Sphere

Skeleton for PHysical and Engineering REsearch

Ver 1.0.0

SPL ライブラリ（低レベルライブラリ）　マニュアル

2010 年 8 月 01 日

もくじ

1. ライブラリの概要

2. ユーティリティ関数

2.1. Boolian 変数

SPL ライブラリは C 言語で実装されているため、Boolian 変数を以下のように定義しています。

型 ： SplBool
値 ： （真） SplTrue （ = 1 ）
（偽） SplFalse （ = 0 ）

2.2. エラー処理

ライブラリ関数内でエラーが発生した場合、SPL ライブラリはエラーコード、エラーメッセージを以下に格納します。

```
int spl_err_code                          ：エラー番号
char spl_err_info[SPL_MAX_STR_SIZE]  ：エラーメッセージ
```

spl_err_code、spl_err_info は外部変数として定義されており、プログラム内のどこからでも直接参照可能です。

または、以下の関数により取得可能です。

```
int spl_get_errcode();                    ：エラー番号取得関数
const char* spl_get_errinfo();            ：エラーメッセージ取得関数
```

またエラーメッセージを出力する関数として

```
void spl_perror(const char* s)
```

を提供します。spl_perror 関数は最後にエラー終了した SPL ライブラリ関数のエラーを説明するメッセージを作成し、標準エラー出力に出力します。

2.3. メッセージ出力

SPL ライブラリは以下に示す 3 種類のメッセージ出力関数を提供します。

```
void spl_msg(const char* format, ...)
void spl_errmsg(const char* format, ...)
```

void spl_wrnmsg(const char* format, ...)

これらのメッセージ出力関数は可変長引数の関数として定義されており、引数の指定方法は printf 文と同じ文法を採用しています。また spl_msg 関数は標準出力に出力し、spl_errmsg 関数と spl_wrnmsg 関数は標準エラー出力にメッセージを出力します。

2.4. 時刻取得

時刻取得を行う関数として以下の関数を提供します。

double splGetTime(void)

splGetTime 関数は内部で gettimeofday 関数を呼び出して、取得した timeval 構造体より紀元（1970 年 1 月 1 日 00:00:00 UTC）からの経過時間を double で返します。単位はマイクロ秒です。

3. メッセージパッシング機能

SPL ライブラリのメッセージパッシング機能は MPICH を元に作成され、インターフェイスは MPICH が提供するそれに１対１に対応しています。
ただし、SPL ライブラリのメッセージパッシング関数群の接頭辞は SPL（MPICH では MPI）となります。

例えば、同期送信関数（Send）は、MPICH で

```
int MPI_Send(void *buf, int count, MPI_Datatype dtype,
             int dest, int tag, MPI_Comm comm.)
```

と定義されていますが、SPL ライブラリでは、

```
int SPL_Send(void *buf, int count, SPL_Datatype dtype,
             int dest, int tag, SPL_Comm comm.)
```

と定義されます。

4. ファイル入出力機能

SPL ライブラリは SBX ファイル、SPX ファイルの並列入出力をサポートしています。

SBX　：　整数のボクセルデータを格納するためのファイルフォーマット
SPX　：　実数のボクセルデータを格納するためのファイルフォーマット

なお、各ファイルのフォーマットは 5.3. を参照ください。

4.1. ファイル構造体

分散出力されたファイルに対して並列実行環境で入出力処理を行うために、ファイル情報を一括管理できる機能を提供します。
Sphere では複数のファイルの情報（ファイル名や属性）をまとめてリスト構造で保持する構造体を提供します。

```
typedef struct _spl_sv_file_ {
  int set_flag;
  size_t start_idx[3];
  const char fname[SPL_MAX_STR_SIZE];
  struct _spl_sv_file_* next;
} splSvFile

typedef struct _spl_sv_file_list_ {
  size_t list_size;
  splSvFile* top;
} splSvFileList
```

splSvFile 構造体は 1 ファイルの情報を格納するための構造体です。また、splSvFileList 構造体は splSvFile 構造体のリストを管理する構造体です。
ユーザは主に splSvFileList 構造体に以下の関数を用いてファイル情報を登録することで、ファイル情報を作成することができます。

```
SplBool splInitSvFileList(splSvFileList* obj)
SplBool splFinalizeSvFileList(splSvFileList* obj)
```

```
SplBool splAddSvFile(splSvFileList* obj, const char* fn,
                     const size_t sidx[3])
```

関数の使用方法に関する詳細は 5.2.2. を参照ください。

4.2. SBX ファイルの入出力

4.2.1. splSbx 構造体

SBX ファイルに記述される各レコードの情報を格納するための構造体です。

```
typedef struct _spl_sbx_ {...} splSbx
```

splSbx 構造体の各メンバは SBX ファイルの各レコードに 1 対 1 に対応します。
メンバの詳細は 5.1.3. を参照ください。

4.2.2. splSbx 構造体の初期化

splSbx 構造体を初期化します。

```
SplBool splInitSbx(splSbx* obj)
```

関数の詳細は 5.2. を参照ください。

4.2.3. splSbx 構造体のファイナライズ

必要の無くなった splSbx 構造体のファイナライズを行います。

```
SplBool splFinalizeSbx(splSbx* obj, SplBool mem_free_flag)
```

関数の詳細は 5.2. を参照ください。

4.2.4. splSbx 構造体への情報の設定

splSbx 構造体に情報を設定します。

```
SplBool splSbxSetValues(splSbx* obj, unsigned int dims,
                        unsigned int vlen, unsigned int dtype,
                        size_t gc, size_t* sta_idx,
                        size_t* size, unsigned char* dt)
```

関数の詳細は 5.2. を参照ください。

4.2.5. ファイルの読込み

SBX ファイルの読込みを行います。

```
SplBool splReadSbx(splSvFileList* file_list, int mode,
                   splSbx* obj, int readRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.2.6. ファイルへの書出し

SBX ファイルへの書出しを行います。

```
SplBool splWriteSbx(const char* fname, int mode,
                    SplBool cmp_flag, SplBool ext_rec_flag,
                    size_t block_size, const splSbx* obj,
                    int writeRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.2.7. SBX ファイルからの情報の取得

指定された SBX ファイルに記述されているヘッダレコードの情報を splSbx 構造体に取得します。

```
SplBool splGetSbxInfo(const char* fname, splSbx* obj,
                      int mode, int readRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.3. SPX ファイルの入出力

4.3.1. splSpx 構造体

SPX ファイルに記述される各レコードの情報を格納するための構造体です。

typedef struct _spl_spx_ {...} splSpx

splSpx 構造体の各メンバは SPX ファイルの各レコードに 1 対 1 に対応します。
メンバの詳細は 5.1.4. を参照ください。

4.3.2. splSpx 構造体の初期化

splSpx 構造体を初期化します。

SplBool splInitSpx(splSpx* obj)

関数の詳細は 5.2. を参照ください。

4.3.3. splSpx 構造体のファイナライズ

必要の無くなった splSpx 構造体のファイナライズを行います。

SplBool splFinalizeSpx(splSpx* obj, SplBool mem_free_flag)

関数の詳細は 5.2. を参照ください。

4.3.4. splSpx 構造体への情報の設定

splSpx 構造体に情報を設定します。

SplBool splSpxSetValues(splSpx* obj, unsigned int dims,
unsigned int vlen, unsigned int dtype,
size_t gc, size_t* sta_idx,
size_t* size, unsigned char* dt)

関数の詳細は 5.2. を参照ください。

4.3.5. ファイルの読込み

SPX ファイルの読込みを行います。

```
SplBool splReadSpx(splSvFileList* file_list, int mode,
                   splSpx* obj, int readRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.3.6. ファイルへの書出し

SPX ファイルへの書出しを行います。

```
SplBool splWriteSpx(const char* fname, int mode,
                    SplBool cmp_flag, SplBool ext_rec_flag,
                    size_t block_size, const splSpx* obj,
                    int writeRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.3.7. SPX ファイルからの情報の取得

指定された SPX ファイルに記述されているヘッダレコードの情報を splSpx 構造体に取得します。

```
SplBool splGetSpxInfo(const char* fname, splSpx* obj,
                      int mode, int readRank, SPL_Comm grp)
```

関数の詳細は 5.2. を参照ください。

4.4. splData 共用体

splSbx 構造体、splSpx 構造体の共通メンバとして splData data;を持ちます。

splData 共用体は、SBX、SPX ファイルの読込み、書出しデータを設定する共用体で以下の書式となっています。

union splData

（用　途）

ボクセルファイル（SBX, SPX）のデータを格納する共用体

（書　式）

