受け取るバッファへのポインタです。
（Z_0～Z_{36}）

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

・本関数を実行して測定値を取得する前に、予め s3_CalcAberration()関数を実行して測定値を計算しておく必要があります。
・pDataバッファの格納順序は、pData[i]：ゼルニケ係数Z_i項となります。
（i＝0，1，…，36）Z_0，Z_1は無効な要素になります。

例

```
double Data[S3_C_MAX_ZERNIKE_NUM];

s3_StartCapture();                                  //測定開始
if(s3_CalcAberration()) {                           //収差計算の実行
    if(!s3_GetZernikeAberration(Data)) {            //ゼルニケ収差測定結果の取得
        //計算結果の取得に失敗した場合の処理
    }
}
```

7.5 7. s3_GetSeidelAberration()

形式

BOOL s3_GetSeidelAberration(S3_T_SEIDEL_ABERRATION **pData*)

機能

ザイデル収差の測定結果を取得します。

引数

pData　　[OUT] ザイデル収差測定値を受け取る構造体へのポインタです。

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

本関数を実行して測定値を取得する前に、予め s3_CalcAberration()関数を実行して測定値を計算しておく必要があります。

例

```
S3_T_SEIDEL_ABERRATION Data;

s3_StartCapture();                          //測定開始
if(s3_CalcAberration()) {                   //収差計算の実行
    if(!s3_GetSeidelAberration(&Data)) {    //ザイデル収差測定結果の取得
        //計算結果の取得に失敗した場合の処理
    }
}
```

7.5 8. s3_GetTotalAberration()

形式

BOOL s3_GetTotalAberration(S3_T_TOTAL_ABERRATION **pData*)

機能

総合波面収差の測定結果を取得します。

引数

pData　　[OUT] 総合波面収差測定値を受け取る構造体へのポインタです。

戻り値

TRUE　　正常終了
FALSE　　異常終了

備考

本関数を実行して測定値を取得する前に、予め s3_CalcAberration()関数を実行して測定値を計算しておく必要があります。

例

```
S3_T_TOTAL_ABERRATION Data;

s3_StartCapture();                          //測定開始
if(s3_CalcAberration()) {                   //収差計算の実行
    if(!s3_GetTotalAberration(&Data)) {     //総合波面収差測定結果の取得
        //計算結果の取得に失敗した場合の処理
    }
}
```

7.5 9. s3_GetCapturedFrameHandle()

形式

BOOL s3_GetCapturedFrameHandle(S3_T_hFRAME **hFrame*)

機能

現在のキャプチャフレームオブジェクトへのハンドルを取得します。

引数

hFrame　　[OUT] キャプチャフレームオブジェクトのハンドルを受け取る変数へのポインタです。

戻り値

TRUE　　正常終了
FALSE　　異常終了

例

```
S3_T_hFRAME hFrame;
if(!s3_GetCapturedFrameHandle(&hFrame)) {
    //フレームのハンドル値の取得に失敗した場合の処理
}
```

7.60. s3_GetFrameBmpInfo()

形式

BOOL s3_GetFrameBmpInfo(S3_T_hFRAME *hFrame*, BITMAPINFO **pInfo*)

機能

指定したキャプチャフレームオブジェクトからビットマップ情報を取得します。

引数

hFrame	[IN]	キャプチャフレームオブジェクトへのハンドルを指定します。
pInfo	[OUT]	ビットマップ情報を受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hFRAME hFrame;
BITMAPINFO *pInfo;

s3_GetCapturedFrameHandle(hFrame);              //フレームハンドル値の取得
if(!s3_GetFrameBmpInfo(hFrame, &pInfo)) {       //ビットマップ情報の取得
    //ビットマップ情報の取得に失敗した場合の処理
}
```

7.6 1. s3_GetFrameSize()

形式

BOOL s3_GetFrameSize(S3_T_SIZE **pSize*)

機能

指定したキャプチャフレームオブジェクトから、フレームのサイズを取得します。

引数

hFrame	[IN]	キャプチャフレームオブジェクトへのハンドルを指定します。
pSize	[OUT]	フレームサイズを受け取る構造体へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_SIZE Size;

if(!s3_GetFrameSize(&Size)) {
    //フレームサイズの取得に失敗した場合の処理
}
```

7.6.2. s3_GetFramePixel()

形式

BOOL s3_GetFramePixel(S3_T_hFRAME *hFrame*, BYTE **pPixels*)

機能

指定したキャプチャフレームオブジェクトから、ピクセルデータを取得します。

引数

hFrame	[IN]	キャプチャフレームオブジェクトのハンドルを指定します。
pPixels	[OUT]	ピクセルデータを受け取るバッファへのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

例

```
S3_T_hFRAME hFrame;
BYTE *pPixels;

s3_GetCapturedFrameHandle(&hFrame);
if(!s3_GetFramePixel(hFrame, &pPixels)) {
    //ピクセルデータの取得に失敗した場合の処理
}
```

7.6 3. s3_GetFramePixelBufferSize()

形式

BOOL s3_GetFramePixelBufferSize(S3_T_hFRAME *hFrame*, unsigned long * *pSize*)

