第5章 開発ツールとソフトウェアの基礎

第2章でも軽くふれたように，Analog DevicesはDSPプログラムの開発環境として，VisualDSP++を中心とするシステムを供給しています．VisualDSP++はプロジェクト管理，エディタ，コンパイラ，アセンブラ，リンカ，ROM化ツール，シミュレータ，コンパイル済みシミュレータ，性能測定ツールを1パッケージにまとめた強力な統合開発環境で，プロジェクト管理統合開発環境プラグインとしてICEを接続することもできます．また，ICEをもっていなくてもUSBケーブル越しにPCとAnalog Devices製の評価基板を接続して簡単な評価を行うこともできます．

この章では，VisualDSP++と評価基板であるEZ-KIT Lite BF533を組み合わせて使う場合に最低限知っておくべきことを説明します．一般にツールの使い方を細かく書くと，バージョン・アップと同時にその情報の価値が落ちてしまいます．そのため，本書ではマニュアル本的な情報は極力少なくしたいと考えていますが，この章と第10章，第11章だけは例外としてツールの使い方を細かく説明します．

5-1 EZ-KIT Lite版VisualDSP++のインストール

● 三つのステップに分かれるVisualDSP++のインストール

EZ-KIT Lite版VisualDSP++のインストールは，三つのステップに分かれています．すなわち，

- VisualDSP++のインストール
- EZ-KIT Lite用ドライバのインストール
- ライセンスのインストール

の3ステップです．これらのうち最初の2ステップは，順番どおりに行えば難しいことはありません．

最初にVisualDSP++をCD-ROMからインストールします．インストールが終了したら，念のためにPCを再起動してください．次にEZ-KIT Liteに電源を入れ，USBケーブルでPCと接続します．環境に問題がなければこれでデバイス・ドライバがインストールされます．EZ-KIT Lite BF533のUSBポート横のUSB MONITOR LEDが点灯すればインストールは成功しています(**写真5-1**)．インス

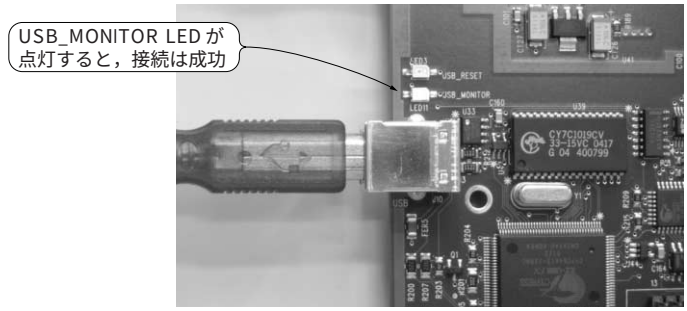

写真5-1　EZ-KIT Lite BF533のUSB MONITOR LED

トール後に念のためUSBケーブルをはずして再びPCを再起動し，ログインして再度USBケーブルを挿してください．

● やや面倒なライセンスのインストール

インストールが終わったら，最後にライセンスを設定します．ライセンスの設定はやや面倒で，以下のステップをふみます．

1. シリアル番号を入力してライセンスを仮設定状態にする．
2. Analog Devicesに登録してバリデーション・コードを取得する．
3. バリデーション・コードを入力してライセンスを恒久設定状態にする．

VisualDSP++にシリアル番号を入力するとライセンスが仮設定状態となり，10日間利用可能になります．この間にAnalog DevicesのWebサイトからユーザ登録を済ませ，バリデーション・コードを取得します．バリデーション・コードをVisualDSP++に入力すると，恒久的に利用できるようになります．

Column … 5-A　インストール・ディレクトリ

VisualDSP++のインストール・ディレクトリは，メジャー・バージョンごとに違います．たとえば筆者のシステムでは次のようになっています．

```
C:¥Program Files¥Analog
    Devices¥VisualDSP 3.5 16-Bit
C:¥Program Files¥Analog
    Devices¥VisualDSP 4.0
```

いうまでもなく，上がVisualDSP++ 3.5で，下が4.0です．VisualDSP++ 4.0では16/32ビット製品群の開発ツールが統合されたため，以前あった“16-Bit”という言葉が取れています．

本書で「インストール・ディレクトリ」と呼ぶときには上記のようなVisualDSP++関連ディレクトリの基点となる場所を指しますので，覚えておいてください．また，ファイル・パスはとくに断らないかぎりインストール・ディレクトリを基点としたものとします．