```
typedef union _spl_data_ {
  unsigned char* c;          // データ型:char
  unsigned short* s;         // データ型:short
  unsigned int* i;           // データ型:int
  float* f;                  // データ型:float
  double* d;                 // データ型 double
} splData;
```

SBX、SPX ファイルからの読込み時には、splData 共用体メンバに対して、読込みを行うデータサイズ分の領域の確保、初期化をしなければなりません。

SBX、SPX ファイルからの書込み時には、書込みを行うデータを splData 共用体メンバに設定しなければなりません。

5. 付録

5.1. 型／変数／構造体　リファレンス

5.1.1. ユーティリティ

SplBool

（用　途）

Boolian 型

（書　式）

#include　“spl/spl.h”

SplBool　　：　真　SplTrue（ = 1 ）

偽　SplFalse（ = 0 ）

（説　明）

SPL ライブラリで定義された Boolian 型。

真のとき SplTrue（ = 1 ）、偽のとき SplFalse（ = 0 ）とする

spl_err_code

（用　途）

エラーコード格納用外部変数

（書　式）

int　　spl_err_code

（説　明）

SPL ライブラリ関数内でエラーが発生した場合、spl_err_code にエラーコードが格納される。spl_err_code は外部変数として定義されているので、参照する場合はソースプログラム中に

extern int spl_err_code

を記述する必要がある。

spl_err_info

（用　途）

エラーメッセージ格納用外部変数

（書　式）

```
char    spl_err_code    [SPL_MAX_STR_SIZE]
```

（説　明）

SPL ライブラリ関数内でエラーが発生した場合、spl_err_code にエラーメッセージが格納される。エラーメッセージにはエラー原因を示す情報が付加されている。spl_err_info は外部変数として定義されているので、参照する場合はソースプログラム中に

```
extern int spl_err_info
```

を記述する必要がある。

5.1.2. ファイル情報関連

struct splSvFile

（用　途）

ボクセルファイル（SBX, SPX）のファイル情報を格納する構造体

（書　式）

```
typedef struct _spl_sv_file_ {
  int set_flag;
  size_t start_idx[3];
  const char fname[SPL_MAX_STR_SIZE];
  struct _spl_sv_file_* next;
} splSvFile
```

（説　明）

- int set_flag
  構造体の各変数の値設定の有無を示すフラグ。
  　0：値設定無し　　1：値設定有り
  ユーザは set_flag にアクセスしてはならない。
- size_t start_idx[3]
  始点グローバルインデックス
- const char fname[SPL_MAX_STR_SIZE]
  ファイル名
  ファイル名は SPL_MAX_STR_SIZE 文字より少ない文字列でなければならない。SPL_MAX_STR_SIZE は spl.h に定義される。
- struct _spl_sv_file_* next
  ファイルリスト作成時に必要となるポインタ変数。
  通常、ユーザは next にアクセスしない。

struct splSvFileList

（用　途）

splSvFile 構造体のリストを格納するための構造体

（書　式）

```
typedef struct _spl_sv_file_list_ {
  size_t list_size;
  splSvFile* top;
} splSvFileList
```

（説　明）

- size_t list_size
  リストに登録されている splSvFile（ファイル情報）の数
- splSvFile* top
  splSvFile のリスト

これらの構造体メンバにユーザが直接アクセスする必要はない。

5.1.3. SBX ファイル関連

struct splSbx

（用　途）

SBX ファイルに記述された情報を格納するための構造体

（書　式）

```
typedef struct _spl_sbx_ {
  size_t wsize[3];
  size_t start_idx[3];
  unsigned int dims;
  unsigned int vlen;
  unsigned int dtype;
  size_t gc;
  unsigned int rlen;
  unsigned int crddef;
  unsigned int aux;
  size_t size[3];
  double origin[3];
  double pitch[3];
  splData data;
} splSbx
```

（説　明）

- size_t wsize[3]
  計算領域全体のボクセルサイズ
- size_t start_idx[3]
  始点グローバルインデックス
- unsigned int dims
  次元数
- unsigned int vlen
  ベクトル数
- unsigned int dtype
  データタイプ
- size_t gc

仮想セルサイズ

- unsigned int rlen
  浮動小数点数のサイズ
- unsigned int crddef
  定義点情報
- unsigned int aux
  未使用
- size_t size[3]
  ボクセルサイズ
- double origin[3]
  原点座標
- double pitch[3]
  ピッチサイズ
- splData data
  データポインタ

union splData

（用　途）

ボクセルファイル（SBX, SPX）のデータを格納する共用体

（書　式）

```
typedef union _spl_data_ {
  unsigned char* c;          // データ型:char
  unsigned short* s;         // データ型:short
  unsigned int* i;           // データ型:int
  float* f;                  // データ型:float
  double* d;                 // データ型 double
} splData;
```

（説　明）

- unsigned char* c
  データ型:char
- unsigned short* s
  データ型:short
- unsigned int* i

データ型:int

- float* f
  データ型:float
- double* d
  データ型 double

SBX ファイルからのデータ読込み時には、ボクセルサイズ(size[3])、データタイプ(dtype)分の領域を確保、初期化し、splData メンバにデータポインタを設定しなければならない。

読込み実行後、SBX ファイルのデータは splData メンバにコピーされる。

SBX ファイルからのデータ書込み時には、ボクセルサイズ(size[3])、データタイプ(dtype)分の書込みデータを splData メンバにデータポインタを設定しなければならない。

書込み実行にて splData メンバのデータが SBX ファイルに出力される。

5.1.4. SPX ファイル関連

struct splSpx

（用　途）

SPX ファイルに記述された情報を格納するための構造体

（書　式）

```
typedef struct _spl_spx_ {
  size_t wsize[3];
  size_t start_idx[3];
  unsigned int dims;
  unsigned int vlen;
  unsigned int dtype;
  size_t gc;
  unsigned int rlen;
  unsigned int crddef;
  unsigned int aux;
  size_t size[3];
  double origin[3];
  double pitch[3];
  size_t step;
  double time;
  splData data;
} splSpx
```

（説　明）

- size_t wsize[3]
  計算領域全体のボクセルサイズ
- size_t start_idx[3]
  始点グローバルインデックス
- unsigned int dims
  次元数
- unsigned int vlen
  ベクトル数
- unsigned int dtype

データタイプ

- size_t gc
  仮想セルサイズ
- unsigned int rlen
  浮動小数点数のサイズ
- unsigned int crddef
  定義点情報
- unsigned int aux
  未使用
- size_t size[3]
  ボクセルサイズ
- double origin[3]
  原点座標
- size_t step
  ステップ数
- double time
  時刻
- double pitch[3]
  ピッチサイズ
- splData data
  データポインタ

union splData

（用　途）

ボクセルファイル（SBX, SPX）のデータを格納する共用体

（書　式）

```
typedef union _spl_data_ {
  unsigned char* c;          // データ型:char
  unsigned short* s;         // データ型:short
  unsigned int* i;           // データ型:int
  float* f;                  // データ型:float
  double* d;                 // データ型 double
} splData;
```

（説　明）

- unsigned char* c
  データ型:char
- unsigned short* s
  データ型:short
- unsigned int* i
  データ型:int
- float* f
  データ型:float
- double* d
  データ型 double

SPX ファイルからのデータ読込み時には、ボクセルサイズ(size[3])、浮動小数点数のサイズ(rlen)、ベクトル数(vlen)分の領域を確保、初期化し、splData メンバにデータポインタを設定しなければならない。

読込み実行後、SPX ファイルのデータは splData メンバにコピーされる。

SPX ファイルからのデータ書込み時には、ボクセルサイズ(size[3])、浮動小数点数のサイズ(rlen)、ベクトル数(vlen)分の書込みデータを splData メンバにデータポインタを設定しなければならない。

書込み実行にて splData メンバのデータが SPX ファイルに出力される。

5.2. 関数　リファレンス

5.2.1. ユーティリティ関数

spl_get_errcode

（用　途）

エラーコードの取得

（書　式）

#include “spl/spl.h”

int　　spl_get_errcode　（　）

（説　明）

エラーコードを取得する。

（返り値）

エラーコード

（エラー）

無し

spl_get_errinfo

（用　途）

エラーメッセージの取得

（書　式）

#include “spl/spl.h”

const char*　　spl_get_errinfo　（　）

（説　明）

エラーメッセージを取得する。

（返り値）

エラーメッセージ

（エラー）

無し

spl_perror

（用　途）

エラーメッセージの出力

（書　式）

```
#include “spl/spl.h”
  void      spl_perror      （    const char* s  ）
```

（説　明）

最後にエラー終了した SPL ライブラリ関数のエラーを説明するメッセージを作成し、標準エラー出力に出力される。

（返り値）

無し

（エラー）

無し

spl_msg, spl_errmsg, spl_wrnmsg

（用　途）

メッセージ出力

（書　式）