機能

指定したキャプチャフレームオブジェクトから、ピクセルデータのバッファサイズを取得します。

引数

hFrame	[IN]	指定するキャプチャフレームオブジェクトのハンドル
pSize	[OUT]	ピクセルデータのバッファサイズを受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

7.6 4. s3_isMeasuring()

形式

BOOL s3_isMeasuring(void)

機能

現在、測定中かどうかの確認をします。

引数

なし。

戻り値

TRUE	測定中
FALSE	非測定中

例

```
if(s3_isMeasuring()) {
    //測定中の場合の処理
}
```

7.6 5. s3_isCalibDone()

形式

BOOL s3_isCalibDone(void)

機能

現在、校正が完了しているかどうかの確認をします。

引数

なし。

戻り値

TRUE	校正完了
FALSE	校正未完了

例

```
if(s3_isCalibDone()) {
    //校正が完了している場合の処理
}
```

7.6 6. s3_isUserCalibDone()

形式

BOOL s3_isUserCalibDone(void)

機能

現在、ユーザー校正が完了しているかどうかの確認をします。

引数

なし。

戻り値

TRUE	校正完了
FALSE	校正未完了

7.67. s3_isMaskDone()

形式

BOOL s3_isMaskDone(void)

機能

現在の校正データに対して、マスクデータが登録されているかどうかの確認をします。

引数

なし。

戻り値

TRUE	登録済み
FALSE	未登録

備考

マスクデータが登録されていても、S3_T_OPTION 構造体の UseMask 設定値が 1 に設定されていないと、マスクデータは測定に使用されません。

7.68. s3_isSnapshotEmpty()

形式

BOOL s3_isSnapshotEmpty(void)

機能

スナップショットデータが存在するかどうかの確認を行います。

引数

なし。

戻り値

TRUE	存在しない
FALSE	存在する

例

```
if(s3_isSnapshotEmpty()) {
    //スナップショットデータが存在しない場合の処理
}
```

7.69. s3_TakeSnapshot()

形式

BOOL s3_TakeSnapshot(void)

機能

現在のフレーム画像のスナップショットを取得します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

本関数を実行してスナップショットを取得する前に、予め s3_StartCapture()関数を実行して測定を開始しておく必要があります。

例

```
char filename[]="sample.bmp";

s3_StartCapture();                          //測定開始
if(!s3_TakeSnapshot()) {                    //スナップショット画像の取得
    //スナップショット画像の取得に失敗した場合の処理
}
if(!s3_SaveSnapshot(filename)) {            //スナップショット画像の保存
    //スナップショット画像の保存に失敗した場合の処理
}
```

7.70. s3_GetSnapshotHandle()

形式

BOOL s3_GetSnapshotHandle(S3_T_hFRAME **hFrame*)

機能

現在のスナップショットオブジェクトへのハンドル値を取得します。

引数

hFrame　　[OUT] スナップショットオブジェクトのハンドルを受け取る変数へのポインタです。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

スナップショットデータから、ビットマップ情報やピクセル情報を取得するためのハンドル値を取得します。

例

```
S3_T_hFRAME hFrame;
BITMAPINFO *pInfo;
BYTE *pPixels;

if(!s3_GetSnapshotHandle(&hFrame)) {          //スナップショットハンドルの取得
    //スナップショットハンドルの取得に失敗した場合の処理
}
if(!s3_GetFrameBmpInfo(hFrame, &pInfo)) {     //ビットマップ情報の取得
    //ビットマップ情報の取得に失敗した場合の処理
}
if(!s3_GetFramePixel(hFrame, &pPixels)) {     //ピクセルデータの取得
    //ピクセルデータの取得に失敗した場合の処理
}
```

7.7 1. s3_CalcSnapAberration()

形式

BOOL s3_CalcSnapAberration(void)

機能

スナップショット画像から波面収差を計算します。

引数

なし。

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・次項の s3_ReadSnapshot()関数で読み込まれたスナップショット画像データに対し、波面収差を計算します。
・この関数を実行する前に、s3_StartCapture()関数を実行して測定を開始しておく必要があります。また、本関数を実行して得た測定結果を取得するには、s3_GetMeasData()関数を使用します。

例

```
char filename[]="sample.bmp";
S3_T_MEAS_INFO Info;

if(!s3_ReadSnapshot(filename)) {          //スナップショット画像の読み込み
    //スナップショット画像の読み込みに失敗した場合の処理
}
s3_StartCapture();                        //測定開始
if(!s3_CalcSnapAberration()) {            //収差計算実行
    //収差計算に失敗した場合の処理
}
if(!s3_GetMeasData(&Info)) {              //測定結果の取得
    //計算結果の取得に失敗した場合の処理
```

```
}
```

7.7 2. s3_ReadSnapshot()

形式

BOOL s3_ReadSnapshot(char ***pSnapshotFileName*)

機能

指定したスナップショットファイルから、フレーム画像データを読み込みます。

引数

pSnapshotFileName [IN] 読み込み元のスナップショットファイルのパス名を指定する文字列へのポインタです。
（拡張子 “ .BMP ” または “ .PNG ” ）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