● ライセンス・インストール手順の実際

実際の手順を追ってみましょう．EZ-KIT Liteに電源を入れ，PCに接続します．USB MONITOR LEDが点灯したのを確認してVisualDSP++を立ち上げます．最初の起動であれば，ライセンスの登録が必要であると知らせるダイアログが表示されます(**図5-1**)．そこでLicenses…ボタンを押すと，ライセンス管理ダイアログが現れます(**図5-2**)．ライセンス管理ダイアログが現れない場合は，メニュー・バーからHelp⇒About VisualDSP++…項目を選び，Licensesタブをクリックしてライセンス管理ダイアログを出してください．ダイアログのNewボタンをクリックしてEZ-KIT Liteに同梱されているCD-ROMジャケット裏のシリアル番号を入力します(**図5-3**)．シリアル番号は，KIT-で始まる文字列です．

ユーザ登録は先のシリアル番号の入力から10日以内に済ませます．10日を越えると一時的にVisualDSP++を使えなくなるので注意が必要です．また，登録からバリデーション・コードの発行まで，3日ほど見ておくほうがよいでしょう．ユーザ登録は，Analog DevicesのWebページから行い

Column…5-B　ライセンスについて

VisualDSP++には，ライセンス形態に応じて三つの亜種があります．

・正規製品版
・EZ-KIT lite版
・お試し版(Test Drive)

これらの三つの亜種は，すべて同じバイナリからなっています．つまり，インストールされるプログラムはまったく同じなのです．それぞれの版の違いは，登録するシリアル番号とバリデーション・コードからなります．

ライセンスごとの違いは，VisualDSP++4.0においては表5-Aのようになっています．

バイナリがすべての版で共通であるため，どのライセンスでもアップデートを適用することができます．

表5-A　ライセンスごとの違い

ライセンス形　態	シリアル番号			バリデーション・コード入力後の制限
	形　式	入手法	入力後の制限	
正規製品版	ADI-###…	ADI-###… シリアル番号は製品CDのジャケットに貼ってある．	機能制限なし．バリデーション・コードを入力しないと30日間で使用不能になる．	制限なし
EZ-KIT Lite版	KIT-###…	KIT-###… シリアル番号は製品CD-ROMのジャケットに貼ってある．	機能制限なし．バリデーション・コードを入力しないと10日間で使用不能になる．	インストールから90日間は機能制限なし．その後，シミュレータとエミュレータが使用不能になる．USBケーブルによるEZ-KIT LiteとPCの接続は制限されない．また，ビルドできるコード・サイズに制限がかかる．
お試し版(Test Drive)	TST-###…	TST-###… シリアル番号はAnalog Devicesのウェブサイトから取得	機能制限なし．90日間で使用不能になる．	入力できない．