```
#include “spl/spl.h”
  void    spl_msg     (   const char* format, ... )
  void    spl_errmsg  (   const char* format, ... )
  void    spl_wrnmsg  (   const char* format, ... )
```

（説　明）

メッセージ出力関数は可変長引数の関数として定義され、引数の指定方法は printf 文と同じ文法を採用している。また spl_msg 関数は標準出力に出力し、spl_errmsg 関数と spl_wrnmsg 関数は標準エラー出力にメッセージを出力する。

（返り値）

無し

（エラー）

無し

splGetTime

（用　途）

（書　式）

#include “spl/spl.h”

double　splGetTime　（　）

（説　明）

splGetTime 関数は内部で gettimeofday 関数で得られる timeval 構造体より、紀元（1970 年 1 月 1 日 00:00:00 UTC）からの経過時間を取得する。

（返り値）

紀元（1970 年 1 月 1 日 00:00:00 UTC）からの経過時間を返す。単位はマイクロ秒。

（エラー）

無し

5.2.2. ファイル情報関連関数

splInitSvFile

（用　途）

splSvFile 構造体の初期化

（書　式）

#include “spl/splfio.h”

SplBool　splInitSvFile　（　splSvFile* obj　）

（説　明）

obj を初期化する。

（返り値）

初期化に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

・ SPL_ERR_INVALID_PARAMETER

obj が NULL

splFinalizeSvFile

（用　途）

splSvFile 構造体のファイナライズ

（書　式）

#include “spl/splfio.h”

SplBool　splFinalizeSvFile　（　splSvFile* obj　）

（説　明）

必要の無くなった obj にファイナライズ処理を行う。

（返り値）

ファイナライズに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj が NULL

splSetSvFile

（用　途）

splSvFile 構造体にファイル情報をセットする。

（書　式）

```
#include “spl/splfio.h”
  SplBool   splSetSvFile   (   splSvFile* obj,
                               const char* fn,
                               const size_t sidx[3]  )
```

（説　明）

- splSvFile* obj
  splSvFile 構造体へのポインタ
- const char* fn
  ファイル名
- const size_t sidx[3]
  始点インデックス情報

（返り値）

obj への情報のセットに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、fn または sidx のいづれかが NULL
- SPL_ERR_STR_SIZE_TOO_LONG
  fn に指定されたファイル名の長さが SPL_MAX_STR_SIZE を超えている。

splInitSvFileList

（用　途）

splSvFileList 構造体の初期化

（書　式）

#include “spl/splfio.h”

SplBool　splInitSvFileList　（　splSvFileList* obj　）

（説　明）

・ splSvFileList* obj

初期化する splSvFileList 構造体へのポインタ

（返り値）

初期化に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

・ SPL_ERR_INVALID_PARAMETER

obj が NULL

splFinalizeSvFileList

（用　途）

splSvFileList 構造体のファイナライズ

（書　式）

#include “spl/splfio.h”

SplBool　splFinalizeSvFileList　（　splSvFileList* obj　）

（説　明）

必要の無くなった obj にファイナライズ処理を行う。

（返り値）

ファイナライズに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

・ SPL_ERR_INVALID_PARAMETER

obj が NULL

splAddSvFile

（用　途）

splSvFileList 構造体にファイル情報を追加する。

（書　式）

```
#include “spl/splfio.h”
  SplBool   splAddSvFile   (   splSvFileList* obj,
                               const char* fn,
                               const size_t sidx[3]   )
```

（説　明）

- splSvFileList* obj
  情報を追加する splSvFileList 構造体
- const char* fn
  ファイル名情報
- const size_t sidx[3]
  始点インデックス情報

（返り値）

obj への情報の追加に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、fn または sidx のいづれかが NULL
- SPL_ERR_STR_SIZE_TOO_LONG
  fn に指定されたファイル名の長さが SPL_MAX_STR_SIZE を超えている
- SPL_ERR_MEMORY_ALLOCATE
  メモリ領域の確保に失敗

splAddSvFileName

（用　途）

splSvFileList 構造体にファイル名を追加する。
始点インデックスは、ファイル内の拡張レコードの始点インデックスに従う。

（書　式）

```
#include “spl/splfio.h”
  SplBool   splAddSvFile   (   splSvFileList* obj,
                               const char* fn
```

（説　明）

- splSvFileList* obj
  情報を追加する splSvFileList 構造体
- const char* fn
  ファイル名情報

（返り値）

obj への情報の追加に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、fn または sidx のいづれかが NULL
- SPL_ERR_STR_SIZE_TOO_LONG
  fn に指定されたファイル名の長さが SPL_MAX_STR_SIZE を超えている
- SPL_ERR_MEMORY_ALLOCATE
  メモリ領域の確保に失敗

5.2.3. SBX ファイル操作関連関数

splInitSbx

（用　途）

splSbx 構造体の初期化

（書　式）

```
#include “spl/splfio.h”
  SplBool   splInitSbx     （   splSbx* obj   ）
```

（説　明）

・ splSbx* obj
  初期化する splSbx 構造体

（返り値）

初期化に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

・ SPL_ERR_INVALID_PARAMETER
  obj が NULL

splFinalizeSbx

（用　途）

splSbx 構造体のファイナライズ

（書　式）

```
#include “spl/splfio.h”
  SplBool   splFinalizeSbx   （   splSbx* obj,
                                  SplBool mem_free_flag   ）
```

（説　明）

- splSbx* obj
  ファイナライズを行う splSbx 構造体
- SplBool mem_free_flag
  splSbx 構造体の内部で動的に確保されたメモリ領域の開放処理を行うかどうかのフラグ。
  SplTrue：開放する
  SplFalse：開放しない

（返り値）

ファイナライズに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj が NULL

splSbxSetValues

（用　途）

splSbx 構造体に値をセットする

（書　式）

```
#include “spl/splfio.h”
  SplBool   splSbxSetValues   (   splSbx* obj,
                                  unsigned int dims,
                                  unsigned int vlen,
                                  unsigned int dtype,
                                  size_t gc,
                                  size_t* sta_idx,
                                  size_t* size,
                                  unsigned char* dt   )
```

（説　明）

- splSbx* obj
  値設定の対象となる splSbx オブジェクト
- unsigned int dims
  次元数
- unsigned int vlen
  ベクトル長
- unsigned int dtype
  データタイプ
- size_t gc
  仮想セル数
- size_t* sta_idx
  始点インデックス
- size_t* size
  ボクセルサイズ
- unsigned char* dt
  データ配列への参照

（返り値）

obj への情報の設定に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、sta_idx、size または dt のいづれかが NULL

splGetSbxInfo

（用　途）

SBX ファイルに記述されている値を splSbx 構造体に取得する

（書　式）

```
#include “spl/splfio.h”
  SplBool   splGetSbxInfo   (   const char* fname,
                                splSbx* obj,
                                int mode,
                                int readRank,
                                SPL_Comm grp   )
```

（説　明）

- const char* fname
  情報を取得するファイルの名前
- splSbx* obj
  splSbx 構造体
- int mode
  ファイルのオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 readRank のノードがファイルをオープンし、
  　　読み込んだ値を他のノードにブロードキャストする。
  　　このとき、識別子 readRank のノードも情報を保持する。
- int readRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

情報の取得に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、sta_idx、size または dt のいづれかが NULL

splReadSbx

（用　途）

SBX ファイルかたらの読込み

（書　式）

```
#include “spl/splfio.h”
  SplBool   splReadSbx   (   splSvFileList* file_list,
                             int mode,
                             splSbx* obj,
                             int readRank,
                             SPL_Comm grp   )
```

（説　明）

- splSvFileList* file_list
  読み込むファイルのリスト
- int mode
  ファイルオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 readRank のノードがファイルをオープンし、
  　　読み込んだ値を他のノードにブロードキャストする。
  　　このとき、識別子 readRank のノードも情報を保持する。
- splSbx* obj
  読込先の splSbx 構造体
- int readRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

ファイルの読込みに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  file_list もしくは obj が NULL

splWriteSbx

（用　途）

SBX ファイルへの書出し

（書　式）

```
#include “spl/splfio.h”
  SplBool   splWriteSbx   (   const char* fname,
                              int mode,
                              SplBool cmp_flag,
                              SplBool ext_rec_flag,
                              size_t block_size,
                              const splSbx* obj,
                              int writeRank,
                              SPL_Comm grp   )
```

（説　明）

- const char* fname
  書出すファイルの名前
- int mode
  ファイルのオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 readRank のノードがファイルをオープンし、
  　　他のノードから情報を集めてデータを書き出す。
  　　このとき、識別子 readRank のノードも出力情報を持っている。
- SplBool cmp_flag
  圧縮フラグ
  　SplTrue：圧縮データとして書き出す。
  　SplFalse：非圧縮データとして書き出す
- SplBool ext_rec_flag
  拡張レコードの記述フラグ
  　SplTrue：拡張レコードを出力する
  　SplFalse：拡張レコードを出力しない
- size_t block_size
  ブロックサイズ
- const splSbx* obj

splSbx オブジェクト

- int writeRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

ファイルの読込みに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  fname もしくは obj が NULL

5.2.4. SPX ファイル操作関連関数

splInitSpx

（用　途）

splSpx 構造体の初期化

（書　式）

```
#include “spl/splfio.h”
  SplBool   splInitSpx     （   splSpx* obj   ）
```

（説　明）

・ splSpx* obj
  初期化する splSpx 構造体

（返り値）

初期化に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

・ SPL_ERR_INVALID_PARAMETER
  obj が NULL

splFinalizeSpx

（用　途）

splSpx 構造体のファイナライズ

（書　式）

