Intel® Integrated Performance Primitives Version 8.1 （Windows 版）

～ 活用ガイド ～

XLSOFT

2014 年 8 月 8 日 作成版

インテル IPP について

XLSOFT

- IPP 関数はその種類によって各ドメインにグループ分けされている。
- 静的リンクと動的リンクが可能。
- SSE 命令、AVX命令を使用し、プロセッサー・アーキテクチャー単位で異なる関数名で定義されている。
- 実行時に動作プロセッサーを自動認識し、最適な関数コードを使用する。（自動ディスパッチャー）
- IPP ライブラリーには、マルチスレッドバージョンとシリアルバージョンが用意されている。（マルチスレッドバージョンの使用は非推奨）
- マルチスレッド化は OpenMP で実装され、インテルで提供される OpenMP ライブラリーを使用。
- IPP ライブラリーのリンクは、IPP 専用のプリプロセッサーの定義を利用することで簡略化されている。

ドメイン

XLSOFT

- IPP 関数は、それぞれのドメインに分けられ管理される。
- ドメイン単位でヘッダー名、静的ライブラリー名、DLL 名が定められている。
- IPP 関数名は、入力データの種類によって Prefix（接頭辞）が定められる。
- IPP 関数が定義されているヘッダー名は IPPリファレンス・ガイドに記載されている。

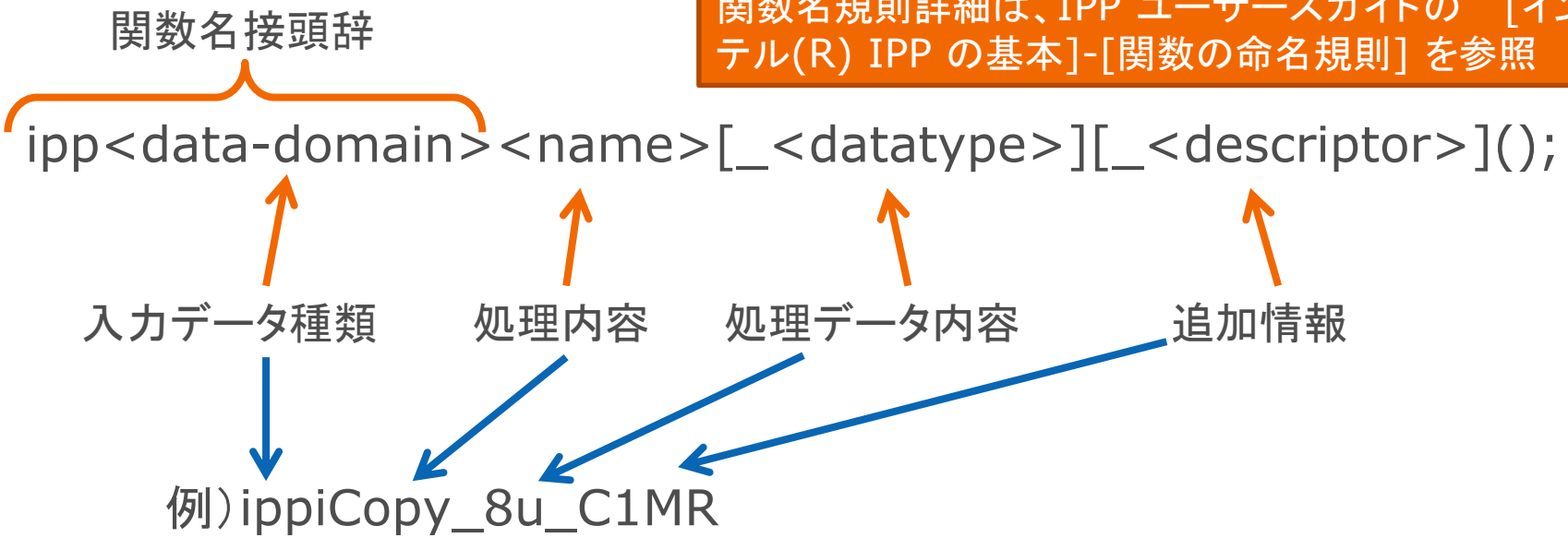
ドメイン コード	ヘッダー	静的リンク ライブラリー	DLL	関数名 接頭辞	内容
ippAC	ippac.h	ippac[*].lib	ippac[**]-x.x.dll	ipps	audio coding
ippCC	ippcc.h	ippcc[*].lib	ippcc[**]-x.x.dll	ippi	color conversion
ippCH	ippch.h	ippch[*].lib	ippch[**]-x.x.dll	ipps	string operations
ippCP	ippcp.h	ippcp[*].lib	ippcp[**]-x.x.dll	ipps	cryptography
ippCV	ippcv.h	ippcv[*].lib	ippcv[**]-x.x.dll	ippi	computer vision
ippDC	ippdc.h	ippdc[*].lib	ippdc[**]-x.x.dll	ipps	data compression
:	:	:	:	:	:

ドメイン詳細は、IPP ユーザーズガイドの [インテル(R) IPP の基本]-[ドメイン] を参照

* は、mt が付くものと付かないものがある
** は、プロセッサー識別子（例：s8）
x.x は、製品バージョン番号（例：8.1）

IPP 関数の名前規則

関数名規則詳細は、IPP ユーザーズガイドの　[インテル(R) IPP の基本]-[関数の命名規則] を参照



要素	内容	例
入力データ種類 (関数名接頭辞)	1D、2D、3D、マトリックス、ベクトルなどの種類を、s,g,I,m,r で区別	ipps, ippi, ippm, ippr
処理内容	関数の処理内容の略語表記	Add, FFTFwd, LuDecomp
処理データ内容	データのビット幅と符合	8u, 32f, 64f
追加情報	データレイアウトなど・・・	ISfs, C1R, P

※ ただし、コア関数(ippcore.h)はこの規則に従わない。（例：ippGetStatusString）

IPPで使用するデータ型

IPPで使用するすべてのデータ型は ippdefs.h ファイル内で定義されており、
データ型の名前には以下の規則がある。

接頭辞	ビット幅と符号	(複素数)
Ipp	8s	C
	16s	
	32s	
	64s	
	8u	
	16u	
	32u	
	64u	
	32f	
	64f	

符号文字
　s：符号あり
　u：符号なし
　f：浮動小数点

例） Ipp16sc の場合
　　意味:符号あり16ビット幅の複素数型

(ippdefs.h ファイル内)

```
251
252 typedef unsigned char   Ipp8u;
253 typedef unsigned short  Ipp16u;
254 typedef unsigned int    Ipp32u;
255
256 typedef signed char    Ipp8s;
257 typedef signed short   Ipp16s;
258 typedef signed int     Ipp32s;
259 typedef float   Ipp32f;
260 typedef __INT64 Ipp64s;
261 typedef __UINT64 Ipp64u;
262 typedef double  Ipp64f;
263
264 typedef struct {
265     Ipp8s  re;
266     Ipp8s  im;
267 } Ipp8sc;
268
269 typedef struct {
270     Ipp16s  re;
271     Ipp16s  im;
272 } Ipp16sc;
273
274 typedef struct {
275     Ipp16u  re;
276     Ipp16u  im;
277 } Ipp16uc;
278
```

ディレクトリー／ファイル構成（IPP ルート外）

XLSOFT

◆ IPP ルートディレクトリ

<インストールディレクトリ>¥Composer XE 2013 SP1¥IPP

◆ 各種マニュアル

英語表記
<インストールディレクトリ>¥Composer XE 2013 SP1¥Documentation¥en_US¥ipp
日本語表記
<インストールディレクトリ>¥Composer XE 2013 SP1¥Documentation¥ja_JP¥ipp

◆ IA32 用 DLL ファイル（Intel64用 DLL ファイルを含む）

<インストールディレクトリ>¥Composer XE 2013 SP1¥redist¥intel64¥ipp

◆ サンプルコード

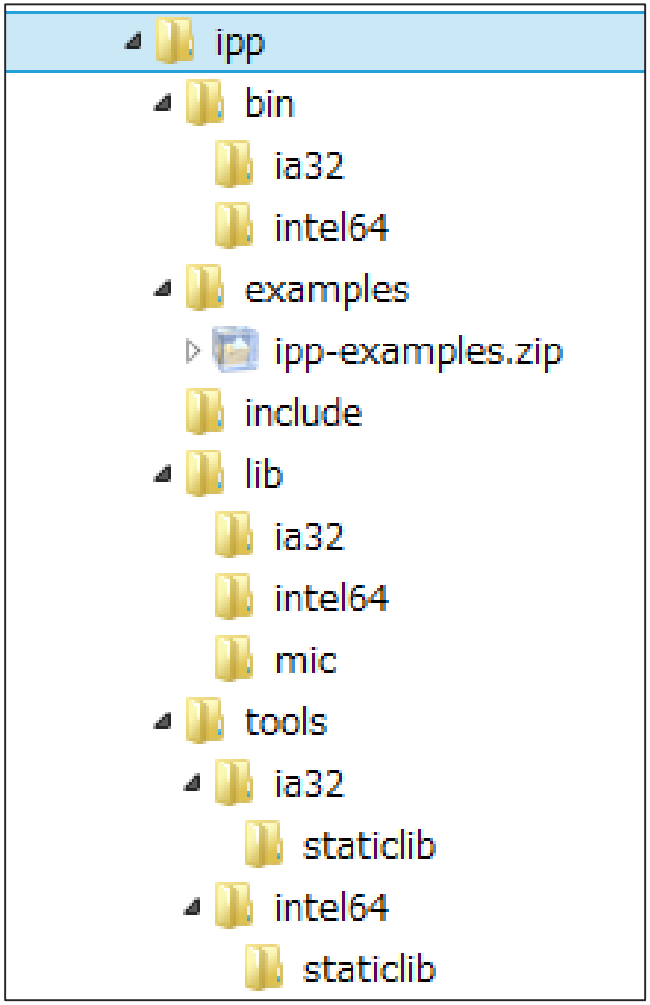
<インストールディレクトリ>¥Composer XE 2013 SP1¥Samples¥en_US¥ipp
※その他のサンプルコードは、ipp ルート内または
インテル社のサイトから別途ダウンロードが必要（「関連情報」ページ参照）

