



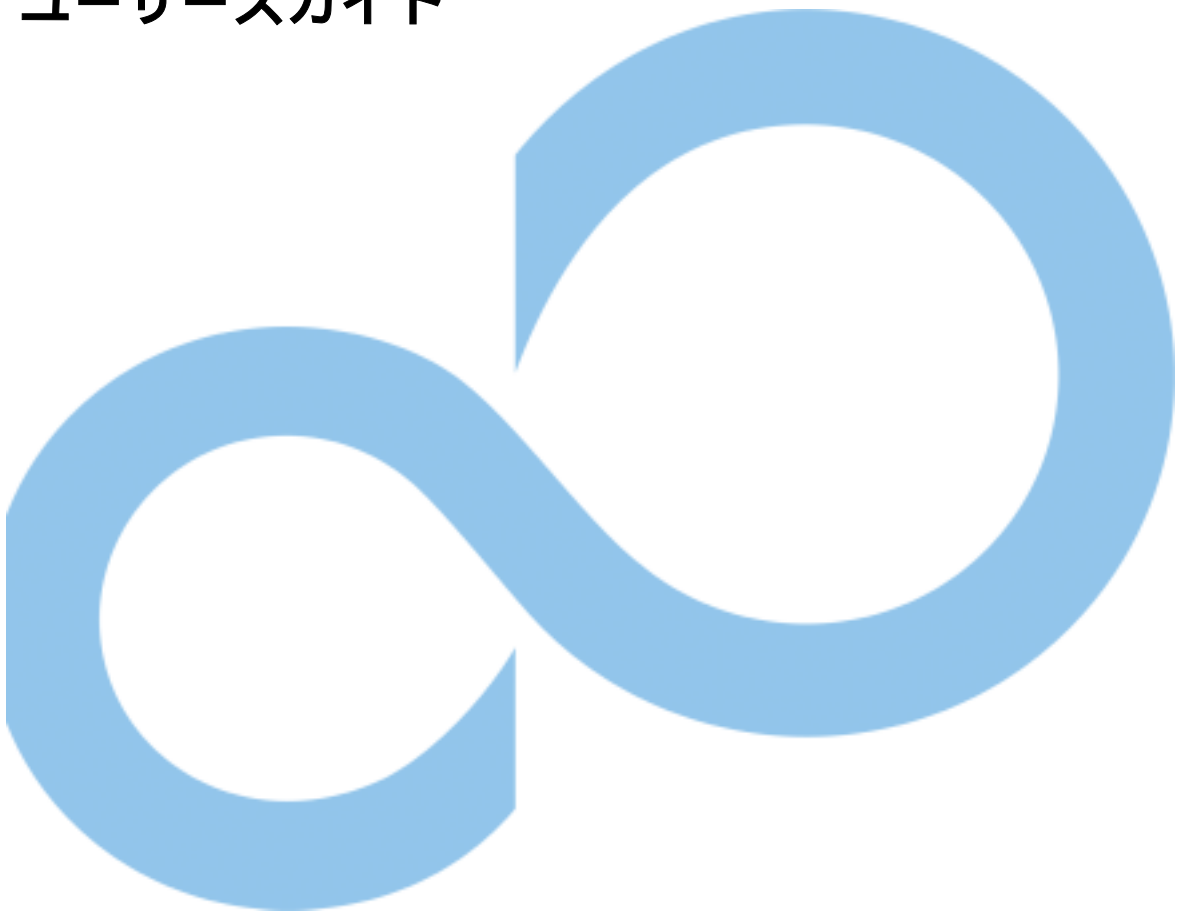
Microsoft® Windows® 95
Microsoft® Windows® 98
Microsoft® Windows® Me

Microsoft® Windows NT®
Microsoft® Windows® 2000
Microsoft® Windows® XP

B1JW-5311-01

PowerCOBOL V7.0

ユーザーズガイド



FMVシリーズ, FMV-DESKPOWER, FMV-BIBLO

Net  OBOL

 FUJITSU

はじめに

このたびは、「NetCOBOL シリーズ V7.0」をお買い上げいただき、まことにありがとうございます。

「PowerCOBOL V7.0 (以降、PowerCOBOLと略します)」は、COBOLプログラマを対象とした、GUI(Graphical User Interface)アプリケーション開発ツールです。PowerCOBOLは、「NetCOBOL シリーズ」を構成するコンポーネントの1つです。

