１．SSPライブラリのデータ構造

SSPライブラリでは以下に列挙するデータ構造を持つ。

（１）関数の返値、およびエラー
基本関数はSSP_RESULT型の実行結果を返す。以下の返値が用意されている。

返値	意味
SSP_SUCCESS	基本関数の実行に成功
SSP_ERR_NOMEM	メモリ確保に失敗した
SSP_ERR_THREAD	スレッドの作成に失敗した
SSP_ERR_EBAYERBITS	Bayerの画素のビット数指定が間違っている
SSP_ERR_EINIT	システムの初期化に失敗した
SSP_ERR_ECAMSETUP	カメラのセットアップに失敗した
SSP_ERR_EMMAL	MMALの初期化に失敗した
SSP_ERR_EFRAMTYPE	画像フレームの形式が間違っている
SSP_ERR_EFATAL	致命的エラーが発生した

（２）画像フレームタイプ
SSPライブラリでは以下の画像フレームタイプをカメラからの出力画像として取り出すことができる。この型をssp_frame_preprocess_options構造体のFormatConvertTypeに指定することで、ユーザレベルで受け取る画像フレームタイプを規定できる。

画像フレームタイプ	説明
SSP_FRAME_BAYER10	10ビットBayerフォーマットであり、偶数ラインにRGRGRG…、奇数ラインにGBGBGB…と10ビットの画素が並ぶ。
SSP_FRAME_BAYER8	8ビットBayerフォーマットであり、偶数ラインにRGRGRG…、奇数ラインにGBGBGB…と8ビットの画素が並ぶ。
SSP_FRAME_BAYER16	16ビットBayerフォーマットであり、偶数ラインにRGRGRG…、奇数ラインにGBGBGB…と8ビットの画素が並ぶ。10ビットまたは、8ビットのBayerしかサポートしないデバイスの場合は、それらのデータを16ビットに変換している。
SSP_FRAME_RGB24	24ビットRGBフォーマットであり、8ビットにパックされた、R、G、Bの要素がこの順で並ぶ。Bayerからの変換は、ACPI方式を用いている。
SSP_FRAME_YUV444	YUV444フォーマットであり、8ビットにパックされた、Y、Cb、Crがこの順で並ぶ。 YUV変換はITU-R BT.601変換をRGB24から行う。
SSP_FRAME_YUV422	YUV422フォーマットであり、8ビットにパックされた、Y0、Cb00、Y1、Cr01がこの順で並ぶ。 YUV変換はITU-R BT.601変換をRGB24から行う。

２．SSPライブラリの構造体

（１）ssp_handle構造体
SSPライブラリを利用する際には、ユーザアプリケーションからはハンドルと呼ばれる構造体を元にして、基本関数を呼び出す。ハンドルは基本関数の初期化関数で作成する。以下のようなメンバを持つ。

struct ssp_handleのメンバー	説明
void *user_data;	ユーザ側で利用できるポインタであり、例えば、フレーム処理関数に渡されたハンドルでもこのポインタの値は共通に保持される。SSPライブラリを通して、グローバル変数のように用いる事ができる。基本関数の初期関数が呼ばれた直後に、このポインタは設定されなくてはならない。
struct ssp_settings *settings;	ユーザから与えられた設定情報であり、それをもとにユーザに自動認識したハードウェア情報なを返す構造体である。
void *internal_data;	SSPライブラリが内部的に用いるデータへのポインタ

（２）ssp_frame 構造体
ユーザ側への画像フレームはこの構造体で返される。この構造体を操作する際は専用のマクロと基本関数を使うべきである。

struct ssp_frameのメンバー	説明
int FrameType;	画像フレームのタイプを表す。
int Width;	画像フレームの幅を画素数を保持する。単位が画素数であるため、バイト数に注意する。
int Height;	画像フレームの高さを画素数で保持する。
unsigned char *FrameData;	画像フレームのデータを保持する

（３）struct ssp_settings構造体
SSPライブラリでは、ユーザでこの構造体を作成し、初期化関数に渡すことで環境設定を行う。この構造体は静的、動的を問わず、ユーザ側で作成され、さらに、SSPライブラリの終了まで保持されなければいけない。この構造体には複数の構造体を含むので、参照する際に注意が必要である。さらに、ユーザ側で初期値を提供するべき変数と、システム側で値を設定する変数があるので、注意が必要である。下記表ではユーザが初期化する変数については「（ユーザ）」とマークしている。

struct ssp_settings構造体のメンバ	説明
struct ssplib_settings lib_settings;	ライブラリの基本設定のための構造体である。
struct ssp_frame_preprocess_options frame_preprocess_options;	フレーム画像前処理の設定である。