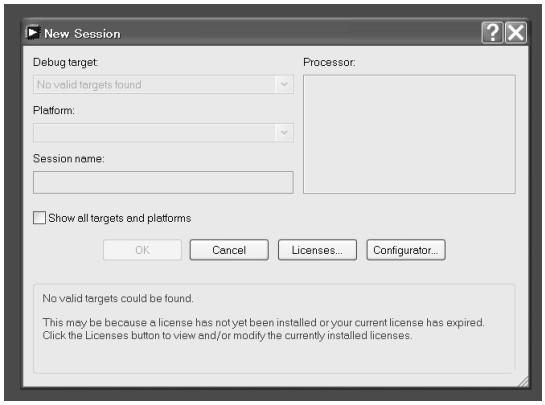

図5-1　初回立ち上げ時のダイアログ

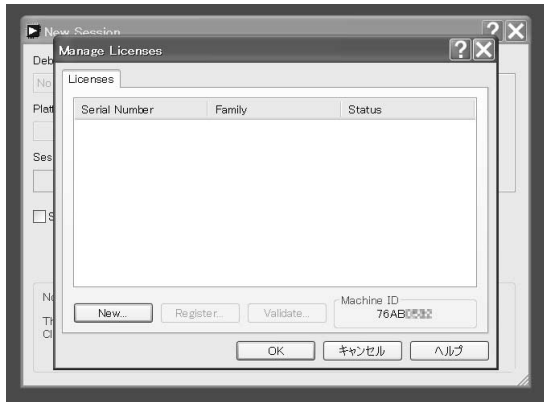

図5-2　ライセンス管理ダイアログ

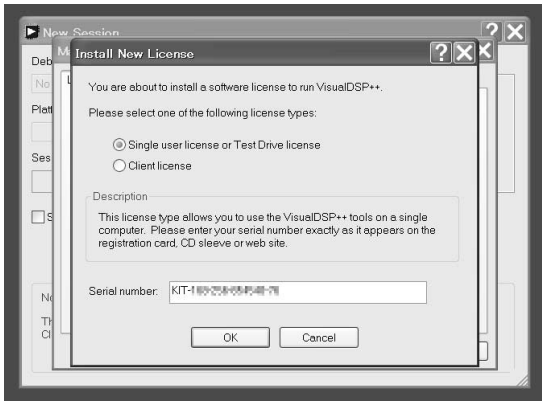

図5-3　シリアル番号の入力

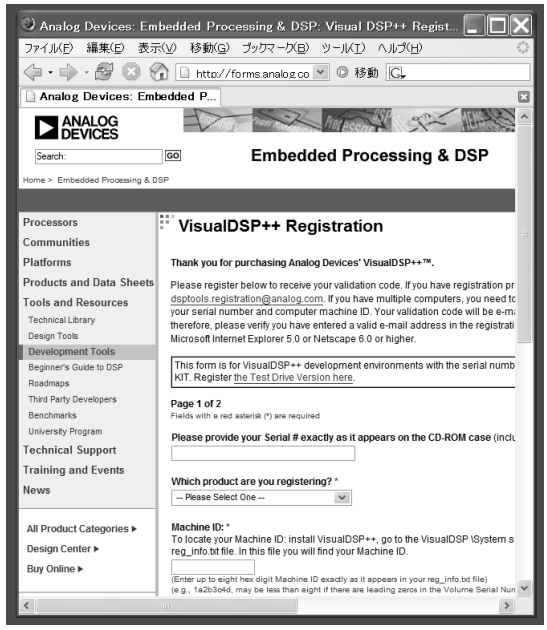

図5-4　ライセンスの入力

ます．このページには**図5-2**のライセンス管理ダイアログのRegisterボタンをクリックしてアクセスできます．**図5-2**ではRegister…ボタンは無効になっていますが，シリアル番号を入力してリストの中で選択するとボタンが使用可能になります．Webページが開いたら，シリアル番号と**図5-2**の右下に表示されているマシンIDのほか，ユーザ情報を登録します(**図5-4**)．登録が完了すると数日でバリデーション・コードが送られてきます．

バリデーション・コードは，シリアル番号同様にライセンス管理ダイアログから登録できます．バリデーション・コードを登録するとライセンスがPCに固定され，恒久的にVisualDSP++を利用できるようになります(**図5-5**)．

● 最新アップデートの適用

ライセンスのインストールが成功したら，最新のアップデートを適用します．アップデート・ファイルはVisualDSP++の最新版とともにAnalog DevicesのWebサイトからダウンロードできます[注1]．このファイルは本書の執筆時点では不定期に更新されています．バグが取れていきますのでなるべくアップデート・ファイルを適用することをおすすめします．

アップデート・ファイルを入手するには，Windowsのコントロール・パネルから，「プログラムの追加と削除」を開きます．一覧の中からVisualDSP++を探して「変更と削除」ボタンをクリックします．するとダイアログが現れる(**図5-6**)ので，“Go to Analog Devices web site”を選んでからNextボタンをクリックします．VisualDSP++のアップデート・ページが開くので，最新のアップデート・ファイルをダウンロードしてください．ファイルの拡張子は.vduです．

アップデート・ファイルをダウンロードしたらWebページを閉じ，再びコントロール・パネルのリストからVisualDSP++を探して「変更と削除」をクリックします．ダイアログが開いたら，今度は“Apply a downloaded update”を選んでNextボタンをクリックします．あとは，ウィザードの指示にしたがってアップデートを進めます．

5-2　統合環境でプログラムを走らせる

ライセンスの設定が終わったら，いよいよVisualDSP++を使えるようになります．

最初にEZ-KIT Lite版のVisualDSP++を使うときの注意を書いておきます．正規版のVisualDSP++を買わずにEZ-KIT Lite版のみを買った場合には，EZ-KIT Liteなしで VisualDSP++を使うことはできません．VisualDSP++を立ち上げる前に必ずEZ-KIT Liteの電源を入れ，PCに接続してからMONITOR_LEDが点灯するのを待ちます．点灯を確認(**写真5-1**)してからVisualDSP++を起動します．