次項の、s3_SaveSnapshot()関数で保存された、フレーム画像データを読み込みます。

7.7 3. s3_SaveSnapshot()

形式

BOOL s3_SaveSnapshot(char ***pSnapshotFileName*)

機能

指定したファイル名で、フレーム画像のスナップショットを保存します。
s3_TakeSnapshot()関数を実行した瞬間のフレーム画像を、画像ファイルとして保存します。

引数

pSnapshotFileName [IN] 保存先のスナップショットファイルのパス名を指定する文字列へのポインタです。
（拡張子“.BMP”または“.PNG”）

戻り値

TRUE	正常終了
FALSE	異常終了

備考

・スナップショット画像が存在しない時（s3_isSnapshotEmpty()関数の戻り値が 1 の時）、この関数は失敗します。
・指定可能なファイルフォーマットは、BMP と PNG の 2 種類になります。
ファイルフォーマットは、指定したファイル名の拡張子により判別されます。

8. 定数

8.1. S3_C_MAX_ZERNIKE_NUM

形式

```
#define S3_C_MAX_ZERNIKE_NUM        (37)
```

概要

最大ゼルニケ多項式係数項目数

8.2. S3_C_MAX_PATH_LEN

形式

```
#define S3_C_MAX_PATH_LEN           (260)
```

概要

最大パス文字列バッファ長

8.3. S3_T_CALIB_MODE

形式

```
typedef enum {
    S3_C_STANDERD_CALIB=0,
    S3_C_USER_CALIB=1,
} S3_T_CALIB_MODE;
```

概要

校正モードを表す定数

メンバ

S3_C_STANDERD_CALIB　…　デフォルト校正
S3_C_USER_CALIB　…　ユーザー校正

8.4. S3_T_USER_CALIB_MODE

形式

```
typedef enum {
    S3_C_PLANE_WAVE_CALIB=0,
    S3_C_SPHERICAL_WAVE_CALIB=1,
} S3_T_USER_CALIB_MODE;
```

概要

ユーザー校正モードを表す定数

メンバ

S3_C_PLANE_WAVE_CALIB	…	平面波校正
S3_C_SPHERICAL_WAVE_CALIB	…	球面波校正

8.5. S3_T_UNIT_TYPE

形式

```
typedef enum {
    S3_C_UNIT_LAMBDA=0,
    S3_C_UNIT_MICRO=1,
} S3_T_UNIT_TYPE;
```

概要

測定値の出力単位モードを表す定数

メンバ

S3_C_UNIT_LAMBDA	…	λ
S3_C_UNIT_MICRO	…	μm

8.6. S3_T_ANG_UNIT_TYPE

形式

```
typedef enum {
    S3_C_ANG_DEG=0,
    S3_C_ANG_MRAD=1,
    S3_C_ANG_MIN=2,
} S3_T_ANG_UNIT_TYPE;
```

概要

測定値の角度出力単位モードを表す定数

メンバ

S3_C_ANG_DEG	…	度
S3_C_ANG_MRAD	…	ミリラジアン
S3_C_ANG_MIN	…	分

8.7. S3_T_FORMAT_TYPE

形式

```
typedef enum {
    S3_C_FMT_PV=0,
    S3_C_FMT_RMS=1,
} S3_T_FORMAT_TYPE;
```

概要

測定値の出力形式モードを表す定数

メンバ

S3_C_FMT_PV	…	ＰＶ
S3_C_FMT_RMS	…	ＲＭＳ

8.8. S3_T_MASK_STATE

形式

```
typedef enum {
    S3_C_MSK_OFF=0,
    S3_C_MSK_ON=1,
    S3_C_MSK_NO_PAIR=2,
} S3_T_MASK_STATE;
```

概要

マスク状態値を表す定数

メンバ

S3_C_MSK_OFF	…	マスク無効
S3_C_MSK_ON	…	マスク有効
S3_C_MSK_NO_PAIR	…	マスク有効（対応点なし）

9. 構造体

9.1. S3_T_OPTION

形式

```
typedef struct _S3_T_OPTION {
        struct T_CAMERA {
                double Gain;
                double Shutter;
        } CAMERA;
        struct T_LIGHT {
                double Lambda;
                int OpticalPath;
        } LIGHT;
        struct T_IMAGE {
                BOOL UseFrameAve;
                int FrameAveNum;
                int CenterType;
                S3_T_dPOINT AreaCenter;
                int AreaType;
                int BinLevel;
                double AreaDiameter;
                S3_T_dSIZE AreaSize;
                BOOL AutoAreaSize;
                BOOL UseMask;
        } IMAGE;
        struct T_OUTPUT {
                int UnitType;
                int UnitFormat;
                int AngUnitType;
        } OUTPUT;
        struct T_CALIB {
                int CalibMode;
                double Lambda;
                double PropagationDistance;
```

```
    } CALIB;
    struct T_CALC {
        BOOL UseZernikeTerm[S3_C_MAX_ZERNIKE_NUM];
    } CALC;
} S3_T_OPTION;
```

概要

光学ヘッドの制御および測定に必要なパラメータ値を格納します。
この構造体で与えられた設定値を API に反映するには、s3_SetParam()関数を使用します。

