[第5章]

使い方を学ぶ練習プログラミング

FED WIZ-Cの使用法の習得

見本

ディジタル・クロックを作る前に，この章では実際にFED WIZ-Cで評価・練習用プログラムを作って動作させてみます．WIZ-Cの使い方を習得するために，LEDを点滅させたり，割り込みで正確な1秒を作り出すプログラムを作り，WIZ-Cのシミュレーション機能を使ってパソコン上で模擬的に動作させます．

この章で作るプログラムは本番で使うものもあるので，ここでWIZ-Cの操作手順やプログラムの内容をよく理解しておきましょう．

● はじめに

ここで作成するサンプル・プログラムは，できれば，簡単な回路を組み立てて実際に動かしてみたほうがよいと思います．けれども，単純なLEDの点滅やスイッチ入力は，ハードウェアを製作しなくてもWIZ-Cのシミュレーション機能を使うことで，パソコンだけでデバッグ，シミュレーションができるので，評価回路は無理に製作しなくてもかまいません．

5-1　WIZ-Cを使った開発の手順

WIZ-Cにはテキスト・エディタも内蔵されているため，別にエディタを用意する必要もなく，コンパイルとソース・ファイル修正の繰り返し作業には便利です．また，実際にPICにプログラムを書き込む前に，ある程度のシミュレーション（パソコン上でのソフトウェアによる模擬的な動作の確認）もできます．WIZ-Cのシミュレータは強力なので，場合によってはシミュレーションだけでデバッグが完了する場合もあるでしょう．

WIZ-Cによる開発の大まかな手順は次のようになります．

(1) プロジェクトを新規に作成し，WIZ-C内蔵ツールでI/OピンやPIC内蔵モジュールを設定します．
(2) WIZ-Cのエディタでソース・ファイルを作成または修正します．一通り完成したらソース・コードを見直します．
(3) WIZ-Cからコンパイル操作を行います．
(4) コンパイル・エラーが発生したら(2)に戻ってソース・ファイルを修正し，再びコンパイルします．
(5) コンパイル・エラーがなくなったら，必要に応じてシミュレーションを行います．

このアイコンは，章末に用語解説があります

(6) シミュレーションの結果に不具合がある場合は(2)に戻ってソース・ファイルを修正し，再びコンパイルを行い，シミュレーションを繰り返します．
(7) シミュレーションの結果がうまくいった場合，生成されているHEXAファイルをPICプログラマを使ってPICへ書き込みます．
(8) プログラムされたPICを製作したセットへ挿入し，電源を入れて実際に動作させ，動作を確認します．
(9) 不具合がある場合は(2)に戻ってプログラムを修正し，これを繰り返します．

なお，シミュレーションではI/Oポートの入出力は限定的に模擬できるだけなので，すべての処理を確認できるわけではありませんが，ブレーク・ポイントの設定やシングル・ステップでの実行，割り込み処理や演算結果の確認といったことが可能です．WIZ-CではLEDのほかに，LCD(液晶表示器)，16進キーパッド，アナログ入力などのシミュレーション・デバイスが用意されていて，ハードウェアがなくてもパソコン上で仮想のPICを模擬的に動かすことができます．また，出力ポートの出力波形をロジック・アナライザ風にグラフ状に表示させることもできます．

Pic-Keyを別に用意すれば，製作した実機セットを実際に動かしながら動作の確認をすることができます．

図5-1　WIZ-Cのメイン・ウィンドウ
このメイン・ウィンドウ上でソース・コードの作成やコンパイル，シミュレーション実行などが操作できる．

5-2　WIZ-Cの操作の概要

● メイン・ウィンドウ

図5-1にWIZ-Cのメイン・ウィンドウのサンプルを示します．大きく四つのウィンドウに分かれています．ソース・エディタ・ウィンドウはソース・ファイルを作成したり，修正するためのテキスト・エディタです．環境設定でUNICODEを使用するようにしておけば，コメントに日本語を使うことができます．プロジェクト・ウィンドウは，そのプロジェクトに含まれるソース・ファイルの一覧を示しています．また，情報ウィンドウ(Information)にはコンパイル状況やコンパイル・エラーなどを表示します．デバッガ・ウィンドウではレジスタや変数の内容を表示させたり，シミュレーション・デバイスの表示や操作を行います．