<インストールディレクトリ>
C:¥Program Files (x86)¥Intel　　(x64システム)
C:¥Program Files¥Intel　　(x86システム)

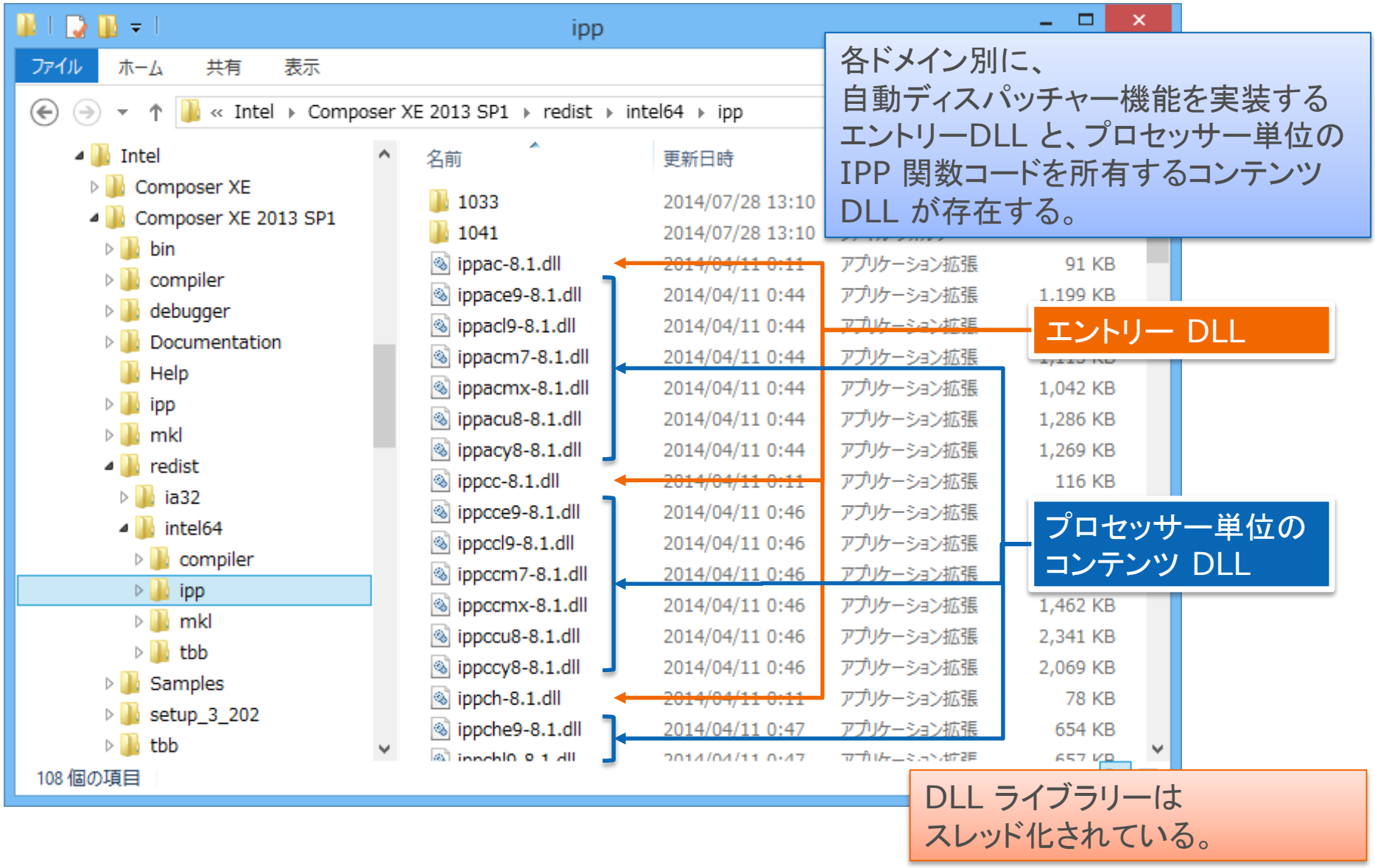
ディレクトリー／ファイル構成（IPP ルート内）

ディレクトリ名	格納ファイル内容
bin	IPPビルド環境設定用バッチファイル (ia32、intel64)
example	サンプルプログラム
include	IPP ヘッダーファイル
lib	静的リンク・ライブラリー (ia32、intel64、mic)
tools	プロセッサー・アーキテクチャー別の 静的リンク・ライブラリー (ia32、intel64)