メンバ

・CAMERA構造体

Gain　光学ヘッドのゲイン値[dB]を格納します。
ゲイン値を高くすることで、光学ヘッドの感度は上がりますが、その分ノイズが増え測定精度が悪化します。通常はこの値を 0 に設定して下さい。
- 設定範囲：　0．00～24．00（0．05dB 単位）
- 初期値：　0．00

Shutter　光学ヘッドのシャッタースピード値[ms]を格納します。
シャッタースピードを短く指定すると、ショットノイズの影響により測定値がバラツキます。逆に長く指定すると振動の影響を受けやすくなります。
- 設定範囲：　0．01～16000．00（0．01ms 単位）
- 初期値：　9．00

・LIGHT構造体

Lambda　測定λ値[nm]を格納します。
測定時の入射ビームの波長を指定します。
- 設定範囲：　400．0～1000．0
- 初期値：　658．0

OpticalPath　光路タイプを格納します。
- 設定範囲：　0：シングルパス，1：ダブルパス
- 初期値：　0

・IMAGE構造体

UseFrameAve　　フレーム平均処理有無を格納します。
フレーム平均処理を有効にするにはこの設定値を 1 に設定してください。
・設定範囲：　0：無，1：有
・初期値：　　1

FrameAveNum　　フレーム平均化回数を格納します。
フレーム平均処理を適用すると、フレーム重ね合わせ処理により個々のフレームのノイズが軽減され、安定した測定が可能となります。ノイズの軽減量はフレーム平均処理の適用回数が多いほど大きくなりますが、一方で、あまり回数を増やすと振動の影響を受けやすくなります。
・設定範囲：　1～150
・初期値：　　9

CenterType　　計測中心モードを格納します。
計測中心を「固定」と「追従」のどちらにするかを選択します。
・設定範囲：　0：固定，1：追従
・初期値：　　0

AreaCenter　　計測中心座標[mm]を格納します。
計測中心のＸ座標とＹ座標を設定します。フレームの中心が原点（0，0）になります。
・設定範囲：　－1．00～1．00
・初期値：　　（0．00，0．00）

AreaType　　計測エリアモードを格納します。
収差計算の対象となるエリアの形状を選択します。円形と矩形の２種類の形状が選択できます。
・設定範囲：　0：円形，1：矩形
・初期値：　　0

BinLevel　　２値化レベルを格納します。

ここで設定した値以下のピーク値を持つスポットは、計算から除外されます。スポットのベースラインより高い位置に２値化レベルを設定して下さい。

・設定範囲：　1～255
・初期値：　　40

AreaDiameter　　計測エリア直径[mm]を格納します。
AreaType メンバを 0 に設定した場合、ここで設定した値を直径とする円形エリアの内側が計測エリアになります。

・設定範囲：　1．80～6．00
・初期値：　　3．00

AreaSize　　計測エリアサイズ[mm]を格納します。
AreaType メンバを１に設定した場合、ここで設定した値を幅および高さとする矩形エリアの内側が計測エリアになります。

・設定範囲：　1．30～3．60
・初期値：　　3．00

AutoAreaSize　　自動エリアサイズ調整モードを格納します。
この値を１に指定すると、計測エリアサイズをビームサイズに合わせて自動的に決定します。０に設定した場合は、AreaDiameter メンバ、あるいは AreaSize メンバで指定したサイズが使用されます。

・設定範囲：　0：手動，1：自動
・初期値：　　0

UseMask　　マスク処理有無を格納します。

・設定範囲：　0：無，1：有
・初期値：　　0

※ CenterType メンバが 1（追従）に設定されているとき、または、AutoAreaSize メンバが 1（自動）に設定されているときは、マスク処理を使用することができません。

・OUTPUT構造体

UnitType　出力単位モードを格納します。
- 設定範囲：　0：λ，1：μm
- 初期値：　0

UnitFormat　出力形式モードを格納します。
- 設定範囲：　0：PV，1：RMS
- 初期値：　0

AngUnitType　出力角度単位モードを格納します。
- 設定範囲：　0：deg，1：mrad，2：分
- 初期値：　0

・CALIB 構造体

CalibMode　校正モードを格納します。
校正モードを「デフォルト校正」と「ユーザー校正」のどちらにするか選択します。
- 設定範囲：　0：デフォルト校正，1：ユーザー校正
- 初期値：　0

Lambda　校正λ値[nm]を格納します。
校正時の入射ビームの波長を指定します。
- 設定範囲：　400．0～1000．0
- 初期値：　658．0

PropagationDistance　伝播距離[mm]を格納します。
- 設定範囲：　100．0～100000．0
- 初期値：　811．0

・CALC 構造体

UseZernikeTerm[]　ゼルニケ係数の収差計算使用有無を格納します。
ここで“有効”に指定されたゼルニケ係数を使用して、総合波面収差計算および収差マップの計算を行います。
- 設定範囲：　0：無，1：有
- 初期値：　1（i＝2，3，…，36）

※UseZernikeTerm配列の格納順序は、UseZernikeTerm[i]：ゼルニケ係数Z_i項の収差計算有無となります。（ｉ＝0，1，…，36）

9.2. S3_T_dPOINT

形式

```
typedef struct _S3_T_dPOINT {
        double x;
        double y;
} S3_T_dPOINT;
```

