SXF Ver.3.1 レベル 2 対応
共通ライブラリ機能仕様書

SFC 編

平成 27 年 5 月

一般社団法人 オープン CAD フォーマット評議会

改訂履歴

日付	改訂内容
2008/02	発行
2011/03	Visual C++ 6.0 から Visual C++ 2008 対応に変更
2015/05	関数仕様の引数の型を修正

目　　次

はじめに

本書は、SXF Ver.3.1 レベル 2 対応共通ライブラリについて、SFC ファイルの入出力機能の仕様について記載したものである。

なお、SXF Ver.3.0 レベル 2 に対応した共通ライブラリ仕様書からの主な変更は以下の通りである。

- クロソイドフィーチャの対応
- 弧長寸法フィーチャの対応
- SXF ファイルバージョンの対応

1 開発方針

SXF Ver.3.1 レベル 2 対応共通ライブラリの開発方針は、以下の通りである。

（1）SXF Ver.3.0 レベル 2 対応共通ライブラリとの互換

SXF Ver.3.0 レベル 2 対応共通ライブラリより、利用可能な資産は利用する。

（2）SXF Ver.3.0 レベル 2 対応共通ライブラリインターフェースとの互換

- SXF Ver.3.1 レベル 2 対応共通ライブラリは、SXF Ver.3.0 レベル 2 対応共通ライブラリにクロソイドフィーチャと弧長寸法フィーチャの処理機能の追加を行う。
- SXF 仕様のバージョン情報の入出力機能の追加を行なう。その他の関数のインターフェースは変更しない。
- SXF Ver.2.0 および Ver.3.0 レベル 2 対応共通ライブラリで出力した SFC ファイルも読込み可能とする。

2 用語の定義

(1) SXF 仕様

Scadec data eXchange Format の略で CAD データ交換標準開発コンソーシアムの開発した CAD データ交換仕様全体を指す。

また、SXF 仕様に基づいて出力される物理ファイルは、以下の（2）と（3）の２種類が存在する。

※但し、共通ライブラリでは、物理ファイルの拡張子のは意識していない。

SXF 仕様は、現在改訂を加え Ver.3.1 となった。

(2) STEP ファイル（拡張子.p21）

STEP のルールに準拠したファイル形式で国際的に通用するもの（電子納品時の正式ファイル）

(3) フィーチャコメントファイル（拡張子.sfc）

：コメントの形で書かれたファイル形式。上記ファイル形式での交換を補うもの（Scadec Feature Comment file）

従来の SXF 仕様 Ver.1.0 では、拡張子が .sxf であったが、仕様の総称ＳＸＦと混同する可能性があるため、SXF 仕様 Ver.2.0 より、拡張子を .sfc とした。

(4) フィーチャ仕様

フィーチャ仕様は、ある意味付けをもつ単位で、フィーチャ要素を定義したものである。ＣＡＤシステムの「線分」「円」や「直線寸法」にあたると考えてよい。

(5) フィーチャ構造体

フィーチャ構造体は、フィーチャ仕様に従い、それをデータ化した構造体である。ＣＡＤシステムと共通ライブラリは、このフィーチャ構造体にてデータの入出力を行う。

(6) フィーチャモデル

フィーチャ仕様では、フィーチャ要素同士の階層関係の仕様も定義している。このフィーチャ要素同士の階層関係をフィーチャモデルと呼ぶ。

3 開発する機能の範囲

3-1 機能概要図

図 1 に SXF Ver.2.0 レベル 2 対応共通ライブラリ、図 2 に SXF Ver.3.0 および Ver.3.1 .レベル 2 対応共通ライブラリの機能概要図を記載する。

（1）SXF Ver.2.0 レベル 2 対応共通ライブラリ

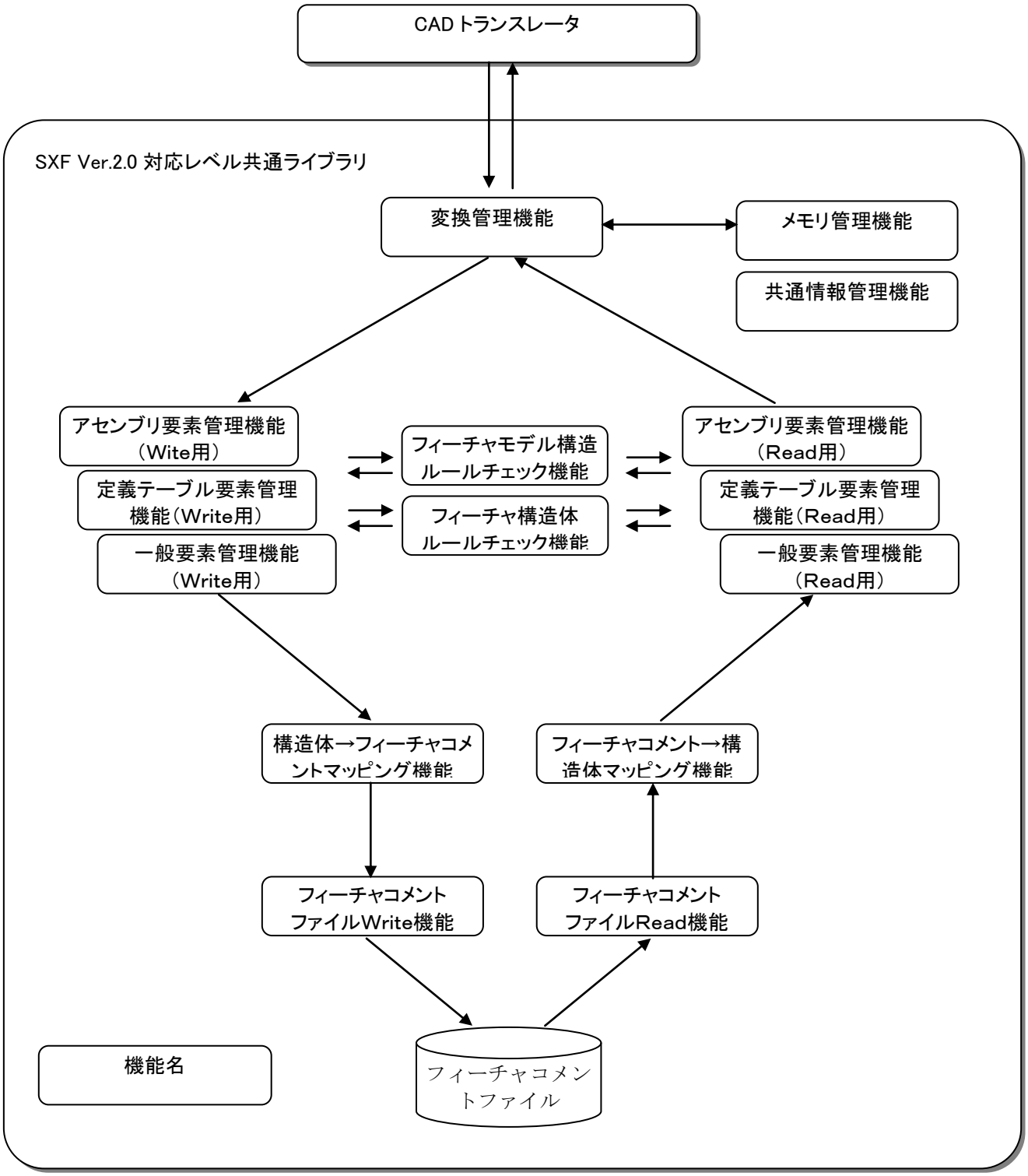


図 1　SXF Ver.2.0 レベル 2 対応共通ライブラリ機能概要図

（2）SXF Ver.3.0 および Ver.3.1 レベル 2 対応共通ライブラリ

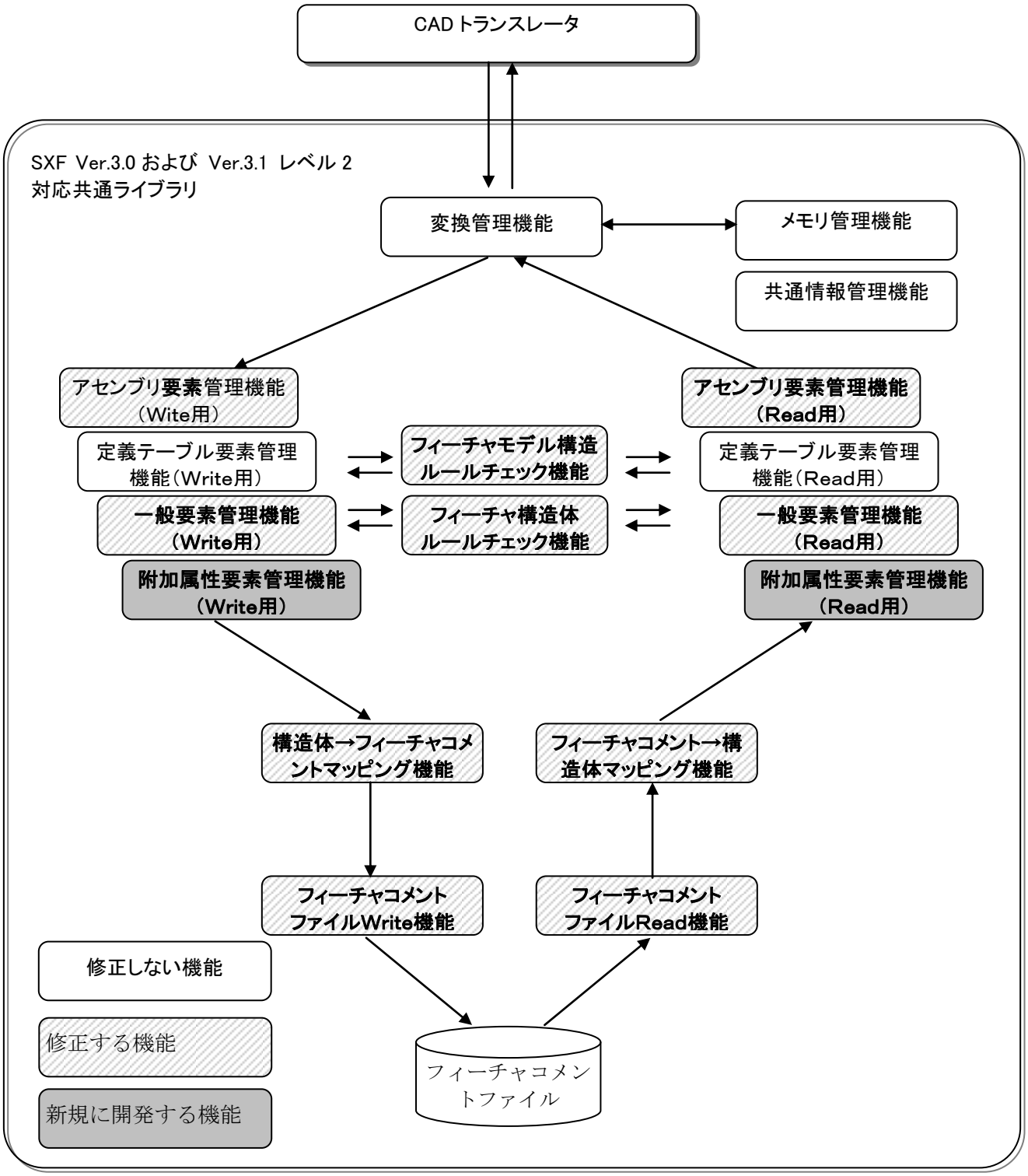


図 2　SXF Ver.3.0 および Ver.3.1 .レベル 2 対応共通ライブラリ機能概要図

3-2 SXF Ver.3.1 レベル 2 対応 SFC 共通ライブラリの機能項目

レベル２用フィーチャコメント用共通ライブラリは、以下の機能項目からなる。

（1）変換管理機能

本機能は、関数を介して、ＣＡＤシステムとフィーチャ構造体のやり取りを行う。関数は、Read/Write 共通で使用するオープン関数・クローズ関数・メッセージ問合せ関数、Read 時に利用するアセンブリ要素 Read 関数・定義テーブル要素 Read 関数・一般要素 Read 関数、フィーチャ構造体削除関数、Write 時利用するアセンブリ要素 Write 関数・定義テーブル要素 Write 関数・一般要素 Write 関数等の、13 の関数から成る。（後述の表 2 参照）

（2）メモリ管理機能（ＣＡＤ側よりの制御を可能とする）

本機能は、Read 時のフィーチャ構造体の管理を行う。

（3）共通情報管理機能

本機能は、STEP ファイルのヘッダーへ書込むまたは、ヘッダーから読み込んだ情報の管理を行う。

（4）フィーチャモデル構造ルールチェック機能

本機能は、レベル２のフィーチャモデル構造仕様のチェックを行う。

（5）フィーチャ構造体ルールチェック機能

本機能は、フィーチャ構造体毎に、フィーチャ毎パラメータの上限値・下限値などルールのチェックを行う。

（6）フィーチャコメント用アセンブリ要素管理機能（Write 用）

本機能は、Write 時に、レベル２の用紙・複合図形定義・複合曲線要素の管理を行う。

（7）フィーチャコメント用アセンブリ要素管理機能（Read 用）

本機能は、Read 時に、レベル２の用紙・複合図形定義・複合曲線要素の管理を行う。

（8）フィーチャコメント用定義テーブル管理機能（Write 用）

本機能は、Write 時に、レベル２のレイヤ・既定義線種・ユーザ定義線種・既定義色・ユーザ定義色・線幅・文字フォントの管理を行う。

（9）フィーチャコメント用定義テーブル管理機能（Read 用）

本機能は、Read 時に、レベル２のレイヤ・既定義線種・ユーザ定義線種・既定義色・ユーザ定義色・線幅・文字フォントの管理を行う。

（10）フィーチャコメント用一般要素管理機能（Write 用）

本機能は、Write 時に、一般要素の管理を行う。

（11）フィーチャコメント用一般要素管理機能（Read 用）

本機能は、Read 時に、一般要素の管理を行う。

（12）フィーチャ構造体→フィーチャコメントマッピング機能(Write)

本機能は、Write 時に、フィーチャ構造体からフィーチャコメントファイル形式へのマッピングを行う。

（13）フィーチャコメント→フィーチャ構造体マッピング機能(Read)

本機能は、Read 時に、フィーチャコメントファイル形式からフィーチャ構造体へのマッピングを行う。

（14）フィーチャコメントファイル Write 機能

本機能は、Write 時に、フィーチャコメントファイルへの書き出しを行う。

（15）フィーチャコメントファイル Read 機能

本機能は、Read 時に、フィーチャコメントファイルの読み込みを行う。

（16）フィーチャコメント用附加属性要素管理機能（Write 用）

本機能は、Write 時に、附加属性要素の管理を行う。

（17）フィーチャコメント用附加属性要素管理機能（Read 用）

本機能は、Read 時に、附加属性要素の管理を行う。

3-3 SXF 仕様 Ver.3.1 での追加仕様への対応項目

（1）クロソイドフィーチャの対応

SXF 仕様 Ver.3.1 で追加された、クロソイドフィーチャの入力処理及び出力処理を追加する。

（2）弧長寸法フィーチャの対応

SXF 仕様 Ver.3.1 で追加された、弧長寸法フィーチャの入力処理及び出力処理を追加する。

4 稼動環境

4-1 ハードウェア

①ＣＰＵ　　1.6GHz 以上

②メモリ　　Windows XP は 384MB 以上、Windows Vista は 768MB 以上

③ハードディスク容量　　2.7GB 以上

4-2 ソフトウェア

①OS　　Windows XP(x86 および x64)SP2 以降、Windows Vista、Windows 7

②開発言語　Windows のプログラム開発言語である Visual C++ 2008 SP1（米マイクロソフト社製）

4-3 .提供内容

（1）共通ライブラリ dll,lib

①内容　　SFC 用共通ライブラリ dll と lib。

②ファイル名　　common_lib.dll、common_lib.lib

（2）フィーチャ構造体用 .h ファイル

- SXFStruct.h
- max.h

（3）ライブラリ関数定義用 .h ファイル

- SXFAPIFuncExt.h

4-4 組込み方法

①dll 呼び出し側のプロジェクト作成

- 新規作成のプロジェクトで「Win32 コンソールアプリケーション」を選び、
- アブリケーションの設定で「共通ヘッダーファイルを追加」に「MFC」を選ぶ。

②dll のコピー

- 以下の２つのファイルを、作成したプロジェクト配下に移動する。

  common_lib.dll

  common_lib.lib

③main への include 追加

- トランスレータ側のメイン関数に以下の１行を追加。

  #include "SXFStruct.h"

  #include "SXFAPIFuncExt.h"

④ビルドする。

- 作成したプロジェクト配下にトランスレータのソースを移動し
- プロジェクトに追加した後、ビルドする。

5 フィーチャモデルについて

5-1 フィーチャ要素の種類

フィーチャモデルを構成するフィーチャ要素は、「定義テーブル型要素」「アセンブリ要素」「一般要素」「附加属性要素」の４つの種類に分別される。この４つのフィーチャ要素の種類によって、共通ライブラリの利用方法が異なる。以下にそれぞれの種類に属するフィーチャ要素を記載する。

各々のフィーチャの仕様は「フィーチャ仕様書」を参照のこと。

5-1-1 定義テーブル要素

定義テーブル要素とは、図形の属性を定義するものである。この要素に属するフィーチャ要素には、以下のものがある。

要素名		フィーチャ型
(1)	レイヤコード	1
(2)	既定義色コード	2
(3)	ユーザ定義色コード	3
(4)	既定義線種コード	4
(5)	ユーザ定義線種コード	5
(6)	線幅コード	6
(7)	文字フォントコード	7

5-1-2 アセンブリ要素

アセンブリ要素とは、用紙や部分図ように、幾何図形要素を配置する要素である。この要素に属するフィーチャ要素には、以下のものがある。

要素名			フィーチャ型
(1)	用紙		1
(2)	複合図形定義	(a) 部分図定義	2
		(b) 作図グループ定義	
		(c) 作図部品定義	
(3)	複合曲線定義		3

5-1-3 一般要素

一般要素とは、線分や寸法線のように図面を表記に用いる要素である。この要素に属するフィーチャ要素には、以下のものがある。

要素名		フィーチャ型
(1)	点マーカ	POINT_MARKER
(2)	線分	LINE
(3)	折線	POLYLINE
(4)	円	CIRCLE
(5)	円弧	ARC
(6)	楕円	ELLIPSE
(7)	楕円弧	ELLIPSE_ARC
(8)	文字要素	TEXT
(9)	スプライン	SPLINE
(10)	クロソイド	CLOTHOID
(11)	複合図形配置	SFIG_LOCATE
(12)	既定義シンボル	EXTERNALLY_DEFINED_SYMBOL
(13)	直線寸法	LINEAR_DIMENSION
(14)	弧長寸法	CURVE_DIMENSION
(15)	角度寸法	ANGULAR_DIMENSION
(16)	半径寸法	RADIUS_DIMENSION
(17)	直径寸法	DIAMETER_DIMENSION
(18)	引き出し線	LABEL
(19)	バルーン	BALLOON
(20)	ハッチング（既定義(外部定義)）	EXTERNALLY_DEFINED_HATCH
(21)	ハッチング（塗り）	FILL_AREA_STYLE_COLOUR
(22)	ハッチング（ユーザ定義）	FILL_AREA_STYLE_HATCHING
(23)	ハッチング（パターン）	FILL_AREA_STYLE_TILES

5-1-4 附加属性要素

附加属性要素とは、図面の属性を定義するものである。この要素に属するフィーチャ要素には、以下のものがある。

要素名		フィーチャ型
(1)	図面表題欄	DRAWING_ATTRIBUTE

5-2 レベル２のフィーチャモデル構造

フィーチャ仕様で定義されたフィーチャモデル構造は、アセンブリ要素を定義し、その上に一般要素を配置する形である。定義テーブル要素は、アセンブリ要素を定義、一般要素を配置する際の属性として使用する。

アセンブリ要素同士の階層が定義されており、アセンブリ要素の用紙を最下層とし、複合図形（部分図）の配置、複合図形（作図グループ）の配置、複合図形（作図部品）の順で配置する必要がある。

なお、複合図形は、部分図・作図グループ・作図部品の３種類に分別されるが、いずれもアセンブリ要素の複合図形定義で定義し、その上に他の要素を配置する。他のアセンブリ要素上に複合図形を配置する際は、一般要素の複合図形配置で配置する。

表 １にフィーチャ要素種別毎の階層関係を示す。

表 １　フィーチャ要素種別毎の階層関係

配置する側 / 配置される側	用紙	複合図形（部分図）	複合図形（作図グループ）	複合図形（作図部品）	一般要素	附加属性要素
用紙		○	○	○	○	○
複合図形（部分図）			○	○	○	○
複合図形（作図グループ）			○：作図ｸﾞﾙｰﾌﾟ同士の階層化が可能	○	○	
複合図形（作図部品）				○：作図部品同士の階層化が可能	○	
一般要素						
補足説明	用紙は、１要素必ず必要。最下層である	部分図は、用紙上に、複数配置可能	作図グループは、用紙・部分図・作図グループ上に複数配置可能	作図部品は、用紙・部分図・作図グループ・作図部品上に複数配置可能	一般要素は、用紙・部分図・作図グループ上・作図部品上に複数配置可能	附加属性要素は、用紙・部分図上で利用可能