実際の操作はプログラムを作成するときに詳しく説明します．ここでは，簡単に概要を説明します．

● アプリケーション・デザイナを使ったプロジェクトの新規作成

アプリケーション・デザイナは，**図5-2**に示すようなI/OやPIC内蔵モジュールを視覚的に定義するツールです．

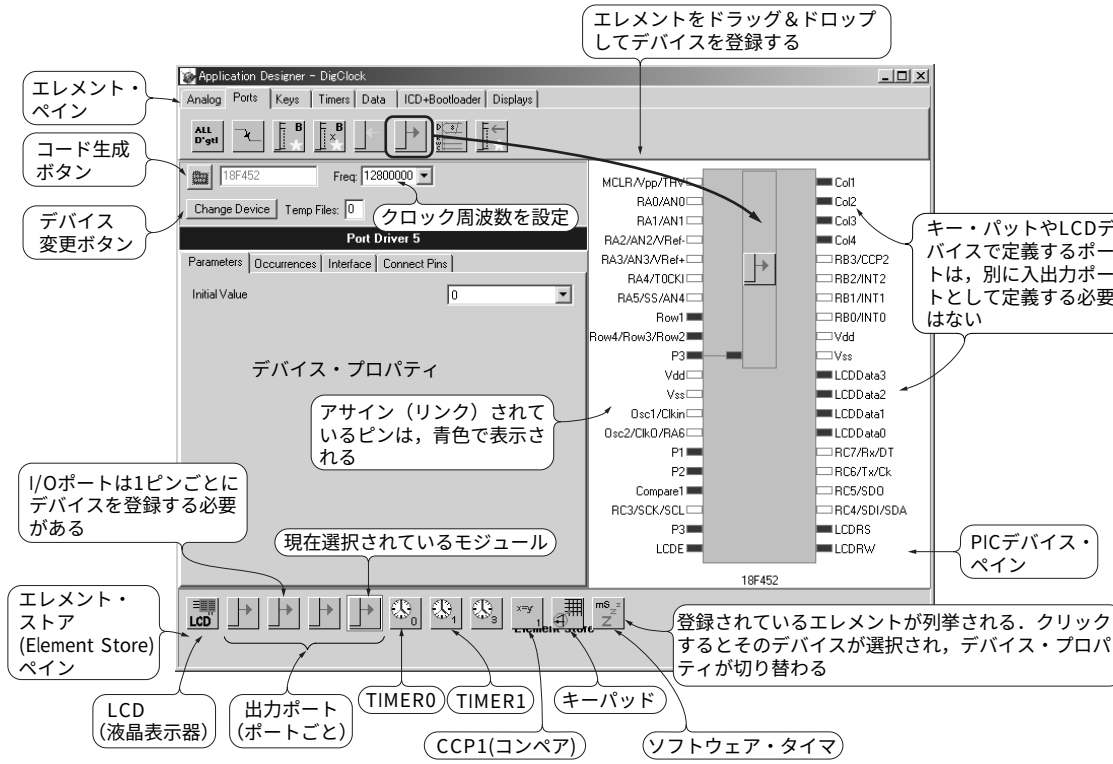
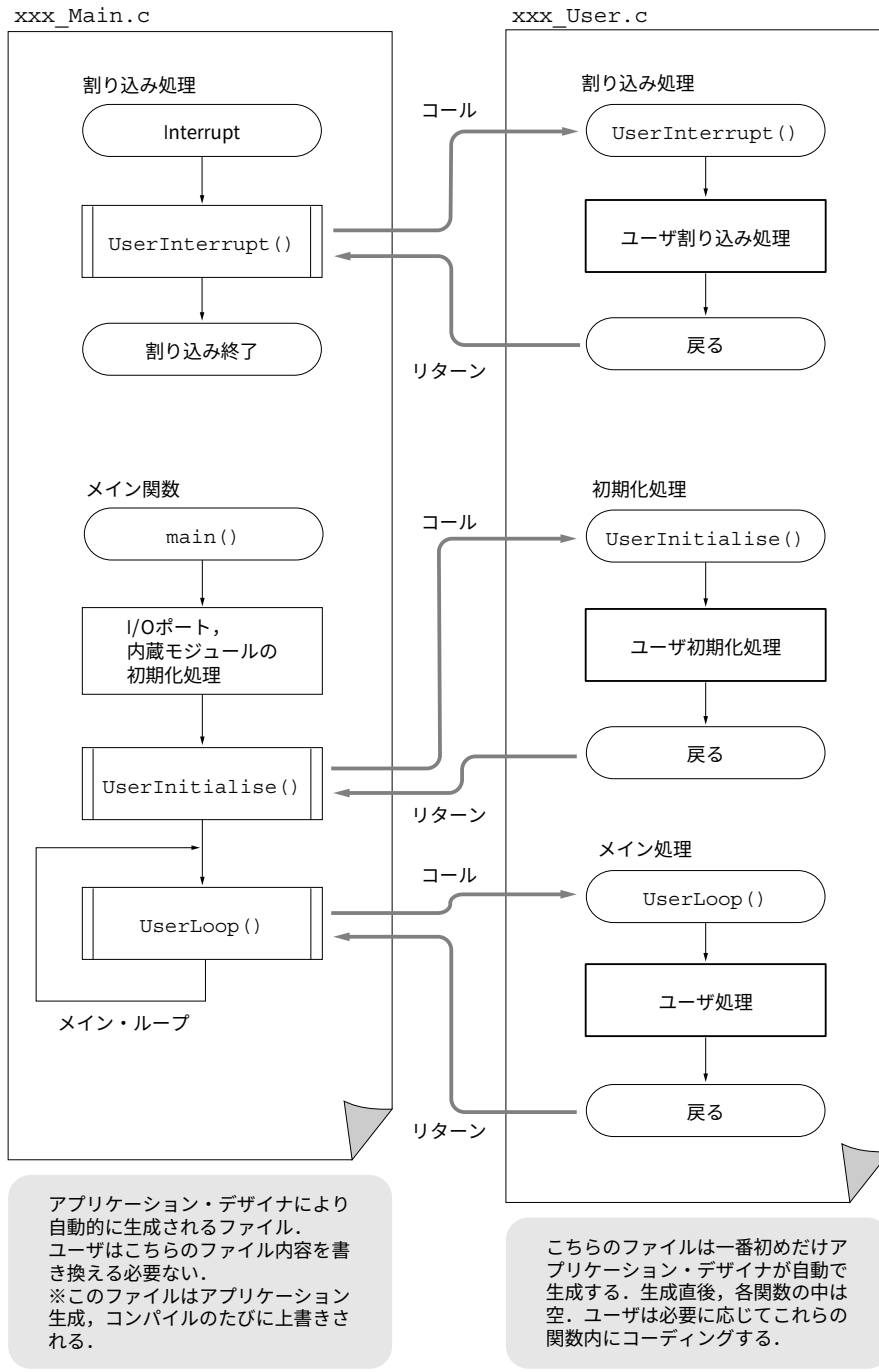


