MDIを用いて開発したマルチドキュメント型実習環境

大 西 孝 臣*

Multi-document experimental environment developed through MDI

Takaomi OHNISHI

Abstract

This article shows one of key technologies developing a multi-document experimental environment for learning digital signal processing. This environment is an MDI (Multi-Document Interface) Windows application with two document types, each of which is embedded in an MDI child frame window.

Although both the Application Wizard and the Class Wizard are very powerful tools of Visual C++ development environment, neither of two provides how to properly create "MDI Windows applications with multi document types." So, the author provides the technique of implementation.

1．はじめに

筆者は、ディジタル信号処理の実習を行うための、2つのドキュメントタイプをマルチドキュメントのWindowsアプリケーションとして統合した実習環境をVisual C++を用いて開発した。

Visual C++に付属するApplication WizardやClass Wizardを使用することにより、シングルドキュメントのMDIアプリケーションを開発することは可能である。しかし、MDIが持つ本来の機能を使った、マルチドキュメントのMDIアプリケーションを開発するための常套手段や正攻法は用意されていない。本稿では、その開発の技術面を紹介する。

2．実習環境の概要

2.1 構 成

本実習環境においては、学習者は、図1に示すような、プログラミング言語を扱った実習環境を提供する「コンパイル環境」と、数値データ処理としてのFFT及び逆FFTの性質を学ばせることに重点を置いた「数値データ・グラフ表示環境」の、計2つのドキュメントタイプと実質的に向きあうこととなる。

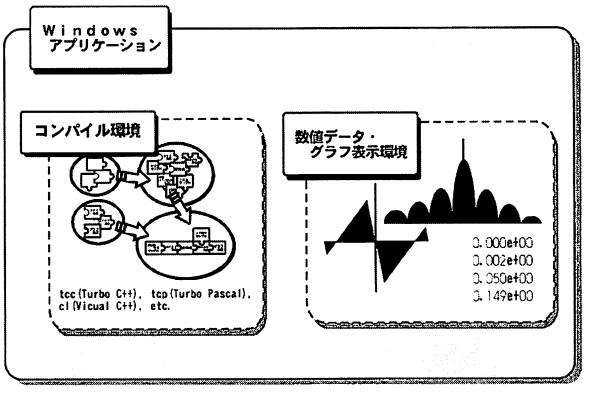

図1 実習環境の構成

2つのドキュメントタイプは、図2に示すように、1つのアプリケーションについて複数のドキュメントに対応した子ウィンドゥを同時に表示することのできるMDI（Multi-Document Interface）という形態のWindowsアプリケーションの下に統合されている。

2.2 シングルドキュメントのアプリケーションの組み合わせ

Visual C++を用いてMDIアプリケーションを開発する場合、常套手段として最初に行うことはApplication Wizardを用いてスケルトンプログラムを生成させることである。スケルトンプログラムとは、Windowsアプリケーションとしての

* 助 手 情報工学科

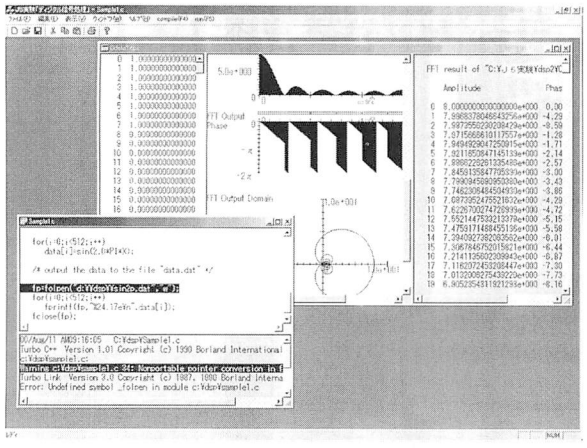

図2　実習環境の概観

標準的なGUIを備えていることの他には何1つの機能をもたない“空の”アプリケーションである。MDIアプリケーションのスケルトンプログラムには1つのドキュメントテンプレートが関連付けられている。言い換えると、1つのドキュメントタイプを扱うことのできるアプリケーションである。