6 関数仕様

6-1 関数一覧

表５．１に、レベル２共通ライブラリの関数一覧を記載する。

表 2　関数一覧

No.	関数名	内容	機能概要
1	SXFopen_part21	STEP ファイルのオープン	処理の一番始めに STEP ファイルをオープンする。この際、読み込み(Read)か書き込み(Write)かの方向も指定する。
2	SXFclose_part21	STEP ファイルのクローズ	処理の最後に STEP ファイルをクローズする。
3	SXFread_table	定義テーブル要素の読み込み	読み込みたい定義テーブル要素の型を指定し、フィーチャ構造体とコードを受け取る。この関数は定義テーブル要素が無くなるまで繰り返し呼び出す。この処理は、SXFopen_part21 に続いて、SXFread_assembly_name，SXFread_next_feature より前に必ず行う必要がある。
4	SXFread_assembly_name	アセンブリ要素の読み込み	読み込みたいアセンブリ要素の型を指定し、フィーチャ構造体とコードを受け取る。この関数に続いて SXFread_next_feature を呼び出す事により、そのアセンブリ要素に配置された一般要素または附加属性要素を読み込む事が可能である。この関数は、SXFopen_part21，SXFread_table の処理に続いて呼び出す。また、アセンブリ要素が無くなるまで、アセンブリ要素の読み出し→一般要素または附加属性要素の読み出しを繰り返し行う。
5	SXFread_next_feature	一般要素または附加属性要素の読み込み	SXFread_assembly_name に続いて呼び出す事で、アセンブリ要素に配置された一般要素または附加属性要素を読み込む事が可能である。この関数は、該当のアセンブリ要素上の一般要素または附加属性要素が無くなるまで繰り返し呼び出す。
6	SXFwrite_table	定義テーブル要素の書き込み	書き込みたい定義テーブル要素の型とフィーチャ構造体を指定し、コードを受け取る。この関数は書き込みたい定義テーブル要素が無くなるまで繰り返し呼び出す。この処理は、SXFopen_part21 に続いて、SXFwrite_assembly_name，SXFwrite_next_feature より前に必ず行う必要がある。
7	SXFwrite_assembly_name	アセンブリ要素の書き込み	書き込みたいアセンブリ要素の型とフィーチャ構造体を指定し、コードを受け取る。この関数に続いて SXFwrite_next_feature を呼び出す事により、そのアセンブリ要素に配置したい一般要素を書き込む事が可能である。この関数は、SXFopen_part21，SXFwrite_table の処理に続いて呼び出す。また、書き込みたいアセンブリ要素が無くなるまで、アセンブリ要素の書き込み→一般要素の書き込みを繰り返し行う。
8	SXFwrite_next_feature	一般要素または附加属性要素の書き込み	SXFwrite_assembly_name に続いて呼び出す事で、アセンブリ要素に配置したい一般要素または附加属性要素を書き込む事が可能である。この関数は、該当のアセンブリ要素上に書き込みたい一般要素が無くなるまで繰り返し呼び出す。

9	SXFPopMsg	エラーメッセージ問い合せ	各関数の戻り値が負の場合、共通ライブラリが保持しているメッセージ内容の問い合せを行う
10	SXFdelete_ assembly	読み込み処理時のフィーチャ構造体の削除を行う	SXFread_assembly_name のパラメータである、アセンブリ型とフィーチャ構造体へのポインタを指定し、その構造体を削除する
11	SXFdelete_feature	読み込み処理時のフィーチャ構造体の削除を行う	SXFread_next_feature のパラメータである、フィーチャ型とフィーチャ構造体へのポインタを指定し、その構造体を削除する
12	SXFset_file_version _part21	バージョンの設定	SXF ファイルのバージョンを設定する
13	SXFget_file_version _part21	バージョンの取得	SXF ファイルのバージョンを取得する

6-2 .関数仕様

（1）SXFopen_part 21

機　能	STEP ファイルのオープン 処理の一番始めに STEP ファイルをオープンする。この際、読み込み(Read)か書き込み(Write)かの方向も指定する。
呼び方	SXFopen_part 21 (file_name, rw_flg, tolerance1, tolerance2, divide, level, mode, author , organization, trans_name,version)

引　数	パラメータ	型	説明	
	入力			
	file_name	chr[257]	ファイル名	
	rw_flg	int(4byte)	読/書フラグ	0:読込み、1:書込み
	tolerance1	double(8byte)	長さ用許容誤差（誤差判定に使用）	
	tolerance2	double(8byte)	角度用許容誤差（誤差判定に使用）	
	devide	double(8byte)	許容誤差（レベルダウン時の曲線の分割に使用）	※未使用
	level	int(4byte)	レベル指定	2:ﾚﾍﾞﾙ 2 のみ有功
	mode	int(4byte)	モード指定	1:ﾌｨｰﾁｬｺﾒﾝﾄﾓｰﾄﾞのみ有功
	書込み時入力/読込み時出力			
	author	chr[257]	ファイル作成者	
	organization	chr[257]	作成者所属	
	trans_name	chr[257]	トランスレータ名	
	書込み時無効／読込時出力			
	version	chr[257]	共通ライブラリバージョン	書込み時の指定は無視される

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	【読み込み時】 ・レベル指定は２のみ有効。１が指定された場合はログファイルにワーニングを出力の上処理は続行。それ以外が指定された場合は戻り値にエラーを返却。 ・モード指定は１(フィーチャコメントモード)のみ有効。0: 併用、2:AP202 が指定された場合にはログファイルにワーニングを出力の上処理は続行。それ以外が指定された場合には戻り値にエラーを返却 ・ファイル作成者、作成者所属、トランスレータ名にはカンマ「,」は使用できない

（2）SXFclose_part 21

機　能	STEP ファイルのクローズ 処理の最後に STEP ファイルをクローズする。
呼び方	SXFclose_part 21 ();

引　数	パラメータ	型	説明	
	なし			

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	書き込みの際は最後に必須

（3）SXFread_table

機　能	定義テーブル要素の読み込み 読み込みたい定義テーブル要素の型を指定し、フィーチャ構造体と各コードを受け取る。この関数は定義テーブル要素が無くなるまで繰り返し呼び出す。この処理は、SXFopen_part21 に続いて、SXFread_assembly_name, SXFread_next_feature より前に必ず行う必要がある。
呼び方	SXFread_table(table_type, STRUCT, end_flg)

引　数	パラメータ	型	説明	
	入力			
	table_type	int(4byte)	定義テーブル型	1:レイヤ 2:既定義色 3:ユーザ定義色 4 既定義線種 5:ユーザ定義線種 6:線幅 7:文字フォント
	出力			
	STRUCT *1	void *	フィーチャ構造体変数ポインタ	*1：構造体は予め定義テーブル型に合わせて準備。構造体の詳細は、10 フィーチャ構造体仕様を参照。
	end_flg	int *(4byte)	終了フラグ	0:継続，１：後続なし

戻り値	正数　:コード（正常）コードは共通ライブラリが振る番号である －XXXX: エラーメッセージ参照
制　限	・レイヤコード、ユーザ定義線種コード、ユーザ定義色コード、線幅コードのユーザ定義領域、文字フォントは、本関数が呼び出される毎に昇順な番号が返却される。 ・また、既定義線種コード、既定義色コード、線幅コードは、名称または線幅にあった既定義コードが返却される。

（4）SXFread_assembly_name

機　能	アセンブリ要素の読み込み 読み込みたいアセンブリ要素の型を指定し、フィーチャ構造体と各コードを受け取る。この関数に続いて SXFread_next_feature を呼び出す事により、そのアセンブリ要素に配置された一般要素を読み込む事が可能である。この関数は、SXFopen_part21,　SXFread_table の処理に続いて呼び出す。また、アセンブリ要素が無くなるまで、アセンブリ要素の読み出し→一般要素の読み出しを繰り返し行う。
呼び方	SXFread_assembly_name(assembly_type, STRUCT, end_flg)

引　数	パラメータ	型	説明	
	入力			
	assembly_type	int(4byte)	アセンブリ型	1:用紙 2:複合図形 3:複合曲線
	出力			
	STRUCT *1	void *&	フィーチャ構造体変数ポインタ	*1 : 構造体は予め void*で準備。 構造体の詳細は、10 フィーチャ構造体仕様を参照。
	end_flg	int *(4byte)	終了フラグ	0:継続，１：後続なし

戻り値	正数　:コード（正常）コードは共通ライブラリが振る番号である －XXXX: エラーメッセージ参照
制　限	

（5）SXFread_next_feature

機　能	一般要素または附加属性要素の読み込み SXFread_assembly_name に続いて呼び出す事で、アセンブリ要素に配置された一般要素または附加属性要素を読み込む事が可能である。この関数は、該当のアセンブリ要素上の一般要素または附加属性要素が無くなるまで繰り返し呼び出す。
呼び方	SXFread_next_feature(feature_type ,STRUCT, end_flg)

引　数	パラメータ	型	説明	
	出力			
	feature_type	char *	フィーチャ型	
	STRUCT *1	void *&	フィーチャ構造体変数ポインタ	*1 : 構造体は予め void*で準備。 構造体の詳細は、10 フィーチャ構造体仕様を参照。
	end_flg	int *(4byte)	終了フラグ	0:継続，１：後続なし

戻り値	正数　:フィーチャ ID（正常）フィーチャ ID は、STEP ファイル上のインスタンス ID －XXXX: エラーメッセージ参照
制　限	

（6）SXFwrite_table

機　能	定義テーブル要素の書き込み 書き込みたい定義テーブル要素の型とフィーチャ構造体を指定し、各コードを受け取る。この関数は書き込みたい定義テーブル要素が無くなるまで繰り返し呼び出す。この処理は、SXFopen_part21 に続いて、SXFwrite_assembly_name，SXFwrite_next_feature より前に必ず行う必要がある。
呼び方	SXFwrite_table(table_type, STRUCT)

引　数	パラメータ	型	説明	
	入力			
	table_type	int(4byte)	定義テーブル型	1:レイヤ 2:既定義色 3:ユーザ定義色 4 既定義線種 5:ユーザ定義線種 6:線幅 7:文字フォント

	STRUCT *1	void *	フィーチャ構造体変数ポインタ	*1：構造体は予め定義テーブル型に合わせて準備。構造体の詳細は、10 フィーチャ構造体仕様を参照。

戻り値	正数　　:コード（正常）コードは共通ライブラリが振る番号である －XXXX: エラーメッセージ参照
制　限	

（7）SXFwrite_assembly_name

機　能	アセンブリ要素の書き込み 書き込みたいアセンブリ要素の型とフィーチャ構造体を指定し、コードを受け取る。この関数に続いて SXFwrite_next_feature を呼び出す事により、そのアセンブリ要素に配置したい一般要素を書き込む事が可能である。この関数は、SXFopen_part21，SXFwrite_table の処理に続いて呼び出す。また、書き込みたいアセンブリ要素が無くなるまで、アセンブリ要素の書き込み→一般要素の書き込みを繰り返し行う。
呼び方	SXFwrite_assembly_name(assembly_type,STRUCT)

引　数	パラメータ	型	説明	
	入力			
	assembly_type	int(4byte)	アセンブリ型	1:用紙 2:複合図形 3::複合曲線
	STRUCT *1	void *	フィーチャ構造体変数ポインタ	*1：構造体は予めアセンブリ型に合わせて準備。構造体の詳細は、10 フィーチャ構造体仕様を参照。

戻り値	正数　　:コード（正常）コードは共通ライブラリが振る番号である －XXXX: エラーメッセージ参照
制　限	

（8）SXFwrite_next_feature

機　能	一般要素または附加属性要素の書き込み SXFwrite_assembly_name に続いて呼び出す事で、アセンブリ要素に配置したい一般要素または附加属性要素を書き込む事が可能である。この関数は、該当のアセンブリ要素上に書き込みたい一般要素または附加属性要素が無くなるまで繰り返し呼び出す。
呼び方	SXFwrite_next_feature(feature_type, STRUCT)

引　数	パラメータ	型	説明	
	入力			
	feature_type	char *	フィーチャ型	グループは当機能を使用しない
	STRUCT *1	void *	フィーチャ構造体変数ポインタ	*1：構造体は予めフィーチャ型に合わせて準備。構造体の詳細は、10 フィーチャ構造体仕様を参照。

戻り値	正数　:フィーチャ ID（正常）フィーチャ ID は、STEP ファイル上のインスタンス ID －XXXX: エラーメッセージ参照
制　限	

（9）SXFPopMsg

機　能	エラーメッセージの問い合せ 各関数の戻り値が負の場合、共通ライブラリが保持しているメッセージ内容の問い合せを行う
呼び方	SXFPopMsg(feature_type, msgno,message)

引　数	パラメータ	型	説明	
	出力			
	feature_type	char *&	フィーチャ型	グループは当機能を使用しない
	msgno	int *(4byte)	メッセージ番号	
	message	char *&	メッセージ内容	

戻り値	正数　：
制　限	

（10）SXFdelete_assembly

機　能	読み込み処理時にフィーチャ構造体の削除を行う SXFread_assembly_name のパラメータである、アセンブリ型とフィーチャ構造体へのポインタを指定し、その構造体を削除する。
呼び方	SXFdelete_assembly (assembly_type , STRUCT,)

引　数	パラメータ	型	説明	
	入力			
	assembly_type	int(4byte)	アセンブリ型	1:用紙 2:複合図形 3::複合曲線
	STRUCT	void *&	フィーチャ構造体変数ポインタ	構造体の詳細は、10 フィーチャ構造体仕様を参照。

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	本関数で削除できるのは､読み込み処理時のアセンブリ要素のフィーチャ構造体のみである。一般要素のフィーチャ構造体は”SXFdelete_feature”で削除する。また定義テーブル要素のフィーチャ構造体の削除は、従来通り共通ライブラリ側で行う。

（11）SXFdelete_feature

機　能	読み込み処理時にフィーチャ構造体の削除を行う SXFread_next_feature のパラメータである、フィーチャ型とフィーチャ構造体へのポインタを指定し、その構造体を削除する。
呼び方	SXFdelete_feature (feature_type , STRUCT,)

引　数	パラメータ	型	説明	
	入力			
	feature_type	char *	フィーチャ型	
	STRUCT	void *&	フィーチャ構造体変数ポインタ	構造体の詳細は、10 フィーチャ構造体仕様を参照。

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	本関数で削除できるのは、読み込み処理時の一般要素のフィーチャ構造体のみである。アセンブリ要素のフィーチャ構造体は”　SXFdelete_assembly”で削除する。定義テーブル要素のフィーチャ構造体の削除は、従来通り共通ライブラリ側で行う。

（12）SXFset_file_version_part21

機　能	SXF ファイルのバージョンを設定する。
呼び方	SXFset_file_version_part21(file_version)

引　数	パラメータ	型	説明	
	入力			
	file_version	char[257]	SXF ファイルバージョン	

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	SXFopen_part21 の関数コール前に呼び出す。

（13）SXFget_file_version_part21

機　能	SXF ファイルのバージョンを取得する。
呼び方	SXFget_file_version_part21(file_version)

引　数	パラメータ	型	説明	
	出力			
	file_version	char[257]	SXF ファイルバージョン	

戻り値	0　　:正常 －XXXX: エラーメッセージ参照
制　限	SXFopen_part21 の関数コール後に呼び出す。

6-3 関数と機能の対応

表４．２に、レベル２共通ライブラリの関数と機能の対応を示す。

表 3　関数と機能の関係

No.	関数名	関数の内容		機能名
1	SXFopen_part21	STEP ファイルのオープン	（1）	変換管理機能
			（3）	共通情報管理機能
			（１５）	フィーチャコメントファイル Read 機能
			（7）	フィーチャコメント用アセンブリ要素管理機能（Read 用）
			（9）	フィーチャコメント用定義テーブル管理機能（Read 用）
			（１１）	フィーチャコメント用一般要素管理機能（Read 用）
2	SXFclose_part21	STEP ファイルのクローズ	（1）	変換管理機能
3	SXFread_table	定義テーブル要素の読み込み	（1）	変換管理機能
			（9）	フィーチャコメント用定義テーブル管理機能（Read 用）
			（１３）	フィーチャコメント→フィーチャ構造体マッピング機能(Read)
4	SXFread_assembly_name	アセンブリ要素の読み込み	（1）	変換管理機能
			（7）	フィーチャコメント用アセンブリ要素管理機能（Read 用）
			（１３）	フィーチャコメント→フィーチャ構造体マッピング機能(Read)
5	SXFread_next_feature	一般要素･附加属性要素の読み込み	（1）	変換管理機能
			（１１）	フィーチャコメント用一般要素管理機能（Read 用）
			（１７）	フィーチャコメント用附加属性要素管理機能（Read 用）
			（１３）	フィーチャコメント→フィーチャ構造体マッピング機能(Read)
6	SXFwrite_table	定義テーブル要素の書き込み	（1）	変換管理機能
			（8）	フィーチャコメント用定義テーブル管理機能（Write 用）
			（１２）	フィーチャ構造体→フィーチャコメントマッピング機能(Write)
			（１４）	フィーチャコメントファイル Write 機能
7	SXFwrite_assembly_name	アセンブリ要素の書き込み	（1）	変換管理機能
			（6）	フィーチャコメント用アセンブリ要素管理機能（Write 用）
			（１２）	フィーチャ構造体→フィーチャコメントマッピング機能(Write)
			（１４）	フィーチャコメントファイル Write 機能
8	SXFwrite_next_feature	一般要素･附加属性要素の書き込み	（1）	変換管理機能
			（１０）	フィーチャコメント用一般要素管理機能（Write 用）
			（１６）	フィーチャコメント用附加属性要素管理機能（Write 用）
			（１２）	フィーチャ構造体→フィーチャコメントマッピング機能(Write)
			（１４）	フィーチャコメントファイル Write 機能
9	SXFPopmsg	エラーメッセージ問い合せ	（1）	変換管理機能
10	SXFdelete_assembly	読み込み処理時のフィーチャ構造体の削除を行う	（1）	変換管理機能
			（2）	メモリ管理機能
11	SXFdelete_feature	読み込み処理時のフィーチャ構造体の削除を行う	（1）	変換管理機能
			（2）	メモリ管理機能

12	SXFset_file_version_part21	バージョンの設定	（３）	共通情報管理機能
13	SXFget_file_version_part21	バージョンの取得	（３）	共通情報管理機能

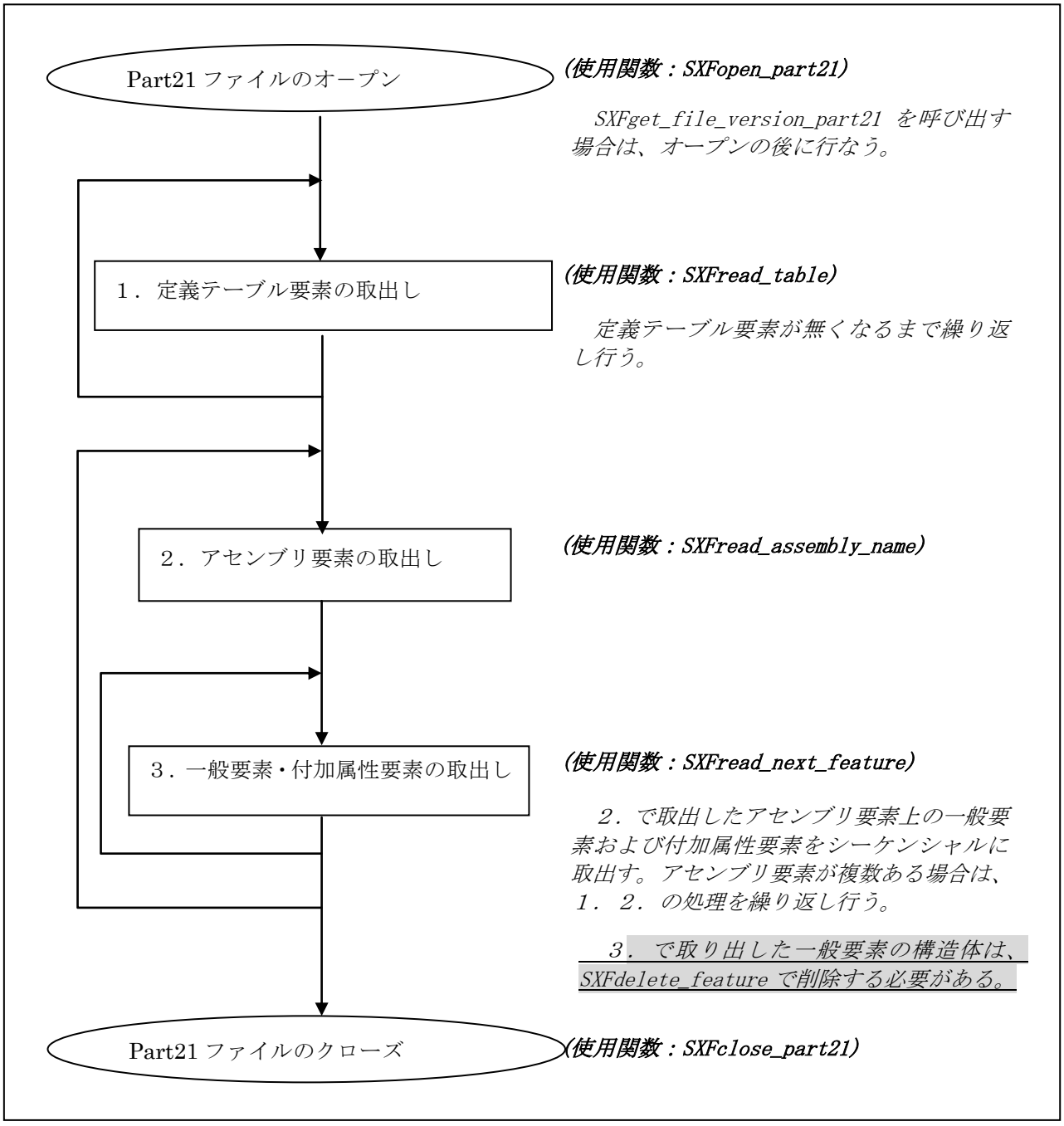
※機能名の番号は、２.２SXF Ver.3.1 レベル 2 対応 SFC 共通ライブラリの機能項目の見出し番号である。