```
#include “spl/splfio.h”
  SplBool   splFinalizeSpx   (   splSpx* obj,
                                 SplBool mem_free_flag   )
```

（説　明）

- splSpx* obj
  ファイナライズを行う splSpx 構造体
- SplBool mem_free_flag
  splSpx 構造体の内部で動的に確保されたメモリ領域の開放処理を行うかどうかのフラグ。
  SplTrue：開放する
  SplFalse：開放しない

（返り値）

ファイナライズに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj が NULL

splSpxSetValues

（用　途）

splSpx 構造体に値をセットする

（書　式）

```
#include “spl/splfio.h”
  SplBool   splSpxSetValues   (   splSpx* obj,
                                  unsigned int dims,
                                  unsigned int vlen,
                                  unsigned int dtype,
                                  size_t gc,
                                  size_t* sta_idx,
                                  size_t* size,
                                  unsigned char* dt   )
```

（説　明）

- splSpx* obj
  値設定の対象となる splSpx オブジェクト
- unsigned int dims
  次元数
- unsigned int vlen
  ベクトル長
- unsigned int dtype
  データタイプ
- size_t gc
  仮想セル数
- size_t* sta_idx
  始点インデックス
- size_t* size
  ボクセルサイズ
- unsigned char* dt
  データ配列への参照

（返り値）

obj への情報の設定に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、sta_idx、size または dt のいづれかが NULL

splGetSpxInfo

（用　途）

SPX ファイルに記述されている値を splSpx 構造体に取得する

（書　式）

```
#include “spl/splfio.h”
  SplBool   splGetSpxInfo   (   const char* fname,
                                splSpx* obj,
                                int mode,
                                int readRank,
                                SPL_Comm grp   )
```

（説　明）

- const char* fname
  情報を取得するファイルの名前
- splSpx* obj
  splSpx 構造体
- int mode
  ファイルのオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 readRank のノードがファイルをオープンし、
  　　読み込んだ値を他のノードにブロードキャストする。
  　　このとき、識別子 readRank のノードも情報を保持する。
- int readRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

情報の取得に成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  obj、sta_idx、size または dt のいづれかが NULL

splReadSpx

（用　途）

SPX ファイルかたらの読込み

（書　式）

```
#include “spl/splfio.h”
  SplBool   splReadSpx   (   splSvFileList* file_list,
                             int mode,
                             splSpx* obj,
                             int readRank,
                             SPL_Comm grp   )
```

（説　明）

- splSvFileList* file_list
  読み込むファイルのリスト
- int mode
  ファイルオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 readRank のノードがファイルをオープンし、
  　　読み込んだ値を他のノードにブロードキャストする。
  　　このとき、識別子 readRank のノードも情報を保持する。
- splSpx* obj
  読込先の splSpx 構造体
- int readRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

ファイルの読込みに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  file_list もしくは obj が NULL

splWriteSpx

（用　途）

SPX ファイルへの書出し

（書　式）