（３－１）struct ssplib_settings構造体

struct ssplib_settings構造体のメンバ	説明
int NumFrameFIFOSize;	フレームイベントFIFOの深さを指定する（ユーザ）
int NumThreadForBuildInPreprocess;	フレーム画像前処理で使うスレッド数の最大数を指定する（ユーザ）
void (*frame_drop_cam_user_func)(struct ssp_handle *handle);	フレームイベントFIFOがFULLの際にSSPライブラリがフレームをFIFOに書き込もうとする際、または、メモリ確保に失敗したときに、この関数が呼ばれる。落とされたフレームはSSPライブラリ側で破棄される。（ユーザ）
void (*frame_drop_pre_user_func)(struct ssp_handle *handle);	フレーム画像前処理の際にメモリ確保に失敗すると、この関数が呼ばれる。落とされたフレームはSSPライブラリ側で破棄される。（ユーザ）
void (*fame_preprocess_user_func)(struct ssp_handle *handle, struct ssp_frame *frame);	フレーム画像前処理の最後に呼び出されるユーザ側のフレーム処理関数を設定する。この関数を介して、フレーム画像データをユーザ側で受け取り、処理できる。（ユーザ）

（３－３）ssp_frame_preprocess_options構造体

struct ssp_frame_preprocess_optionsのメンバ	説明
int FormatConvertType;	画像前処理後の画像フレームのフォーマットを指定する。（ユーザ）
int GammaCorrectionEnable;	ガンマ補正を必要とするかをSSP_ENABLE、SSP_DISABEで指定する。（ユーザ）
float GammaVal;	ガンマ補正に用いる補正値（ユーザ）

フレーム画像の前処理は、FormatConvertType、及び、GammaCorrectionEnableの値に従って、（１）Bayer→RGB変換、（２）ガンマ補正、（３）RGB→YUV変換、の順で実行され、ユーザ側のフレーム処理イベント関数fame_preprocess_user_funcが呼び出される。

３．SSPライブラリの関数

SSPライブラリでは、以下の関数を用いて、イメージセンサーを制御する。

（１）初期化関数
SSP_RESULT ssp_initialize(struct ssp_handle **handle, struct ssp_profile *profile, struct ssp_settings *settings);
SSPライブラリを用いる際に最初に呼び出されなければいけない。Settingsに関しては、struct ssp_settings構造体はユーザ側で用意し、そのポインタを渡す。profileに関しては、struct ssp_profile構造体をssp_read_profile関数で作成して渡す。
この関数の実行後、struct ssp_handleへのポインタが返される。

（２）終了関数
SSP_RESULT ssp_finalize(struct ssp_handle *handle);
SSPライブラリの終了時には、この関数が呼びされなければいけない。
この関数の引数のhandleは初期関数で返されたポインタを渡す。

（３）イメージセンサープロファイル読み込み関数
イメージセンサープロファイルは、XMLで提供される必要があり、そのファイル名を以下の関数に渡し、struct ssp_profile構造体を作成する。
SSP_RESULT ssp_read_profile(char *profile_uri, struct ssp_profile **profile) ;

（４）イメージセンサーストリーム開始関数
SSP_RESULT ssp_start_streaming(struct ssp_handle *handle);
イメージセンサーをアクティブにし、画像フレームを受け取りはじめる。

（５）イメージセンサーストリーム停止関数
SSP_RESULT ssp_stop_streaming(struct ssp_handle *handle);
イメージセンサーを停止し、画像フレーム取得を停止する。

（６）イベントFIFO同期関数
SSP_RESULT ssp_sync_event(struct ssp_handle *handle);
フレームイベントのFIFOの中身が全て、フレーム画像前処理スレッドで処理されるまで実行をブロックする関数である。

（７）イベントFIFOフラッシュ関数
SSP_RESULT ssp_flush_event(struct ssp_handle *handle);
フレームイベントのFIFOの中身を全て削除する。この場合、処理されなかったイベントに含まれるフレームは全て破棄される。

（８）フレーム画像のバイト数取得関数
SSP_RESULT ssp_get_frame_size(struct ssp_frame *frame, int *size);
フレーム画像のバイト数を取得する。

４．SSPライブラリのマクロ

（１） ssp_get_frame_data(frame)はフレーム画像のFrameDataを返す

（２） ssp_get_frame_type(frame)はフレーム画像のFrameTypeを返す。

（３） ssp_get_frame_height(frame)はフレーム画像のHeightを返す。

（４） ssp_get_frame_width(frame)はフレーム画像のWidthを返す。

（５） ssp_release_frame(frame)はフレーム画像を解放する。この解放マクロは、ユーザが提供するフレーム処理関数の最後で必ず呼び出されなければいけない。このマクロが呼び出されない場合、フレームを解放する手段がSSPライブラリには用意されていないので、メモリリークとなる。