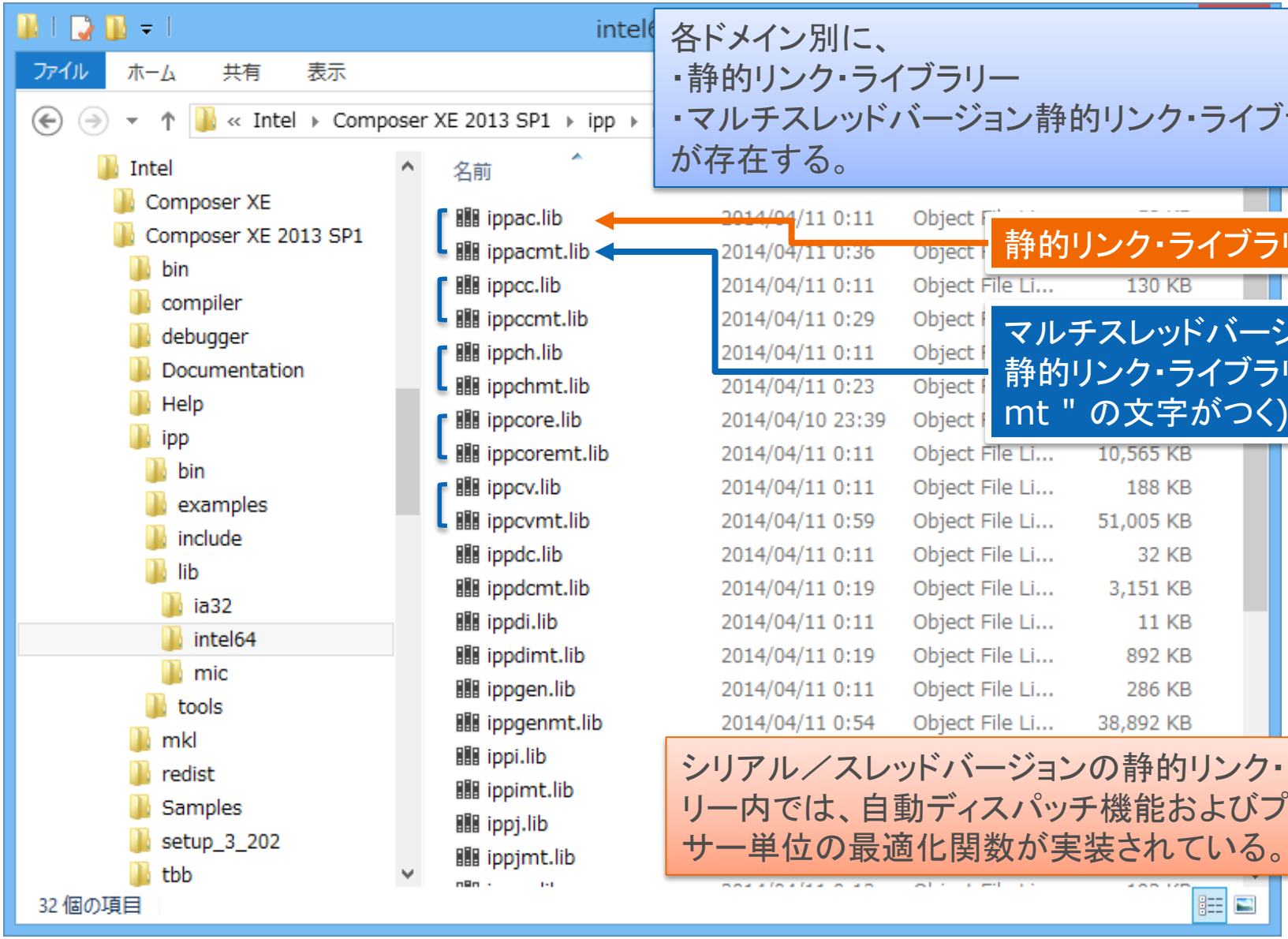
ipp ルートディレクトリ内
(場所は前ページ参照)



