第10章 ソフトウェア開発の流れとCode Composer Studio（CCS）の基本的な機能

第2部ではハードウェアを中心に解説してきました．この第3部では，ソフトウェアの開発について説明します．最初に，ソフトウェア開発全体の流れを解説します．そのあと，統合開発環境Code Composer Studio（CCS）の基本的な機能について，コンパイルして実行ファイルを作り，デバッグを行ってからベンチマークをとるまでの流れで説明します．この章を理解すると，CCSがもっているさまざまな機能を使って，DSP上でプログラムを動作させ，デバッグ/ベンチマーク計測ができるようになります．

10-1 ソフトウェア開発の流れ

ディジタル信号処理の新しいアルゴリズムは，通常，研究所で研究/作成され，C/C++のプログラムで，パソコンやワークステーション上で評価している場合が多く見受けられます．研究者はアルゴリズムを考えることが仕事ですから，その検証手段は特定はされません．しかし，実際の組み込みシステムを構築する場合，どうしても，いろいろと制約が付いてしまいます．ときにはシステムに搭載しているメモリが少なかったり，ときには筐体が小さく発熱（消費電力）を抑えることが必要だったりします．そこで，パソコンより低消費電力で高性能なDSPによるシステムが登場するわけです．

この場合組み込みユーザは，パソコンやワークステーション上で実現しているC/C++のプログラムをC6000 DSP上でコンパイル，実行させることから始まります．もちろん，検証用の入力データとその結果を用意し，正しく動作するまでデバッグしていきます．

正しい動作を確認した後，プログラム最適化とベンチマーク（実行速度）計測を繰り返し行い，満足のいく性能まで向上させます（**図10-1**）．プログラム最適化は，コンパイラの最適化オプションを付け/キャッシュをONにするといった，基本的な最適化から始めます．その後，さまざまな最適化手法や，Code Composer Studioに付属しているツールで高速化していきます（第11章，第12章参照）．プログラムの高速化はC6000 DSPの場合，ほぼ100%，C言語上での最適化となります．

C6000 DSPの場合，プログラム最適化前は，思っていたような性能が出ていないはずです．しか

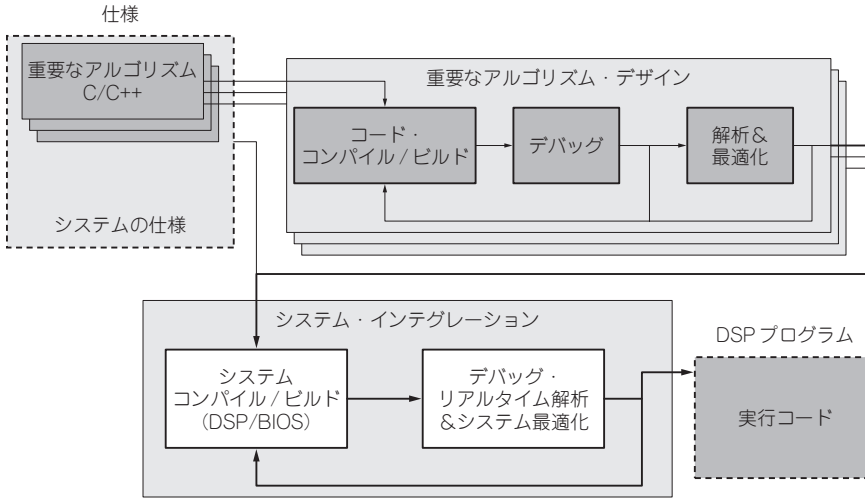

図10-1　ソフトウェア開発フロー

し最適化後は(アルゴリズムに依存するが)数十倍の速度を得ることも十分可能です.

これらのアルゴリズムが満足のいく性能まで向上させるとともに，これらを含んだシステムとして，プログラムを構築します．TI社ではDSP/BIOSというリアルタイムOS(RTOS)が無償提供されています．その後，システムとしてのデバッグ/最適化を行います．システム全体でのリアルタイム解析機能も用意されています．これで，プログラムは完成です.

まとめると，次のようなソフトウェア開発の流れとなります.

①パソコン/マイコンで作成したC言語プログラムを，DSP上でコンパイル/デバッグ

②ベンチマークを取りながら最適化を行い，満足のいく性能まで向上させる

③これらのプログラムを組み込み，システムとして構築，デバッグ，システム全体の最適化

第3部では，上記の流れにそって，CCSの基本的な使い方から，プログラム高速化の方法，DSP/BIOSを使ったシステム構築/最適化まで解説します.

10-2　シミュレータ，エミュレータ，DSKの違いの詳細

DSPのプログラムを評価するためには，大きく分けて次の3種類があります.