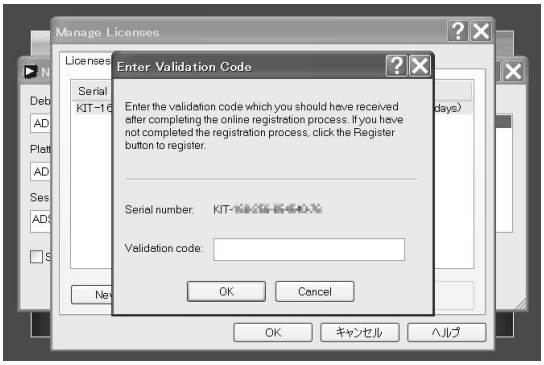

図5-5　バリデーション・コードの入力

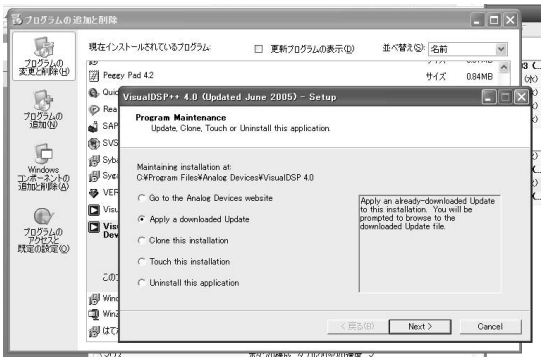

図5-6　アップデート・ダイアログ

注1：http://www.analog.com/processors/processors/Blackfin/crosscore/toolsUpgrades/

● さっそくプログラムを作ってみる

VisualDSP++が立ち上がったら，さっそくプログラムを作ってみましょう．VisualDSP++では，プログラムは「プロジェクト」という単位で管理されています．プロジェクトとは，ソース・コードやライブラリからなるツリー構造で，VisualDSP++上でグラフィカルに管理することができます．新しいプロジェクトを作るには，メニュー・バーのFile⇒New⇒Project…項目を選びます．そうすると，プロジェクト・ウィザードが現れてプロジェクトの初期設定が始まります(**図5-7**)．ここではプロジェクトの保存場所とプロジェクトに付ける名前を指定します．保存場所とプロジェクト名には日本語を含まないように気をつけてください．英語圏で作られたソフトウェア全体にいえることですが，う

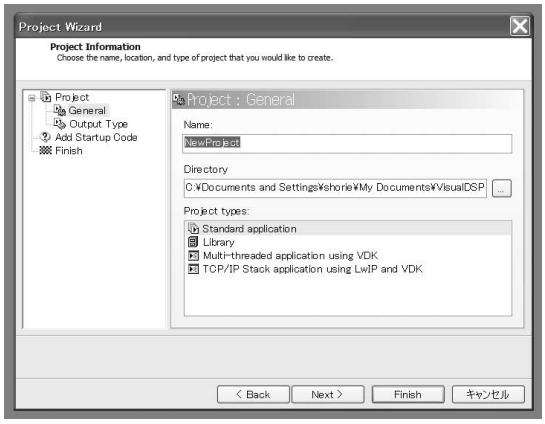

図5-7　プロジェクト・ウィザード

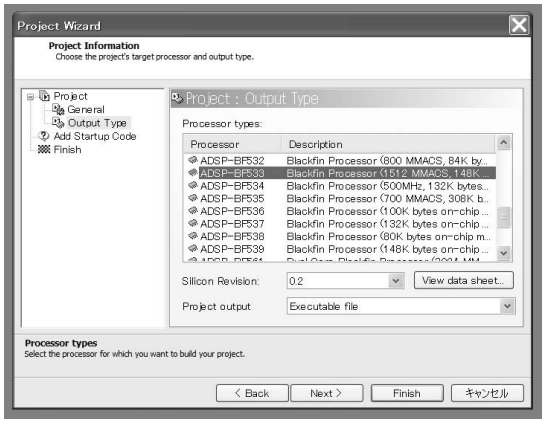

図5-8　プロジェクトの出力設定

Column…5-C　日本語ディレクトリについて

VisualDSP++は，日本語をはじめとする2バイト・コードからなるディレクトリをうまく扱えません．この結果，次のような場合には動作が非常に怪しくなります．

一つは，プロジェクト・ディレクトリを日本語ディレクトリの下に置いた場合です．典型例がデスクトップで，ここにプロジェクト・ディレクトリを作ると，ビルドで失敗してしまいます．プロジェクト・ディレクトリには漢字やひらがなをはじめとして2バイト・コードを使用しないでください．Windows 2000やXPの場合，マイドキュメント・フォルダの下にプロジェクトを作っても正しく動くため，日本語ディレクトリが許されているように錯覚してしまいます．しかし，マイドキュメント・フォルダのパス名はMy Documentsであるため，実際には2バイト・コードを含んでいません．