```
#include “spl/splfio.h”
  SplBool   splWriteSpx   (   const char* fname,
                              int mode,
                              SplBool cmp_flag,
                              SplBool ext_rec_flag,
                              size_t block_size,
                              const splSpx* obj,
                              int writeRank,
                              SPL_Comm grp   )
```

（説　明）

- const char* fname
  書出すファイルの名前
- int mode
  ファイルのオープンモード
  　０：全てのノードがファイルをオープンする
  　１：識別子 writeRank,のノードがファイルをオープンし、
  　　他のノードから情報を集めてデータを書き出す。
  　　このとき、識別子 writeRank,のノードも出力情報を持っている。
- SplBool cmp_flag
  圧縮フラグ
  　SplTrue：圧縮データとして書き出す。
  　SplFalse：非圧縮データとして書き出す
- SplBool ext_rec_flag
  拡張レコードの記述フラグ
  　SplTrue：拡張レコードを出力する
  　SplFalse：拡張レコードを出力しない
- size_t block_size
  ブロックサイズ
- const splSpx* obj

splSpx オブジェクト

- int writeRank
  ファイルをオープンするノード識別子
  （mode が１もしくは２のとき有効）
- SPL_Comm grp
  コミュニケーショングループ

（返り値）

ファイルの読込みに成功すると SplTrue、失敗すると SplFalse を返す。

（エラー）

- SPL_ERR_INVALID_PARAMETER
  fname もしくは obj が NULL

5.2.5.

（用　途）

（書　式）

#include “”

(

（説　明）

（返り値）

（エラー）

5.3. ファイルフォーマット

5.3.1. SBX ファイルフォーマット

概略

SBX ファイルフォーマットは、シンプルボクセル構造におけるマスク情報を記述するためのファイルフォーマットです。３次元（1 次元、2 次元も表現可能）の計算空間を１ボクセル辺り 1、2、４バイトのいづれかのデータ容量で表現することができます。また SBX ファイルはデータレコードに対して圧縮処理を施した部分圧縮ファイル（非圧縮データの記述も可）です。

レコード名	意味	データ記述
データ属性レコード	データの属性情報を記述する	必須
サイズレコード	ボクセルサイズを記述する	必須
原点座標レコード	計算領域の原点座標を記述する	必須
ピッチレコード	ボクセルピッチを記述する	必須
データレコード	データを記述する	必須
付随情報レコード	計算領域全体のボクセルサイズと計算領域全体に対する相対位置を特定するための始点インデックスを記述する（領域分割時）	任意

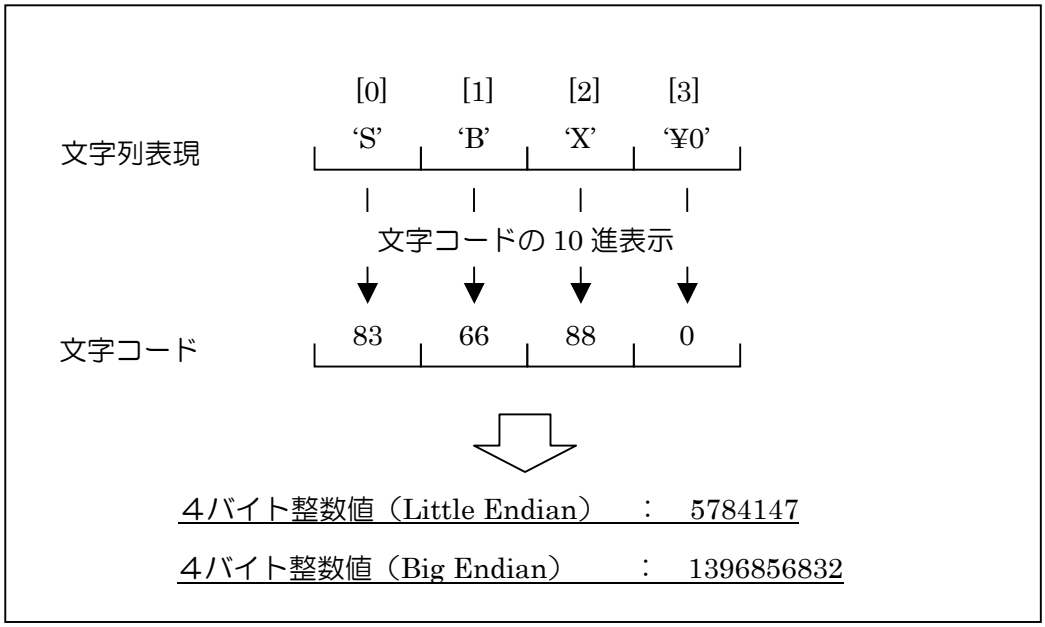
データ属性レコード

データの属性を記述するレコードです。

名称	表現	サイズ	説明
FMT	整数	4 byte	ファイルフォーマット識別コード（※１）
DIMS	整数	4 byte	次元数（＝１ or ２ or ３　※２）
VLEN	整数	4 byte	ベクトル長（＝１　※３）
DTYPE	整数	4 byte	データ種別（※４）
GC	整数	4 byte	ガイドセル数
RLEN	整数	4 byte	実数データのサイズ（※５）
CRDDEF	整数	4 byte	原点座標の定義位置（※６）
AUX	整数	4 byte	拡張仕様（※７）
BLKSIZE	整数	8 byte	ブロックデータサイズ（※８）

（※１）ファイルフォーマット識別コード

ファイルフォーマット識別コードにはデータの記述フォーマットを識別するための識別子を指定します。SBX ファイルでは 4 バイトの整数値に Little Endian の環境では「5784147」が、Big Endian の環境では「1396856832」が設定されます。



ファイルフォーマット識別コード

（※２）次元数

次元数は空間定義を以下で設定します。

空間定義	設定値
１次元	１
２次元	２
３次元	３

（※３）ベクトル長

１定義点がもつデータの数を設定します。ＳＢＸデータは常に固定値（＝１）となります。

（※４）データ種別

データの種別を記述します。データ種別とは１ボクセル辺りのデータ量を示す値です。SBX ファイルでは１ボクセル辺りのデータ量を 1、２、４バイトのいづれかとしています。設定値は以下です。

データ型種別	設定値
1 バイト（unsigned char）	１
2 バイト（unsigned short）	２
4 バイト（unsigned int）	４

1 バイトの場合は設定を”1”としてメモリ上に uchar 型の配列として読込みます。
2 バイトの場合は設定を”2”としてメモリ上に ushort 型の配列として読込みます。
4 バイトの場合は設定を”4”としてメモリ上に uint 型の配列として読込みます。

（※５）実数データのサイズ

SBX ファイルでの実数データとは、

- 原点座標レコードに記述される座標値
- ピッチレコードに記述されるボクセルピッチの値

を指します。以下の値を設定します。

実数データのサイズ	設定値
単精度浮動小数点数（float：4byte）	４
倍精度浮動小数点数（double：8byte）	８

（※６）原点座標の定義位置

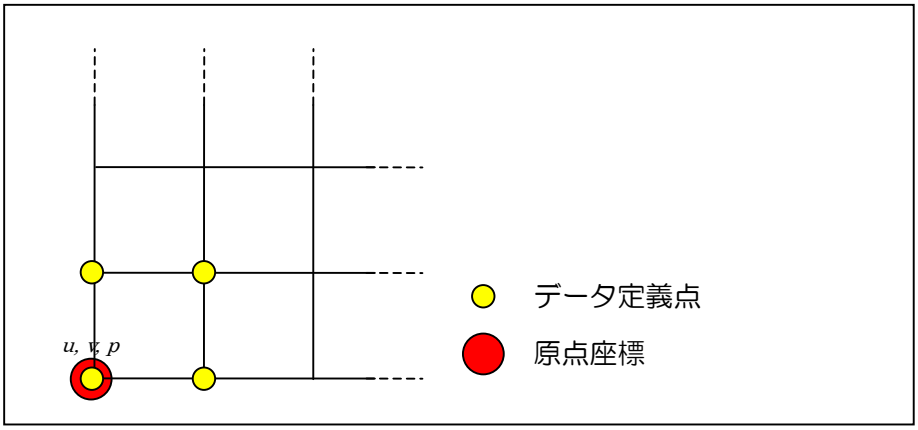
ＳＢＸファイルは原点座標の定義点を以下のいづれかで定義します。

原点座標の定義位置	設定値
Regular	１
Collocate	２

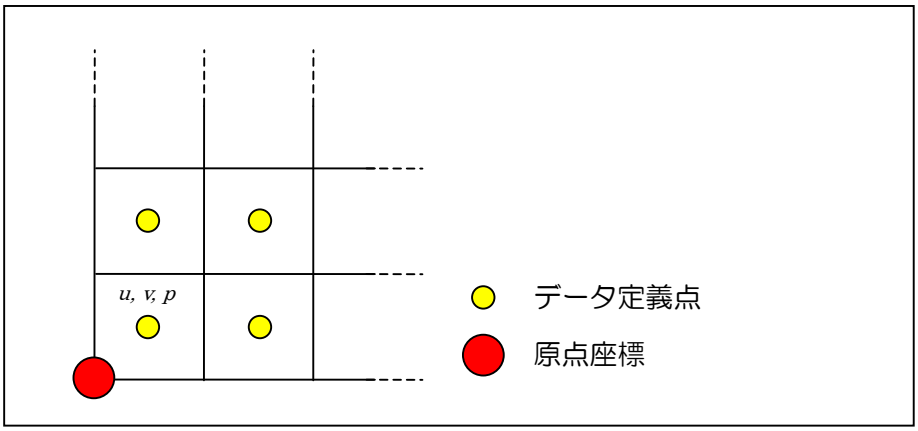
Staggered（1）	3
Staggered（2）	4

各定義での原点座標定義点は以下になります。

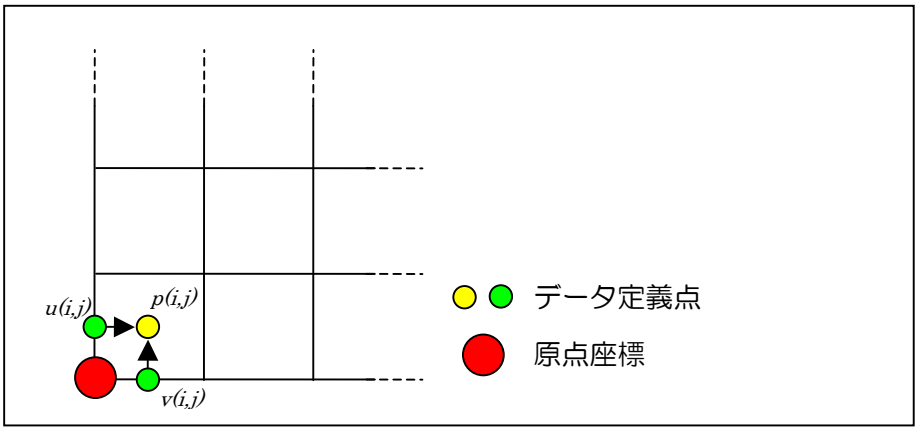
（1） Regular



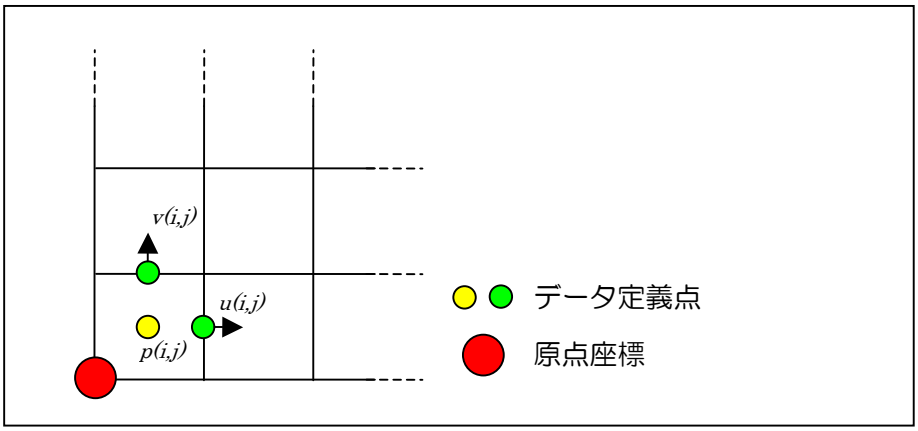
（2） Collocate



（３） Staggered（１）



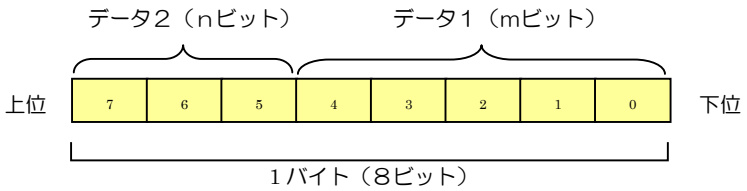
（４） Staggered（２）



（※７）SBX ファイルの拡張仕様

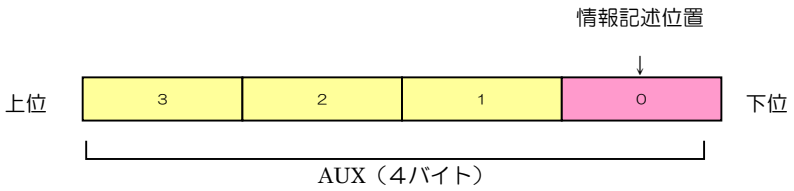
（１）データ種別（DTYPE）の値が１の場合の拡張仕様