1. パソコン上で動作するシミュレータを使って評価する場合
2. 実機上でエミュレータを使って評価する場合(**図10-2**)
3. TIが提供する安価な評価環境 DSPスタータ・キット(DSK)を使用する場合

1番目は,「シミュレータ」で，パソコン上でDSPのプログラムをシミュレーションします.

2番目は，ユーザが作成したDSPボード上にJTAG端子を出してXDS510-USBやXDS560などのJTAG対応のエミュレータを接続します．このエミュレータ経由で，ボードに搭載されているDSP上

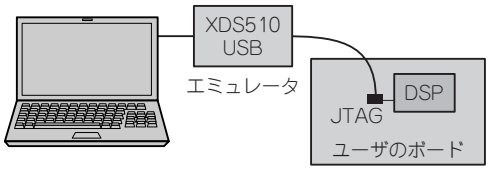

図10-2　エミュレータ経由での評価環境

での評価を行うことができます．この場合，動作するDSP搭載ボードとエミュレータが必要です．

3番目は，DSPスタータ・キット(DSK)を使った場合です．このDSKは，ボード上にエミュレータの回路が組み込まれていると考えてください．ユーザは，USBケーブルでパソコンとDSKを接続するだけで，DSK上に搭載されているDSPの評価を行うことができます．

これら3種類とも，まったく同じTI DSPの統合開発環境 Code Composer Studioの環境で評価できます．ドライバを選択するだけで，シミュレータやエミュレータ，DSKに対応できます．この選択は「CCS Setup」で，CCSを起動する前に行います．

10-3 CCS Setup

とりあえず，CCS Setupを立ち上げてみましょう．CCS3.1では，**図10-3**のような画面が表示されます．

一番左の「System Configuration」欄には，現在選択しているドライバ(この図では，DM642のSimulatorを選択)，真ん中の「Available Factory Boards」には，インストールされているドライバ

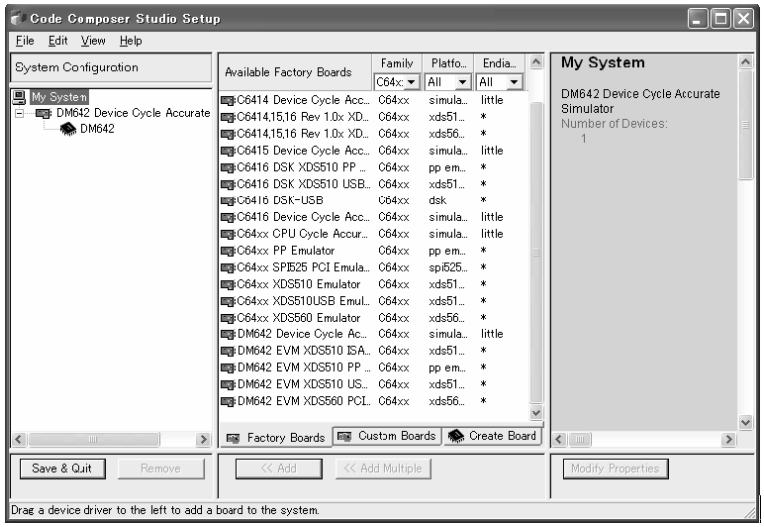

図10-3　CCS Setupの画面

が表示されます．新しくモードを切り替える/ドライバを入れ替える場合は，現在選択しているドライバを消去し，新しいドライバを追加します．たとえば，C6416-DSK上でCCSを立ち上げたいときには，「System Configuration」内のMy Systemsにあるドライバを選択し下のRemoveボタンを押して現在のドライバを削除した後，「Available Factory Boards」内のC6416-DSK-USBを選択して，下の<< Addボタンを押すと，My Systemsに組み込まれます．マウスを使って，C6416-DSK-USBドライバをドラッグ＆ドロップでMy Systemsにもっていっても組み込み可能です．これで，シミュレータ/エミュレータ/DSKの選択は終了です．

10-4 GEL(General Extension Language)ファイル

CCSを立ち上げる前に，もう一つ行うことがあります．それは，CCS起動時の各ボードの設定や，ボードに合わせたCCSの設定です．これは，C言語ライクなスクリプト言語 GEL(General Extension Language)を用いて設定できます．C6416T-DSKやC6713-DSKの場合，ボードに合わせて設定を記述したファイル「GELファイル」が定義されているので，何もする必要はありません．しかし，ユーザが作成したボードの場合，設定を変更したいときは，その設定にあったGELファイルを作成する必要があります．この作成したGELファイルは，ドライバで選択しているプロセッサのPropertiesで指定しています(**図10-4**)．

GELはC言語ライクなスクリプト言語になっており，C言語を知っている人であれば簡単に理解できるようになっています(ユーザ定義の関数やif文も可能)．CCSへの設定は，「GEL_」から名前が始