概要

double 型の座標を格納します。

メンバ

x	x 座標
y	y 座標

9.3. S3_T_SIZE

形式

```
typedef struct _S3_T_SIZE {
        int cx;
        int cy;
} S3_T_SIZE;
```

概要

int 型のサイズを格納します。

メンバ

cx	幅
cy	高さ

9.4. S3_T_dSIZE

形式

```
typedef struct _S3_T_dSIZE {
        double cx;
        double cy;
} S3_T_dSIZE;
```

概要

double 型のサイズを格納します。

メンバ

cx	幅
cy	高さ

9.5. S3_T_RECT

形式

```
typedef struct _S3_T_RECT {
        int x;
        int y;
        int width;
        int height;
} S3_T_RECT;
```

概要

int 型の矩形情報を格納します。

メンバ

x	int 型の矩形領域の原点（左上隅）の x 座標を格納します。
y	int 型の矩形領域の原点（左上隅）の y 座標を格納します。
width	int 型の矩形領域の幅を格納します。
height	int 型の矩形領域の高さを格納します。

9.6. S3_T_OFS_GAIN

形式

```
typedef struct _S3_T_OFS_GAIN {
        double Gain;
        double Offset;
        BOOL Use;
} S3_T_OFS_GAIN;
```

概要

ゼルニケ係数の各項のオフセット・ゲイン補正値を格納します。

メンバ

Gain	ゲイン値を格納します。
Offset	オフセット値を格納します。
Use	オフセット・ゲイン処理有無を格納します。（0：無，1：有）

備考

この情報を設定/取得するには、s3_SetZernikeOffsetGain()関数，
s3_GetZernikeOffsetGain()関数を使用します。

9.7. S3_T_MEAS_INFO

形式

```
typedef struct _S3_T_MEAS_INFO {
        BOOL Status;
        int SpotNum;
        int PairNum;
        int PeakVal;
        int Saturation;
        S3_T_dPOINT BeamCenter;
        S3_T_ABERRATION Aberration;
} S3_T_MEAS_INFO;
```

概要

測定結果を格納します。

メンバ

Status	ステータス（0：計算失敗，1：計算成功）
SpotNum	ビームスポット数
PairNum	収差計算に使用するビームスポット数
PeakVal	最大階調値（0～255）
Saturation	飽和ピクセル数（0～921600）
BeamCenter	ビーム中心座標（単位[mm]）
Aberration	波面収差測定値

備考

この情報を取得するには、s3_ GetMeasData()関数を使用します。

9.8. S3_T_ABERRATION

形式

```
typedef struct _S3_T_ABERRATION {
        double Zernike[S3_C_MAX_ZERNIKE_NUM];
        S3_T_SEIDEL_ABERRATION Seidel;
        S3_T_TOTAL_ABERRATION Total;
} S3_T_ABERRATION;
```

概要

波面収差測定値を格納します。

メンバ

Zernike	ゼルニケ収差測定値
Seidel	ザイデル収差測定値
Total	総合波面収差測定値

備考

・ゼルニケ収差測定値の格納順序は、Zernike[i]：Z_i （i＝0，1，…，36）となります。Z_0，Z_1は無効な要素になります。

・ゼルニケ収差測定値の出力単位は、S3_T_OPTION 構造体の UnitType メンバと UnitFormat メンバで設定された単位となります。

9.9. S3_T_SEIDEL_ABERRATION

形式

```
typedef struct _S3_T_SEIDEL_ABERRATION {
        double Tilt;
        double Focus;
        double AS;
        double Coma;
        double SA;
        double Tilt_Angle;
        double AS_Angle;
        double Coma_Angle;
} S3_T_SEIDEL_ABERRATION;
```

概要

ザイデル収差測定値を格納します。

メンバ

Tilt	ザイデル Tilt マグニチュード項
Focus	ザイデル Focus マグニチュード項
AS	ザイデル AS マグニチュード項
Coma	ザイデル Coma マグニチュード項
SA	ザイデル SA マグニチュード項
Tilt_Angle	ザイデル Tilt アングル項
AS_Angle	ザイデル AS アングル項
Coma_Angle	ザイデル Coma アングル項

備考

・ザイデル収差測定値のマグニチュード項の出力単位は、S3_T_OPTION 構造体の UnitType メンバと UnitFormat メンバで設定された単位となります。

・ザイデル収差測定値のアングル項の出力単位は、“ 度 ” となります。

9.10. S3_T_TOTAL_ABERRATION

形式

```
typedef struct _S3_T_TOTAL_ABERRATION {
        double PV;
        double RMS;
        double Pwr;
        double RoC;
        double Rho;
        double Strehl;
        struct T_ANGLE {
                double x;
                double y;
                double D;
        } Angle;
} S3_T_TOTAL_ABERRATION;
```

概要

総合波面収差測定値を格納します。

メンバ

PV	総合波面収差 PV 値
RMS	総合波面収差 RMS 値
Pwr	Pwr 値（単位[λ PV]または[μm PV]）
RoC	波面曲率半径 RoC 値（単位[mm]）
Rho	波面曲率ρ値（単位[1/m]）
Strehl	ストレール比
x	入射角度 x 成分
y	入射角度 y 成分