問題が発生するもう一つの場合は，VisualDSP++を日本語ディレクトリにインストールしてしまった場合です．中にはデスクトップの下にプログラムをインストールして「動かない」とおっしゃる方もあります．VisualDSP++はなるべくデフォルト・ディレクトリにインストールしてください．

っかり日本語のファイルやディレクトリを使うと理解に苦しむエラーを起こすことがあるので注意が必要です．

NEXTボタンを押すと，ウィザードが次の設定に進みます(**図5-8**)．ここでは，プロジェクトのターゲット・プロセッサなどコードの出力に関する設定を行います．EZ-KIT BF533の場合，Processor TypeはADSP-BF533にしてください．Silicon Revisionは，使用しているチップの版です．これは**写真5-2**のように，パッケージ上に印字されているものを入力してください．Project Typeは，Executable Fileを選びます．設定が終わったら，Finishボタンを押してください．まだ設定できる項目もありますが，今は無視してかまいません．

以上の設定を終えると，**図5-9**のように空プロジェクトによるVisualDSP++の画面が現れます．左側にはプロジェクト・ウィンドウ，右側にはディスアセンブル・ウィンドウ，下側に出力ウィンドウが配置された構成です．インストール直後と似ていますが，左側のプロジェクト・ウィンドウに新しいプロジェクトが表示されている点が異なります．

写真5-2　シリコン・リビジョン

Column…5-D　アノーマリはチェックしてください

Analog Devicesはチップの不具合のうち設計に起因するものを「アノーマリ」として公開しており，製品ページからダウンロードして読むことができます．

アノーマリの中には軽度のものもあれば，かなりきついものもあり，さまざまです．たいていのものには回避策がありますが，回避が困難なものの場合は，次のリビジョンで変更されます．