本稿では、開発時において、まずは図3に示すように、コンパイル環境を提供する「compile05(旧)」と、数値データ・グラフ表示環境を提供する「fft01」という名称の別個のMDIアプリケーションを実現した。

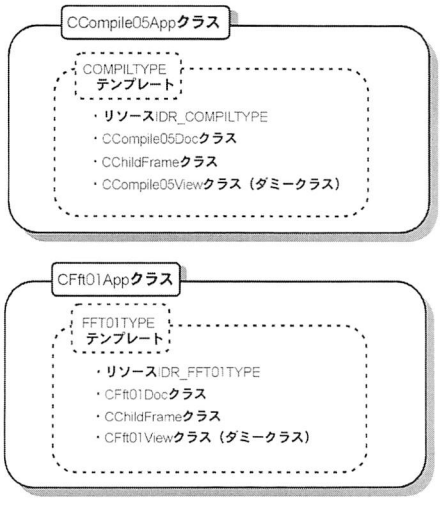

図3　別個のアプリケーションとして実現した2つのドキュメントタイプ

2つのアプリケーションそれぞれには、アプリケーションクラス、ドキュメントクラス、チャイルドフレーム用のフレームウィンドゥクラス、ビュークラスというMFCライブラリからの4つの派生クラスとリソースからなる構成要素がドキュメントテンプレートを通じて互いに関連付けられている。なお、2つのアプリケーションにおける子ウィンドゥはそれぞれ分割ウィンドゥであるため、ドキュメントテンプレートによって関連付けられたビュークラスはダミーとして扱われる。

続いて、2つのアプリケーションの一方であるfft01をもう一方のcompile05（旧）に組み込む形で両者を組み合わせて、1つのWindowsアプリケーション「compile05（新）」と実現する。

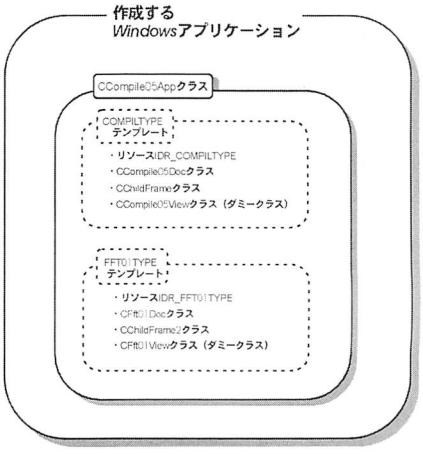

図4　1つのマルチドキュメントアプリケーションへの統合

図4に示された、ドキュメントテンプレートの構築、派生クラスやリソースの関連付けは、ソースレベルで次頁冒頭のように行われる。

アプリケーションクラスCCompile05AppのメソッドInitInstanceにおいて、CMultiDocTemplateクラスのインスタンスに、コンパイル環境ドキュメントタイプについては、チャイルドフレーム用のリソースIDR__COMPILETYPE、ドキュメントクラスCCompile05Doc、チャイルドフレーム用のフレームウィンドゥクラスCChildFrame、ダミーのビュークラスCCompile05Viewが関連付けられ、数値データ・グラフ表示環境ドキュメントタイプについては、チャイルドフレーム用のリソースIDR__FFT01TYPE、ドキュメントクラスCFft01Doc、チャイルドフレーム用のフレームウィンドゥクラスCChildFrame2（図3におけるクラス名を改称）、ダミーのビュークラスCFft01Viewが関連付けられる。

```
// compile05.cpp : アプリケーション用クラスの機能定義を行います。
//

   ...

BOOL CCompile05App::InitInstance()
{

   ...

        CMultiDocTemplate* pDocTemplate;

        pDocTemplate = new CMultiDocTemplate(
                IDR_COMPILTYPE,
                RUNTIME_CLASS(CCompile05Doc),
                RUNTIME_CLASS(CChildFrame), // カスタム MDI 子フレーム
                RUNTIME_CLASS(CCompile05View));
        AddDocTemplate(pDocTemplate);

        pDocTemplate = new CMultiDocTemplate(
                IDR_FFT01TYPE,
                RUNTIME_CLASS(CFft01Doc),
                RUNTIME_CLASS(CChildFrame2), // カスタム MDI 子フレーム
                RUNTIME_CLASS(CFft01View));
        AddDocTemplate(pDocTemplate);

        // メイン MDI フレーム ウィンドウを作成
        CMainFrame* pMainFrame = new CMainFrame;
        if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
                return FALSE;
        m_pMainWnd = pMainFrame;

        // DDE、file open など標準のシェル コマンドのコマンドラインを解析します。
        CCommandLineInfo cmdInfo;
        ParseCommandLine(cmdInfo);

        // 起動時に新しい MDI 子ウィンドウを表示しない!!!
        cmdInfo.m_nShellCommand = CCommandLineInfo::FileNothing;

        // コマンドラインでディスパッチ コマンドを指定します。
        if (!ProcessShellCommand(cmdInfo))
                return FALSE;

   ...

}

   ...
```