7 関数の利用法

7-1 Ｐａｒｔ２１ファイルの Read 処理

7-1-1 処理の流れ

Read 側の処理の流れを以下の図に示す。処理は必ず以下の流れである必要がある。



7-1-2 定義テーブル要素の取出し(使用関数：SXFread_table)

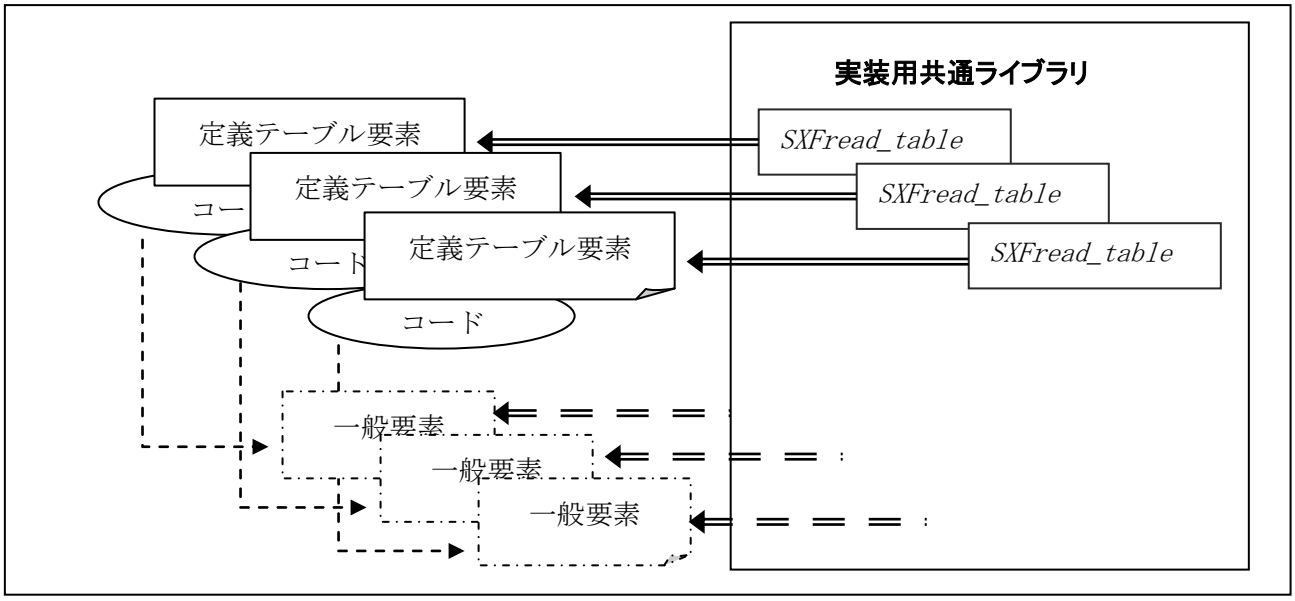
①定義テーブル要素取出し関数を呼び出し、「定義テーブル要素」構造体と「コード」をシーケンシャルに継続要素がなくなるまで受取る。構造体は予め定義テーブル型に合わせ準備。

②シーケンシャル読み出し中に、他の関数を呼び出した場合、それ以降継続して定義テーブル要素は呼び出せない。

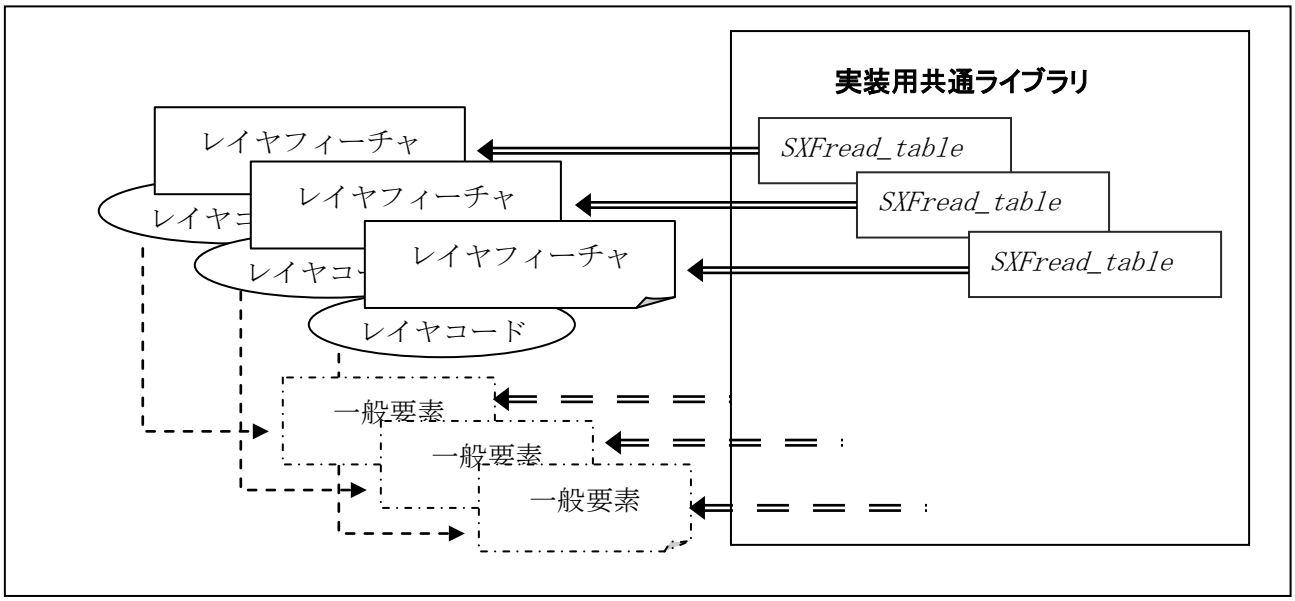
③受取ったコードは、一般要素のパラメータに設定されているコードである。

④コードには、レイヤコード、色コード、線種コード、線幅コード、文字フォントコードがある。

以下に定義テーブル要素の取出し手順を示す。



以下に定義テーブル要素がレイヤコードの場合を示す。



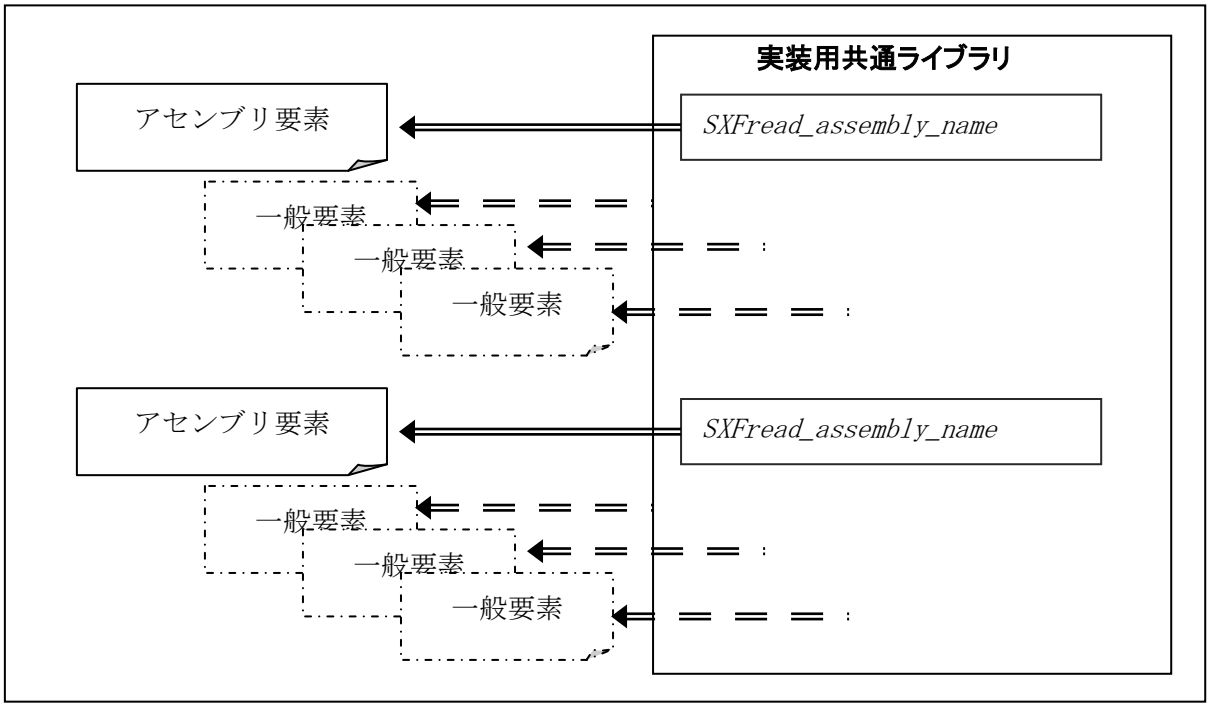
7-1-3 アセンブリ要素の取出し(使用関数：SXFread_assembly_name)

①アセンブリ要素取出し関数を呼び出し、「アセンブリ要素」構造体を受取る。構造体は予め void*で準備。取出した構造体は SXFdelete_ assembly で削除する必要がある。

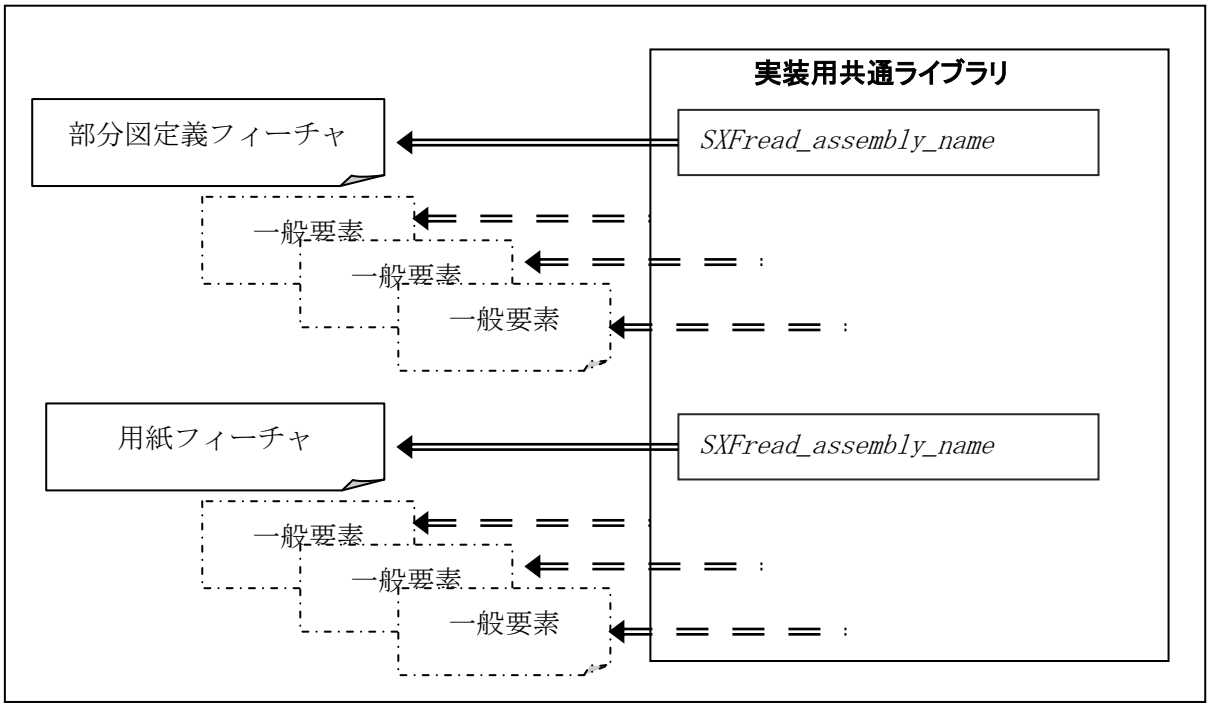
②①に続いて、①上の「一般要素」構造体をシーケンシャルに取出す。

③アセンブリ要素が複数或る場合は、①②の処理をアセンブリ要素の数だけ繰り返す。

以下にアセンブリ要素の取出し手順を示す。



以下にアセンブリ要素が部分図と用紙の場合を示す。



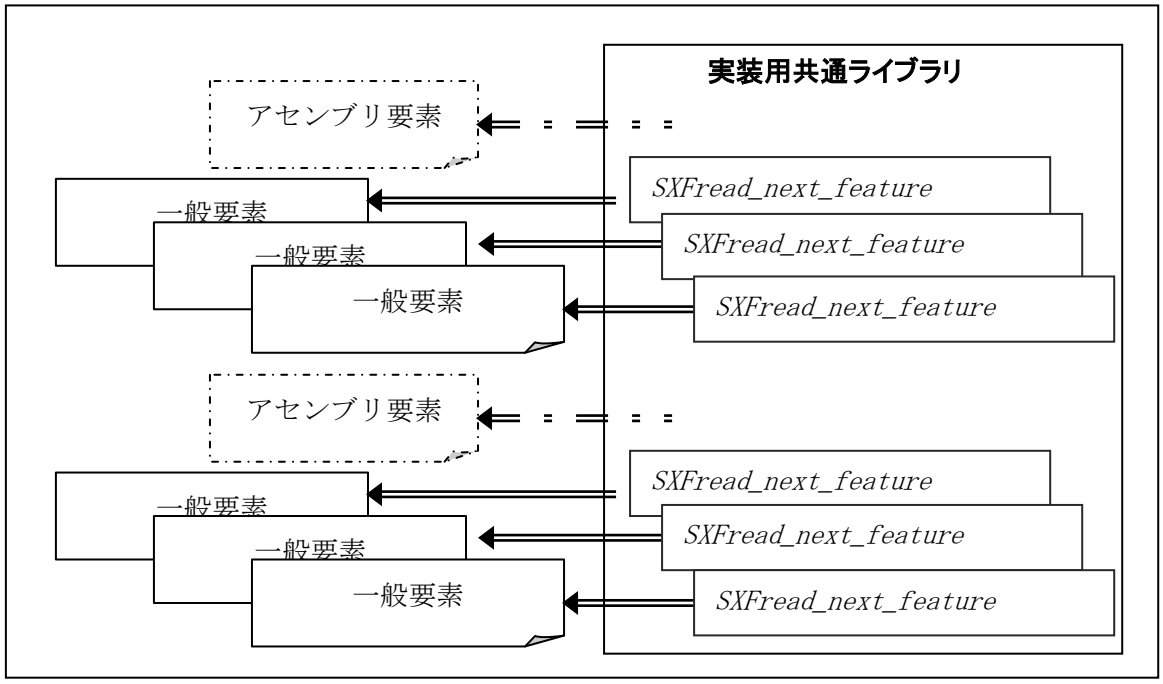
7-1-4　一般要素・附加属性要素の取出し(使用関数：SXFread_next_feature)

①アセンブリ要素取出し関数を呼び出し、「アセンブリ要素」構造体を受取る。

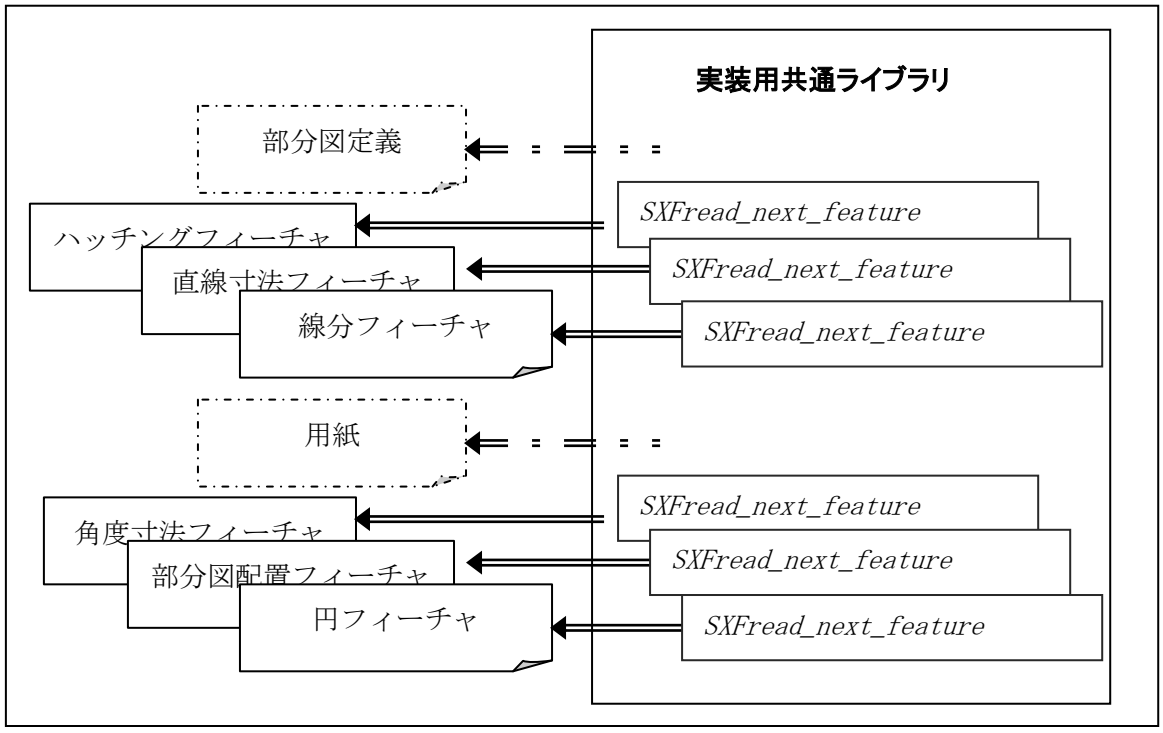
②①に続いて、①上の「一般要素」又は「附加属性要素」の構造体をシーケンシャルに無くなるまで取出す。構造体は予め void*で準備。取出した構造体は SXFdelete_feature で削除する必要がある。

③アセンブリ要素が複数或る場合は、①②の処理をアセンブリ要素の数だけ繰り返す。

以下に一般要素の取出手し順を示す。



以下に一般要素取出しの例を示す。



7-1-5 処理の例

```
/*************************************/
/****** フィーチャ構造体の準備 ******/
/*************************************/
          void* Struct = NULL;
          Layer_Struct las;
          Predefined_Colour_Struct pcs;
          Userdefined_Colour_Struct ucs;
          Predefined_Linetype_Struct pls;
          Userdefined_Linetype_Struct uls;
          Line_Width_Struct lws;
          Text_Font_Struct tfs;

//***********************************/
/****** STEP ファイルオープン ******/
/***********************************/
if (SXFopen_part21(
          in_file_name,      //ファイル名
          rw_flg,            //read/write フラグ
          tolerance1,        //長さ用許容誤差
          tolerance2,        //角度用許容誤差
          divide,            //スプライン分解用許容誤差
          level,             //レベル指定                (入力)
          mode,              //変換モード                (入力)
          author,            //ファイル作成者            (write 入力／read 出力)
          org,               //ファイル作成者所属        (write 入力／read 出力)
          translator_name,   //トランスレータ名          (write 入力／read 出力)
          version,           //共通ライブラリバージョン (read 時のみ出力)
) < 0){
          //エラー問い合わせ
          SXFPopMsg(name,&msgno,text);
          cout << name << " : " << msgno << " : " << text << endl << endl;
          msgno = 0;
          name = "";
          text = "";
          return ;
}
if (SXFget_file_version_part21(file_version) < 0)
{
          return ;
}
/*****************************************/
/****** 定義テーブル情報読み込み ******/
/*****************************************/
/****** レイヤコード読み込み ******/
EndFlag = 0;
while(!EndFlag){
          int ret_layer =SXFread_table(1,&las,&EndFlag);
          if (ret_layer > 0){
                    /****** ここに要素の処理を記載 ******/
          }
}
/****** 既定義色コード読み込み ******/
EndFlag = 0;
while(!EndFlag){
          int ret_precolor = SXFread_table(2,&pcs,&EndFlag);
          if (ret_precolor > 0){
                    /****** ここに要素の処理を記載 ******/
          }
}
```

```
/******以下省略（定義テーブル読み込み）******/
/************************************************/
/****** アセンブリ要素(複合曲線)読み込み ******/
/************************************************/
EndFlag = 0;
while(!EndFlag){
        ret = SXFread_assembly_name( CCURVE_ORG , Struct,&EndFlag);
        if (ret < 0){
                continue;
        }
        else{
                /****** ここに要素の処理を記載 ******/

                /** アセンブリ要素フィーチャ構造体の削除 **/
                SXFdelete_assembly(CCURVE_ORG, Struct);
        }
        /***************************************/
        /****** 複合曲線上の要素を読込む ******/
        /***************************************/
        FeatureEndFlag=0;
        while(!FeatureEndFlag){
                /** 終了フラグが立つまで要素を読込む **/
                error_code = SXFread_next_feature(FeatureName,Struct,&FeatureEndFlag);
                if (error_code  > 0){
                        if (!strcmp(FeatureName, "ARC")){//円弧読みこみ
                                /****** ここに要素の処理を記載 ******/

                                /** 一般要素フィーチャ構造体の削除 **/
                                SXFdelete_feature(FeatureName, Struct);
                                continue;
                        }
                        else if (!strcmp(FeatureName, "ELLIPSE_ARC")){//楕円弧読みこみ
                                /****** ここに要素の処理を記載 ******/

                                /** 一般要素フィーチャ構造体の削除 **/
                                SXFdelete_feature(FeatureName, Struct);
                                continue;
                        }
                        else if (!strcmp(FeatureName, "POLYLINE")){

                                /******以下省略（一般要素読み込み）******/


                        }
                }
        }
}
/************************************************/
/****** アセンブリ要素(複合図形)の読み込み ******/
/************************************************/
EndFlag = 0;
while(!EndFlag){
        ret = SXFread_assembly_name(SFIG_ORG,Struct,&EndFlag);//複合図形
        if (ret < 0){
                continue;
        }
        else{
                /****** ここに要素の処理を記載 ******/

                /** アセンブリ要素フィーチャ構造体の削除 **/
                SXFdelete_assembly(SFIG_ORG, Struct);
```