データ種別（DTYPE）の値が１（１ボクセルあたりのデータ量が１バイト）の場合には、８ビットを１つのデータで占有する使用方法のほかに、２つのデータをｍビットとｎビット（ｍ＋ｎ＝８）で記述することが可能です（8 ビットを１つのデータ占有するのはｎ＝０の場合になります）。



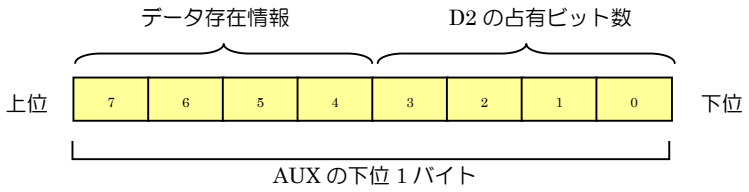
１ボクセル（１バイト）データへの２データの記述イメージ

情報は AUX の下位１バイトを使用して記述されます。



情報の記述位置

AUX の下位１バイトはその内部を 4 ビットずつ２つのフィールドに分割してそれぞれのフィールドに情報を記述します。
ここで、１ボクセルあたりのデータ（１バイト）に下位mビットにデータ D1、上位ｎビットにデータ D2 を記述する場合、以下の情報が各ビットフィールドに記述されます。



１ボクセル（１バイト）データへの２データの記述イメージ

AUX の下位１バイトの記述仕様

下位４ビット	D２の占有ビット数（n）
上位４ビット	データ存在情報（ ０ or １ or ２ ）（※）

（※）データ存在情報

データ存在情報には、D1、D2 共に有効な情報が記述されている場合は０、D1 のみに有効な情報が記述されている場合は１、D2 のみに有効な情報が記述されている場合は２が設定されます。

（※８）ブロックデータサイズ

データレコードの１ブロックに記述されるデータの数（データ定義点の数）を指定します。データレコードは任意数のデータを１ブロックとして圧縮されており、それら圧縮ブロックを圧縮後ブロックサイズとともに複数記述する構成をとります。ただし非圧縮の場合はデータブロックサイズには”0”が設定されます。

	設定値	意味
非圧縮	0	データは非圧縮で記述されている
圧縮	１～	ブロックは設定値のデータ数を １ブロックとして圧縮されている

例えば、データ種別が４（１データが４バイト）でブロックデータサイズに”1024”と記述されている場合、１ブロックあたりの圧縮前のデータ量は”４０９６”（１０２４×４×１）バイトになります。

サイズレコード

ボクセルサイズ（計算空間のボクセル分割数）を記述するレコードです。

名称	表現	サイズ	説明
IMAX	整数	8 byte	I 方向ボクセル数
JMAX	整数	8 byte	J 方向ボクセル数（＝※１）
KMAX	整数	8 byte	K 方向ボクセル数（＝※２）

（※１）J 方向ボクセル数

1 次元データの場合には J 方向ボクセル数は”1”を記述します。

（※２）K 方向ボクセル数

1 次元データもしくは 2 次元データの場合は K 方向ボクセル数は”1”を記述します。

ここで、サイズレコードに記述されている IMAX,JMAX および KMAX の各値にはデータ属性レコード中に記述されているガイドセルの数（GC）が含まれています。またサイズレコードの各値には、以下の計算式の結果が負の数になることがないように記述されている必要があります。

（計算式） ： IMAX（もしくは JMAX,KMAX） ― GC ＊ ２

ただし、1 次元データの JMAX,KMAX、2 次元データの KMAX には GC の値に関係なく“１”が入ります。

原点座標レコード

計算領域の原点座標を記述するレコードです。

名称	表現	サイズ	説明
XORG	実数	4 or 8 byte（※１）	原点の X 座標
YORG	実数	4 or 8 byte（※１）	原点の Y 座標（※２）
ZORG	実数	4 or 8 byte（※１）	原点の Z 座標（※３）

（※１）データサイズ

属性レコードに記述されている実数データのサイズ(RLEN)によって決まります。RLEN が１（単精度浮動小数点数）の場合は４byte、RLEN が２（倍精度浮動小数点数）の場合は８byte となります。

（※２）原点の Y 座標

1 次元データの場合には原点の Y 座標は不定値となります。

（※３）原点の Z 座標

1 次元データもしくは 2 次元データの場合は原点の Z 座標は不定値となります。

ピッチレコード

ボクセルピッチを記述するレコードです。

名称	表現	サイズ	説明
DX	実数	4 or 8 byte（※１）	I 方向のボクセルピッチ
DY	実数	4 or 8 byte（※１）	J 方向のボクセルピッチ（※２）
DZ	実数	4 or 8 byte（※１）	K 方向のボクセルピッチ（※３）

（※１）データサイズ

属性レコードに記述されている実数データのサイズ(RLEN)によって決まります。RLEN が１（単精度浮動小数点数）の場合は４byte、RLEN が２（倍精度浮動小数点数）の場合は８byte となります。

（※２）J 方向のボクセルピッチ

1 次元データの場合には J 方向ボクセルピッチは不定値となります。

（※３）K 方向のボクセルピッチ

1 次元データもしくは 2 次元データの場合は K 方向ボクセルピッチは不定値となります。

データレコード

データレコードはデータ本体を記述するレコードです。記述するデータはデータ属性レコードのブロックデータサイズ（BLKSIZE）単位で複数個のデータブロックに分割し、それぞれのブロックを個別に圧縮したものが圧縮後のブロックサイズとともに記述されます。ただし非圧縮データを記述する場合（BLKSIZE＝０）は全データを１ブロックとして圧縮せずに記述されます。

データブロックの非圧縮時のデータ長計算式

（非圧縮ブロックのデータ長[byte]）＝（BLKSIZE）×（DTYPE）×（ＶＬＥN）

BLKSIZE：データ属性レコードのデータブロックサイズ

DTYPE　：データ属性レコードのデータ種別

VLEN　　：データ属性レコードのベクトル長（ＳＢＸでは固定値”１”）

データレコードは下記に示すフォーマットで記述されます。

データレコード（ｎブロックの場合）

名称	表現	サイズ	説明
COMPSIZE1	整数	8 byte	N=1 ブロックの圧縮サイズ
COMPDATA1(N=1)	―	COMPSIZE1 byte	N=1 ブロックの圧縮データ
COMPSIZE2	整数	8 byte	N=2 ブロックの圧縮サイズ
COMPDATA2(N=2)	―	COMPSIZE2 byte	N=2 ブロックの圧縮データ
…			
COMPSIZEn(N=n)	整数	8 byte	N=n ブロックの圧縮サイズ
COMPDATAn(N=n)	―	COMPSIZEn byte	N=n ブロックの圧縮データ
END	整数	8 byte	終端（＝０）

また、圧縮データの伸長時のデータ並びは以下となります。

名称	表現	サイズ	説明
DATA(0,0,0)	（※１）	DTYPE の設定による（※２）	格子点(0,0,0)のデータ
DATA(1,0,0)	（※１）	DTYPE の設定による（※２）	格子点(1,0,0)のデータ
DATA(0,1,0)	（※１）	DTYPE の設定による（※２）	格子点(0,1,0)のデータ
DATA(1,1,0)	（※１）	DTYPE の設定による（※２）	格子点(1,1,0)のデータ
…			
DATA(IMAX-1,JMAX-1,KMAX-1)	（※１）	DTYPE の設定による（※２）	格子点(IMAX-1,JMAX-1,KMAX-1)のデータ

（※１）データの表現