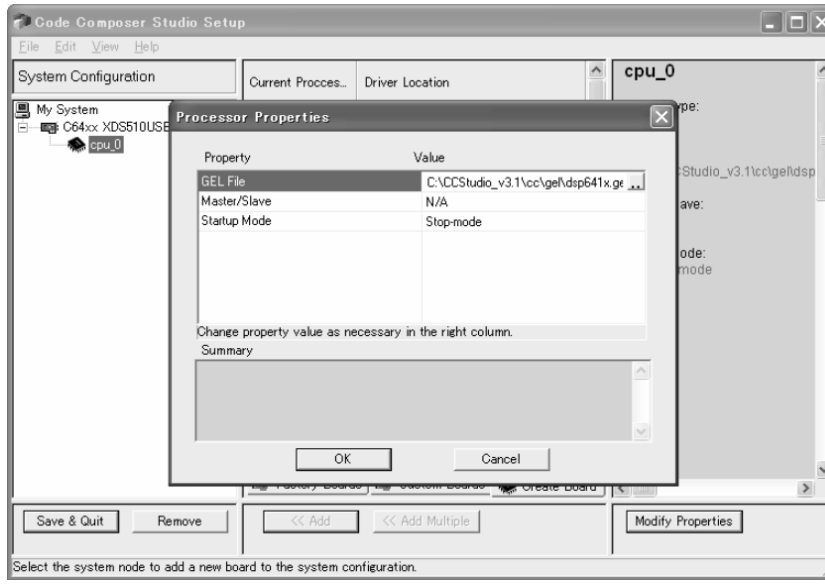

図10-4　ドライバで選択しているプロセッサのプロパティ

表10-1　GEL callback関数

関　数	説　明
StartUp()	CCS起動時に実行される
OnTargetConnect()	CCS上からConnectを行ったときに実行される
OnReset(int)	CCS上からDSP RESETを行ったときに実行される
OnPreFileLoaded()	CCSからプログラムをロードする前に実行される
OnFileLoaded(int , int)	CCSからプログラムをロードした後に実行される
OnRestart(int)	CCS上からRESTARTを行ったときに実行される
OnHalt()	CCS上からHaltを行ったときに実行される

まるGEL関数で行います．このGEL関数はCCS上でのコマンドに対応した関数を用意しており，CCSへの各種設定を行えます．GELの文法を組み合わせ，ユーザのボードに合わせた初期設定を記載したり，よく使用するコマンド群をユーザ定義のCCSメニューに入れることもできます．

また**表10-1**のように，CCS上で特定のコマンド実行時に起動されるGEL callback関数が規定されています．これらの関数内にはCCSの設定やボードの設定を記載します(Connectなどのコマンドについては後で説明する)．

ここで，設定が必要な二つの関数について説明します．一つ目はStartUP関数で，CCS起動時に実行されます．この関数では，必ずDSPから見たメモリ・マップの設定をCCSに教える必要があります．これには，内部RAMや内部ペリフェラルの設定空間もすべて必要です．メモリ空間をCCSに教えないと，ロード時にエラーが生じたり正しくメモリの内容を表示しなかったりするので，必ず設定してください．このメモリ・マップの設定を行うために，GEL_MapOn/GEL_MapReset/GEL_MapAddなどのGEL関数が定義されています．**リスト10-1**に例を示します．

二つ目はOnTargetConnect()関数で，エミュレータがDSPに接続しにいく(Connect)ときに実行されます．通常は，エミュレータを接続したらすぐにすべてのメモリ空間を表示できるように，EMIF

Column…10-A　CCSのHelpはお勧め

ソフトウェアを開発しているときにDSPに関して調べたい内容があった場合は，Code Composer StudioのHelp機能をお勧めします．CCS上のメニューのHelp->Contentsもしくは Windowsのスタート・メニューのTexas Instruments->Code Composer Studio 3.1->Documentation->TMS320C64x Help(CCS3.1の場合)で立ち上がります．

このHelpには，GELを含むCCSの機能はもちろん，C標準ライブラリの説明やチュートリアル，後で説明するDSP/BIOSやChip Support LibraryのAPI，ディジタル信号処理ライブラリや画像処理ライブラリのAPI，C6000コアの命令セットまで幅広く記載されています．とても便利なHelp機能です．何か分からない単語があれば，Helpで調べることをお勧めします．CCS3.1から，C6000 DSPのHelpは3分割され，C62x，C64x，C67xそれぞれのHelpとなっています．