公表されているアノーマリのリストには，それぞれの回避策が提示されています．アノーマリは定期的にアップデートされるので，ときどき目を通すようにしてください．

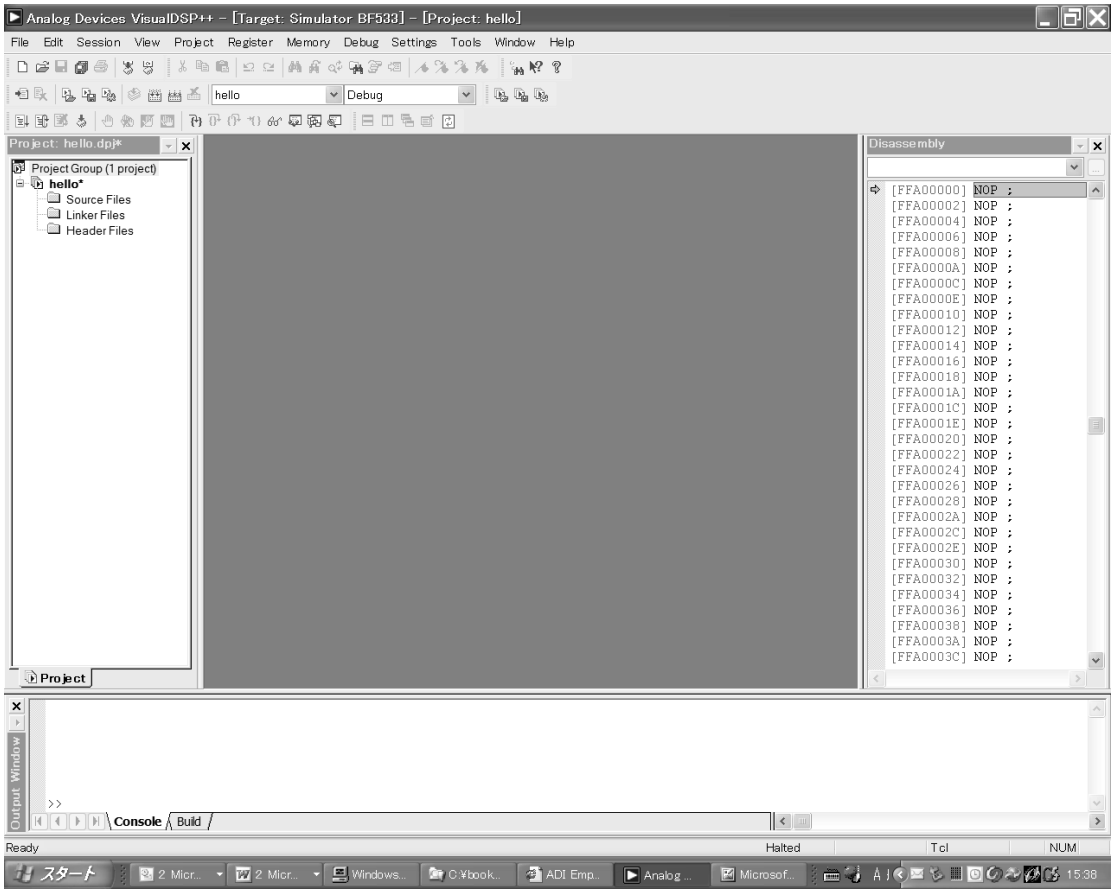

図5-9　新しいプロジェクト

● ソース・ファイル作成

さっそく何かプログラムを作ってみましょう．まずはソース・ファイルを作ります．ソース・ファイルは，メニュー・バーからFile⇒New…⇒File項目を選ぶと作られます．ソース・ファイルを作ったら，名前をつけて保存します．ここで注意すべきこととして拡張子の指定があります．ワープロソフトなどは拡張子を勝手に付けてくれるので，拡張子なしでファイルを保存することに慣れている人がいるかもしれません．しかし，VisualDSP++は各種のファイルを作る都合上，デフォルトの拡張子は「なし」になっています．そのため，名前をつけてファイルを保存する場合，必ずユーザが拡張子をつけて保存しなければなりません．とりあえず今は`main.cpp`という名前でファイルを保存してください．

● プロジェクトへのファイル追加

ファイルを保存したら，今度はプロジェクトに追加します．プロジェクトにファイルを追加すると，

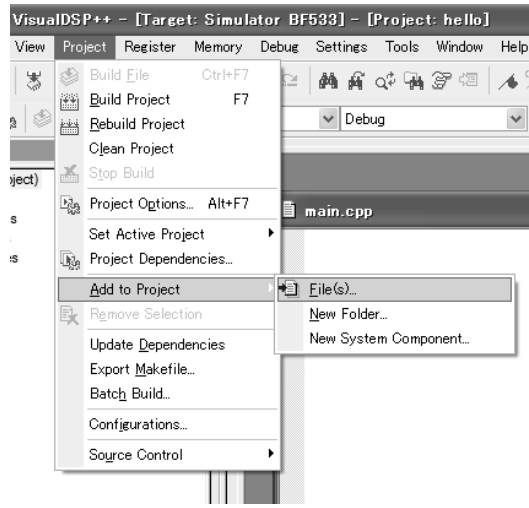

図5-10　プロジェクトへのファイル追加

ビルド時に統合環境が自動的に適切なコード・ジェネレータを呼び出してくれます．プロジェクトへの追加は，Projectメニューから Add to Projectサブメニューの Files…項目を選んでください(**図5-10**)．ファイル選択ダイアログが現れるので，先ほどの main.cpp を選択します．すると，プロジェクト・ツリーに追加したファイルが現れます．

ファイルの追加が終わったら，空のままのソース・ファイルにプログラムを書き込みましょう．何でもよいのですが，定番の Hello, World!にしておきましょう．LEDの制御などの組み込みらしいプログラムは後の節で紹介します．

```
#include <iostream.h>

int main(void)
{
    cout << "Hello, World!" << endl;
}
```

入力したら，保存してビルドをかけます．ビルドは Projectメニューから Rebuild All項目を選んでください．プログラムのビルドが進み，エラーがなければ実行ファイルの EZ-KIT Liteへのダウンロードが始まります．ダウンロードには少し時間がかかりますが，やがて main関数の頭でブレークがかかって停止します．

この状態で EZ-KIT Liteは停止しており，VisualDSP++はユーザの操作待ちです．そこで Debugメニューの Run項目を選択すると実行が始まり，VisualDSP++の出力ウィンドウにメッセージが表示され，実行が終了します．