2002年7月

Microsoft、Windows、Windows NT、Visual Basic、ActiveXおよびMSDNは、米国Microsoft Corporationの米国およびその他の国における登録商標です。

Oracleは、ORACLE Corporationの登録商標です。

その他の会社名または製品名は、それぞれ各社の商標または登録商標です。

Microsoft Corporationのガイドラインに従って画面写真を使用しています。

All Rights Reserved, Copyright 富士通株式会社 1993-2002

このマニュアルの使いかた

本書の目的

本書は、PowerCOBOLを利用して応用プログラムを作成するにあたり、PowerCOBOLの基本的な使いかたを習得することを目的としています。

本書を利用することにより、PowerCOBOLの機能を使った、いろいろなアプリケーションの開発方法を習得することができます。

第1章からひととおりお読みいただき、使用方法を身に付けるためのガイドとしてお使いください。

PowerCOBOLをお使いになるうえで、本書がささやかな足がかりとなることを願っています。

前提知識

本書を読むにあたって、以下の知識が必要です。

Windows に関する基本的な知識

COBOLに関する知識

なお、本書とあわせて、『ソフトウェア説明書』(PCOB.TXT)をお読みください。

本書に記載できなかった重要な情報が記載されています。

また、必要に応じて以下のマニュアルを参照してください。

マニュアル名称	記載内容
PowerCOBOL V7.0 リファレンス (HTMLヘルプ)	PowerCOBOLに関する以下の説明 ダイアログボックスの機能 コントロールの仕様
COBOL文法書	COBOL文法の説明
NetCOBOL 使用手引書 for Windows V7.0	COBOLに関する以下の説明 翻訳オプション 翻訳時メッセージ 実行時環境情報 実行時メッセージ ファイルの処理 印刷処理 Unicode
Microsoft Developer Network ライブラリ	Windows のデータアクセス サービスに関する以下の説明 ADO(Microsoft ActiveX Data Objects)

『Microsoft Developer Network ライブラリ』は、Microsoft CorporationのWebサイトを参照してください。

本書の構成

第1部 入門編

PowerCOBOLをはじめてお使いになるお客様を対象に、概要、アプリケーション作成の流れ、および開発に利用するウィンドウについて説明しています

第2部 開発環境編

実際にアプリケーションを作成しながら、PowerCOBOLの開発環境がもつ機能について説明しています。

第3部 プログラミング編

コントロールへのアクセス方法や、いろいろなアプリケーションを作成するためのプログラミング方法について説明しています。

V1.0からV3.0までのPowerCOBOLをお使いのお客様へ

従来、「PowerCOBOL V3.0」または、それ以前のバージョンのPowerCOBOLをお使いの場合は、まず「[V3.0以前のPowerCOBOLをご利用の方へ](#) (p307) 」をお読みください。

本書の表記、説明内容について

各製品の略称表記

本書では、製品名を以下のように省略して記述しています。

Microsoft	Windows	95 operating system
	Windows	95
Microsoft	Windows	98 operating system
	Windows	98
Microsoft	Windows	Millennium Edition
	Windows	Me
Microsoft	Windows NT	Workstation operating system Version 4.0
	Windows NT	4.0
Microsoft	Windows NT	Server Network operating system Version 4.0
	Windows NT	4.0
Microsoft	Windows NT	Server Network operating system Version 4.0, Terminal Server Edition
	Windows NT	4.0
Microsoft	Windows NT	Server Network operating system, Enterprise Edition Version 4.0
	Windows NT	4.0
Microsoft	Windows	2000 Professional operating system
	Windows	2000