2.3　プロジェクトに対するクラス追加の通知

２つのアプリケーションの一方をもう一方に組み込むためには、まず、組み込む側のアプリケーションのプロジェクト用のディレクトリにあるドキュメントクラス、チャイルドフレーム用のフレームウィンドウクラス、ビュークラスそれぞれのヘッダファイル（拡張子が.hのファイル）およびインプリメンテーションファイル（あるいは（狭義の）ソースファイル、拡張子が.cppのファイル）を、組み込まれる側のアプリケーションのプロジェクト用のディレクトリへとコピーする。

続いて、コピーされたファイルにより定義されたクラスが追加されたことを、組み込まれる側のアプリケーションのプロジェクトにおいて通知するために、Visual C＋＋のメニューより『プロジェクト』→『プロジェクトへ追加』→『ファイル』を選択して、図５に示すダイアログにおいて、コピーされたソースファイルをプロジェクトに追加させる。

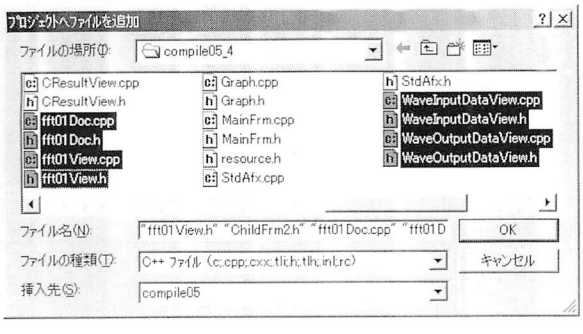

図５　プロジェクトへのファイルの追加

2.4　Class Wizardに対するクラス追加の通知

前節おいて述べた手続きにより、プロジェクトはクラスの追加を認識することができるが、Visual C＋＋のClass Wizardはクラスの追加をまだ認識していないため、追加したクラスとリソースとの連携ができておらず、ユーザなどが起こすイベントに対するドライバ関数が作成できないなどの障害が起きる状況のままである。Visual C＋＋の開発環境においては、ソースファイルのコピーによって追加されたクラスに対するClass Wizardへの通知の手段は用意されていないが、プロジェクトのディレクトリにあるCLWファイルを手作業で操作することを通じて、Class Wizardにクラスの追加を通知することができる。操作した後のCLWファイルcompile05.clwの一部を次に示す。

```
; CLW ファイルは MFC ClassWizard の情報を含んでいます。

[General Info]

   ...

ClassCount=17
Class1=CCompile05App
Class2=CCompile05Doc
Class3=CCompile05View
Class4=CMainFrame

   ...

Class13=CChildFrame2
Class14=CWaveInputDataView
Class15=CFft01Doc
Class16=CFft01View
Class17=CWaveOutputDataView

   ...
   ...

[CLS:CCompile05App]
Type=0
HeaderFile=compile05.h
ImplementationFile=compile05.cpp
Filter=N

   ...

[CLS:CChildFrame2]
Type=0
HeaderFile=ChildFrm2.h
ImplementationFile=ChildFrm2.cpp
Filter=M
BaseClass=CMDIChildWnd
VirtualFilter=mfWC
LastObject=ID_AMPLITUDE_PHASE

   ...

[CLS:CWaveInputDataView]
Type=0
HeaderFile=WaveInputDataView.h
ImplementationFile=WaveInputDataView.cpp
BaseClass=CEditView
Filter=C
LastObject=ID_FILE_PRINT
VirtualFilter=VWC

   ...
```