データはデータ属性レコードに記述される DTYPE（データ種別）の値によりプログラム中で読み込まれるデータ型が異なります。

DTYPE の値	データ量	表現（データ型）
1	1 byte	unsigned char
2	2 byte	unsigned short
4	4 byte	unsigned int

（※２）１データのサイズとデータ型

データ属性レコードに記述される DTYPE（データ種別）の値です。

付随情報レコード

領域分割されたデータが SBX ファイルに記述されている場合、計算領域全体のボクセルサイズを記述するレコードです。本レコードの記述は任意です。

名称	表現	サイズ	説明
WIMAX	整数	8 byte	I 方向全体ボクセル数
WJMAX	整数	8 byte	J 方向全体ボクセル数（＝※１）
WKMAX	整数	8 byte	K 方向全体ボクセル数（＝※２）
ISTA	整数	8 byte	I 方向の始点インデックス
JSTA	整数	8 byte	J 方向の始点インデックス（＝※３）
KSTA	整数	8 byte	K 方向の始点インデックス（＝※４）

（※１）J 方向ボクセル数

1 次元データの場合には J 方向全体ボクセル数は”1”を記述します。

（※２）K 方向ボクセル数

1 次元データもしくは 2 次元データの場合は K 方向全体ボクセル数は”1”を記述します。

（※３）J 方向始点インデックス

1 次元データの場合には J 方向始点インデックスは”0”を記述します。

（※４）K 方向始点インデックス

1 次元データもしくは 2 次元データの場合は K 方向始点インデックスは”0”を記述します。

5.3.2. SPX ファイルフォーマット

概略

SPX ファイルフォーマットは、シンプルボクセル構造における計算結果等浮動小数点数データを記述するためのファイルフォーマットです。１ファイルに１タイムスライスのデータを１つ記述します。３次元（1 次元、2 次元も表現可能）の計算空間を１ボクセル辺り単精度もしくは倍精度のデータで表現することができます。また SPX ファイルはデータレコードに対して圧縮処理を施した部分圧縮ファイルです。

レコード名	意味	データ記述
データ属性レコード	データの属性情報を記述する	必須
サイズレコード	ボクセルサイズを記述する	必須
原点座標レコード	計算領域の原点座標を記述する	必須
ピッチレコード	ボクセルピッチを記述する	必須
時刻レコード	タイムステップと時刻を記述する	必須
データレコード	データを記述する	必須
付随情報レコード	計算領域全体のボクセルサイズと計算領域全体に対する相対位置を特定するための始点インデックスを記述する（領域分割時）	任意

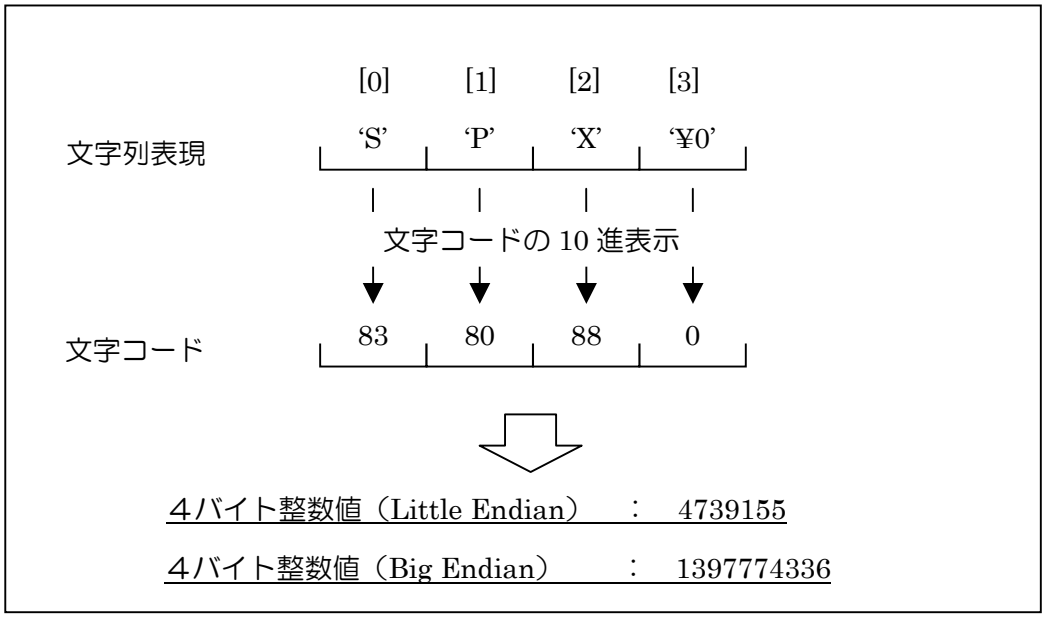
データ属性レコード

データの属性を記述するレコードです。

名称	表現	サイズ	説明
FMT	整数	4 byte	ファイルフォーマット識別コード（※１）
DIMS	整数	4 byte	次元数（＝１ or ２ or ３ ※２）
VLEN	整数	4 byte	ベクトル長（※３）
DTYPE	整数	4 byte	データ種別（※４）
GC	整数	4 byte	ガイドセルの数
RLEN	整数	4 byte	実数データのサイズ（※５）
CRDDEF	整数	4 byte	原点座標の定義位置（※６）
AUX	整数	4 byte	未使用
BLKSIZE	整数	8 byte	データブロックサイズ（※７）

（※１）ファイルフォーマット識別コード

ファイルフォーマット識別コードにはデータの記述フォーマットを識別するための識別子を指定します。SPX ファイルでは 4 バイトの整数値に Little Endian の環境では「5787731」が、Big Endian の環境では「1397774336」が設定されます。



ファイルフォーマット識別コード

（※２）次元数

次元数は空間定義を以下で設定します。

空間定義	設定値
1次元	1
2次元	2
3次元	3

（※3）ベクトル長

1定義点がもつデータの数を設定します。

（※4）データ種別

SPX ファイルでは、スカラーデータもしくはベクトルデータのいづれかを記述できます。データ種別は記述されるデータがスカラーデータなのかベクトルデータなのかを区別するための識別子です。それぞれの識別子は以下になります。

データ種別	値
スカラーデータ	1
ベクトルデータ	3

（※5）実数データのサイズ

SPX ファイル内に記述できる実数データは以下です。

- 原点座標レコードに記述される座標値
- ピッチレコードに記述されるボクセルピッチの値
- 時刻レコードの時刻の値
- データレコードの値

RLEN は、これらの実数値が単精度なのか倍精度なのかを区別するための識別子です。RLEN には以下の値を設定します。

実数データのサイズ	設定値
単精度浮動小数点数（float：4byte）	4
倍精度浮動小数点数（double：8byte）	8

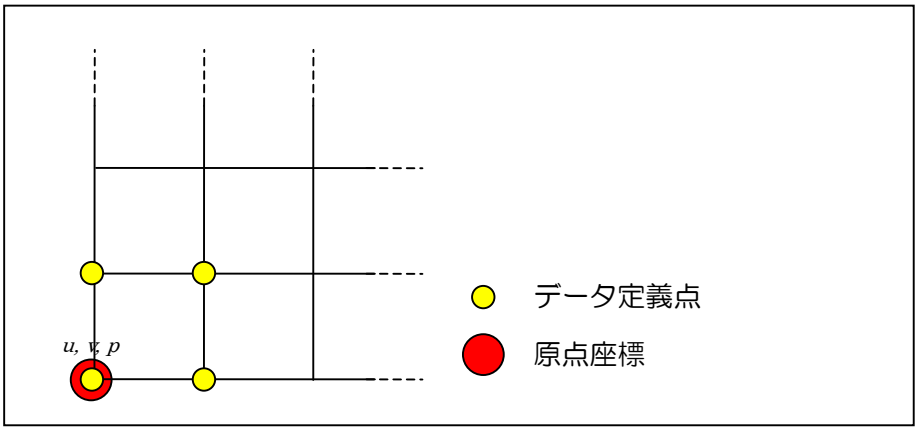
（※６）原点座標の定義位置

SPX ファイルは原点座標の定義点を以下のいづれかで定義します。

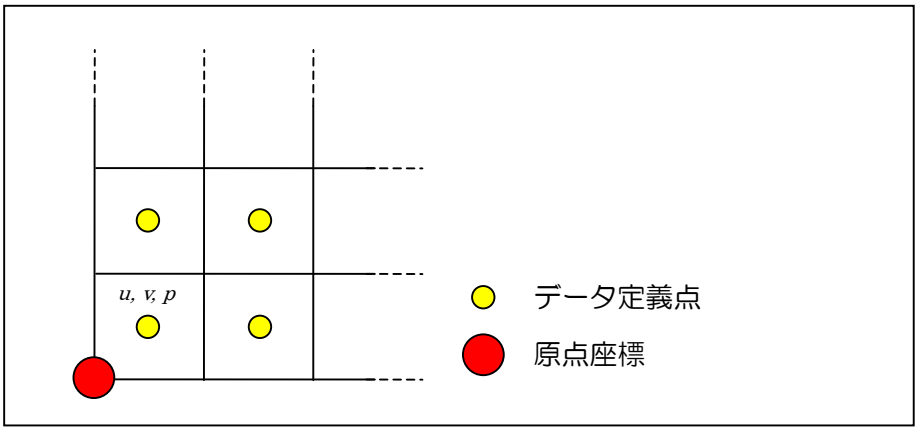
原点座標の定義位置	設定値
Regular	1
Collocate	2
Staggered（1）	3
Staggered（2）	4

各定義での原点座標定義点は以下になります。

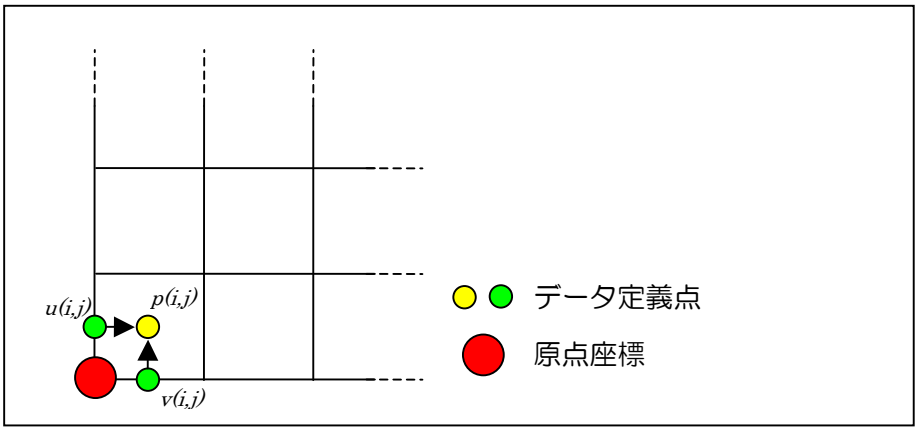
（5） Regular



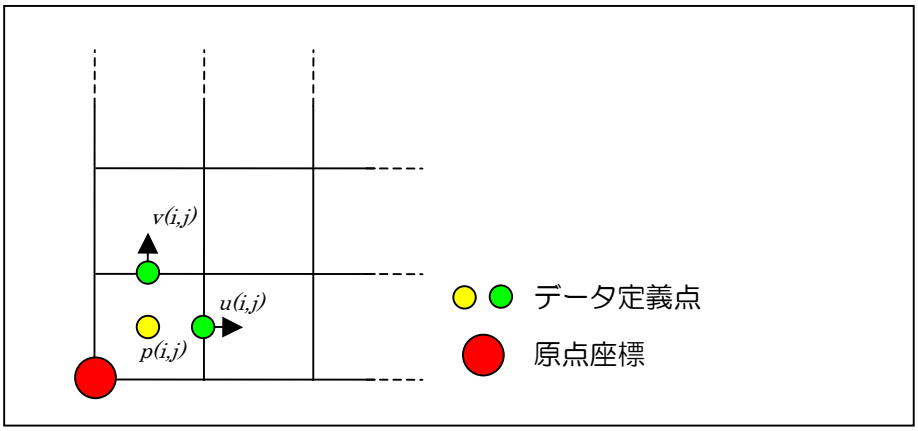
（6） Collocate



（7） Staggered（1）



（8） Staggered（2）



（※7）ブロックデータサイズ

データレコードの１ブロックに記述されるデータの数（データ定義点の数）を指定します。データレコードは任意数のデータを１ブロックとして圧縮されており、それら圧縮ブロックを圧縮後ブロックサイズとともに複数記述する構成をとります。ただし非圧縮の場合はデータブロックサイズには”0”が設定されます。

	設定値	意味
非圧縮	0	データは非圧縮で記述されている
圧縮	1～	ブロックは設定値のデータ数を １ブロックとして圧縮されている

例えば、データ種別が２（ベクトルデータ）で、ベクトル長が３、実数データサイズが８（倍精度）、かつブロックデータサイズに”1024”と記述されている場合、

１ブロックあたりの圧縮前のデータ量は”２４５７６”（１０２４×８×３）バイトになります。

サイズレコード

ボクセルサイズ（計算空間のボクセル分割数）を記述するレコードです。

名称	表現	サイズ	説明
IMAX	整数	8 byte	I 方向ボクセル数
JMAX	整数	8 byte	J 方向ボクセル数（＝※１）
KMAX	整数	8 byte	K 方向ボクセル数（＝※２）

（※１）J 方向ボクセル数

1 次元データの場合には J 方向ボクセル数は”1”を記述します。

（※２）K 方向ボクセル数

1 次元データもしくは 2 次元データの場合は K 方向ボクセル数は”1”を記述します。