Microsoft Windows 2000 Server operating system
Windows 2000
Microsoft Windows 2000 Advanced Server operating system
Windows 2000
Microsoft Windows XP Professional operating system
Windows XP
Microsoft Windows XP Home Edition operating system
Windows XP
Windows 95、Windows 98、Windows Me、Windows NT 4.0、Windows 2000
および Windows XP
Windows

参照マニュアルの略称表記

本書では、参照先マニュアル名を以下のように省略して記述しています。

『PowerCOBOL V7.0 リファレンス』 『リファレンス』

『NetCOBOL 使用手引書 for Windows V7.0』 『NetCOBOL 使用手引書』

『Microsoft Developer Network ライブラリ』 『MSDN 』

サンプルプログラムについて

本書の説明で使用するサンプルプログラム（例題プログラム）は、PowerCOBOLをインストールしたフォルダ配下のSamples¥PowerCOBフォルダに格納されています。たとえば、"C:¥Program Files¥NetCOBOL"にPowerCOBOLをインストールした場合、

"C:¥Program Files¥NetCOBOL¥Samples¥PowerCOB"フォルダ内の該当するサンプルプログラムを参照してください。

また、PowerCOBOLで提供しているサンプルプログラムの一覧および概要については、付録の「[サンプルプログラムについて](#)（ p321）」に記載されていますので、あわせてご利用ください。

注意事項や参考情報

このマニュアルでは、以下の記述形式を使用しています。



注意

とくに注意していただきたいことを記述しています。必ずお読みください。



知っていると役に立つ情報を記述しています。

図（イメージ）

本書に記載されている図（イメージ）は、Windows の種類、動作する機種、解像度および本製品のバージョンアップやレベルアップなどにより、実際に表示される内容と多少異なる場合があります。

キーの表記

本書のキーに関する説明は、「106/109 日本語キーボード」を基準として記述されています。また、[Ctrl+Home]といった表記は、[Ctrl]キーと[Home]キーを同時に押すことを示しています。

なお、アプリケーション開発時のキーボードによる操作方法については、『リファレンス』を参照してください。

メニューの選択方法に関する表記

本書のメニュー選択方法に関する説明で、「ポップアップメニュー」と表記されている場合、マウスの右ボタンを押して（クリックして）表示されるメニューを表しています。それ以外の場合は、ウィンドウ上部にあるメニューバーからの選択を表しています。たとえば、[編集 - オブジェクト - ユーティリティ]メニューの[開く]コマンドといった表記は、[編集]メニューの[オブジェクト]サブメニュー（右端に右向きの黒三角を持つ項目です）さらにその[ユーティリティ]サブメニューの[開く]というメニュー項目を選択することを示しています。

ダイアログボックスに関する説明

PowerCOBOLを使ってアプリケーションを開発する場合、さまざまなダイアログボックスが表示されます。ダイアログボックス内の各項目の説明については、『リファレンス』を参照してください。

コントロールの詳細仕様

PowerCOBOLが提供しているコントロール（ウィンドウを作成するための部品）の詳細仕様（プロパティ、メソッド、イベントなど）については、『リファレンス』を参照してください。

目次

第1部 入門編	1
第1章 PowerCOBOLの概要	3
1.1 PowerCOBOLとは	4
1.2 イベント駆動型プログラムとは	6
第2章 アプリケーション作成の流れ	11
2.1 アプリケーションの作成手順	12
2.2 プロジェクトの作成	15
2.3 フォームの編集	18
2.4 イベント手続きの編集	24
2.5 実行可能プログラムの作成（ビルド）と実行	28
2.6 プロジェクトの構成	31
第3章 PowerCOBOLの基本操作	33
3.1 ウィンドウの名称	34
3.1.1 プロジェクトウィンドウ	34
3.1.2 フォーム編集ウィンドウ	37
3.1.3 手続き編集ウィンドウ	39
3.1.4 プロパティ設定ダイアログボックス	40
3.2 その他の基本操作	42
3.2.1 メニューバー	42
3.2.2 ポップアップメニュー	42
3.2.3 キーボード操作によるメニューコマンドの実行	43
3.2.4 ツールバー	44
3.2.5 キーボードによるプロパティ設定ダイアログボックスの操作	45
第2部 開発環境編	47
第4章 アプリケーションを作成しよう	49
4.1 プロジェクトを作成する	51
4.1.1 新規にプロジェクトを作成する	51
4.1.2 プロジェクトのプロパティを設定する	52
4.1.3 プロジェクトの構成要素を編集する	52
4.2 フォームを編集する	55
4.2.1 フォーム編集ウィンドウを表示する	55
4.2.2 フォームのプロパティを設定する	55
4.2.3 フォームにコントロールを配置する	56
4.2.4 コントロールのプロパティを設定する	57
4.2.5 コントロールの位置とサイズを調整する	62
4.2.6 コントロールの色を変更する	62
4.2.7 コントロールの文字のフォントを変更する	64
4.2.8 フォーム編集ウィンドウを閉じる	65