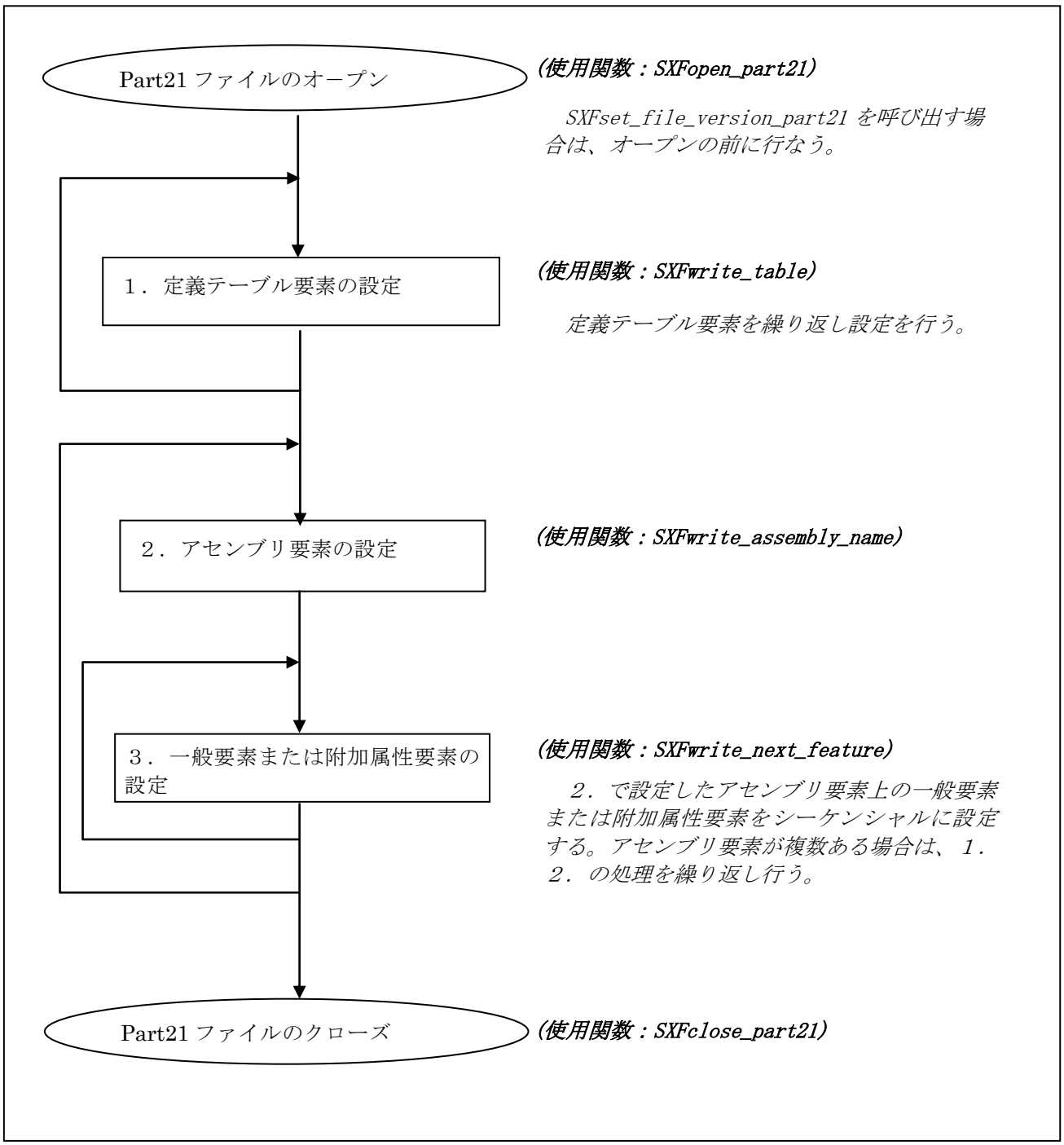
```
		}
		/*******************************************/
		/****** 複合図形上の要素の読み込み ******/
		/*******************************************/
		FeatureEndFlag=0;
		while(!FeatureEndFlag){
			/** 終了フラグが立つまで要素を読込む **/
			error_code = SXFread_next_feature(FeatureName,Struct,&FeatureEndFlag);
			if (error_code > 0){
				if(!strcmp(FeatureName,"LINE")){// 線分の読み込み
					/****** ここに要素の処理を記載 ******/

					/** 一般要素フィーチャ構造体の削除 **/
					SXFdelete_feature(FeatureName, Struct);
					continue;
				}
				else if (!strcmp(FeatureName, "CIRCLE")){

					/******以下省略（一般要素読み込み）******/

				}
		}
}
/*************************************************/
/****** アセンブリ要素(用紙)の読み込み ******/
/*************************************************/
EndFlag = 0;
while(!EndFlag){
		ret = SXFread_assembly_name(SHEET,Struct,&EndFlag);//シート
		if (ret < 0){
			EndFlag = 1;
			break;
		}
		else{
			/****** ここに要素の処理を記載 ******/

			/** アセンブリ要素フィーチャ構造体の削除 **/
			SXFdelete_assembly((SHEET, Struct);
		}
		/*************************************/
		/****** 用紙上の要素の読み込み ******/
		/*************************************/
		FeatureEndFlag=0;
		while(!FeatureEndFlag){
			/** 終了フラグが立つまで要素を読込む **/
			error_code = SXFread_next_feature(FeatureName,Struct,&FeatureEndFlag);
			if (error_code > 0){
				if(!strcmp(FeatureName,"LINE")){// 線分の読み込み
					/****** ここに要素の処理を記載 ******/

					/** 一般要素フィーチャ構造体の削除 **/
					SXFdelete_feature(FeatureName, Struct);
					continue;
				}
				else if (!strcmp(FeatureName, "CIRCLE")){

					/******以下省略（一般要素読み込み）******/

				}
			}
		}
```

```
/************************************/
/****** STEP ファイルクローズ ******/
/************************************/
if (SXFclose_part21() < 0){
          //エラー問い合わせ
          SXFPopMsg(name,&msgno,text);
          cout << name << " : " << msgno << " : " << text << endl << endl;
          msgno = 0;
          name = "";
          text = "";
          return ;
}
```

7-2　Ｐａｒｔ２１ファイル Write 処理

7-2-1　処理の流れ

write の処理の流れを以下の図に示す。処理は必ず以下の流れである必要がある。



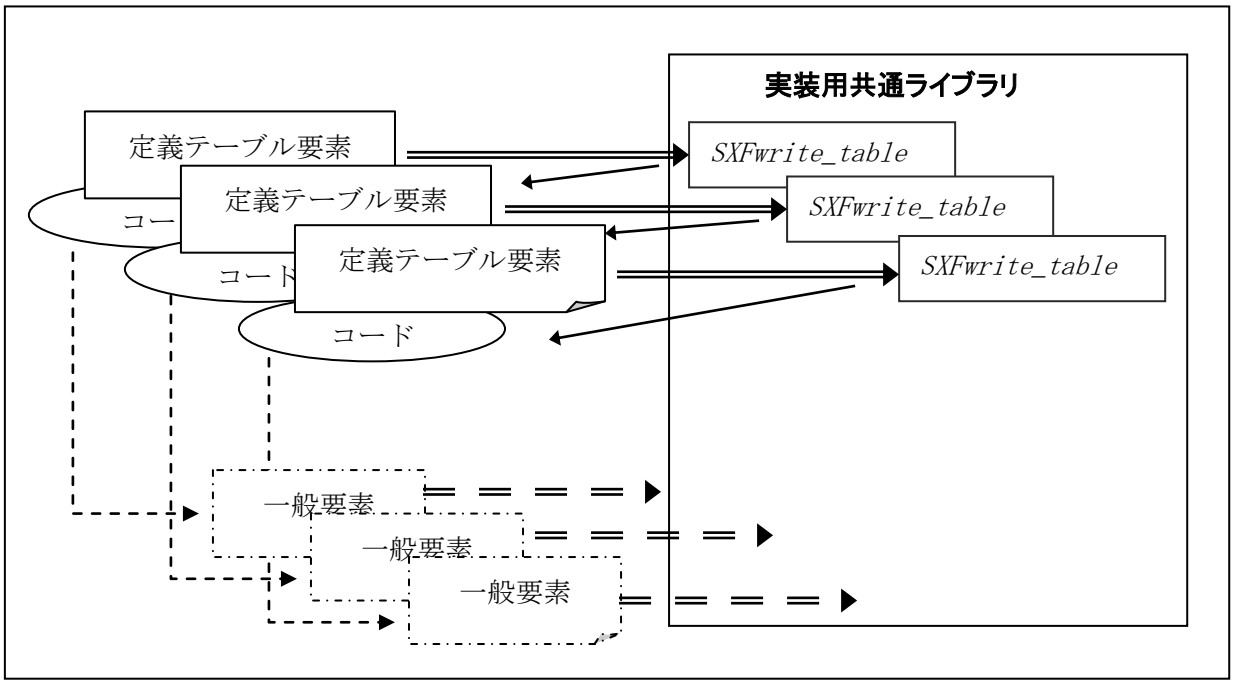
7-2-2 定義テーブル要素の設定(使用関数：SXFwrite_table)

①定義テーブル要素設定関数に「定義テーブル要素」を設定し「コード」を受取る。この処理を全ての定義テーブル要素に対し、シーケンシャルに設定する。構造体は予め定義テーブル型に合わせ準備。

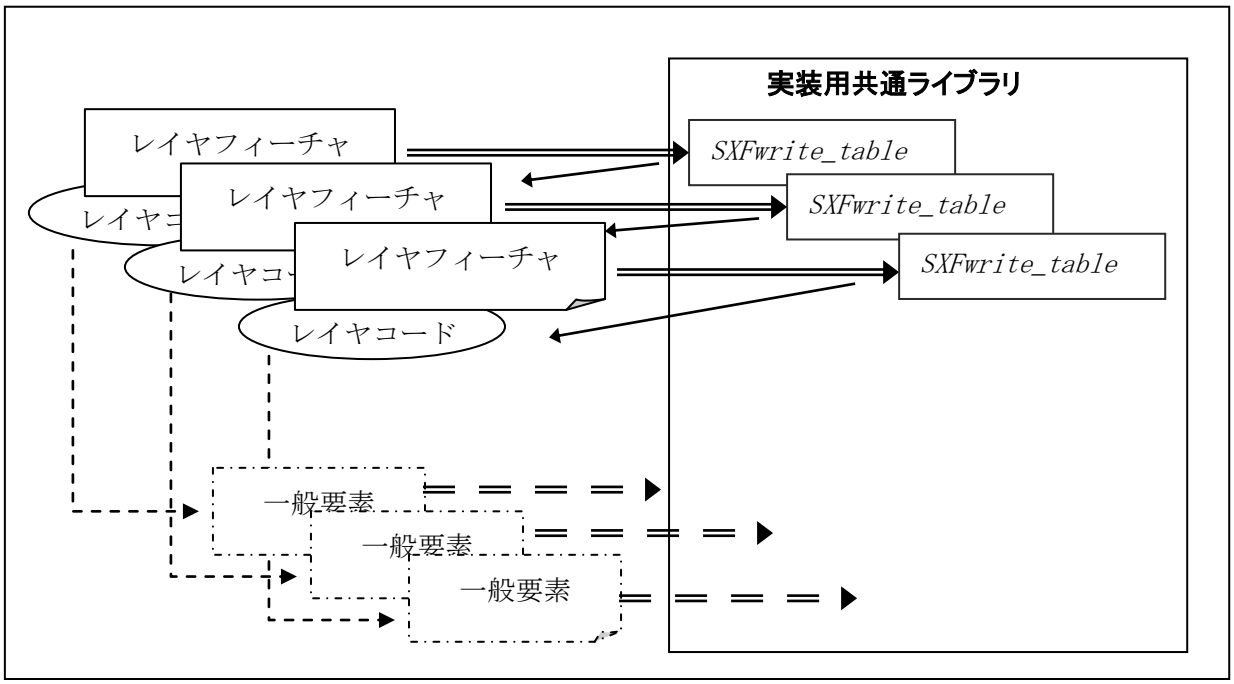
②受取ったコードを、一般要素のパラメータに設定する。

③コードには、レイヤコード、色コード、線種コード、線幅コード、文字フォントコードがある。

以下に定義テーブル要素の設定手順を示す。



以下に定義テーブル要素がレイヤコードの場合を示す。



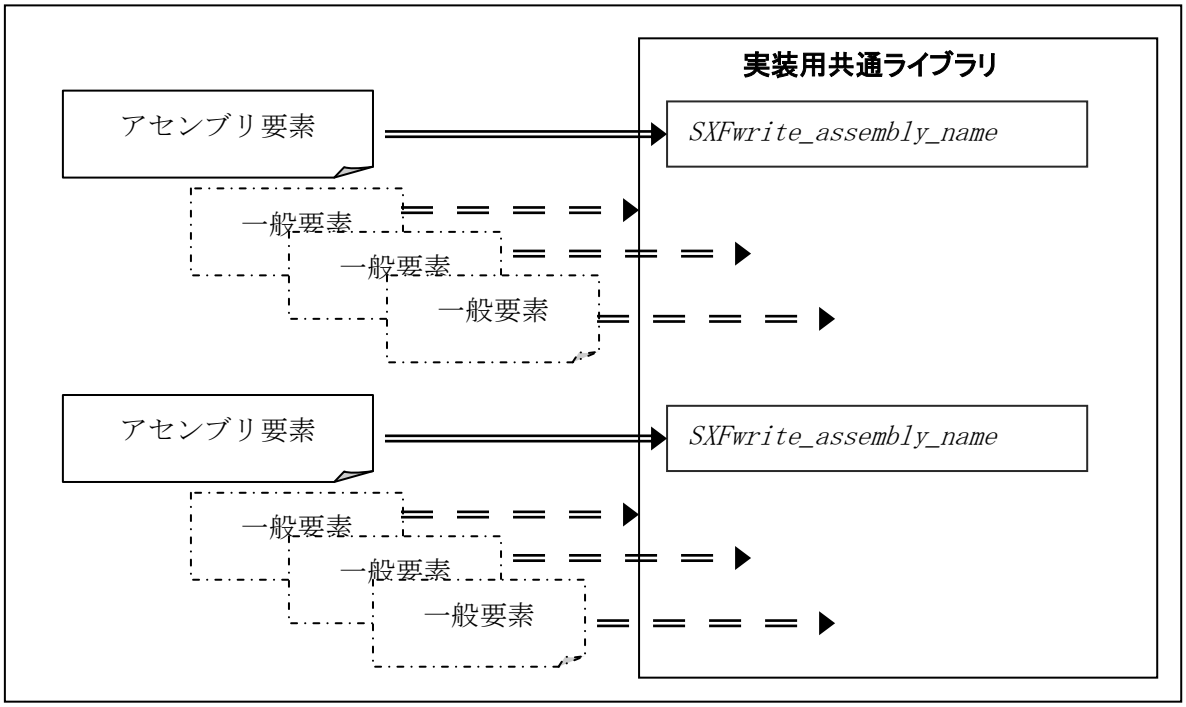
7-2-3 アセンブリ要素の設定(使用関数：SXFwrrite_assembly_name)

①アセンブリ要素設定関数で「アセンブリ要素」構造体を設定する。構造体は予めアセンブリ型に合わせ準備。

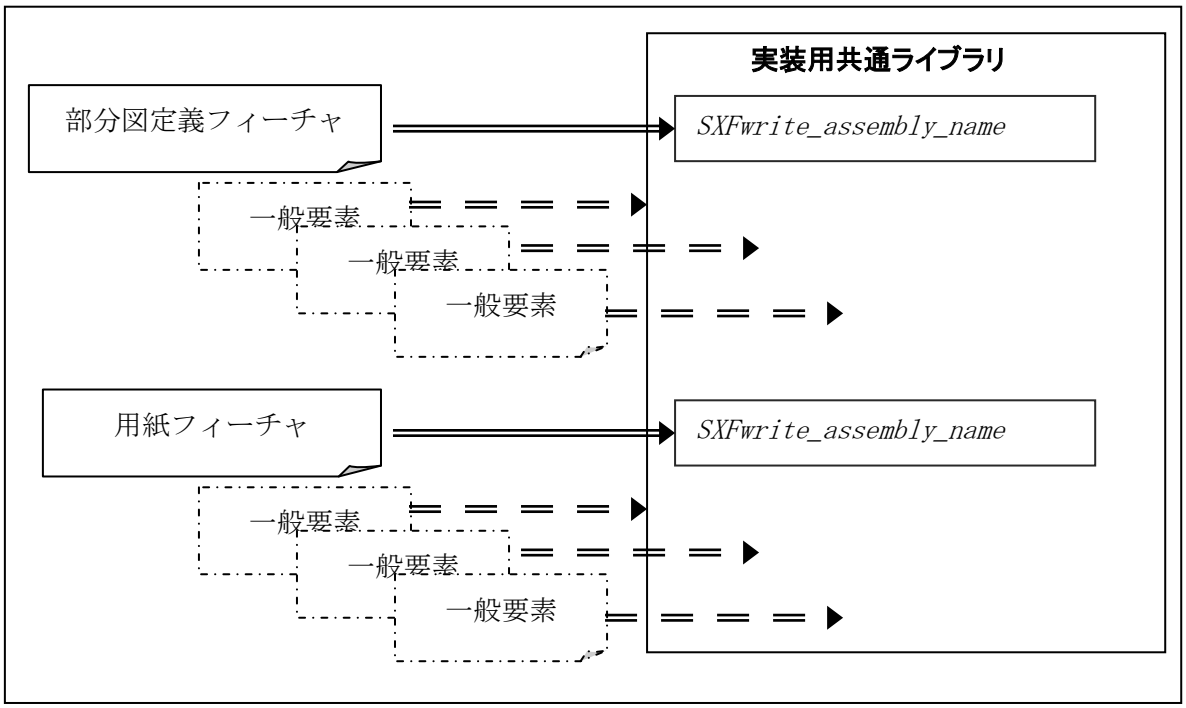
②①に続いて、①上の「一般要素」構造体をシーケンシャルに設定する。

③アセンブリ要素が複数或る場合は、①②の処理をアセンブリ要素の数だけ繰り返す。

以下にアセンブリ要素の設定手順を示す。



アセンブリ要素が部分図と用紙の場合を以下に示す。



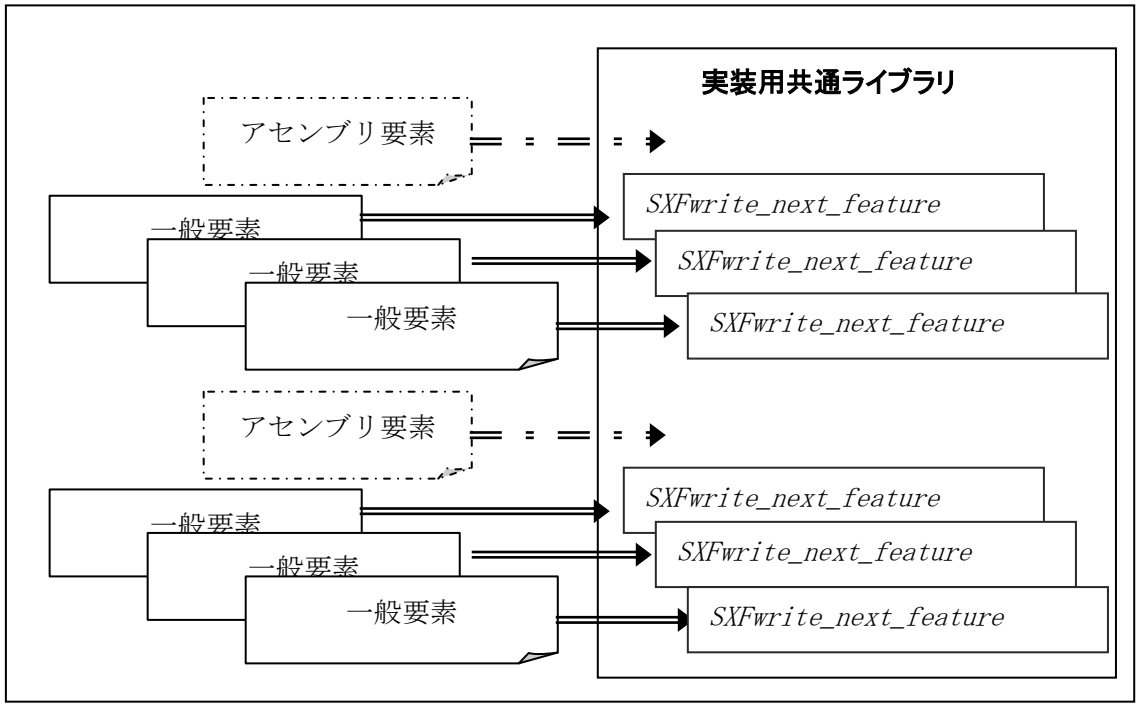
7-2-4　一般要素・附加属性要素の設定(使用関数：SXFwrite_next_feature)