ここで、サイズレコードに記述されている IMAX,JMAX および KMAX の各値にはデータ属性レコード中に記述されているガイドセルの数（GC）が含まれています。またサイズレコードの各値には、以下の計算式の結果が負の数になることがないように記述されている必要があります。

（計算式） ： IMAX（もしくは JMAX,KMAX） ― GC ＊ ２

ただし、1 次元データの JMAX,KMAX、2 次元データの KMAX には GC の値に関係なく“１”が入ります。

原点座標レコード

計算領域の原点座標を記述するレコードです。

名称	表現	サイズ	説明
XORG	実数	4 or 8 byte（※１）	原点の X 座標
YORG	実数	4 or 8 byte（※１）	原点の Y 座標（※２）
ZORG	実数	4 or 8 byte（※１）	原点の Z 座標（※３）

（※１）データサイズ

属性レコードに記述されている実数データのサイズ(RLEN)によって決まります。RLEN が１（単精度浮動小数点数）の場合は４byte、RLEN が２（倍精度浮動小数点数）の場合は８byte となります。

（※２）原点の Y 座標

1 次元データの場合には原点の Y 座標は不定値となります。

（※３）原点の Z 座標

1 次元データもしくは 2 次元データの場合は原点の Z 座標は不定値となります。

ピッチレコード

ボクセルピッチを記述するレコードです。

名称	表現	サイズ	説明
DX	実数	4 or 8 byte（※１）	I 方向のボクセルピッチ
DY	実数	4 or 8 byte（※１）	J 方向のボクセルピッチ（※２）
DZ	実数	4 or 8 byte（※１）	K 方向のボクセルピッチ（※３）

（※１）データサイズ

属性レコードに記述されている実数データのサイズ(RLEN)によって決まります。RLEN が１（単精度浮動小数点数）の場合は４byte、RLEN が２（倍精度浮動小数点数）の場合は８byte となります。

（※２）J 方向のボクセルピッチ

1 次元データの場合には J 方向ボクセルピッチは不定値となります。

（※３）K 方向のボクセルピッチ

1 次元データもしくは 2 次元データの場合は K 方向ボクセルピッチは不定値となります。

時刻レコード

ボクセルピッチを記述するレコードです。

名称	表現	サイズ	説明
STEP	整数	8 byte	タイムステップ数
TIME	実数	4 or 8 byte（※１）	時刻

（※１）データサイズ

属性レコードに記述されている実数データのサイズ(RLEN)によって決まります。RLEN が１（単精度浮動小数点数）の場合は４byte、RLEN が２（倍精度浮動小数点数）の場合は８byte となります。

データレコード

データレコードはデータ本体を記述するレコードです。記述するデータはデータ属性レコードのブロックデータサイズ（BLKSIZE）単位で複数個のデータブロックに分割し、それぞれのブロックを個別に圧縮したものが圧縮後のブロックサイズとともに記述されます。ただし非圧縮データを記述する場合（BLKSIZE＝０）は全データを１ブロックとして圧縮せずに記述されます。

データブロックの非圧縮時のデータ長計算式

（非圧縮ブロックのデータ長[byte]）＝（BLKSIZE）×（VLEN）×（RLEN）

BLKSIZE：データ属性レコードのデータブロックサイズ

VLEN　：データ属性レコードのベクトル長

RLEN　：データ属性レコードの実数データサイズ

データレコードは下記に示すフォーマットで記述されます。

データレコード（ｎブロックの場合）

名称	表現	サイズ	説明
COMPSIZE1	整数	8 byte	N=1 ブロックの圧縮サイズ
COMPDATA1(N=1)	―	COMPSIZE1 byte	N=1 ブロックの圧縮データ
COMPSIZE2	整数	8 byte	N=2 ブロックの圧縮サイズ
COMPDATA2(N=2)	―	COMPSIZE2 byte	N=2 ブロックの圧縮データ
…			
COMPSIZEn(N=n)	整数	8 byte	N=n ブロックの圧縮サイズ
COMPDATAn(N=n)	―	COMPSIZEn byte	N=n ブロックの圧縮データ
END	整数	8 byte	終端（＝０）

また、圧縮データの伸長時のデータ並びは以下となります。

スカラーデータ

名称	表現	サイズ	説明
DATA(0,0,0)	実数	RLEN（※１）	格子点(0,0,0)のデータ
DATA(1,0,0)	実数	RLEN（※１）	格子点(1,0,0)のデータ
DATA(0,1,0)	実数	RLEN（※１）	格子点(0,1,0)のデータ
DATA(1,1,0)	実数	RLEN（※１）	格子点(1,1,0)のデータ
…			
DATA(IMAX-1,JMAX-1,KMAX-1)	実数	RLEN（※１）	格子点(IMAX-1,JMAX-1,KMAX-1)のデータ

ベクトルデータ（VLEN＝３のとき）

名称	表現	サイズ	説明
DATA(0,0,0,0)	実数	RLEN（※１）	格子点(0,0,0,0)のデータ
DATA(1,0,0,0)	実数	RLEN（※１）	格子点(1,0,0,0)のデータ
DATA(2,0,0,0)	実数	RLEN（※１）	格子点(2,0,0,0)のデータ
DATA(0,1,0,0)	実数	RLEN（※１）	格子点(0,1,0,0)のデータ
DATA(1,1,0,0)	実数	RLEN（※１）	格子点(1,1,0,0)のデータ
DATA(2,1,0,0)	実数	RLEN（※１）	格子点(2,1,0,0)のデータ
DATA(0,2,0,0)	実数	RLEN（※１）	格子点(0,2,0,0)のデータ
…			
DATA(VLEN-1,IMAX-1,JMAX-1,KMAX-1)	実数	RLEN（※１）	格子点(VLEN-1,IMAX-1,JMAX-1,KMAX-1)のデータ

（※１）１データのサイズとデータ型

データ属性レコードに記述される RLEN（実数データサイズ）の値です。

付随情報レコード

領域分割されたデータが SPX ファイルに記述されている場合、計算領域全体のボクセルサイズを記述するレコードです。本レコードの記述は任意です。

名称	表現	サイズ	説明
WIMAX	整数	8 byte	I 方向全体ボクセル数
WJMAX	整数	8 byte	J 方向全体ボクセル数（＝※１）
WKMAX	整数	8 byte	K 方向全体ボクセル数（＝※２）
ISTA	整数	8 byte	I 方向の始点インデックス
JSTA	整数	8 byte	J 方向の始点インデックス（＝※３）
KSTA	整数	8 byte	K 方向の始点インデックス（＝※４）

（※１）J 方向ボクセル数

1 次元データの場合には J 方向全体ボクセル数は”1”を記述します。

（※２）K 方向ボクセル数

1 次元データもしくは 2 次元データの場合は K 方向全体ボクセル数は”1”を記述します。

（※３）J 方向始点インデックス

1 次元データの場合には J 方向始点インデックスは”0”を記述します。

（※４）K 方向始点インデックス

1 次元データもしくは 2 次元データの場合は K 方向始点インデックスは”0”を記述します。