D　　入射角度 距離 D

備考

・PV 項，RMS 値の出力単位は、S3_T_OPTION 構造体の UnitType メンバと UnitFormat メンバで設定された単位となります。

・角度 x 成分 / y 成分 / D の出力単位は、S3_T_OPTION 構造体の AngUnitType メンバで設定された単位となります。

9.11. S3_T_MASK_ELEMENT

形式

```
typedef struct _S3_T_MASK_ELEMENT {
        S3_T_dPOINT Spot;
        S3_T_MASK_STATE State;
} S3_T_MASK_ELEMENT;
```

概要

個々のマスク要素の情報を格納します。

メンバ

Spot	マスク座標
State	マスク状態

9.12. S3_T_MASK_INFO

形式

```
typedef struct _S3_T_MASK_INFO {
        S3_T_CALIB_MODE CalibMode;
        int AreaType;
        S3_T_dPOINT AreaCenter;
        double AreaDiameter;
        S3_T_dSIZE AreaSize;
} S3_T_MASK_INFO;
```

概要

マスク生成時のパラメータ情報を格納します。

メンバ

CalibMode	校正モード（0:標準，1:ユーザー）
AreaType	計測エリア（0:円形，1:矩形）
AreaCenter	計測中心座標[mm]
AreaDiameter	計測エリア直径[mm]
AreaSize	計測エリアサイズ[mm]

備考

この情報を取得するには、s3_GetMaskInfo()関数を使用します。

9.13. S3_T_DEV_INFO

形式

```
typedef struct _S3_T_DEV_INFO {
        double CCD_PixSize;
        double MLA_LensPitch;
        double MLA_Focus;
} S3_T_DEV_INFO;
```

概要

光学ヘッドのデバイス情報を格納します。

メンバ

CCD_PixSize	ＣＣＤ ピクセルサイズ[um]
MLA_LensPitch	ＭＬＡ レンズピッチ[mm]
MLA_Focus	ＭＬＡ 焦点距離[mm]

備考

この情報を取得するには、s3_GetDeviceInfo()関数を使用します。

10. エラーコード

形式

```
typedef enum{
        S3_ERR_SUCSESS = 0,
        S3_ERR_UNEXP_CAMERA_ERROR = -1,
        S3_ERR_UNEXP_API_ERROR = -2,
        S3_ERR_BAD_HANDLE = -3,
        S3_ERR_BAD_PARAM = -4,
        S3_ERR_CAMERA_NOT_CONNECTED = -5,
        S3_ERR_CAMERA_UNINITIALIZED = -6,
        S3_ERR_CANNOT_START_CAPTURE = -7,
        S3_ERR_CANNOT_STOP_CAPTURE = -8,
        S3_ERR_BAD_USER_CALIB_FILE = -9,
        S3_ERR_USER_CALIB_FAIL = -10,
        S3_ERR_NO_USER_CALIB_DATA = -11,
        S3_ERR_STD_CALIB_FAIL = -12,
        S3_ERR_NO_MASK_DATA = -13,
        S3_ERR_BAD_MASK_FILE = -14,
        S3_ERR_NO_FRAME_DATA = -15,
        S3_ERR_FILE_ACCESS_FAIL = -16,
        S3_ERR_TIME_OUT = -17,
        S3_ERR_CALC_FAIL = -18,
        S3_ERR_MASK_FAIL = -19,
        S3_ERR_PARAM_FAIL = -20,
        S3_ERR_INVALID_CAMERA_TYPE = -21,
        S3_ERR_MEASURMENT_NOT_STARTED = -22,
        S3_ERR_NO_AUTHORITY_TO_WRITE = -23,
        S3_ERR_NOT_ENOUGTH_MEMORY = -24,
        S3_ERR_BAD_SNAPSHOT_FILE = -25,
        S3_ERR_NO_SNAPSHOT_DATA = -26,
        S3_ERR_DONGLE_KEY_NOT_FOUND = -9998,
        S3_ERR_UNDEF_FAIL = -9999,
} S3_T_ERR_CORD;
```