動的リンク･ライブラリー（DLL）



静的リンク・ライブラリー（LIB）



自動ディスパッチャー

◆ IPP で使用されるプロセッサー別認識記号

記号	意味	例
IA-32 Architecture		
px	汎用IA-32プロセッサー	ippig9-7.0.dll ippip8-7.0.dll ipp_p8.h ipp_s8.h p8_ippiCopy_16s_C1C3R s8_ippiXorC_8u_C4IR
v8	SSSE3 命令搭載プロセッサー	
p8	SSE4.1 命令搭載プロセッサー	
g9	AVX 命令搭載プロセッサー	
s8	ATOM プロセッサー	
Intel 64 Architecture		
mx	汎用 Intel64 プロセッサー	ippse9-7.0.dll ippsm7-7.0.dll ipp_n8.h ipp_y8.h m7_ippiWTFwd_32f_C1R n8_ippsIIRSetTaps_32fc
u8	SSSE3 命令搭載プロセッサー	
y8	SSE4.1 命令搭載プロセッサー	
e9	AVX 命令搭載プロセッサー	
n8	ATOM プロセッサー	

リンクモデル

XLSOFT

1. 動的リンク

一般的なライブラリーの動的リンク処理
デフォルトで自動ディスパッチ有り、スレッドバージョンとなる。

2. 静的リンク（自動ディスパッチ有り）

一般的なライブラリーの静的リンク処理
自動ディスパッチ有り、スレッドかシリアルバージョンかはリンクするライブラリーで選択できる。

3. 静的リンク（自動ディスパッチ無し）

特定のプロセッサー向けライブラリーのみのリンク処理
スレッドかシリアルバージョンを選択できる。

4. カスタム DLL によるリンク

必要な IPP 関数のみを集めた DLL による動的リンク処理
自動ディスパッチの有り無し、またスレッドかシリアルバージョンかの選択も可能。

リンクモデルの比較

	DLL	静的リンク (ディスパッチ有り)	静的リンク (ディスパッチ無し)	カスタムDLL
プロセッサーの更新	自動	再コンパイル 再配布	ヘッダーの入れ替え 再コンパイル 再配布	再コンパイル 再配布
最適化	全ての CPU	全ての CPU	特定の単一 CPU	全ての CPU
ビルド	インポートライブラリーのリンク	ディスパッチャーと静的ライブラリーのリンク	静的ライブラリーのリンク	静的ライブラリーから特別DLLの作成
呼び出し	通常の関数名	通常の関数名	指定プロセッサーに特化した関数名	通常の関数名
合計バイナリーサイズ	大きい (IPPのDLLファイルを配布するため)	小さい (使用IPP関数のみのため)	非常に小さい (特定のプロセッサーのみのため)	小さい (使用IPP関数のみのため)
実行ファイルのサイズ	非常に小さい	小さい	小さい	非常に小さい
カーネルモード	非サポート	サポート	サポート	非サポート
マルチスレッド	サポート	スレッドバージョンをリンクした場合はサポート	スレッドバージョンをリンクした場合はサポート	スレッドバージョンをリンクした場合はサポート

動的リンク手順

XLSOFT

① ヘッダー "ipp.h" をインクルードする
（ipp.h 内ですべてのIPP ドメイン・ヘッダーファイルをインクルードしているため）

② IPP 関数をコールする（プロセッサー識別子がついていない関数を使用）

③ ソースコードのコンパイルを行う
a. 対応した IPP ドメインのインポート・ライブラリーをリンカーに指定する
（例えば、ippsCopy_8u 関数であれば ipps.lib をリンクする）
b. "/D_IPP_SEQUENTIAL_DYNAMIC" を指定する
c. Intel Compiler を使用し /Qipp を指定する

④ 実行環境で DLL へのパスが通っていることを確認する。

```
例）Visual C++(cl) または Intel Compiler(icl) を使用する場合
> cl  ipp-test.cpp  ipps.lib ippcore.lib                    ← ライブラリーの直接指定
> cl  ipp-test.cpp  /D_IPP_SEQUENTIAL_DYNAMIC    ← Define の利用

例）Intel Compiler(icl) を使用する場合
> icl  ipp-test.cpp /Qipp   /MD                  ← Intel Compiler と /Qipp の利用
```

静的リンク手順（自動ディスパッチャー有り）

XLSOFT

① ヘッダー "ipp.h" をインクルードする
（"ipp.h" 内ですべてのIPP ドメイン・ヘッダーファイルをインクルードしているため）

② ippInit 関数 または ippInitCpu 関数を使用して自動ディスパッチャーを初期化する。

③ IPP 関数をコールする（プロセッサー識別子がついていない通常の関数をコールする）

④ ソースコードをコンパイルする
a. 対応した IPP ドメインの静的ライブラリーをリンカーに指定する
（例えば、ippsCopy_8u 関数であれば ippsmt.lib をリンクする）
（※ ライブラリー間の依存関係を考慮してライブラリーを指定する必要がある。）
b. "/D_IPP_SEQUENTIAL_STATIC"（ /D_IPP_PARALLEL_STATIC ）を指定してリンクする。
c. Intel compiler を使用し /Qipp を指定し OpenMP ライブラリーをリンクする