VisualDSP++によるプログラミングの例に関しては第6章や第7章にもう少し詳しく説明していますので，そちらも参照してください．

5-3 C/C++言語のコンパイラ，アセンブラ，リンカ

● 高級言語とアセンブリ言語の混用を考えた設計

VisualDSP++には，C/C++言語のコンパイラ，アセンブラ，リンカが一式付属してきます．これらのツールはC++言語のマクロ・プロセッサを共用しており，アセンブラとコンパイラがプロセッサ

Column…5-E エディタの日本語対応化

VisualDSP++のエディタは，日本語対応化することができます．日本語対応によって日本語のコメントを入力，編集，表示できるようになります．ただし，この機能は正式には未サポートとなっているので注意が必要です．

日本語への対応は，エディタのフォント変更によって行います．フォントの設定はSettingsメニューのPreferences項目を選ぶとダイアログが現れるので，ツリーの中からEditorをクリックすることで設定画面を呼び出せます．設定画面のTypeコンボボックスを選んで，どのファイルのフォントを設定するのか選びます．普通はC/C++のソース・ファイルと，アセンブリ言語のソース・ファイルだけで十分でしょう．ファイル・タイプを選んだら，font…ボタンをクリックしてフォントを選択します．

フォントは日本語を表示できるものでなければなりませんが，筆者は普通Fixedsysを使用しています．このフォントは等幅であるため，プログラムの作成に適しています．それ以外のフォントではうまく日本語が表示されない場合もあるので気をつけてください．

なお，以上の変更を行っても，アセンブリ言語のソースの場合は，コメントが正しく表示されません．これはシンタックス・カラーリング機能のせいです．コメントの扱いに問題があるらしく，1文字の予約語と同じキャラクタが2バイト・コードの中に存在すると，日本語文字が崩れてしまいます．仕方がないので，シンタックス・カラーリングの機能を変更することでこの問題を解決します．具体的には，カラーリング対象文字列から1文字のものをすべて取り除きます．カラーリング文字列の宣言は，インストール・ディレクトリの下のsys¥blackfinasm.iniファイルで行われているので，このファイルをテキスト・エディタで開きます．そして次のように，1文字予約語の宣言をすべてコメントアウトします．

```
JUMP=Mnemonic
;L=Mnemonic     ;予約語‘L’の宣言をコメントアウト
LE=Mnemonic
```

コメントアウトは‘;’(セミコロン)で行います．1文字予約語をすべてコメントアウトすれば，文字化けはなくなります．

以上の作業で，日本語のコメントを入力・表示できるようになります．なお，エディタ自身は日本語に対応しているわけではなく，無理やり日本語を表示させているだけです．そのため，2バイト文字の削除は削除キーを2回押さなければならないという制限があります．

繰り返しになりますが，これはAnalog Devicesが公式にすすめている方法ではないので，対応は自己の責任の元で行ってください．

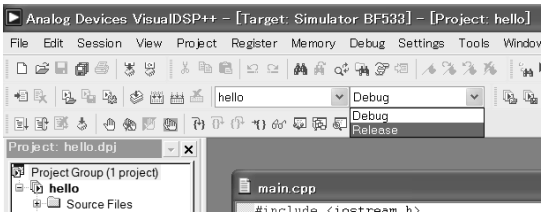

図5-11　コンフィグレーション・コンボ・ボックス

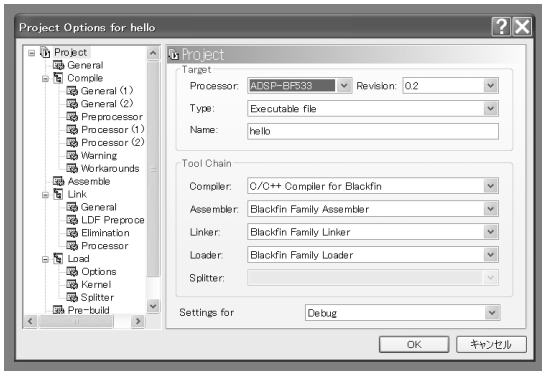

図5-12　プロジェクト・オプション

のレジスタ・アドレスを定義したマクロを共用できるなど，はじめから高級言語とアセンブリ言語の混用を考えた設計になっています．

コンパイラは，とくに16ビット演算の最適化に力点を置いて設計されています．これは，Blackfinには32ビットRISCプロセッサとしての「顔」のほかに，16ビットDSPの「顔」もあるからです．一方でC/C++言語は，本来は信号処理アルゴリズムのような極度に性能に敏感な分野での応用を考えて作られたものではありません．そのためVisualDSP++には，コンパイラに最適化のヒントを与えるためのpragmaや，特殊な命令を活用するための組み込み関数が用意されています．とくに組み込み関数は，ライブラリ関数と異なり，動作をコンパイラが理解しているためかなり本格的な最適化を行うことができます．