概要

エラーコードを規定します。

メンバ

S3_ERR_SUCCESS	正常終了
S3_ERR_UNEXP_CAMERA_ERROR	予期しないカメラエラー
S3_ERR_UNEXP_API_ERROR	予期しないエラー
S3_ERR_BAD_HANDLE	不正なハンドル値
S3_ERR_BAD_PARAM	不正なパラメータ値
S3_ERR_CAMERA_NOT_CONNECTED	カメラが接続されていない
S3_ERR_CAMERA_UNINITIALIZED	カメラが未だ初期化されていない
S3_ERR_CANNOT_START_CAPTURE	測定開始エラー
S3_ERR_CANNOT_STOP_CAPTURE	測定停止エラー
S3_ERR_BAD_USER_CALIB_FILE	不正なユーザー校正ファイル
S3_ERR_USER_CALIB_FAIL	ユーザー校正実行エラー
S3_ERR_NO_USER_CALIB_DATA	ユーザー校正データが設定されていない
S3_ERR_STD_CALIB_FAIL	デフォルト校正実行エラー
S3_ERR_NO_MASK_DATA	マスクデータが設定されていない
S3_ERR_BAD_MASK_FILE	不正なマスクファイル
S3_ERR_NO_FRAME_DATA	フレームデータが存在しない
S3_ERR_FILE_ACCESS_FAIL	ファイルアクセスエラー
S3_ERR_TIME_OUT	タイムアウトエラー
S3_ERR_CALC_FAIL	計算不可
S3_ERR_MASK_FAIL	マスク登録エラー
S3_ERR_PARAM_FAIL	パラメータ設定エラー
S3_ERR_INVALID_CAMERA_TYPE	無効なカメラ機種
S3_ERR_MEASUREMENT_NOT_STARTED	測定が未だ開始されていない
S3_ERR_NO_AUTHORITY_TO_WRITE	書き込み権限がない
S3_ERR_NOT_ENOUGTH_MEMORY	メモリ不足エラー
S3_ERR_BAD_SNAPSHOT_FILE	不正なスナップショット画像ファイル
S3_ERR_NO_SNAPSHOT_DATA	スナップショットデータが設定されていない
S3_ERR_DONGLE_KEY_NOT_FOUND	ドングルキーの認識エラー
S3_ERR_UNDEF_FAIL	未定義のエラー

11. 特記事項

11.1. 測定上の注意

11.1.1. センサへの入射光量について

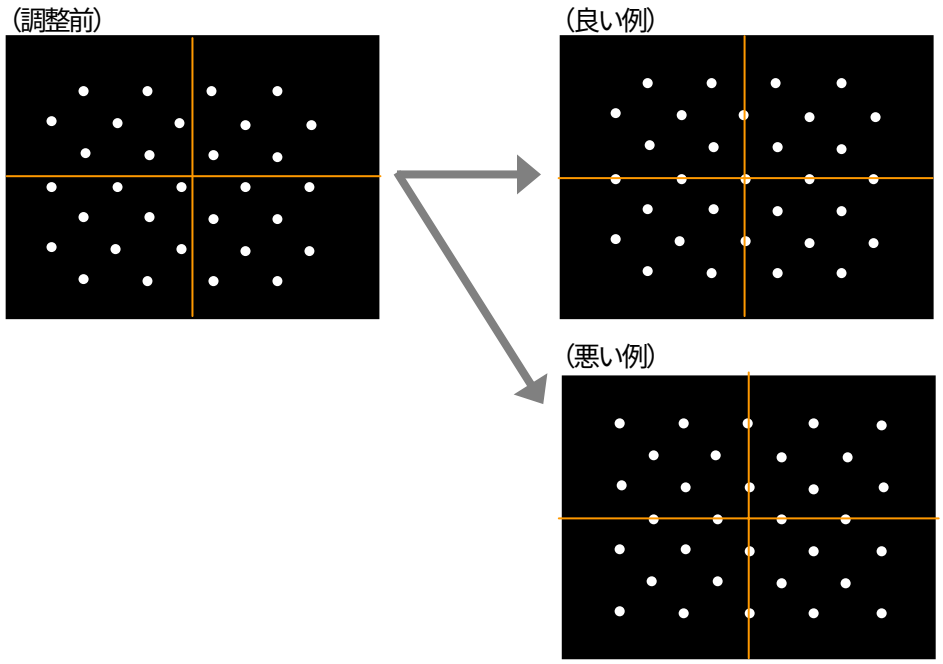
波面センサへの入射光量は 1.06 mW/mm^2（3 μW@φ3.0 mm）以下となるよう調整して下さい。

11.1.2. シャッタースピードについて

シャッタースピードは 0.5 msec 以上 5 msec 未満を推奨します。
入射光のピーク階調値が 180 以上、飽和ピクセル数 300 以下となるよう調整して下さい。

11.1.3. スポット調整について

スポット位置は波面センサ面へのビームの入射角により変化します。
フレームの中心にスポットが一致するよう調整して下さい。交点にスポットが無い状態になると測定ができません。