例）Visual C++(cl) または Intel Compiler(icl) を使用する場合
> cl ipp-test.cpp ippsmt.lib ippcoremt.lib ← シリアルバージョン
> cl ipp-test.cpp ippsmt.lib ippcoremt.lib libiomp5md.lib ← スレッドバージョン
> cl ipp-test.cpp /D_IPP_SEQUENTIAL_STATIC ← シリアルバージョン
➢ cl ipp-test.cpp /D_IPP_PARALLEL_STATIC ← スレッドバージョン

例）Intel Compiler(icl) を使用する場合
> icl ipp-test.cpp /Qipp /Qopenmp /MT ← スレッドバージョン
> icl ipp-test.cpp /Qipp libiomp5md.lib /MT ← スレッドバージョン

静的リンク手順（自動ディスパッチャー無し）

XLSOFT

① ipp¥tools フォルダー内からプロセッサー別に対応したヘッダーをインクルードする。

② ヘッダー "ipp.h" をインクルードする
（"ipp.h" 内ですべてのIPP ドメイン・ヘッダーファイルをインクルードしているため）

③ IPP 関数をコールする(プロセッサー識別子がついていない通常の関数をコールする）

④ ソースコードをコンパイルする
a. 対応した IPP ドメインの静的ライブラリーをリンカーに指定する
（例えば、ippsCopy_8u 関数であれば ippsmt.lib をリンクする）
b. " /D_IPP_SEQUENTIAL_STATIC"(/D_IPP_PARALLEL_STATIC)を指定してリンクする。
c. Intel compiler を使用し/Qipp を指定しOpenMP ライブラリーをリンクする

```
例）Visual C++(cl) または Intel Compiler(icl) を使用する場合
> cl  ipp-test.cpp  ippsmt.lib  ippcoremt.lib                    ← シリアルバージョン
> cl  ipp-test.cpp  ippsmt.lib  ippcoremt.lib  libiomp5md.lib    ← スレッドバージョン
> cl  ipp-test.cpp  /D_IPP_SEQUENTIAL_STATIC      ← シリアルバージョン
> cl  ipp-test.cpp  /D_IPP_PARALLEL_STATIC        ← スレッドバージョン

例）Intel Compiler(icl) を使用する場合
> icl  ipp-test.cpp /Qipp  /MT                    ← シリアルバージョン
> icl  ipp-test.cpp /Qipp  libiomp5md.lib  /MT    ← スレッドバージョン
```

カスタム DLL 作成方法

1. 以下のサイトから IPP サンプル集(Main Samples)をダウンロードする。
   https://software.intel.com/en-us/articles/intel-ipp-version-81-custom-dll

2. ダウンロード・ファイルを展開して、以下の sln ファイルを Visual studio プロジェクトに追加する。
   <解凍先ディレクトリ>¥custom_dll_solution¥custom_dll.sln

手順

① custom_dll¥export.def ファイルに使用する IPP 関数を記述する
　（プロセッサー識別子がついた関数を指定してもよい）

② Visual studio 上からソリューションをビルドする

③ 生成物の確認(Debugディレクトリ内)

④ 作成したインポート・ライブラリーを指定してアプリのビルドを実施

(exports.txtファイル内)