CLWファイルはApplication Wizardがスケルトンプログラムを作成する時から存在している、Class Wizard において取り扱う情報を管理するためのファイルである。Windowsアプリケーション開発時におけるClass Wizardへの操作は、CLWファイルへの項目として反映されるが、ソースファイルのコピーなどはClass Wizardの外における操作であるため、手作業によるクラスの追加が必要になる。

［General Info］の項目においては、Class Wizardが認識しているクラスやリソースが混在した状態で列挙されている。compile05.clwの例では、ClassCountが12であったものを17に改め、Class13からClass17として、追加を通知すべきドキュメントクラス、チャイルドフレーム用のフレームウィンドゥクラス、子ウィンドゥにおけるペイン（分割ウィンドゥにおける、分割された各表示領域）毎のビュークラスを書き加えた。

［CLS: … ］の項目においては、［General Info］の項目に並べられた順番に、各クラスに関する、ヘッダファイルやインプリメンテーションファイルの名称や基本クラスの名称などの詳細な情報が記載されている。compile05.clwの例では、［General Info］の項目において書き加えた5つのクラスの内の最初の2つである、チャイルドフレーム用のフレームウィンドウクラスCChildFrame2および子ウィンドウにおけるペインに対するビューファイルの1つであるCWaveInputDataViewの詳細を示した。追加するクラスの詳細については、組み込む側のfft01アプリケーションにおけるCLWファイルでの情報を用いるか、同じ基本クラスを持つ派生クラスの情報を操作することにより作成する。

結果、図6に示すように、Class Wizardにおける表示に、クラスの追加が反映される

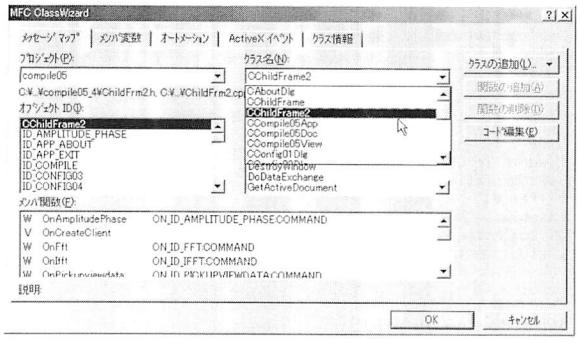

図6　Class Wizardにおけるクラス表示
（CChildFrame2の追加が反映されている。）

3．おわりに

本稿では、2つのシングルドキュメントのMDIアプリケーションの一方を他方に組み合わせる手法を用いて、マルチドキュメントのMDIアプリケーションを実現することにより、ドキュメントを統合させるアプローチについて紹介した。

本手法を用いることにより、ディジタル信号処理の実習における数値データ処理という主目的をより明確にする統合環境の実現ができるようになった。

参考文献

1）大西　孝臣、杉岡　一郎共著、ディジタル信号処理の学習を目的とした実習環境－コマンドラインコンパイラ組み込み型教材による実現－、コンピュータ＆エデュケーション、柏書房、Vol.11，pp.94-99、2001

2）大西　孝臣、杉岡　一郎共著、Windowsアプリケーションにコマンドラインコンパイラを編入させた実習教材の開発、教育工学関連学協会連合第6回全国大会講演論文集、pp.821-822、2000

3）David J. Kruglinski、George Shepherd、Scot Wingo共著、（有）デジタルアドバンテージ訳、プログラミング　Microsoft Visual C++ 6.0、日系BPソフトプレス、1999

4）MSDN Subscriptions Library 日本語版 October 2001、Microsoft Corporation、2001

（平成13年11月30日受理）