11.2. 参考資料

11.2.1. ゼルニケ多項式定義式

#	name	RMS	Polynominal
Z01	Piston	$1/\sqrt{2}$	1
Z02	Tilt X	$1/2$	$\rho \cos\theta$
Z03	Tilt Y	$1/2$	$\rho \sin\theta$
Z04	Defocus	$1/\sqrt{3}$	$2\rho^2-1$
Z05	AS 0/90°	$1/\sqrt{6}$	$\rho^2 \cos 2\theta$
Z06	AS 45°	$1/\sqrt{6}$	$\rho^2 \sin 2\theta$
Z07	Coma X	$1/2\sqrt{2}$	$(3\rho^3-2\rho)\cos\theta$
Z08	Coma Y	$1/2\sqrt{2}$	$(3\rho^2-2\rho)\sin\theta$
Z09	3rd SA	$1/\sqrt{5}$	$6\rho^4-6\rho^2+1$
Z10	5th Trefoil X	$1/2\sqrt{2}$	$\rho^3 \cos 3\theta$
Z11	5th Trefoil Y	$1/2\sqrt{2}$	$\rho^3 \sin 3\theta$
Z12	5th AS X	$1/\sqrt{10}$	$(4\rho^4-3\rho^2)\cos 2\theta$
Z13	5th AS Y	$1/\sqrt{10}$	$(4\rho^4-3\rho^2)\sin 2\theta$
Z14	5th Coma X	$1/2\sqrt{3}$	$(10\rho^5-12\rho^3+3\rho)\cos\theta$
Z15	5th Coma Y	$1/2\sqrt{3}$	$(10\rho^5-12\rho^3+3\rho)\sin\theta$
Z16	5th SA	$1/\sqrt{7}$	$20\rho^6-30\rho^4+12\rho^2-1$
Z17	7th Tetrafoil X	$1/\sqrt{10}$	$\rho^4 \cos 4\theta$
Z18	7th Tetrafoil Y	$1/\sqrt{10}$	$\rho^4 \sin 4\theta$
Z19	7th Trefoil X	$1/2\sqrt{3}$	$(5\rho^5-4\rho^3)\cos 3\theta$
Z20	7th Trefoil Y	$1/2\sqrt{3}$	$(5\rho^5-4\rho^3)\sin 3\theta$
Z21	7th AS X	$1/\sqrt{14}$	$(15\rho^6-20\rho^4+6\rho^2)\cos 2\theta$
Z22	7th AS Y	$1/\sqrt{14}$	$(15\rho^6-20\rho^4+6\rho^2)\cos 2\theta$
Z23	7th Coma X	$1/4$	$(35\rho^7-60\rho^5+30\rho^3-4\rho)\cos\theta$
Z24	7th Coma Y	$1/4$	$(35\rho^7-60\rho^5+30\rho^3-4\rho)\sin\theta$
Z25	7th SA	$1/3$	$70\rho^8-140\rho^6+90\rho^4-20\rho^2-1$
Z26	9th Pentafoil X	$1/2\sqrt{3}$	$\rho^5 \cos 5\theta$
Z27	9th Pentafoil Y	$1/2\sqrt{3}$	$\rho^5 \sin 5\theta$
Z28	9th Tetrafoil X	$1/\sqrt{14}$	$(6\rho^6-5\rho^4)\cos 4\theta$
Z29	9th Tetrafoil Y	$1/\sqrt{14}$	$(6\rho^6-5\rho^4)\sin 4\theta$
Z30	9th Trefoil X	$1/4$	$(21\rho^7-30\rho^5+10\rho^3)\cos 3\theta$
Z31	9th Trefoil Y	$1/4$	$(21\rho^7-30\rho^5+10\rho^3)\sin 3\theta$
Z32	9th AS X	$1/3\sqrt{2}$	$(56\rho^8-105\rho^6+60\rho^4-10\rho^2)\cos 2\theta$
Z33	9th AS Y	$1/3\sqrt{2}$	$(56\rho^8-105\rho^6+60\rho^4-10\rho^2)\sin 2\theta$
Z34	9th Coma X	$1/2\sqrt{5}$	$(126\rho^9-280\rho^7+210\rho^5-60\rho^3+5\rho)\cos\theta$
Z35	9th Coma Y	$1/2\sqrt{5}$	$(126\rho^9-280\rho^7+210\rho^5-60\rho^3+5\rho)\sin\theta$
Z36	9th SA	$1/\sqrt{11}$	$252\rho^{10}-630\rho^8+560\rho^6-210\rho^4+30\rho^2-1$

11.2.2. ザイデル収差定義式

Term	Description	Magnitude	Angle
W_{11}	Tilt	$\sqrt{(Z_2-2Z_7)^2+(Z_3-2Z_8)^2}$	$\tan^{-1}\left(\frac{Z_3-2Z_8}{Z_2-2Z_7}\right)$
W_{20}	Focus	$2Z_4-6Z_9\pm\sqrt{Z_5^2+Z_6^2}$ ※極性は絶対値が最小になるよう選択されます。	
W_{22}	Astigmatism	$\pm 2\sqrt{Z_5^2+Z_6^2}$ ※極性は Focus に対し逆極性になるよう選択されます。	$\frac{1}{2}\tan^{-1}\left(\frac{Z_6}{Z_5}\right)$
W_{31}	Coma	$3\sqrt{Z_7^2+Z_8^2}$	$\tan^{-1}\left(\frac{Z_8}{Z_7}\right)$
W_{40}	Spherical	$6Z_9$	

ミスミグループ
駿河精機株式会社
ＯＳＴ事業部
カスタマーサービス

http://www.surugaost.jp/
TEL．0120－789－446
FAX．0120－789－446
E-mail. ost@suruga－g.co.jp

本社･工場:〒424-8566
静岡県静岡市清水区七ツ新屋 549-1
TEL:054-344-0332／FAX:054-344-0337
東京営業所:〒110-0014
東京都台東区北上野2-18-4 UCJ上野ビル4F
TEL:03-5806-1635／FAX:03-5806-1697
関西営業所:〒569-0071
大阪府高槻市城北町1-5-25 FJYビル4F
TEL:072-661-3500／FAX:072-661-3622