4.3	メニューを作成する	66
4.3.1	メニュー編集ウィンドウを開く	66
4.3.2	メニュー項目を追加する	67
4.3.3	メニュー編集ウィンドウでの編集操作	70
4.3.4	メニュー編集ウィンドウを閉じる	71
4.4	手続きを編集する	72
4.4.1	手続き編集ウィンドウを表示する	73
4.4.2	手続き編集ウィンドウでの編集操作	76
4.4.3	手続きを記述する	76
4.5	ビルド・実行する	88
4.5.1	ビルドとは	88
4.5.2	プロジェクトを保存する	89
4.5.3	ビルドする	89
4.5.4	エラーがあったら	90
4.5.5	実行する	91
4.6	アプリケーション開発時のオプション	92
4.7	プロジェクト構成要素の命名規則	93
4.7.1	モジュール名	93
4.7.2	外部ファイル名	93
4.7.3	フォーム名	93
4.7.4	コントロール名	94
4.7.5	フォームのその他の構成要素の名前	94
第5章	PowerCOBOLを使いこなそう	95
5.1	プロジェクトの便利な機能	96
5.1.1	外部ファイルを使う	96
5.1.2	テンプレートを追加する	98
5.1.3	Unicodeを利用する	99
5.1.4	ユーティリティを利用する	99
5.2	フォーム編集時の便利な機能	102
5.2.1	グリッドを利用する	102
5.2.2	タブ順序とタブグループを設定する	102
5.2.3	コントロールの描画順序を変更する	105
5.2.4	コントロールをまとめて編集する	105
5.2.5	コントロールを配列化して利用する	107
5.2.6	フォームをプレビューする	109
5.2.7	フォームを印刷する	110
5.3	メニューの拡張機能	111
5.3.1	メニュー項目にショートカットキーを割り当てる	111
5.3.2	ポップアップメニュー形式で編集する	112
5.4	手続き編集時の便利な機能	113
5.4.1	コントロール名を挿入する	113
5.4.2	メソッドやプロパティを挿入する	113
5.4.3	文字列を検索・置換する	114
5.4.4	指定行へジャンプする	115