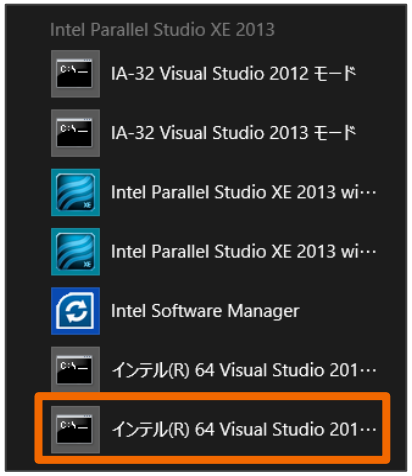
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;            INTEL CORPORATION PROPRIETARY INFORMATION
;     This software is supplied under the terms of a license agreement or
;     nondisclosure agreement with Intel Corporation and may not be copied
;     or disclosed except in accordance with the terms of that agreement.
;         Copyright(c) 2014 Intel Corporation. All Rights Reserved.
;
;         Purpose: List of needed functions from the Intel(R) Integrated
;             Performance Primitives
;
EXPORTS
    ippGetStatusString
    ippsGetLibVersion
    ippiGetLibVersion
    ippsRandUniform_Direct_64f
;;;;;;;;;;;;;;;;;;;;;;;;;; End of file "export.def" ;;;;;;;;;;;;;;;;;;;;;;;;
```

関数を追加する

ビルド環境（コマンドライン）

- Windows の [スタート] メニューからインテル® コンパイラー専用　コマンドプロンプトを起動（必要な環境変数が自動で設定される）

Windows 8



設定内容例）「インテル® 64 Visual Studio 2013 モード」を選択した場合

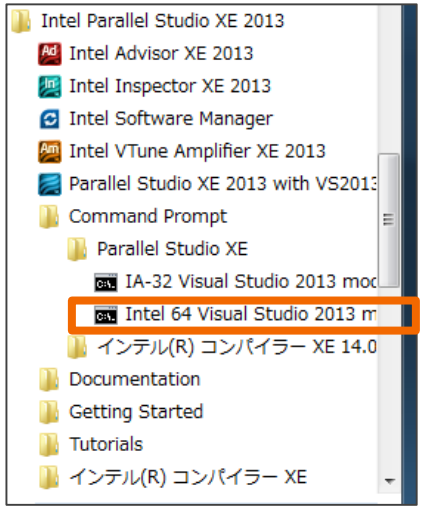
¥Composer XE 2013 SP1¥bin¥ipsxe-comp-vars.bat　intel64

↓（スクリプトコール）

¥Inspector XE 2013¥ inspxe-vars.bat
¥VTune Amplifier XE 2013¥ amplxe-vars.bat
¥compilervars.bat　intel64　vs2013

↓（スクリプトコール）

Windows 7



¥tbb¥bin¥tbbvars.bat　intel64　vs2013
¥mkl¥bin¥mklvars.bat　intel64　vs2013
¥ipp¥bin¥ippvars.bat　intel64　vs2013
　IPPROOT=¥Composer XE 2013 SP1¥ipp
　PATH=¥Composer XE 2013 SP1¥redist¥intel64¥ipp
　　¥Composer XE 2013 SP1¥redist¥intel64¥compiler
　LIB= ¥Intel¥Composer XE 2013 SP1¥ipp¥lib¥intel64
　　¥Intel¥Composer XE 2013 SP1¥compiler¥lib¥intel64
　INCLUDE= ¥Intel¥Composer XE 2013 SP1¥ipp¥include
¥compilervars_arch.bat　intel64　vs2013

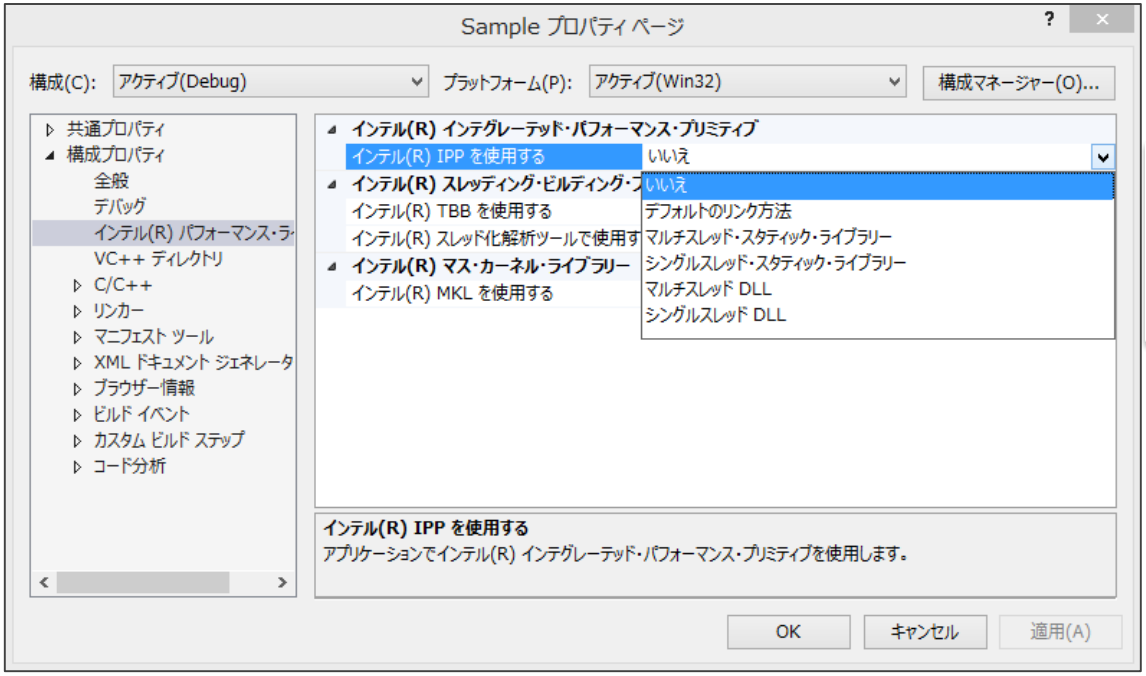
Visual Studio からのリンク方法の指定

① プロジェクトプロパティを開く
(Visual studio 上部メニューの[プロジェクト]→[プロパティ])

② [構成プロパティ][インテル(R)パフォーマンスライブラリー]を選択する。

③「インテル(R) IPPを使用する」からそれぞれ目的に対応したリンク方法を選択する

(Visual Studio プロジェクトプロパティ)



静的か動的、マルチスレッドか
シングルスレッドかどうかで
それぞれ選択する

Visual Studio からのリンク方法の内容

Visual Studio 上での IPP 設定が完了すると、

① IPP インクルード・ディレクトリーが指定される。
例）"C:¥Program Files (x86)¥Intel¥Composer XE 2013 SP1¥ipp¥include"

② プリプロセッサーの定義がセットされる。
_IPP_PARALLEL_DYNAMIC
（"デフォルトのリンク方法" または "共有DLL" が選択された場合）
⇒ マルチスレッド IPP ライブラリーが動的リンクされる。
_IPP_PARALLEL_STATIC
（"マルチスレッド・スタティック・ライブラリー" が選択されている場合）
⇒ マルチスレッド IPP ライブラリーが静的リンクされる。
_IPP_SEQUENTIAL_STATIC
（"シングルスレッド・スタティック・ライブラリー"が選択されている場合）
⇒ シングルスレッド IPP ライブラリーが静的リンクされる。

③ IPP ライブラリー・ディレクトリーが追加される。
例）"C:¥Program Files (x86)¥Intel¥Composer XE 2013 SP1¥lib¥ia32"

※ IPP のプリプロセッサーの定義を使用することにより、IPP ヘッダーファイル内に記述された #pragma comment によって使用ライブラリーが決定される。（詳細は次ページ参照）

IPP ヘッダーにおけるリンクメカニズム

XLSOFT

（ipp.h）