①アセンブリ要素設定関数にて「アセンブリ要素」構造体を設定する。

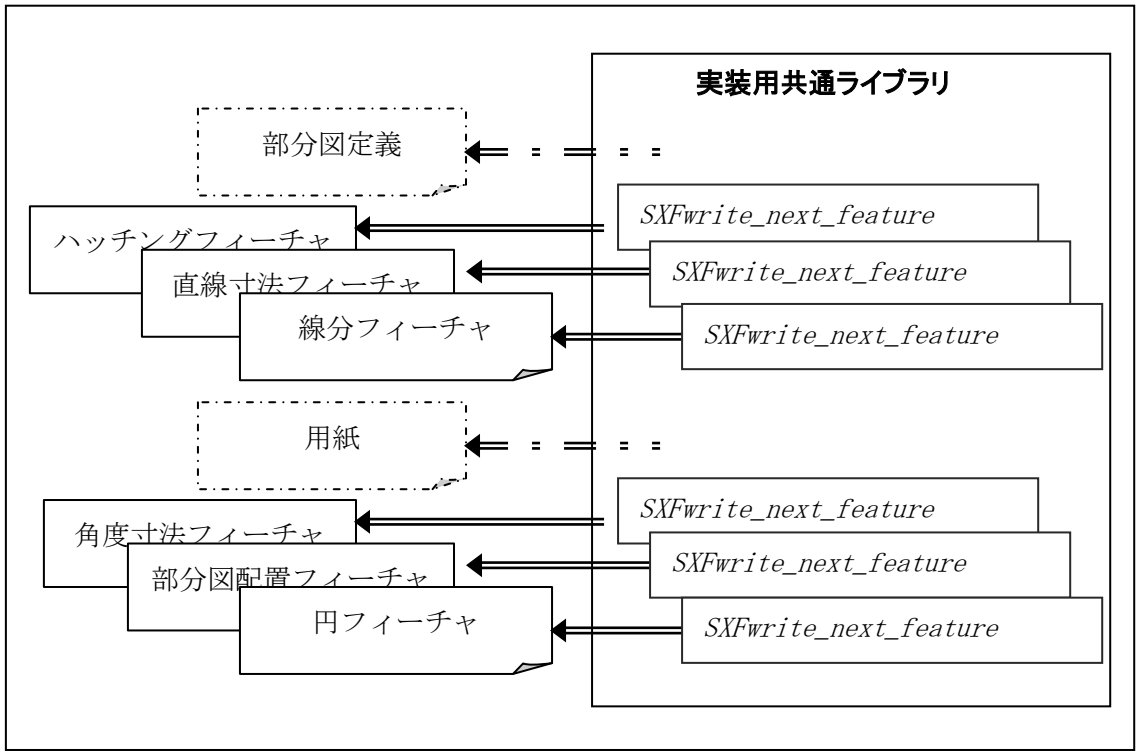
②①に続いて、①上の「一般要素」又は「附加属性要素」の構造体をシーケンシャルに設定する。構造体は予めフィーチャ型に合わせ準備。

③アセンブリ要素が複数或る場合は、①②の処理をアセンブリ要素の数だけ繰り返す。

以下に一般要素の設定手順を示す。



以下に一般要素の設定例を示す。



7-2-5 処理の例

```
/************************************/
/****** STEP ファイルオープン ******/
/************************************/
if (SXFset_file_version_part21(file_version) < 0)
{
       return ;
}
if (SXFopen_part21(
out_file_name,                /** 出力ﾌｧｲﾙ名 **/
          rw_flg,             /** read/write ﾌﾗｸﾞ **/
          tolerance1,         /** 長さ用許容誤差 **/
          tolerance2,         /** 角度用許容誤差 **/
          divide,             /** スプライン分割用許容誤差 **/
          level,              /** ﾚﾍﾞﾙ **/
          mode,               /** 書き込みﾓｰﾄﾞ **/
          author,             /** ﾌｧｲﾙ作成者名 **/
          org,                /** ﾌｧｲﾙ作成者所属 **/
          translator_name,    /** ﾄﾗﾝｽﾚｰﾀ名 **/
          NULL、              /** 共通ﾗｲﾌﾞﾗﾘﾊﾞｰｼﾞｮﾝ(WRITE 時無効) **/
) < 0){
          //エラー問い合わせ
          SXFPopMsg(name,&msgno,text);
          cout << name << " : " << msgno << " : " << text << endl << endl;
          msgno = 0;
          name = "";
          text = "";
          return ;
}

/******************************************************/
/*************** 定義テーブル要素出力 ***************/
/******************************************************/

/************ レイヤコード出力 ************/
/************************************/
/** レイヤコード出力データ設定 **/
/************************************/
          /****** ここに要素の処理を記載 ******/
          /* フィーチャ構造体の定義 */
          Layer_Struct layer_str1;
          /* フィーチャ構造体へのデータ設定 */
          strcpy(layer_str1.name,"layer1");
          layer_str1.lflag = 1;
          /** レイヤコード出力 **/
          layer_code1 = SXFwrite_table(1,&layer_str1);
/************************************/
/** レイヤコード出力データ設定 **/
/************************************/
          /****** ここに要素の処理を記載 ******/
          Layer_Struct layer_str2;
          strcpy(layer_str2.name,"layer2");
          layer_str2.lflag = 0;
          /** レイヤコード出力 **/
          layer_code2 = SXFwrite_table(1,&layer_str2);

/************ 既定義色出力 ************/
/**************************************/
/** 既定義色出力データ設定 **/
/**************************************/
```

A参照

```
        /****** ここに要素の処理を記載 ******/
        Predefined_Colour_Struct color_str2;
        strcpy(color_str2.name,"red");
        /** 既定義色出力 **/
        precolor_code1 = SXFwrite_table(2,&color_str2);
/*********************************/
/** 既定義色出力データ設定 **/
/*********************************/
        /****** ここに要素の処理を記載 ******/
        Predefined_Colour_Struct color_str9;
        strcpy(color_str9.name,"deeppink");
        /** 既定義色出力 **/
        precolor_code2 = SXFwrite_table(2,&color_str9);

/************ ユーザ定義色出力 ************/
/***********************************/
/** ユーザ定義色出力データ設定 **/
/***********************************/
        /****** ここに要素の処理を記載 ******/
        Userdefined_Colour_Struct IN_usercolor1;
        IN_usercolor1.red = 255;
        IN_usercolor1.green = 255;
        IN_usercolor1.blue = 0;
        /** ユーザ定義色出力 **/
        usrcolor_code1 = SXFwrite_table(3,&IN_usercolor1);
/***********************************/
/** ユーザ定義色出力データ設定 **/
/***********************************/
        /****** ここに要素の処理を記載 ******/
        Userdefined_Colour_Struct IN_usercolor2;
        IN_usercolor2.red = 255;
        IN_usercolor2.green = 0;
        IN_usercolor2.blue = 255;
        /** ユーザ定義色出力 **/
        usrcolor_code2 = SXFwrite_table(3,&IN_usercolor2);

/************ 既定義線種出力 ************/
/*********************************/
/** 既定義線種出力データ設定 **/
/*********************************/
        /****** ここに要素の処理を記載 ******/
        Predefined_Linetype_Struct type_str5;
        strcpy(type_str5.name,"long dashed double-dotted");
        /** 既定義線種出力 **/
        pretype_code1 = SXFwrite_table(4,&type_str5);
/*********************************/
/** 既定義線種出力データ設定 **/
/*********************************/
        /****** ここに要素の処理を記載 ******/
        Predefined_Linetype_Struct type_str1;
        strcpy(type_str1.name,"continuous");
        /** 既定義線種出力 **/
        pretype_code2 = SXFwrite_table(4,&type_str1);

/************ ユーザ定義線種出力 ************/
/************************************/
/** ユーザ定義線種出力データ設定 **/
/************************************/
        /****** ここに要素の処理を記載 ******/
        Userdefined_Linetype_Struct IN_usertype1;
        strcpy(IN_usertype1.name,"$$SXF_continuous");
```

A参照

A参照

```
        IN_usertype1.segment = 2;
        IN_usertype1.pitch[0] = 3.5;
        IN_usertype1.pitch[1] = 5.5;
        /** ユーザ定義線種出力 **/
        usrtype_code1 = SXFwrite_table(5,&IN_usertype1);
/*************************************/
/** ユーザ定義線種出力データ設定 **/
/*************************************/
        /****** ここに要素の処理を記載 ******/
        Userdefined_Linetype_Struct IN_usertype2;
        strcpy(IN_usertype2.name,"$$SXF_dashed");
        IN_usertype2.segment = 4;
        IN_usertype2.pitch[0] = 5.5;
        IN_usertype2.pitch[1] = 3.5;
        IN_usertype2.pitch[2] = 2.5;
        IN_usertype2.pitch[3] = 1.5;
        /** ユーザ定義線種出力 **/
        usrtype_code2 = SXFwrite_table(5,&IN_usertype2);

/************ 線幅出力 ************/
/**************************/
/** 線幅出力データ設定 **/
/**************************/
        /****** ここに要素の処理を記載 ******/
        Line_Width_Struct IN_linewidth1;
        IN_linewidth1.width = 0.13;
        /** 線幅出力 **/
        width_code1 = SXFwrite_table(6,&IN_linewidth1);
/**************************/
/** 線幅出力データ設定 **/
/**************************/
        /****** ここに要素の処理を記載 ******/
        Line_Width_Struct IN_linewidth2;
        IN_linewidth2.width = 0.18;
        /** 線幅出力 **/
        width_code2 = SXFwrite_table(6,&IN_linewidth2);

/************ 文字フォント出力 ************/
/**********************************/
/** 文字フォント出力データ設定 **/
/**********************************/
        /****** ここに要素の処理を記載 ******/
        Text_Font_Struct IN_textfont1;
        strcpy(IN_textfont1.name,"MS-ゴシック-5.0-1.0");
        /** 文字フォント出力 **/
        textfont_code1 = SXFwrite_table(7,&IN_textfont1);
/**********************************/
/** 文字フォント出力データ設定 **/
/**********************************/
        /****** ここに要素の処理を記載 ******/
        Text_Font_Struct IN_textfont2;
        strcpy(IN_textfont2.name,"MS-明朝-6.0-2.0");
        /** 文字フォント出力 **/
        textfont_code2 = SXFwrite_table(7,&IN_textfont2);


/********** 以下 定義テーブル要素出力 省略**********/
```

A参照

B参照

```
/*********************************************/
/********* アセンブリ要素出力          **********/
/*********************************************/
/****** 複合曲線データ設定 ******/
        /****** ここに要素の処理を記載 ******/
        /* フィーチャ構造体の定義 */
        Ccurve_Org_Struct Ccurve_str;
        /* フィーチャ構造体へのデータ設定 */
        Ccurve_str.color = precolor_code1;
        Ccurve_str.type  = pretype_code1;
        Ccurve_str.line_width = width_code1;
        Ccurve_str.flag = 1;
        /** 複合曲線出力 **/
        int curve = SXFwrite_assembly_name(3,&Ccurve_str);

        /***************************/
        /*** 一般要素出力         ***/
        /***************************/
        /** 円弧データ設定 **/
                /****** ここに要素の処理を記載 ******/
                /* フィーチャ構造体の定義 */
                Arc_Struct Arc_str;
                /* フィーチャ構造体へのデータ設定 */
                Arc_str.layer = layer_code1;
                Arc_str.color = precolor_code1;
                Arc_str.type  = pretype_code1;
                Arc_str.line_width = width_code1;
                /* 以下省略 */
                /** 円弧出力 **/
                return_code = SXFwrite_next_feature("ARC",&Arc_str);

        /** 折線データ設定 **/
                /****** ここに要素の処理を記載 ******/
                Polyline_Struct Polyline_str1;
                /** 折線出力 **/
                return_code = SXFwrite_next_feature("POLYLINE",&Polyline_str1);

        /******以下省略（一般要素書き込み）******/

/*********************************************/
/********* アセンブリ要素出力テスト **********/
/*********************************************/
/****** 複合曲線データ設定 ******/
        /****** ここに要素の処理を記載 ******/
        Ccurve_Org_Struct Ccurve_str2;
        Ccurve_str2.color = 2;
        Ccurve_str2.type = 5;
        Ccurve_str2.line_width = 2;
        Ccurve_str2.flag = 1;
        /** 複合曲線出力 **/
        int curve2 = SXFwrite_assembly_name(3,&Ccurve_str2);

 /***************************/
 /*** 一般要素出力テスト ***/
 /***************************/
 /** 折線データ設定 **/
        /****** ここに要素の処理を記載 ******/
        Polyline_Struct Polyline_str2;
        /** 折線出力 **/
```

C参照

A

C参照

```
        return_code = SXFwrite_next_feature("POLYLINE",&Polyline_str2);


/*******************************************/
/********* アセンブリ要素出力 ************/
/*******************************************/
/****** 複合図形データ設定 ******/
        /****** ここに要素の処理を記載 ******/
        Sfigorg_Struct Sfig_str;
        strcpy(Sfig_str.name,"sfig");
        Sfig_str.flag = 4;
        /** 複合図形出力 **/
        return_code =SXFwrite_assembly_name(2,&Sfig_str);

        /**********************/
        /*** 一般要素出力 ***/
        /**********************/
        /** 線分データ設定 **/
                /****** ここに要素の処理を記載 ******/
                Line_Struct Line_str;
                /** 線分出力 **/
                return_code = SXFwrite_next_feature("LINE",&Line_str);

        /** 文字要素データ設定 **/
                /****** ここに要素の処理を記載 ******/
                Text_Struct Text_str1;
                Text_str1.font = textfont_code1;
                /** 文字要素出力 **/
                return_code = SXFwrite_next_feature("TEXT",&Text_str1);

        /******以下省略（一般要素書き込み）******/

/*******************************************/
/********* アセンブリ要素出力       ********/
/*******************************************/
/****** 複合図形データ設定 ******/
        /****** ここに要素の処理を記載 ******/
        Sfigorg_Struct Sfig_str2;
        strcpy(Sfig_str2.name,"sfig2");
        Sfig_str2.flag = 1;
        /** 複合図形出力 **/
        return_code = SXFwrite_assembly_name(2,&Sfig_str2);

        /**************************/
        /*** 一般要素出力      ***/
        /**************************/
        /** 複合図形配置データ設定 **/
                /****** ここに要素の処理を記載 ******/
                Sfigloc_Struct Sfigloc_str;
                strcpy(Sfigloc_str.name,"sfig");
                /* 以下省略 */
                /** 複合図形配置出力 **/
                return_code = SXFwrite_next_feature("SFIG_LOCATE",&Sfigloc_str);

        /** 既定義シンボルデータ設定 **/
                /****** ここに要素の処理を記載 ******/
                Externally_Defined_Symbol_Struct ExSymbol_str;
                /** 既定義シンボル出力 **/
                return_code = SXFwrite_next_feature("EXTERNALLY_DEFINED_SYMBOL",&ExSymbol_str);

        /** ユーザ定義ハッチデータ設定 **/
```

D参照

B

E参照

D

```
                    /****** ここに要素の処理を記載 ******/
                    Fill_area_style_hatching_Struct FillHatching_Str;
                    FillTiles_Str.out_id = curve;
                    FillTiles_Str.number = 1;
                    FillTiles_Str.in_id.Add(curve2);
                    /* 以下省略 */
                    /** ユーザ定義ハッチ出力 **/
                    return_code=SXFwrite_next_feature("FILL_AREA_STYLE_HATCHING",&FillHatching_Str);

          /******以下省略（一般要素書き込み）******/

/*****************************************/
/********* アセンブリ要素出力テスト *******/
/*****************************************/
/****** 用紙データ設定 ******/
          /****** ここに要素の処理を記載 ******/
          Sheet_Struct Sheet_str;
          /** 用紙出力 **/
          return_code = SXFwrite_assembly_name(1,&Sheet_str);

          /***************************/
          /*** 一般要素出力テスト ***/
          /***************************/
          /** 複合図形配置データ設定 **/
                    /****** ここに要素の処理を記載 ******/
                    Sfigloc_Struct Sfigloc_str2;
                    strcpy(Sfigloc_str2.name,"sfig2");
                    /* 以下省略 */
                    /** 複合図形配置出力 **/
                    return_code = SXFwrite_next_feature("SFIG_LOCATE",&Sfigloc_str2);

          /******以下省略（一般要素書き込み）******/

/***********************************/
/****** STEP ファイルクローズ ****/
/***********************************/
if (SXFclose_part21() < 0){
          //エラー問い合わせ
          SXFPopMsg(name,&msgno,text);
          cout << name << " : " << msgno << " : " << text << endl << endl;
          msgno = 0;
          name = "";
          text = "";
          return ;
}
```

7-3 フィーチャ構造体に関する注意点

ここでは、フィーチャ構造体の定義に関する注意点を記載します。

フィーチャ型に合わせて動的変数（new）で定義したフィーチャ構造体を delete する際は、構造体でキャストして下さい。

キャストが必要な構造体は以下のとおりです。

①折線
②スプライン
③引き出し線
④バルーン
⑤ハッチング（既定義（外部定義））
⑥ハッチング（塗り）
⑦ハッチング（ユーザ定義）
⑧ハッチング（パターン）

以下に例を示します。

```
    if (f_struct)
        if (strcmp(feature_name,"POLYLINE") == 0)
            delete (Polyline_Struct*)f_struct;
        else if (strcmp(feature_name,"SPLINE") == 0)
            delete (Spline_Struct*)f_struct;
        else if (strcmp(feature_name,"LABEL") == 0)
            delete (Label_Struct*)f_struct;
        else if (strcmp(feature_name,"BALLOON") == 0)
            delete (Balloon_Struct*)f_struct;
        else if (strcmp(feature_name,"FILL_AREA_STYLE_HATCHING") == 0)
            delete (Fill_area_style_hatching_Struct*)f_struct;
        else if (strcmp(feature_name,"EXTERNALLY_DEFINED_HATCH") == 0)
            delete (Externally_Defined_Hatch_Struct*)f_struct;
        else if (strcmp(feature_name,"FILL_AREA_STYLE_COLOUR") == 0)
            delete (Fill_area_style_colour_Struct*)f_struct;
        else if (strcmp(feature_name,"FILL_AREA_STYLE_TILES") == 0)
            delete (Fill_area_style_tiles_Struct*)f_struct;
        else
            delete f_struct;
    f_struct = NULL;
}
```