5.4.5	注記行を設定する	115
5.4.6	インデントを利用する	115
5.4.7	文字の色を変更する	116
5.4.8	文字のフォントを変更する	117
5.4.9	手続きを印刷する	117
5.4.10	外部COBOLファイルを編集する	117
5.4.11	手続き編集用のエディタを変更する	118
5.5	実行可能プログラム作成時および実行時の便利な機能	119
5.5.1	デフォルトモジュールを設定する	119
5.5.2	DLLを作成するには	119
5.5.3	ビルドモードを使い分ける	119
5.5.4	ビルド時に作成されるファイル	120
5.5.5	NetCOBOLのオブジェクト指向プログラミング機能を利用する	122
5.5.6	アプリケーションの多重起動を制御する	122
5.5.7	ビルド用のオプションを設定する	123
5.5.8	実行可能プログラムのバージョンを設定する	128
5.5.9	パッチモードでビルドする	128
5.5.10	アプリケーションの動作環境を設定する	132
5.5.11	インストーラを作成する	133
5.5.12	診断機能を利用する	134
5.5.13	CHECK機能を利用する	140
第6章	アプリケーションをデバッグしよう	141
6.1	デバッグの概要	142
6.1.1	デバッグできる範囲	142
6.1.2	デバッグの進めかた	142
6.2	中断点を設定する	148
6.2.1	中断点を設定するには	148
6.2.2	中断点の一覧を表示する	149
6.3	実行する	152
6.3.1	実行するには	152
6.3.2	実行を中断させるには	153
6.4	データを参照する	154
6.4.1	データチップ	154
6.4.2	クイックウォッチ	155
6.4.3	ウォッチ	156
6.5	呼び出し経路を確認する	158
6.6	いろいろな形態のアプリケーションをデバッグする	159
6.6.1	同一プロジェクト内の複数のモジュールをデバッグする	159
6.6.2	複数プロジェクトにまたがる複数のモジュールをデバッグする	159
6.6.3	COBOLプログラムと組み合わせてデバッグする	160
第3部	プログラミング編	163
第7章	プログラミングの基礎知識	165
7.1	コントロールとフォームの手続き	166
7.1.1	コントロールの手続き	166

7.1.2	フォームの手続き	167
7.2	プロパティへのアクセス方法	171
7.2.1	プロパティの記述形式	171
7.2.2	プロパティの記述例	172
7.2.3	プロパティの参照方法	173
7.2.4	プロパティの設定（変更）方法	177
7.3	メソッドの呼び出し方法	180
7.3.1	メソッドの呼び出し形式	180
7.3.2	メソッドの復帰値	181
7.3.3	メソッドの呼び出し例	181
7.4	登録集ファイルの利用方法	183
7.5	PowerCOBOL固有のデータの取り扱い方法	185
7.5.1	VT_BSTR型の変換方法	185
7.5.2	VT_VARIANT型の変換方法	186
7.5.3	VT_CY型への変換方法	186
7.6	Unicodeの取り扱い方法	188
7.6.1	シフトJISコードからUnicodeへの変換	188
7.6.2	UnicodeからシフトJISコードへの変換	189
7.7	プログラミング上の留意事項	190
第8章	プログラミングテクニック	193
8.1	配列化したコントロールを使ったアプリケーションを作成する	194
8.2	オブジェクトを使ったアプリケーションを作成する	197
8.2.1	オブジェクトへアクセスするには	197
8.2.2	オブジェクトをイベントの引数として受け取るには	202
8.3	複数ウィンドウをもつアプリケーションを作成する	204
8.3.1	OpenFormメソッドを使用する	204
8.3.2	CallFormメソッドを使用する	211
8.3.3	フォーム間での情報の受け渡し方法	213
8.4	コレクションオブジェクトを使ったアプリケーションを作成する	214
8.4.1	POWERGETCONTROLユーティリティを使って操作する	214
8.4.2	NetCOBOLのCOM連携機能を使って操作する	217
8.5	ポップアップメニューを使ったアプリケーションを作成する	219
8.5.1	フォームに新しくメニューを作成する	219
8.5.2	メニュー項目を追加する	219
8.5.3	マウスの右ボタンがクリックされたときの手続きを記述する	221
8.5.4	ポップアップメニューの項目が選択されたときの手続きを記述する	222
8.6	ステータスバーを使ったアプリケーションを作成する	223
8.6.1	フォームにステータスバーを追加する	223
8.6.2	メニュー項目選択時の手続きを記述する	224
8.6.3	メニュー項目の選択が確定（クリック）したときの手続きを追加する	224
8.6.4	メニュー項目の選択がキャンセルされたときの手続きを記述する	225
8.7	ツールバーを使ったアプリケーションを作成する	226
8.7.1	ツールバーのボタン上に表示するイメージを作成する	226
8.7.2	フォームにツールバーを配置する	227