コンパイラとアセンブラに関しては，よほど特殊なことをしないかぎり神経質にオプションをいじくり回す必要はありません．きつい最適化をかける必要がないならば，最適化やデバッグ・オプションの設定はツールバーのConfigurationコンボ・ボックスを切り替えるだけで間に合います(**図5-11**)．Configurationコンボ・ボックスはデバッグ・ビルドとリリース・ビルドを簡単に切り替えるためのもので，このコンボ・ボックスに連動してコンパイラやアセンブラの設定済みオプションが変わります．それぞれのビルドに対応する設定は，プロジェクト・ダイアログから変更可能です．また，デバッグとリリース以外のビルドを追加することもできます．きつい最適化をかける必要がないならば，最適化やデバッグ・オプションの設定はデフォルトのままでよいでしょう．

● 細かな設定はオプション・ダイアログから行う

細かい設定を行いたいときは，コンパイラとアセンブラのオプションの主なものはプロジェクト・オプション・ダイアログから設定可能です(**図5-12**)．このダイアログはメニュー・バー⇒Project⇒Options..項目を選択すると現れます．プロジェクト・ダイアログを使って行うよりも踏み込んだ最適化設定に関しては，第12章で説明します．

コンパイラを使うかアセンブラを使うかの判断は，統合開発環境が自動的に行います．つまりソース・ファイルが.cあるいは.cppという拡張子をもっていればビルド時にコンパイラが呼び出され，ソ

ース・ファイルが.ASMという拡張子をもっていれば，ビルド時にアセンブラが呼び出されます．したがって，ユーザがファイルとツールを明示的に結びつける必要はありません．アセンブリ言語を使う場合の文法などの説明は，第9章で行います．

リンカの呼び出しも，自動的に行われます．リンカに関してはソース・コードだけではなくライブラリの選択や配置の最適化，メモリ領域の調整といった組み込みならではのことが絡んでくるため，製品化の前にはかなりの調整を行わなければならないのが普通です．しかし手始めとしては，デフォルトの設定も十分使えます．リンカについては第10章で説明します．

5-4 ペリフェラルにアクセスする

● MMRにアクセスする

さて，ADSP-BF533の内蔵ペリフェラルはすべてメモリ空間中のMMR（Memory Mapped Register）領域に割り当てられています．このレジスタにアクセスしてみましょう．

レジスタへC/C++言語からアクセスするときには，ポインタからアクセスします．このためのポインタ宣言はVisualDSP++に同梱されているインクルード・ファイルで行われています．cdefBF53x.hというファイルがそれで，#include文で読み込むとポインタ・マクロの宣言が取り込まれます．このマクロ宣言はすべて型付きポインタとなっているので，C/C++言語から比較的安全に使うことができます．一例をあげます．

```
#include <cdefBF53x.h>

volatile unsigned short sw;
int main(void)
{
   *pFIO_DIR = 0x0000;             // FIOを入力に
   *pFIO_INEN = 0x0100;            // PF8をイネーブルに
   sw = *pFIO_FLAG_D & 0x0100;     // PF8の状態を読み込む
}
```

このプログラムはプログラマブル・フラグ（FIO）の8番ピン（PF8）を入力に設定して状態を読み込みます．PF8はEZ-KIT LiteのSW4に接続されているので，プログラムの実行によってSW4の状態を読み込むことができます．SW4を押すと変数swには0x0100が，離すと0x0000が代入されます．

マクロを使ったために，FIO_DIRやFIO_INENレジスタもポインタから指し示される変数としてアクセスできます．そのためプログラムの見通しが非常によくなります．マクロ定義されているポインタは定数なので，ポインタに値を代入して壊してしまう心配はありません．また，ポインタはvolatile unsigned shortへのポインタなので，コンパイラの最適化機構がアクセスを消し去ってしまう心配もありません．