フィーチャ構造体の詳細は、９フィーチャ構造体仕様を参照ください。

8 変換仕様

8-1 Read 処理

- 定義テーブル要素の読み込み（SXFread_table）は、必ず、STEP ファイルのオープン（SXFopen_part21）に続いて、アセンブリ要素の読み込み（SXFread_assembly_name）、一般要素または附加属性要素の読み込み（SXFread_next_feature）より前に、定義テーブル要素が無くなる（＝継続要素がなくなる）まで繰り返し行う必要がある。読み込まれていない定義テーブル要素がある場合、その定義テーブル要素を使用したアセンブリ要素または一般要素は読み込み時にエラーとなる。
- 定義テーブル要素の読み込み（SXFread_table）は、指定された定義テーブル種別に該当する要素を、STEP ファイルの出現順で検索し、その要素がフィーチャルールに則っていれば、定義テーブルコードを附加する。読み込む定義テーブル要素種別の順番には任意である。
- アセンブリ要素読み込み（SXFread_assembly_name）は、定義テーブル要素の読み込み（SXFread_table）に続いて行う必要がある。これに続いて、一般要素または附加属性要素の読み込み（SXFread_next_feature）にて、そのアセンブリ要素に配置された要素を読み出すことができる。アセンブリ要素がなくなる（＝継続要素がなくなる）まで、アセンブリ要素の読み込み（SXFread_assembly_name）と一般要素または附加属性要素の読み込み（SXFread_next_feature）を繰り返し行う必要がある。
- アセンブリ要素読み込み（SXFread_assembly_name）も、指定されたアセンブリ要素種別に該当する要素を、STEP ファイルの出現順で検索し、その要素がフィーチャルールに則っていれば、アセンブリ要素コードを附加する。読み込むアセンブリ要素種別の順番は、複合曲線→複合図形定義→用紙の順番である必要がある。
- 配置されたアセンブリ要素は、事前に STEP ファイル上に出現し、アセンブリ要素読み込み（SXFread_assembly_name）で定義されている必要がある。このアセンブリ要素定義上に配置された一般要素がない場合、これを配置した場合エラーとなる。
- アセンブリ要素読み込み（SXFread_assembly_name）で読み込んだアセンブリ要素の構造体は、フィーチャ要素の削除（SXFdelete_assembly）で削除する必要がある。
- 一般要素または附加属性要素の読み込み（SXFread_next_feature）は、アセンブリ要素読み込み（SXFread_assembly_name）に続いて、一般要素または附加属性要素が無くなる（＝継続要素がなくなる）まで繰り返し行う必要がある。
- 一般要素も、STEP ファイルの出現順に従い、読み込み処理を行う。
- 一般要素の読み込み（SXFread_next_feature）で読み込んだ一般要素の構造体は、フィーチャ要素の削除（SXFdelete_feature）で削除する必要がある。
- SXF 仕様 Ver.1.0 対応の旧ライブラリで出力された SFC ファイルの読み込みは保証されない。

8-2 write 処理

- 定義テーブル要素の書き込み（SXFwrite_table）は、必ず、STEP ファイルのオープン（SXFopen_part21）に続いて、アセンブリ要素の書き込み（SXFwrite_assembly_name）、一般要素または附加属性要素の書き込み（SXFwrite_next_feature）より前に、定義テーブル要素が無くなるまで繰り返し行う必要がある。書き込まれていない定義テーブル要素がある場合、その定義テーブル要素を使用したアセンブリ要素または一般要素は書き込み時にエラーとなる。
- 定義テーブル要素の書き込み（SXFwrite_table）は、フィーチャルールに則っていれば、書き込まれた順番に定義テーブルコードを附加する。書き込む定義テーブル要素種別の順番には任意である。
- アセンブリ要素書き込み（SXFwrite_assembly_name）は、定義テーブル要素の書き込み（SXFwrite_table）に続いて行う必要がある。これに続いて、一般要素または附加属性要素の書き込み（SXFwrite_next_feature）にて、そのアセンブリ要素に配置された要素を書き込むことができる。アセンブリ要素がなくなるまで、アセンブリ要素の書き込み（SXFwrite_assembly_name）と一般要素または附加属性要素の書き込み（SXFwrite_next_feature）を繰り返し行う必要がある。
- アセンブリ要素書き込み（SXFwrite_assembly_name）も、その要素がフィーチャルールに則っていれば、アセンブリ要素が書き込まれた順番に、アセンブリ要素コードを附加する。書き込むアセンブリ要素種別の順番は、複合曲線→複合図形定義→用紙の順番である必要がある。
- アセンブリ要素を配置する場合は、そのアセンブリ要素は、事前にアセンブリ要素書き込み（SXFwrite_assembly_name）で定義されている必要がある。このアセンブリ要素定義上に配置された一般要素がない場合、これを配置した場合エラーとなる。
- 一般要素または附加属性要素の書き込み（SXFwrite_next_feature）は、アセンブリ要素書き込み（SXFwrite_assembly_name）に続いて、一般要素または附加属性要素が無くなるまで繰り返し行う必要がある。
- 一般要素も、書き込まれた順番に処理を行う。

8-3 .ヘッダーファイル仕様

書き出し時に出力するヘッダーの規約は以下の通りである。

（1）ヘッダー規約

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('%FILE_DESCRIPTION%'),
    '1');
FILE_NAME('%FILE_NAME%',
    '%TIME_STAMP%',
    ('%AUTHOR%'),
    ('%ORGANIZATION%'),
    '%PREPROCESSOR_VERSION%',
    '%ORIGINATING_SYSTEM%',
    '');

FILE_SCHEMA(('ASSOCIATIVE_DRAUGHTING'));
ENDSEC;
```

（2）ヘッダーの各パラメータについて

（A）%FILE_DESCRIPTION%

レベルとモードを、 SCADEC $LEVEL$ $MODE$ の形式で出力する。

・$LEVE$　　レベル１：level1、レベル２：level2

・$MODE$　　フィーチャコメントモード：feature_mode、
２０２モード：AP202_mode
併用：ambi_mode

（B）%FILE_NAME%

SXFopen_part21 時の引数の読み込みファイル名を出力する。

（C）%TIME_STAMP%

変換開始時間を出力する。

（D）%AUTHOR%

SXFopen_part21 時の引数のファイル作成者を出力する。

（E）%ORGANIZATION%

SXFopen_part21 時の引数の作成者所属を出力する。

（F）%PREPROCESSOR_VERSION%

共通ライブラリのバージョンと SXF ファイルのバージョンを
API_version$$file_version の形式で出力する。

（G）%ORIGINATING_SYSTEM%

SXFopen_part21 時の引数の CAD システム名を出力する。

（3）ヘッダーの例（レベル１、フィーチャコメントモードの場合）

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('SCADEC level1 feature_mode'),
    '1');
FILE_NAME('TEST DATA',
    '2010-11-09T20:11:29',
    ('%AUTHOR%'),
    ('%ORGANIZATION%'),
    'SCADEC_API_Ver3.20$$3.1',
    '%ORIGINATING_SYSTEM%',
    '');
FILE_SCHEMA(('ASSOCIATIVE_DRAUGHTING'));
ENDSEC;
```

8-4 SXFopen_part21 のパラメータとファイルのヘッダーの関係について

本ライブラリは、レベル２のフィーチャコメントのみ処理可能なライブラリである。

SXFVer.2.0.レベル２対応SFC共通ライブラリで出力されたSTEPファイルの読み込み時は、SXFopen_part21 のパラメータとファイルのヘッダーのパラメータに従い処理を行う。また、書き込み時は、SXFopen_part21 のパラメータに従い処理を行う。

以下に処理内容を記載する。

8-4-1 レベル

（1）Read 側

STEP ファイル上	SXFopen_part21 のレベル	処理内容
１：レベル１	１：レベル１	メッセージをログファイルに出力し、処理続行
	２：レベル２	レベル２で処理続行
	上記以外のレベル	戻り値にエラーを返却
２：レベル２	１：レベル１	メッセージをログファイルに出力し、レベル２で処理続行　ダウンコンバートは行わない
	２：レベル２	レベル２で処理続行
	上記以外のレベル	戻り値にエラーを返却
上記以外	１：レベル１	メッセージをログファイルに出力し、レベル２で処理続行　ダウンコンバートは行わない
	２：レベル２	メッセージをログファイルに出力し、レベル２で処理続行
	上記以外のレベル	戻り値にエラーを返却

（2）Write 側

SXFopen_part21 のレベル	処理内容
１：レベル１	メッセージをログファイルに出力し、レベル２で処理続行
２：レベル２	レベル２で処理
上記以外	戻り値にエラーを返却

8-4-2 モード

（1）Read 側

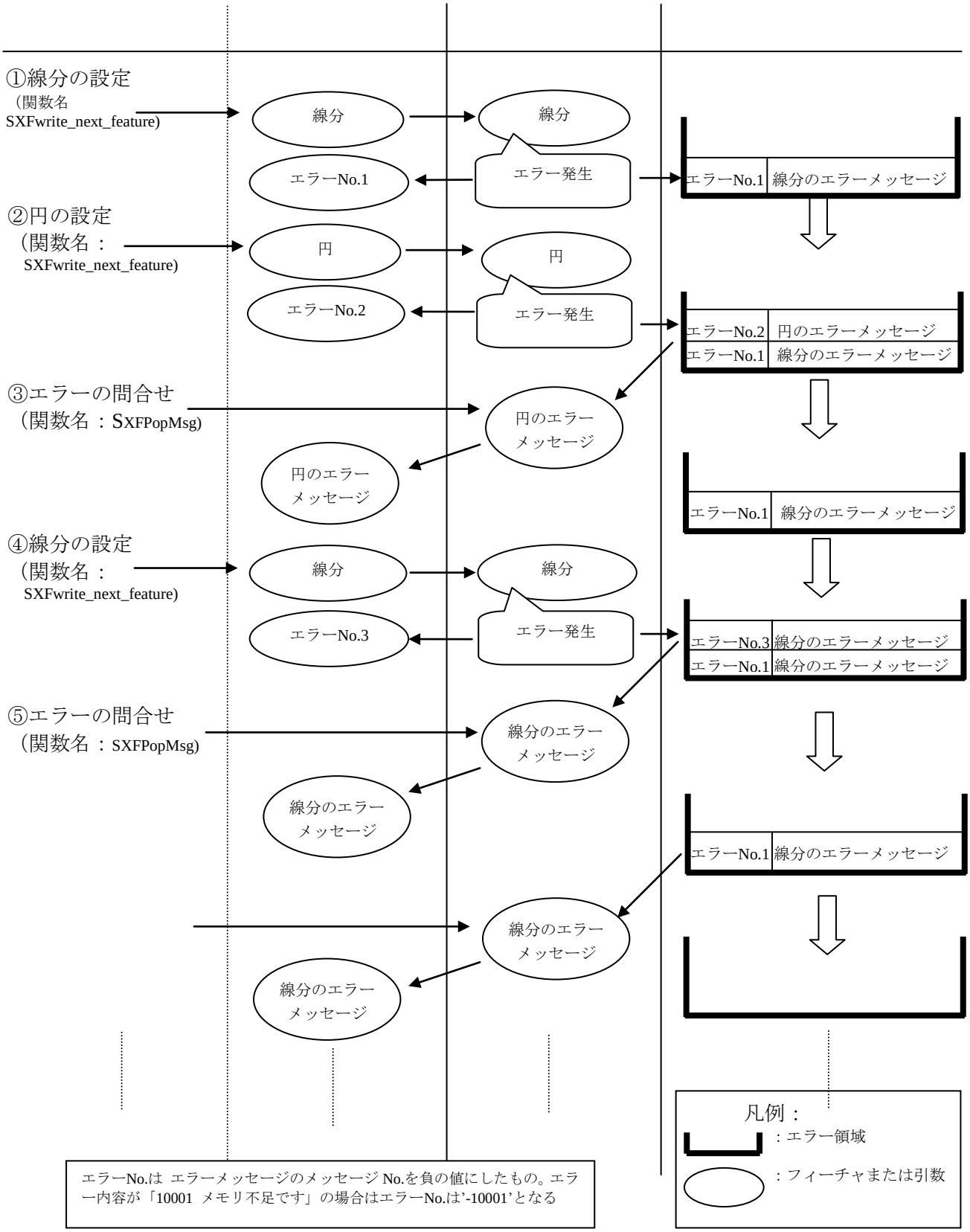
STEP ファイル上	SXFopen_part21 のモード	処理内容
０：併用	０：併用 ２：ＡＰ２０２	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
	１：フィーチャコメント	フィーチャコメントモードで処理
	上記以外	戻り値にエラーを返却
１：フィーチャコメント	０：併用 ２：ＡＰ２０２	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
	１：フィーチャコメント	フィーチャコメントモードで処理
	上記以外	戻り値にエラーを返却
２：ＡＰ２０２	０：併用 ２：ＡＰ２０２	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
	１：フィーチャコメント	フィーチャコメントモードで処理
	上記以外	戻り値にエラーを返却
上記以外のモード	０：併用 ２：ＡＰ２０２	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
	１：フィーチャコメント	メッセージをログファイルに出力し、フィーチャコメントモードで処理
	上記以外	戻り値にエラーを返却

（2）Write 側

SXFopen_part21 のモード	処理内容
０：併用	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
１：フィーチャコメント	フィーチャコメントモードで処理
２：ＡＰ２０２	メッセージをログファイルに出力し、フィーチャコメントモードで処理続行
上記以外のモード	戻り値にエラーを返却

9 エラーメッセージ

9-1 エラー処理のイメージ



9-2 エラーメッセージの種類

（1）システムエラー

OS または汎用ライブラリより返却されるエラー。

① メモリ不足エラー

データのメモリ展開時などに発生。

エラーが発生した API では、エラーメッセージを表示し、処理を中断する。

② ファイルアクセスエラー

STEP ファイル、ワークファイルなどのファイルアクセス時に発生。

エラーが発生した API では、エラーコードを返却し、上位アプリにプロセスを引き渡す。

（2）API 独自エラー

① フィーチャデータの文法チェックエラー

フィーチャ操作時（read_a_feature, write_a_feature）に発生。

エラーが発生した API では、エラーコードを返却し、上位アプリにプロセスを引き渡す。

② フィーチャデータのルールチェックエラー

ルールチェック API（check_rule）操作時に発生。

エラーが発生した API では、エラーコードを返却し、上位アプリにプロセスを引き渡す。

エラー内容は、

③ インスタンス存在チェックの結果、エラーの場合。

対象インスタンスのデータ比較（値比較、サイズ比較、型比較）の結果、エラーの場合。

9-3 エラーメッセージのレベル

エラーメッセージには以下のレベルを設ける。

(1) I レベル

- レベル：Information レベル。
- 説　明：処理の情報を表示するメッセージ。
- 処　置：処理は続行する。

(2) Wレベル

- レベル：warning レベル。
- 説　明：データなどに誤りがあった場合、ＡＰＩで修正可能な場合のエラー情報を表示するメッセージ。
- 処　置：処理は続行する。

(3) E レベル

- レベル：error レベル。
- 説　明：データや環境に謝りがあった場合、ＡＰＩで修正不可能な場合のエラーの情報を表示するメッセージ。
- 処　置：該当の処理は中断する。

(4) Z レベル

- レベル：fatal error エラーレベル。
- 説　明：処理続行不可能な致命的なエラーの情報を表示するメッセージ。
- 処　置：処理続行不可能。

9-4 出力メッセージ

以下に出力されるメッセージを記載する。網掛け部分が新規に追加したメッセージである。

（1）エラーメッセージ

ﾒｯｾｰｼﾞ No	ﾚﾍﾞﾙ	メッセージ	原因・対処方法
10001	Z	メモリ不足です。	実行しようとした操作に十分なメモリがありませんでした
10002	Z	ディスクに書き込めません。	ディスクがいっぱいか書き込めないディスクのファイルを指定しているかなどの原因が考えられます。
10003	Z	システム内で予期せぬエラーが発生しました。処理を中断します	
10004	E	指定されたファイルまたはディレクトリが見つかりません：＜ファイル名またはディレクトリ名＞	
10010	E	STEP ファイルのオープンに失敗しました：＜ファイル名＞	
10011	E	STEP ファイルのクローズに失敗しました：＜ファイル名＞	
10012	W	指定された STEP ファイルは既にオープンされています：＜ファイル名＞	
10013	W	指定された STEP ファイルは既にクローズされています：＜ファイル名＞	
10014	E	STEP ファイルがオープンされていません	
10015	E	モードのエラーです。：ファイルのモード<モード>　オープン時に指定したモード<モード>	
10016	E	レベルは１、２以外指定できません。：レベル<レベル>	
10017	E	モードは０，１，２以外指定できません。モード<モード>	
10018	E	読み書きフラグは０，１以外指定できません。読み書きフラグ<フラグ>"	
10019	I	外部定義ファイルを読み込みます。	
10020	E	展開関数が呼び出せません。	
10021	I	レベル<レベル>が指定されました。	レベル２フィーチャコメント用
10022	I	モードに誤りがあります。モード<モード>　フィーチャコメントモードとして処理します。	レベル２フィーチャコメント用
10023	I	ダウンコンバート処理は行いません。	レベル２フィーチャコメント用
10024	E	処理する要素がありません。	
10025	E	インスタンスデータに誤りがあります。	
10026	I	SCADEC 以外のファイルです。	レベル２フィーチャコメント用
10027	I	AP202 モードのファイルです。	レベル２フィーチャコメント用
10028	E	不正なエンティティ名です。	レベル２フィーチャコメント用
10029	E	ファイルのヘッダー情報が読取れません。	レベル２フィーチャコメント用
20001	W	図面名が指定されていません。	
20002	E	複合図形名が指定されていません。 処理を中断します。	
20003	E	レイヤ名が指定されていません。	

		処理を中断します。	
20004	E	RGB 値に異常があります。 : RGB 値＜RGB 値＞	
20005	E	セグメントの値に異常があります。 : セグメント＜セグメント＞	
20006	E	ピッチに範囲外の値を設定しています。:ピッチ＜ピッチ＞	
20007	E	線幅に範囲外の値を設定しています。:線幅＜線幅＞	
20008	E	最大レイヤ数を超えました。	
20009	E	最大既定義色数を超えました。	
20010	E	最大ユーザ定義色数を超えました。	
20011	E	最大既定義線種数を超えました。	
20012	E	最大ユーザ定義線種数を超えました。	
20013	E	最大線幅数を超えました。	
20014	E	最大文字フォント数を超えました。	
20015	E	表示／非表示フラグの値に異常があります。<フラグ>	
20017	E	用紙は一枚しか設定できません。	
20018	E	レベル 1 では、部分図は一つしか定義できません。	
20019	E	レベル 1 では、複合図形は部分図しか定義できません。	
20020	E	レベル 1 では用紙上には部分図しか配置できません。	
20021	E	レベル 1 では、部分図上には作図グループおよび要素しか配置できません。	
20022	E	アセンブリ要素が開かれていません。	
20023	E	レベル 1 では、用紙上には一つしか部分図を配置できません。	
20024	E	すでに同じ名称がテーブルに存在します。 : 名前<名前>	
20025	E	複合図形種別フラグの値に異常があります。:フラグ：＜フラグ＞ フィーチャ名＜フィーチャ名＞	
20026	E	用紙サイズ種別の値に異常があります。 : 用紙サイズ種別<用紙サイズ種別>	
20027	E	縦/横区分の値に異常があります。 : 縦/横区分<縦/横区分>	
20028	E	自由用紙の長さが 0 以下です。 : 自由用紙長<自由用紙長>	
20029	E	既定義線種ではありません。 : 線種名<線種名>	
20030	E	既定義色ではありません。 : 色名<色名>	
20031	E	既定義線幅ではありません。 : 線幅<線幅>	
20032	E	すでに登録された線幅です。:線幅<線幅>	
20033	E	ベースライン比率の値が異常です。 : Ascent:<ascent> Decent<decent>	
20034	E	文字フォント名称長が 0 です。	
20035	E	自由用紙の長さが設定されています。:自由用紙長<長さ>	
20036	E	アセンブリ要素上にフィーチャ要素が存在しません。	

20037	E	アセンブリ型に異常があります。:アセンブリ型<アセンブリ型>	
20038	E	テーブル型に異常があります。:テーブル型<テーブル型>	
20039	E	複合定義上に用紙は配置できません。	
20040	E	複合曲線上に用紙は配置できません。	
20041	E	読み出すレイヤがありません。	
20042	E	読み出す既定義色がありません。	
20043	E	読み出すユーザ定義色がありません。	
20044	E	読み出す既定義線種がありません。	
20045	E	読み出すユーザ定義線種がありません。	
20046	E	読み出す線幅がありません。	
20047	E	読み出す文字フォントがありません。	
20048	E	読み出すアセンブリ要素がありません。	
30001	E	用紙が設定されていません。処理を中断します。	
30002	E	複合図形が設定されていません。処理を中断します。:複合図形名：＜複合図形名＞ フィーチャ名＜フィーチャ名＞	
30003	E	設定されていないレイヤコードが指定されました。：レイヤコード：＜レイヤコード＞ フィーチャ名＜フィーチャ名＞	
30004	E	色コードの値に異常があります。：＜色コード＞ フィーチャ名；＜フィーチャ名＞	
30005	E	線種コードの値に異常があります。：＜線種コード＞ フィーチャ名；＜フィーチャ名＞	
30006	E	線幅コードの値に異常があります。：＜線幅コード＞ フィーチャ名：＜フィーチャ名＞	
30007	E	文字フォントコードの値に異常があります。：＜文字フォントコード＞ フィーチャ名：＜フィーチャ名＞	
30008	E	レベル１では処理できません。 フィーチャ名<フィーチャ名>	
30009	E	表示／非表示フラグの値に異常があります。<フラグ>	
30010	E	レベル２では作図部品に要素以外を配置できません。 フィーチャ名<フィーチャ名>	
30011	E	作図グループ定義に部分図を配置しています。： フィーチャ ID<フィーチャ ID> フィーチャ名<フィーチャ名>	
30012	E	部分図定義に部分図を配置しています。： フィーチャ ID<フィーチャ ID> フィーチャ名<フィーチャ名>	
30013	E	作図グループには要素以外配置できません。 フィーチャ名<フィーチャ名>	
30014	E	複合曲線定義には配置できません。 フィーチャ名<フィーチャ名>	
30015	E	ヘッダーがありません。ファイル名：＜ファ	