図5-2　WIZ-C アプリケーション・デザイナ(Application Designer)
I/Oポートのピン・アサインやタイマのプリスケーラ値などがこの画面で設定できる．コード生成を実行すると，I/Oの初期化処理がmain()関数に自動的に挿入されたソース・ファイル(xxx_Main.c)が生成される．

リスト5-1　自動生成されるユーザ・メイン・ファイル(xxx_User.C)
WIZ-Cのアプリケーション・デザイナによって自動的に生成されるブランクのテンプレート・ファイル．ユーザはこの中に実際の処理をコーディングしていく．

```
// This file includes all user definable routines. It may be changed at will as
// it will not be regenerated once the application has been generated for the
// first time.
//

//*****************************************************************************
//
// Insert your interrupt handling code if required here.
// Note quick interrupts are used so code must be simple
// See the manual for details of quick interrupts.
//
                                ユーザ割り込み処理
void UserInterrupt( )
{
       // Insert your code here
       この部分に割り込み処理を書き込む


  #asmline goto UserIntReturn      ; PIC Assembler - go back to interrupt routine
}
                                本来の割り込み処理へ戻るためのアセンブリ・コード

//*****************************************************************************

//
// Insert your initialisation code if required here.
// Note that when this routine is called Interrupts will not be enabled - the
// Application Designer will enable them before the main loop
//
                                ユーザ初期化処理
void UserInitialise( )
{
       この部分に初期化処理を書き込む
}

//*****************************************************************************

//
// Insert your main loop code if required here. This routine will be called
// as part of the main loop code
//
                                このUserLoop()関数が実質的なメイン関数になる．
                                この関数はmain関数のループ内からコールされるため，ここでループさせる必要はない
void UserLoop( )
{
       この部分に処理の本体を書き込む
       (ここではループさせる必要はない)
}

//
// User occurrence code
//
アプリケーション・デザイナで定義したイベント・ハンドラ
(Occurrence関数)をここに書く
```



フローチャート5-1
この図は，自動生成されるxxx_Main.cとxxx_User.cファイルの構造と，相互の関係を示している．実際にユーザが手を加えるのはxxx_User.cファイルのほうだけでよい．

I/Oポートやタイマ，CCP，A-Dコンバータなどの PIC 内蔵モジュールは，アプリケーション・デザイナ上ではエレメント（モジュール）として扱われ，それぞれ固有のアイコンで表されます．このアイコンを操作して，それぞれプロパティを設定することにより，使用するI/OやPIC内蔵モジュールを定義し，ピン配置を決定します．

モジュールを登録するには，画面上側のエレメント・ペインから所望のアイコンをPICデバイス・ペインへドラッグ&ドロップします．また，デバイス・プロパティ・ペインでエレメントごとのプロパティなどを設定します．

プロジェクト作成時にアプリケーション・デザイナを使うと，たとえば，LCD（液晶表示器）を使う場合，LCDデータ・バス4本（内3本は自動的にアサイン），制御線3本を画面上のピンをクリックして連結するだけでピン・アサインを定義でき，自動的に初期化コード（LCDの初期化処理も含む）が生成されます．

● 自動生成ファイル

アプリケーション・デザイナでコードを生成させると，xxx_Main.cとxxx_User.cという二つのファイルが自動的に生成されます（xxxはプロジェクト名）．xxx_Main.cにはアプリケーション・デザイナで定義したI/Oやデバイスの初期化処理が自動で生成され，main関数が生成されます．ユーザはこちらのファイルに手を加える必要はまったくありません．

ユーザがプログラムを実際に組み込むのは，xxx_User.cのUserLoopという関数です．この関数はmainのループの中から呼ばれています．このファイルには，その他にUserInterrupt（割り込みの際に呼ばれる割り込み処理），UserInitialise（I/Oや変数などの初期化処理）が生成されるので，それぞれの関数に必要に応じて処理を加えます（**リスト5-1**，**フローチャート5-1**）．

なお，xxx_Main.cファイルのほうは，エディタで書き換えたとしても，アプリケーション・デザイナを使う設定になっている場合，コンパイルのたびに新規にコードが生成され，古いファイルは毎回上書きされてしまい，何にもならないのでご注意ください．

● コンパイル

コンパイルは，アプリケーション・デザイナでコードを生成するたびに実行されています．また，メニューなどからコンパイルを実行することもできます．コンパイル・エラーがある場合は，情報ウィンドウ（Information）にエラーのある行番号とエラーの内容が表示されます．このウィンドウのエラー行をダブル・クリックすると，ソース・コードの該当行へカーソルが移動します．

● シミュレーション

デバッガ・ウィンドウにシミュレーション・デバイスを追加することで，LED，LCD，7セグメントLED，キー・パッド入力，ボタン入力などを模擬することができます．たとえば，LCDデバイスは，本来のプログラムにデバッグ用の修正を加えることなくシミュレーション・デバイスのLCDで初期化や文字を表示させることができます．また，Keypadデバイスはマウスでクリックすることで，キー・パッドからの入力を模擬できます（**図5-1** 右上参照）．