リスト10-1　GEL関数の例

```
StartUp(){
     GEL_MapOn();              /* CCSのメモリ・マップ設定をONにする */
     GEL_MapReset();           /* CCSのメモリ・マップ設定をリセットする */
     GEL_MapAdd( 0x00000000, 0, 0x00100000, 1, 1 );  // Internal Memory
              /* 0番地から0x00100000の大きさ分RAMとしてCCSに教える */
     GEL_MapAdd( 0x01800000, 0, 0x00000058, 1, 1 );  // EMIFA CTL REGS
              /* 0x01800000番地から0x058の大きさ分RAMとしてCCSに教える */
     GEL_MapAdd( 0x01840000, 0, 0x00008300, 1, 1 );  // L2 REGS
     GEL_MapAdd( 0x01880000, 0, 0x0000000C, 1, 1 );  // HPI REGS

     /* … */

}
```

リスト10-2　EMIFの設定例

```
OnTargetConnect()
{
    #define EMIFA_GCTL       0x01800000
    #define EMIFA_CE0        0x01800008
    #define EMIFA_SDRAMCTL   0x01800018
    #define EMIFA_SDRAMTIM   0x0180001c
    #define EMIFA_SDRAMEXT   0x01800020
    *(int *)EMIFA_GCTL     = 0x00052078;
    *(int *)EMIFA_CE0      = 0xffffffd3;  /* CE0 SDRAM                    */
    *(int *)EMIFA_SDRAMCTL = 0x57115000;  /* SDRAM control                */
    *(int *)EMIFA_SDRAMTIM = 0x0000081b;  /* SDRAM timing (refresh)       */
    *(int *)EMIFA_SDRAMEXT = 0x001faf4d;  /* SDRAM extended control       */

    /* … */

}
```

の設定を行います．とくにSDRAMを搭載しているボードは，EMIFの設定をしないと正しく表示できません．EMIFの設定は，C言語のポインタの記述で指定できます．ほかにもPLLやボード固有の設定を行います．**リスト10-2**に例を示します．

ほかの`Callback`関数では，`OnReset`関数にはEMIF設定を記載したり，`OnPreFileLoaded()`関数へは，DSP自身をRESETする`GEL_Reset()`関数とEMIF設定を記載したりします．`GEL`関数やGEL文法の詳細については，CCSのHelp Fileを見てください．また，GELファイルのサンプルは，`¥CCStudio_v3.1¥cc¥gel`（CCS3.1の場合）を参照してください．

10-5　CCSを立ち上げる前に

CCS Setupでドライバ/GELの設定が終わりました．シミュレータの場合は，このほかに動作周波数など，DSP自身の詳細な設定もできます（**図10-5**）．

先ほどのGELファイルの設定時の画面で，**表10-2**のような設定が可能です．

XDS510-USBやXDS560エミュレータの場合，エミュレータとDSP搭載ボートを接続して，ボード

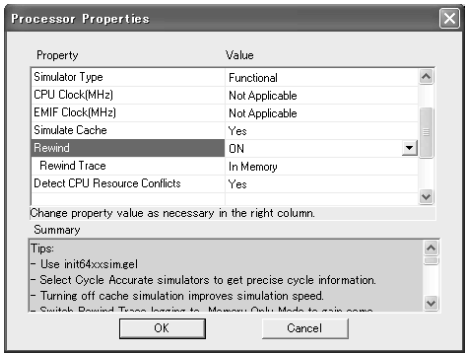

図10-5　シミュレータでのプロセッサのプロパティ

表10-2　シミュレータでの設定

GELファイルの設定	ファイル名の指定
デバイスの設定	このドライバでシミュレーションできるデバイスを選択
シミュレータの動作設定	Cycle Accurate(正しくサイクル数を計測) もしくは，Functional(コアのみシミュレーション)
CPUクロック	CPUコアの動作周波数を設定
EMIFクロック	EMIFクロック周波数を設定
CPUリソース・コンフリクト	検出する / しない
Reserveメモリ・アクセス	検出する / しない
エンディアン	リトル・エンディアン，もしくはビッグ・エンディアン

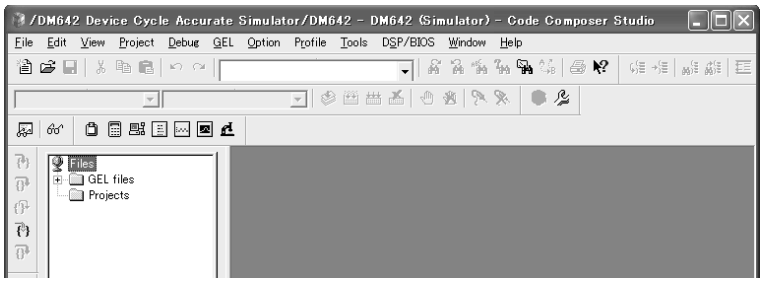

図10-6　CCSの起動直後の画面

の電源を入れてDSPを起動しておく必要があります．DSKの場合も，同じようにDSKとパソコンとをUSBケーブルで接続し，DSKの電源を入れてください．

さあ，CCSを実行してみましょう．**図10-6**のような画面が立ち上がります．