```
"ippversion.h"
"ippdefs.h"
"ippcore.h"
"ippac.h"
"ippcc.h"
"ippdc.h"
"ippch.h"
.....
```

各ヘッダー内で、リンクモデル単位の必要なライブラリーが #pragma comment で指定されているため、ユーザが明示的にリンクする必要はない

(ippdefs.h)

```
#if !defined( _IPP_NO_DEFAULT_LIB )
  #if defined( _IPP_PARALLEL_STATIC )
    #pragma comment( lib, "libircmt" )
    #pragma comment( lib, "libmmt" )
    #pragma comment( lib, "svml_dispmt" )
    #pragma comment( lib, "libiomp5md" )
  #endif
#endif
```

(ippcore.h)

```
#if !defined( _IPP_NO_DEFAULT_LIB )
  #if defined( _IPP_SEQUENTIAL_DYNAMIC )
    #pragma comment( lib, __FILE__ "/../../lib/" _INTEL_PLATFORM "ippcore" )
  #elif defined( _IPP_SEQUENTIAL_STATIC )
    #pragma comment( lib, __FILE__ "/../../lib/" _INTEL_PLATFORM "ippcoremt" )
  #elif defined( _IPP_PARALLEL_DYNAMIC )
    #pragma comment( lib, __FILE__ "/../../lib/" _INTEL_PLATFORM "threaded/ippcore" )
  #elif defined( _IPP_PARALLEL_STATIC )
    #pragma comment( lib, __FILE__ "/../../lib/" _INTEL_PLATFORM "threaded/ippcoremt" )
  #endif
#endif
```

関連情報

XLSOFT

- IPP サンプルコード ダウンロード
  http://www.intel.com/software/products/ipp/samples.htm

- 製品バージョンの確認方法
  ippGetLibVersion() 関数をコールするか、¥include¥ippversion.h で確認する。

- スレッド化された IPP 関数のリスト:
  ¥Composer XE 2013 SP1¥Documentation¥en_US¥ipp¥ThreadedFunctionsList.txt

- IPP ライブラリー間の静的リンクにおける依存関係
  IPP ユーザーズガイドの [インテル® IPP とアプリケーションのリンク]-[アプリケーションで必要なインテル® IPP ライブラリーの選択]-[スタティック・リンク用ライブラリー]-[ドメイン別のライブラリー依存関係] を参照

- OpenCV で IPP ライブラリーの利用
  Cmakeを使用して IPP 8.1 を導入することができる。
  詳細は、インテルUS サイト内で公開されている関連記事を参照。

- DependencyWalker ツールと Dumpbin コマンドの利用
  アプリが必要とする DLL や、オブジェクトが含むシンボルなどを確認できる。