		イル名＞	
30016	E	フィーチャ型に異常があります。:フィーチャ型<フィーチャ型>	
30017	E	図面表題欄は、用紙または、部分図以外には配置できません。 フィーチャ名<%s>	
31001	E	角度の値に異常があります フィーチャ名：＜フィーチャ名＞ 度 ：＜角度＞	
31002	E	配置座標の値に異常があります フィーチャ名：＜フィーチャ名＞ X ：＜X座標＞ Y ：＜Y座標＞	
31003	E	スケールの値に異常があります。：＜スケール＞	
31004	E	線分の長さが 0 以下です。：＜長さ＞ フィーチャ名；＜フィーチャ名＞	
31005	E	半径 0 以下です。：＜半径＞ フィーチャ名；＜フィーチャ名＞	
31006	E	頂点数が 2 未満です。：＜頂点数＞ フィーチャ名：＜フィーチャ名＞	
31007	E	文字範囲幅が 0 以下です。：＜文字範囲幅＞ フィーチャ名：＜フィーチャ名＞	
31008	E	文字範囲高が 0 以下です。：＜文字範囲高＞ フィーチャ名：＜フィーチャ名＞	
31009	E	文字間隔が 0 未満です。：＜文字間隔＞ フィーチャ名：＜フィーチャ名＞	
31010	E	文字配置基点の値に異常があります。：＜文字配置基点＞ フィーチャ名：＜フィーチャ名>	
31011	E	文字書き出し方向の値に異常があります。：＜文字書き出し方向＞ フィーチャ名：＜フィーチャ名＞	
31012	E	向きフラグの値に異常があります。：＜向きフラグ＞ フィーチャ名：＜フィーチャ名＞	
31013	E	開閉フラグの値に異常があります。：＜開閉フラグ＞ フィーチャ名：＜フィーチャ名＞	
31014	E	マーカコードの値に異常があります。：＜マーカコード＞ フィーチャ名：＜フィーチャ名＞	
31016	E	補助線の有無フラグの値に異常があります。：＜フラグ＞ フィーチャ名：＜フィーチャ名＞	
31017	E	矢印コードの値に異常があります。：＜矢印コード＞ フィーチャ名：＜フィーチャ名＞	
31018	E	寸法値の有無フラグの値に異常があります。：＜フラグ＞ フィーチャ名：<フィーチャ名＞	
31019	E	色コードフラグの値に異常があります。：＜色コードフラグ＞ フィーチャ名：＜フィーチャ名＞	

31024	E	ベクトルの大きさが範囲外の値です。： ベクトルの大きさ<ベクトルの大きさ> フィーチャ名<フィーチャ名>	
31025	E	中抜きの閉領域数が 0 未満です。： 中抜きの閉領域数<中抜きの閉領域数> フィーチャ名<フィーチャ名>	
31026	E	ハッチング線のパターン数が範囲外の値です。： ハッチング線のパターン数<数> フィーチャ名<フィーチャ名>	
31028	E	隣合う 2 点が同じ座標です。： X:<X 座標> Y:<Y 座標> フィーチャ名<フィーチャ名>	
31029	E	始角と終角が等しいです。： 始角：<始角> 終角：<終角> フィーチャ名<フィーチャ名>	
31031	E	頂点数が 4 未満です。: 頂点数<頂点数>	
31032	E	ハッチング線の間隔が 0 未満です。： 間隔<間隔> フィーチャ名<フィーチャ名>	
31033	E	ハッチングの外形が定義されていません。: ハッチングの外形<ハッチングの外形> フィーチャ名<フィーチャ名>	
31034	E	ハッチングの中抜きが定義されていません。： ハッチングの中抜き<ハッチングの中抜き> フィーチャ名<フィーチャ名>	
31035	E	一つの複合図形定義に対し、二つ以上配置できません。： フィーチャ ID<フィーチャ ID> 複合図形名<複合図形名> フィーチャ名<フィーチャ名>	
31036	E	矢印配置座標の値が異常です。： フィーチャ ID<フィーチャ ID> 矢印 1 座標 X:<X 座標> Y:<Y 座標> 矢印 2 座標 X:<X 座標> Y:<Y 座標> フィーチャ名<フィーチャ名>	
31037	E	ハッチングの外形で指定したコードを中抜きに指定しています。： フィーチャ ID<フィーチャ ID> 中抜きのコード:<フィーチャ ID > フィーチャ名<フィーチャ名>	
31038	E	頂点数に誤りがあります。： フィーチャ ID<フィーチャ ID> 頂点数:<頂点数 > フィーチャ名<フィーチャ名>	
31039	E	ハッチングの外形で指定したコードを中抜きに指定しています。： フィーチャ ID<フィーチャ ID> 中抜きの閉領域数:<閉領域数 > フィーチャ名<フィーチャ名>	

31040	E	矢印内外コードの値に異常があります。： フィーチャ ID<フィーチャ ID> 矢印内外コード:<矢印内外コード > フィーチャ名<フィーチャ名>	
31041	E	256 バイト以上の文字列がありました。256 バイトに切り取ります。: フィーチャ ID<%s>　フィーチャ名<%s>	
31042	E	日付に範囲外の値を設定しています。: フィーチャ ID<%s> 日付<%d> フィーチャ名<%s>	
31043	E	クロソイドパラメータが 0 未満です。: フィーチャ ID<%s> パラメータ<%lf> フィーチャ名<%s>	
50001	E	"(が存在しません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
50002	E	")が存在しません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
50003	E	",が存在しません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
50004	E	",または)が存在しません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
50006	E	不適切な文字です。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
51001	E	BOOLEAN 型でない値が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
51002	E	INTEGER 型でない値が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
51003	E	LOGICAL 型でない値が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
51004	E	REAL 型でない値が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
51005	E	STRING 型でない値が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
52001	E	DERIVE 宣言していない引数が＊になっています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	

		パラメータ<パラメータ >	
52002	E	OPTIONALE 宣言していない引数が$になっています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
53001	E	参照先のエンティティに誤りがあります。: 参照元インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> 参照先インスタンス ID<インスタンス ID >	
53002	E	参照先のインスタンス ID が存在しません。: 参照元インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> 参照先インスタンス ID<インスタンス ID >	
53003	E	参照先の複合インスタンスが解析できません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
53004	E	参照先の複合インスタンスに不適切なエンティティが指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
53005	E	エンティティが存在しません。: インスタンス ID<インスタンス ID >	
54000	E	SCADEC の対象ではない ENTITY です。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
54001	E	不適切な ENTITY 名が指定されています。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
55001	E	引数の数が足りません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
55002	E	引数の数が多すぎます。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
55003	E	集合型の要素数が足りません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
55004	E	集合型の要素数が多すぎます。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	
56001	E	Enumration 宣言には存在しません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
56002	E	Enumration 形式ではありません。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名> パラメータ<パラメータ >	
57001	E	一度も参照されないインスタンス ID です。: インスタンス ID<インスタンス ID > エンティティ名<エンティティ名>	

10　フィーチャ構造体仕様

ここでは、実装用共通ライブラリのフィーチャ構造体の仕様について説明する。

各々の変数の詳細は「フィーチャ仕様書」を参照下さい。

10-1　フィーチャ構造体一覧

ここでは、フィーチャ要素とフィーチャ構造体の対応を記載します。

10-1-1　定義テーブル要素

定義テーブル要素と構造体の対応を以下に記載する。

要素名		フィーチャ型	フィーチャ構造体名
(1)	レイヤコード	1	Layer_Struct;
(2)	既定義線種コード	2	Predefined_Linetype_Struct;
(3)	ユーザ定義線種コード	3	Userdefined_Linetype_Struct;
(4)	既定義色コード	4	Predefined_Colour_Struct;
(5)	ユーザ定義色コード	5	Userdefined_Colour_Struct ;
(6)	線幅コード	6	Line_Width_Struct;
(7)	文字フォントコード	7	Text_Font_Struct;

10-1-2　アセンブリ要素

要素名		フィーチャ型	フィーチャ構造体名
(1)	用紙	1	Sheet_Struct
(2)	複合図形定義	2	Sfigorg_Struct
(3)	複合曲線定義	3	Ccurve_Org_Struct

10-1-3 一般要素

要素名		フィーチャ型	フィーチャ構造体名
(1)	点マーカ	POINT_MARKER	Point_Marker_Struct
(2)	線分	LINE	Line_Struct
(3)	折線	POLYLINE	Polyline_Struct
(4)	円	CIRCLE	Circle_Struct
(5)	円弧	ARC	Arc_Struct
(6)	楕円	ELLIPSE	Ellipse_Struct
(7)	楕円弧	ELLIPSE_ARC	Ellipse_Arc_Struct
(8)	文字要素	TEXT	Text_Struct
(9)	スプライン	SPLINE	Spline_Struct
(10)	クロソイド	CLOTHOID	Clothoid_Struct
(11)	複合図形配置	SFIG_LOCATE	Sfigloc_Struct
(12)	既定義シンボル	EXTERNALLY_DEFINED_SYMBOL	Externally_Defined_Symbol_Struct
(13)	直線寸法	LINEAR_DIMENSION	LinearDim_Struct
(14)	弧長寸法	CURVE_DIMENSION	CurveDim_Struct
(15)	角度寸法	ANGULAR_DIMENSION	AngularDim_Struct
(16)	半径寸法	RADIUS_DIMENSION	RadiusDim_Struct
(17)	直径寸法	DIAMETER_DIMENSION	DiameterDim_Struct
(18)	引き出し線	LABEL	Label_Struct
(19)	バルーン	BALLOON	Balloon_Struct
(20)	ハッチング(既定義(外部定義))	EXTERNALLY_DEFINED_HATCH	Externally_Defined_Hatch_Struct
(21)	ハッチング(塗り)	FILL_AREA_STYLE_COLOUR	Fill_area_style_colour_Struct
(22)	ハッチング(ユーザ定義)	FILL_AREA_STYLE_HATCHING	Fill_area_style_hatching_Struct
(23)	ハッチング(パターン)	FILL_AREA_STYLE_TILES	Fill_area_style_tiles_Struct

10-1-4 附加属性要素

附加属性要素と構造体の対応を以下に記載する。

要素名		フィーチャ型	フィーチャ構造体名
(1)	図面表題欄	DRAWING_ATTRIBURE	Drawing_Attribute_Struct;

10-2 フィーチャ構造体

ここでは、フィーチャ構造体を記載します。

フィーチャ構造体は、フィーチャ仕様に従い、フィーチャ要素毎に定義した構造体です。フィーチャ要素のパラメータの詳細については「フィーチャ仕様書」を参照下さい。

10-2-1 定義テーブル要素

```
//-------------------------------------------
// レイヤ
//-------------------------------------------
typedef struct Layer_StructDF{
  char name[MAXLAYERNAME2];          /* レイヤ名 */
  int lflag;                         /* 表示/非表示フラグ */
} Layer_Struct;

//-------------------------------------------
// 既定義線種
//-------------------------------------------
typedef struct Predefined_Linetype_StructDF{
  char name[MAXLINETYPENAME];        /* 線種名 */
} Predefined_Linetype_Struct;

//-------------------------------------------
// ユーザ定義線種
//-------------------------------------------
typedef struct Userdefined_Linetype_StructDF{
  char name[MAXLINETYPENAME];        /* 線種名 */
  int segment;                       /* セグメント数 */
  double pitch[MAXPITCH];            /* ピッチ線分の長さ＋空白長さの繰り返し */
} Userdefined_Linetype_Struct;

//-------------------------------------------
// 既定義色
//-------------------------------------------
typedef struct Predefined_Colour_StructDF{
  char name[MAXCOLOURNAME];          /* 色名 */
} Predefined_Colour_Struct;

//-------------------------------------------00
// ユーザ定義色
//-------------------------------------------
typedef struct Userdefined_Colour_StructDF{
  int       red ;                    /* R値 */
  int       green ;                  /* G値 */
  int       blue ;                   /* B値 */
} Userdefined_Colour_Struct ;

//-------------------------------------------
// 線幅
//-------------------------------------------
typedef struct Line_Width_StructDF{
  double width;                      /* 線幅 */
```

```
} Line_Width_Struct;

//---------------------------------------------
// 文字フォント
//---------------------------------------------
typedef struct Text_Font_StructDF{
  char name[MAXTEXTFONTNAME];       /* 文字フォント名 */
} Text_Font_Struct;

//---------------------------------------------
// 図面表題欄
//---------------------------------------------
typedef struct Attribute_StructDF{
  char     p_name[MAXTEXT];      /* 事業名 */
  char     c_name[MAXTEXT];      /* 工事名 */
  char     c_type[MAXTEXT];      /* 契約区分 */
  char     d_title[MAXTEXT];     /* 図面名 */
  char     d_number[MAXTEXT];    /* 図面番号 */
  char     d_type[MAXTEXT];      /* 図面種別 */
  char     d_scale[MAXTEXT];             /* 尺度 */
  int              d_year;                   /* 図面作成年(西暦) */
  int              d_month;                        /* 図面作成月(西暦) */
  int              d_day;                    /* 図面作成日(西暦) */
  char     c_contractor[MAXTEXT];/        * 受注会社名 */
  char     c_owner[MAXTEXT];             /* 発注事業者名 */
} Attribute_Struct;
```

10-2-2 アセンブリ要素

```
//---------------------------------------------
// 用紙
//---------------------------------------------
typedef struct Sheet_StructDF{
  char name[MAXSHEETNAME];          /* 図面名 */
  int type;                         /* 用紙サイズ種別 */
  int orient;                       /* 縦/横区分 */
  int x;                            /* 自由用紙横長 */
  int y;                            /* 自由用紙縦長 */
} Sheet_Struct;

//---------------------------------------------
// 複合図形定義
//---------------------------------------------
typedef struct Sfigorg_StructDF{
  char name[MAXFIGURENAME];         /* 複合図形名 */
  int flag;                         /* 複合図形種別フラグ */
} Sfigorg_Struct;

//---------------------------------------------
//   複合曲線定義
//---------------------------------------------
typedef struct Ccurve_Org_StructDF{
  int color;                        /* 色コード */
  int type;                         /* 線種コード */
```

```
  int line_width;                          /* 線幅コード */
  int flag;                                /* 表示/非表示フラグ */
} Ccurve_Org_Struct;
```

10-2-3 一般要素

```
//---------------------------------------------
// 点マーカ
//---------------------------------------------
typedef struct Point_Marker_StructDF{
  int layer;                               /* レイヤコード */
  int color;                               /* 色コード */
  double start_x;                          /* 配置位置 X 座標 */
  double start_y;                          /* 配置位置 Y 座標 */
  int marker_code;                         /* マーカコード */
  double rotate_angle;                     /* 回転角 */
  double scale;                            /* 尺度 */
} Point_Marker_Struct;

//---------------------------------------------
// 線分
//---------------------------------------------
typedef struct Line_StructDF{
  int layer;                               /* レイヤコード */
  int color;                               /* 色コード */
  int type;                                /* 線種コード */
  int line_width;                          /* 線幅コード */
  double start_x;                          /* 始点X座標 */
  double start_y;                          /* 始点Y座標 */
  double end_x;                            /* 終点X座標 */
  double end_y;                            /* 終点Y座標 */
} Line_Struct;

//---------------------------------------------
// 折線
//---------------------------------------------
typedef struct Polyline_StructDF{
  int       layer ;                        /* レイヤコード */
  int       color ;                        /* 色コード */
  int       type ;                         /* 線種コード */
  int       line_width ;                   /* 線幅コード */
  int number;                              /* 頂点数 */
  CArray<double,double> x;                 /* X 座標 */
  CArray<double, double> y;                /* Y 座標 */
} Polyline_Struct;

//---------------------------------------------
// 円
//---------------------------------------------
typedef struct Circle_StructDF{
  int               layer ;                /* レイヤコード */
  int               color ;                /* 色コード */
  int               type ;                 /* 線種コード */
  int               line_width ;           /* 線幅コード */
```

```
  double   center_x ;                    /* 中心X座標 */
  double   center_y ;                    /* 中心Y座標 */
  double   radius ;                      /* 半径 */
} Circle_Struct;

//-----------------------------------------
// 円弧
//-----------------------------------------
typedef struct Arc_StructDF{
  int        layer ;                     /* レイヤコード */
  int        color ;                     /* 色コード */
  int        type ;                      /* 線種コード */
  int        line_width ;                /* 線幅コード */
  double   center_x ;                    /* 中心X座標 */
  double   center_y ;                    /* 中心Y座標 */
  double   radius ;                      /* 半径 */
  int direction;                         /* 向きフラグ */
  double start_angle;                    /* 始角 */
  double end_angle;                      /* 終角 */
} Arc_Struct;

//-----------------------------------------
// 楕円
//-----------------------------------------
typedef struct Ellipse_StructDF{

  int        layer ;                     /* レイヤコード */
  int        color ;                     /* 色コード */
  int        type ;                      /* 線種コード */
  int        line_width ;                /* 線幅コード */
  double   center_x ;                    /* 中心X座標 */
  double   center_y ;                    /* 中心Y座標 */
  double   radius_x ;                    /* X 方向半径 */
  double   radius_y ;                    /* Y 方向半径 */
  double rotation_angle;                 /* 回転角 */
} Ellipse_Struct;

//-----------------------------------------
// だ円弧
//-----------------------------------------
typedef struct Ellipse_Arc_StructDF{
  int        layer ;                     /* レイヤコード */
  int        color ;                     /* 色コード */
  int        type ;                      /* 線種コード */
  int        line_width ;                /* 線幅コード */
  double   center_x ;                    /* 中心X座標 */
  double   center_y ;                    /* 中心Y座標 */
  double   radius_x ;                    /* X 方向半径 */
  double   radius_y ;                    /* Y 方向半径 */
  int direction;                         /* 向きフラグ */
  double rotation_angle;                 /* 回転角 */
  double start_angle;                    /* 始角 */
  double end_angle;                      /* 終角 */
```

```
} Ellipse_Arc_Struct;

//-------------------------------------------
// 文字要素
//-------------------------------------------
typedef struct Text_StructDF{
  int layer;                          /* レイヤコード */
  int color;                          /* 色コード */
  int font;                           /* 文字フォントコード */
  char str[MAXTEXT];                  /* 文字列 */
  double text_x;                      /* 文字配置基点X座標 */
  double text_y;                      /* 文字配置基点Y座標 */
  double height;                      /* 文字範囲高 */
  double width;                       /* 文字範囲幅 */
  double spc;                         /* 文字間隔 */
  double angle;                       /* 文字回転角 */
  double slant;                       /* スラント角 */
  int b_pnt;                          /* 文字配置基点 */
  int direct;                         /* 文字書き出し方向 */
} Text_Struct;

//-------------------------------------------
// スプライン
//-------------------------------------------
typedef struct Spline_StructDF{
  int       layer ;                   /* レイヤコード */
  int       color ;                   /* 色コード */
  int       type ;                    /* 線種コード */
  int       line_width ;              /* 線幅コード */
  int open_close;                     /* 開閉区分 */
  int number;                         /* 頂点数 */
  CArray<double,double> x;            /* X 座標 */
  CArray<double, double> y;           /* Y 座標 */
} Spline_Struct;

//-------------------------------------------
// クロソイド
//-------------------------------------------
typedef struct Clothoid_StructDF{
  int       layer ;                   /* レイヤコード */
  int       color ;                   /* 色コード */
  int       type ;                    /* 線種コード */
  int       line_width ;              /* 線幅コード */
  double    base_x ;                  /* 配置基点X座標 */
  double    base_y ;                  /* 配置基点Y座標 */
  double    parameter ;               /* クロソイドパラメータ */
  int direction;                      /* 向きフラグ */
  double angle;                       /* 回転角 */
  double start_length;                /* 開始曲線長 */
  double end_length;                  /* 終了曲線長 */
} Clothoid_Struct;

//-------------------------------------------
```

```
// 複合図形配置
//---------------------------------------------
typedef struct Sfigloc_StructDF{
  int layer;                                  /* レイヤコード */
  char name[MAXFIGURENAME];                   /* 複合図形名 */
  double x;                                   /* 配置位置 X 座標 */
  double y;                                   /* 配置位置 Y 座標 */
  double angle;                               /* 回転角 */
  double ratio_x;                             /* X 方向尺度 */
  double ratio_y;                             /* Y 方向尺度 */
} Sfigloc_Struct;

//---------------------------------------------
// 既定義シンボル
//---------------------------------------------
typedef struct Externally_Defined_Symbol_StructDF{
  int layer;                                  /* レイヤコード */
  int color_flag;                             /* 色コードフラグ */
  int color;                                  /* 色コード */
  char name[MAXSYMBOLNAME];                   /* シンボル名 */
  double start_x;                             /* 配置位置 X 座標 */
  double start_y;                             /* 配置位置 Y 座標 */
  double rotate_angle;                        /* 回転角 */
  double scale;                               /* 倍率 */
} Externally_Defined_Symbol_Struct;

//---------------------------------------------
// 直線寸法
//---------------------------------------------
typedef struct LinearDim_StructDF{
  int       layer ;                           /* レイヤコード */
  int       color ;                           /* 色コード */
  int       type ;                            /* 線種コード */
  int       line_width ;                      /* 線幅コード */
  double    sun_x1 ;                          /* 寸法線始点X座標 */
  double    sun_y1 ;                          /* 寸法線始点Y座標 */
  double    sun_x2 ;                          /* 寸法線終点X座標 */
  double    sun_y2 ;                          /* 寸法線終点Y座標 */
  int       flg2 ;                            /* 補助線１の有無フラグ(０:無、１:有) */
  double    ho1_x0 ;                          /* 補助線１基点X座標 */
  double    ho1_y0 ;                          /* 補助線１基点Y座標 */
  double    ho1_x1 ;                          /* 補助線１基点X座標 */
  double    ho1_y1 ;                          /* 補助線１基点Y座標 */
  double    ho1_x2 ;                          /* 補助線１基点X座標 */
  double    ho1_y2 ;                          /* 補助線１基点Y座標 */
  int       flg3 ;                            /* 補助線２の有無フラグ(０:無、１:有) */
  double    ho2_x0 ;                          /* 補助線２基点X座標 */
  double    ho2_y0 ;                          /* 補助線２基点Y座標 */
  double    ho2_x1 ;                          /* 補助線２基点X座標 */
  double    ho2_y1 ;                          /* 補助線２基点Y座標 */
  double    ho2_x2 ;                          /* 補助線２基点X座標 */
  double    ho2_y2 ;                          /* 補助線２基点Y座標 */
  int       arr1_code1 ;                      /* 矢印１コード */
```

```
    int             arr1_code2 ;        /* 矢印１内外コード(0:なし 1:外向き 2:
内向き) */
    double  arr1_x ;                    /* 矢印１配置始点Ｘ座標 */
    double  arr1_y ;                    /* 矢印１配置始点Ｙ座標 */
    double  arr1_r ;                    /* 矢印１配置倍率 */
    int             arr2_code1 ;        /* 矢印２コード */
    int             arr2_code2 ;        /* 矢印２内外コード(0:なし 1:外向き 2:
内向き) */
    double  arr2_x ;                    /* 矢印２配置始点Ｘ座標 */
    double  arr2_y ;                    /* 矢印２配置始点Ｙ座標 */
    double  arr2_r ;                    /* 矢印２配置倍率 */
    int             flg4 ;              /* 寸法値の有無フラグ(０：無、１：有) */
    int             font ;              /* 文字フォントコード */
    char    str[MAXTEXT] ;              /* 文字列 */
    double  text_x ;                    /* 文字列配置基点Ｘ座標 */
    double  text_y ;                    /* 文字列配置基点Ｙ座標 */
    double  height ;                    /* 文字範囲高 */
    double  width ;                     /* 文字範囲幅 */
    double  spc ;                       /* 文字間隔 */
    double  angle ;                     /* 文字列回転角 */
    double  slant ;                     /* スラント角度 */
    int             b_pnt ;             /* 文字配置基点 */
                                        /*  １：左下、２：中下、３：右下、*/
                                        /*  ４：左中、５：中中、６：右中、*/
                                        /*  ７：左上、８：中上、９：右上 */
    int             direct ;            /* 文字書出し方向(１：横書き、２：縦書
き) */
  } LinearDim_Struct;

  //-----------------------------------------------
  //  弧長寸法
  //-----------------------------------------------
  typedef struct CurveDim_StructDF{
    int layer;                          /* レイヤコード */
    int color;                          /* 色コード */
    int type;                           /* 線種コード */
    int line_width;                     /* 線幅コード */
    double sun_x;                       /* 寸法線原点Ｘ座標 */
    double sun_y;                       /* 寸法線原点Ｙ座標 */
    double sun_radius;                  /* 寸法線半径 */
    double sun_angle0;                  /* 寸法線始角 */
    double sun_angle1;                  /* 寸法線終角 */
    int flg2;                           /* 補助線１の有無フラグ(0:無 1:有) */
    double ho1_x0;                      /* 補助線１基点Ｘ座標 */
    double ho1_y0;                      /* 補助線１基点Ｙ座標 */
    double ho1_x1;                      /* 補助線１始点Ｘ座標 */
    double ho1_y1;                      /* 補助線１始点Ｙ座標 */
    double ho1_x2;                      /* 補助線１終点Ｘ座標 */
    double ho1_y2;                      /* 補助線１終点Ｙ座標 */
    int flg3;                           /* 補助線２の有無フラグ(0:無 1:有) */
    double ho2_x0;                      /* 補助線２基点Ｘ座標 */
    double ho2_y0;                      /* 補助線２基点Ｙ座標 */
```

```
  double ho2_x1;                    /* 補助線２始点Ｘ座標 */
  double ho2_y1;                    /* 補助線２始点Ｙ座標 */
  double ho2_x2;                    /* 補助線２終点Ｘ座標 */
  double ho2_y2;                    /* 補助線２終点Ｙ座標 */
  int arr1_code1;                   /* 矢印１コード */
  int arr1_code2;                   /* 矢印１内外コード */
  double arr1_x;                    /* 矢印１配置点Ｘ座標 */
  double arr1_y;                    /* 矢印１配置点Ｙ座標 */
  double arr1_r;                    /* 矢印１配置倍率 */
  int arr2_code1;                   /* 矢印２コード */
  int arr2_code2;                   /* 矢印２内外コード */
  double arr2_x;                    /* 矢印２配置点Ｘ座標 */
  double arr2_y;                    /* 矢印２配置点Ｙ座標 */
  double arr2_r;                    /* 矢印２配置倍率 */
  int flg4;                         /* 寸法値の有無フラグ */
  int font;                         /* 文字フォントコード */
  char str[MAXTEXT];                /* 文字列 */
  double text_x;                    /* 文字列配置基点Ｘ座標 */
  double text_y;                    /* 文字列配置基点Ｙ座標 */
  double height;                    /* 文字範囲高 */
  double width;                     /* 文字範囲幅 */
  double spc;                       /* 文字間隔 */
  double angle;                     /* 文字列回転角 */
  double slant;                     /* スラント角度 */
  int b_pnt;                        /* 文字配置基点 */
  int direct;                       /* 文字書出し方向 */
} CurveDim_Struct;

//-----------------------------------------------
//   角度寸法
//-----------------------------------------------
typedef struct AngularDim_StructDF{
  int layer;                        /* レイヤコード */
  int color;                        /* 色コード */
  int type;                         /* 線種コード */
  int line_width;                   /* 線幅コード */
  double sun_x;                     /* 寸法線原点Ｘ座標 */
  double sun_y;                     /* 寸法線原点Ｙ座標 */
  double sun_radius;                /* 寸法線半径 */
  double sun_angle0;                /* 寸法線始角 */
  double sun_angle1;                /* 寸法線終角 */
  int flg2;                         /* 補助線１の有無フラグ(0:無 1:有) */
  double ho1_x0;                    /* 補助線１基点Ｘ座標 */
  double ho1_y0;                    /* 補助線１基点Ｙ座標 */
  double ho1_x1;                    /* 補助線１始点Ｘ座標 */
  double ho1_y1;                    /* 補助線１始点Ｙ座標 */
  double ho1_x2;                    /* 補助線１終点Ｘ座標 */
  double ho1_y2;                    /* 補助線１終点Ｙ座標 */
  int flg3;                         /* 補助線２の有無フラグ(0:無 1:有) */
  double ho2_x0;                    /* 補助線２基点Ｘ座標 */
  double ho2_y0;                    /* 補助線２基点Ｙ座標 */
  double ho2_x1;                    /* 補助線２始点Ｘ座標 */
```

```
  double ho2_y1;                    /* 補助線２始点Ｙ座標 */
  double ho2_x2;                    /* 補助線２終点Ｘ座標 */
  double ho2_y2;                    /* 補助線２終点Ｙ座標 */
  int arr1_code1;                   /* 矢印１コード */
  int arr1_code2;                   /* 矢印１内外コード */
  double arr1_x;                    /* 矢印１配置点Ｘ座標 */
  double arr1_y;                    /* 矢印１配置点Ｙ座標 */
  double arr1_r;                    /* 矢印１配置倍率 */
  int arr2_code1;                   /* 矢印２コード */
  int arr2_code2;                   /* 矢印２内外コード */
  double arr2_x;                    /* 矢印２配置点Ｘ座標 */
  double arr2_y;                    /* 矢印２配置点Ｙ座標 */
  double arr2_r;                    /* 矢印２配置倍率 */
  int flg4;                         /* 寸法値の有無フラグ */
  int font;                         /* 文字フォントコード */
  char str[MAXTEXT];                /* 文字列 */
  double text_x;                    /* 文字列配置基点Ｘ座標 */
  double text_y;                    /* 文字列配置基点Ｙ座標 */
  double height;                    /* 文字範囲高 */
  double width;                     /* 文字範囲幅 */
  double spc;                       /* 文字間隔 */
  double angle;                     /* 文字列回転角 */
  double slant;                     /* スラント角度 */
  int b_pnt;                        /* 文字配置基点 */
  int direct;                       /* 文字書出し方向 */
} AngularDim_Struct;

//-----------------------------------------------
//   半径寸法
//-----------------------------------------------
typedef struct RadiusDim_StructDF{
  int layer;                        /* レイヤコード */
  int color;                        /* 色コード */
  int type;                         /* 線種コード */
  int line_width;                   /* 線幅コード */
  double sun_x1;                    /* 寸法線始点Ｘ座標 */
  double sun_y1;                    /* 寸法線始点Ｙ座標 */
  double sun_x2;                    /* 寸法線終点Ｘ座標 */
  double sun_y2;                    /* 寸法線終点Ｙ座標 */
  int arr_code1;                    /* 矢印コード */
  int arr_code2;                    /* 矢印内外コード */
  double arr_x;                     /* 矢印配置点Ｘ座標 */
  double arr_y;                     /* 矢印配置点Ｙ座標 */
  double arr_r;                     /* 矢印配置倍率 */
  int flg;                          /* 寸法値の有無フラグ */
  int font;                         /* 文字フォントコード */
  char str[MAXTEXT];                /* 文字列 */
  double text_x;                    /* 文字列配置基点Ｘ座標 */
  double text_y;                    /* 文字列配置基点Ｙ座標 */
  double height;                    /* 文字範囲高 */
  double width;                     /* 文字範囲幅 */
  double spc;                       /* 文字間隔 */
```

```
  double angle;                    /* 文字列回転角 */
  double slant;                    /* スラント角 */
  int b_pnt;                       /* 文字配置基点 */
  int direct;                      /* 文字書出し方向 */
} RadiusDim_Struct;

//-----------------------------------------------
//  直線寸法
//-----------------------------------------------
typedef struct DiameterDim_StructDF{
  int layer;                       /* レイヤコード */
  int color;                       /* 色コード */
  int type;                        /* 線種コード */
  int line_width;                  /* 線幅コード */
  double sun_x1;                   /* 寸法線始点Ｘ座標 */
  double sun_y1;                   /* 寸法線始点Ｙ座標 */
  double sun_x2;                   /* 寸法線終点Ｘ座標 */
  double sun_y2;                   /* 寸法線終点Ｙ座標 */
  int arr1_code1;                  /* 矢印１コード */
  int arr1_code2;                  /* 矢印１内外コード */
  double arr1_x;                   /* 矢印１配置点Ｘ座標 */
  double arr1_y;                   /* 矢印１配置点Ｙ座標 */
  double arr1_r;                   /* 矢印１配置倍率 */
  int arr2_code1;                  /* 矢印２コード */
  int arr2_code2;                  /* 矢印２内外コード */
  double arr2_x;                   /* 矢印２配置点Ｘ座標 */
  double arr2_y;                   /* 矢印２配置点Ｙ座標 */
  double arr2_r;                   /* 矢印２配置倍率 */
  int flg;                         /* 寸法値有無フラグ */
  int font;                        /* 文字フォントコード */
  char str[MAXTEXT];               /* 文字列 */
  double text_x;                   /* 文字列配置基点Ｘ座標 */
  double text_y;                   /* 文字列配置基点Ｙ座標 */
  double height;                   /* 文字範囲高 */
  double width;                    /* 文字範囲幅 */
  double spc;                      /* 文字間隔 */
  double angle;                    /* 文字列回転角 */
  double slant;                    /* スラント角度 */
  int b_pnt;                       /* 文字配置基点 */
  int direct;                      /* 文字書出し方向 */
} DiameterDim_Struct;

//-----------------------------------------------
//  引出し線
//-----------------------------------------------
typedef struct Label_StructDF{
  int layer;                                  /* レイヤコード */
  int color;                                  /* 色コード */
  int type;                                   /* 線種コード */
  int line_width;                             /* 線幅コード */
  int vertex_number;                          /* 頂点数 */
  CArray<double, double> vertex_x;            /* Ｘ座標 */
  CArray<double, double> vertex_y;            /* Ｙ座標 */
```

```
  int arr_code;                                   /* 矢印コード */
  double arr_r;                                   /* 矢印配置倍率 */
  int flg;                                        /* 寸法値の有無フラグ */
  int font;                                       /* 文字フォントコード */
  char str[MAXTEXT];                              /* 文字列 */
  double text_x;                                  /* 文字列配置基点X座標 */
  double text_y;                                  /* 文字列配置基点Y座標 */
  double height;                                  /* 文字範囲高 */
  double width;                                   /* 文字範囲幅 */
  double spc;                                     /* 文字間隔 */
  double angle;                                   /* 文字列回転角 */
  double slant;                                   /* スラント角度 */
  int b_pnt;                                      /* 文字配置基点 */
  int direct;                                     /* 文字書出し方向 */
} Label_Struct;

//-----------------------------------------------
//   バルーン
//-----------------------------------------------
typedef struct Balloon_StructDF{
  int layer;                                      /* レイヤコード */
  int color;                                      /* 色コード */
  int type;                                       /* 線種コード */
  int line_width;                                 /* 線幅コード */
  int vertex_number;                              /* 頂点数 */
  CArray<double,double> vertex_x;                 /* X座標 */
  CArray<double,double> vertex_y;                 /* Y座標 */
  double center_x;                                /* 中心X座標 */
  double center_y;                                /* 中心Y座標 */
  double radius;                                  /* 半径 */
  int arr_code;                                   /* 矢印コード */
  double arr_r;                                   /* 矢印配置倍率 */
  int flg;                                        /* 寸法値の有無フラグ */
  int font;                                       /* 文字フォントコード */
  char str[MAXTEXT];                              /* 文字列 */
  double text_x;                                  /* 文字列配置基点X座標 */
  double text_y;                                  /* 文字列配置基点Y座標 */
  double height;                                  /* 文字範囲高 */
  double width;                                   /* 文字範囲幅 */
  double spc;                                     /* 文字間隔 */
  double angle;                                   /* 文字列回転角 */
  double slant;                                   /* スラント角度 */
  int b_pnt;                                      /* 文字配置基点 */
  int direct;                                     /* 文字書出し方向 */
} Balloon_Struct;

//-----------------------------------------------
//   ハッチング(既定義(外部定義)
//-----------------------------------------------
typedef struct Externally_Defined_Hatch_StructDF{
  int layer;                                      /* レイヤコード */
  char name[MAXHATCHNAME];                        /* ハッチング名 */
  int out_id;                                     /* 外形の複合曲線のフィーチャコード */
```

```
    int number;                                 /* 中抜きの閉領域数 */
    CArray<int, int> in_id;                     /* 中抜きの複合曲線のフィーチャコード */
  } Externally_Defined_Hatch_Struct;

  //-----------------------------------------------
  //   ハッチング(塗り)
  //-----------------------------------------------
  typedef struct Fill_area_style_colour_StructDF{
    int layer;                                  /* レイヤコード */
    int color;                                  /* 色コード */
    int out_id;                                 /* 外形の複合曲線のフィーチャコード */
    int number;                                 /* 中抜きの閉領域数 */
    CArray<int,int> in_id;                      /* 中抜きの複合曲線のフィーチャコード */
  } Fill_area_style_colour_Struct;

  //-----------------------------------------------
  //   ハッチング(ユーザ定義)
  //-----------------------------------------------
  typedef struct Fill_area_style_hatching_StructDF{
    int layer;                                      /* レイヤコード */
    int hatch_number;                               /* ハッチング線のパターン数 */
    int hatch_color[MAXHATCHNUMBER];                /* ハッチング線の色コード */
    int hatch_type[MAXHATCHNUMBER];                 /* ハッチング線の線種コード */
    int hatch_line_width[MAXHATCHNUMBER];           /* ハッチング線の線幅コード */
    double hatch_start_x[MAXHATCHNUMBER];           /* ハッチング線のパターン開始
点 X 座標*/
    double hatch_start_y[MAXHATCHNUMBER];           /* ハッチング線のパターン開始
点 Y 座標 */
    double hatch_spacing[MAXHATCHNUMBER];           /* ハッチング間隔 */
    double hatch_angle[MAXHATCHNUMBER];             /* ハッチング線の角度 */
    int out_id;                                     /* 外形の複合曲線のフィーチャ
コード */
    int number;                                     /* 中抜きの領域数 */
    CArray<int,int> in_id;                          /* 中抜きの複合曲線のフィーチ
ャコード*/
  } Fill_area_style_hatching_Struct;

  //-----------------------------------------------
  //   ハッチング(パターン)
  //-----------------------------------------------
  typedef struct Fill_area_style_tiles_Struct{
    int layer;                                  /* レイヤコード */
    char name[MAXPRESYMBOLNAME];                /* 既定義シンボル名 */
    int hatch_color;                            /* ハッチパターンの色コード */
    double hatch_pattern_x;                     /* ハッチパターン配置位置 X 座標 */
    double hatch_pattern_y;                     /* ハッチパターン配置位置 Y 座標 */
    double hatch_pattern_vector1;               /* ハッチパターンの繰り返しベクトル 1
の大きさ */
    double hatch_pattern_vector1_angle;         /* ハッチパターンの繰り返しベクトル 1
の角度 */
    double hatch_pattern_vector2;               /* ハッチパターンの繰り返しベクトル 2
の大きさ */
    double hatch_pattern_vector2_angle;         /* ハッチパターンの繰り返しベクトル 2
```

```
の角度 */
      double hatch_pattern_scale_x;          /* ハッチパターンの X 尺度 */
      double hatch_pattern_scale_y;          /* ハッチパターンの Y 尺度 */
      double hatch_pattern_angle;            /* ハッチパターンの向きの角度 */
      int out_id;                            /* 外形の複合曲線のフィーチャコード */
      int number;                            /* 中抜きの閉領域数 */
      CArray<int,int> in_id;                 /* 中抜きの複合曲線のフィーチャコード
*/
  } Fill_area_style_tiles_Struct;
```

10-2-4 各フィーチャの制限値

```
//
#define MAXLAYER            256             /*最大レイヤ数*/
#define MAXPRELINETYPE      16              /*最大既定義線種数*/
#define MAXPRECOLOUR        16              /*最大既定義色数*/
#define MAXLINEWIDTH        16              /*最大線幅数*/
#define MAXTEXTFONT         1024            /*最大文字フォント数*/

#define MAXUSERLINETYPE     16              /*最大ユーザ定義線種数*/
#define MAXUSERCOLOUR       240             /*最大ユーザ定義色数*/
#define MAXCOLOUR           256             /*最大色数*/
#define MAXLINETYPE         32              /*最大線種数*/

//最大ピッチ数
#define MAXPITCH            8               /*最大ピッチ数*/
//最大 RGB 値
#define MAXRGB              255             /*最大 RGB 値*/
//最大セグメント数
#define MAXSEGMENT          8
//最小セグメント数
#define MINSEGMENT          2

//double 型の上限と下限
#define MINDOUBLE           -1000000000000000.0     /*double 下限*/
#define MAXDOUBLE           1000000000000000.0      /*double 上限*/

//名前の最大名称長
#define MAXSHEETNAME        257             /*図面名の最大名称長*/
#define MAXLAYERNAME2       257             /*レイヤ名の最大名称長*/
#define MAXLINETYPENAME     257             /*線種名の最大名称長*/
#define MAXCOLOURNAME       257             /*色名の最大名称長*/
#define MAXTEXTFONTNAME     257             /*文字フォント名の最大名称長*/
#define MAXFIGURENAME       257             /*複合図形名の最大名称長*/
#define MAXSYMBOLNAME       257             /*既定義シンボル名の最大名称長
*/
#define MAXHATCHNAME        257             /*ハッチング名の最大名称長*/
#define MAXPRESYMBOLNAME    257             /*既定義シンボル名（ハッチング
パターンに使用）の最大名称長*/
#define MAXTEXT             257             /*最大文字数*/

//
#define MAXHATCHNUMBER      4               /*ハッチング線の最大パターン数*/
```

```
//角度の上限と下限
#define MAXANGLE                360.0
#define MINANGLE                0.0

//スラント角度の上限と下限
#define MAXSLANT                85.0
#define MINSLANT                -85.0

//ヘッダー情報(ファイル名など)を UNICODE に変換後の最大文字数
#define MAXUNICODENAME          1280
```