8.7.3	ツールバーにボタンを追加する	228
8.7.4	ツールバーのボタンがクリックされたときの手続きを記述する	229
8.8	タブコントロールを使ってアプリケーションを作成する	231
8.8.1	ダイアログボックスを開くための手続きを記述する	231
8.8.2	新しいフォームにタブコントロールを配置する	232
8.8.3	その他のコントロールを配置する	233
8.8.4	新しいフォームの手続きを記述する	234
第9章	他のアプリケーションとの連携	237
9.1	DLLを使ったアプリケーションを作成する	238
9.1.1	PowerCOBOLで作成したDLLを使用する	238
9.1.2	COBOLで作成したDLLを使用する	238
9.1.3	PowerCOBOLで作成したDLLをCOBOLから使用する	239
9.1.4	COBOL以外の言語で作成したプログラムから使用する	243
9.1.5	PowerCOBOLとCOBOLで作成したDLLが混在する場合	244
9.1.6	PowerCOBOLで作成したDLLの寿命	244
9.2	ActiveXコントロールを作成する	245
9.2.1	プロジェクトを作成する	246
9.2.2	フォームを編集する	248
9.2.3	ActiveXコントロール用インタフェースを定義する	251
9.2.4	手続きを編集する	255
9.2.5	ツールボックス用ビットマップを定義する	257
9.2.6	システムに登録する	258
9.3	ActiveXコントロールを使ったアプリケーションを作成する	260
9.3.1	フォームを編集する	261
9.3.2	手続きを編集する	263
9.3.3	コマンドボタンコントロールを追加する	264
9.3.4	ボタンがクリックされたときの手続きを記述する	265
9.3.5	Activateメソッドの役割	266
9.3.6	OpenedイベントとClosedイベント	267
9.3.7	Visibleプロパティの取り扱い方法	268
9.4	オートメーションサーバを使ったアプリケーションを作成する	270
9.4.1	フォームにコマンドボタンコントロールを配置する	271
9.4.2	手続きを編集する	271
9.5	他のアプリケーションやバッチファイルを起動する	274
9.6	PowerCOBOLで作成したActiveXコントロールをWeb上で利用する	275
9.6.1	ActiveXコントロールを用意する	275
9.6.2	HTML文書を記述する	275
9.7	DBアクセスコントロールを利用してデータベースと連携する	277
9.7.1	クライアント環境(ODBC)を設定する	277
9.7.2	フォームにコントロールを配置する	278
9.7.3	コントロールの手続きを記述する	282
9.8	ADOデータソースコントロールを利用してデータベースと連携する	287
9.8.1	エディットコントロールと連携する	288
9.8.2	DataGridコントロールとの連携環境を設計時に設定する	294

9.8.3	DataGridコントロールとの連携環境を実行時に設定する	298
付録A	PowerCOBOLが提供するコントロールとオブジェクト	303
付録B	V3.0以前のPowerCOBOLをご利用の方へ	307
B.1	用語	307
B.2	操作性	308
B.3	COMとの関係	309
B.4	アイテム属性名	309
B.5	メソッド呼び出し	309
B.6	資産の移行方法	310
B.6.1	移行の前準備	310
B.6.2	変換	311
B.7	資産移行時の留意事項	311
B.7.1	V3.0以前からの留意事項	311
B.7.2	V2.0以前からの留意事項	315
B.7.3	V1.0以前からの留意事項	315
B.7.4	V1.0L10からの留意事項	316
B.8	非互換項目	318
B.8.1	V3.0以前からの非互換項目	318
B.8.2	V2.0以前からの非互換項目	319
B.8.3	V1.0以前からの非互換項目	319
B.8.4	V1.0L10からの非互換項目	320
付録C	サンプルプログラムについて	321
C.1	サンプルプログラムの使用方法	321
C.2	サンプルプログラム一覧	322
C.3	サンプルプログラムの補足説明	331
C.3.1	ActiveX.ppjの補足説明	331
C.3.2	OpenActiveX.ppjの補足説明	331
C.3.3	CallActiveX.ppjの補足説明	332
C.3.4	ADODataSource2.ppjの補足説明	333
C.3.5	ADODataSource4.ppjの補足説明	334
C.3.6	DBAccess2.ppjの補足説明	334
C.3.7	DDE.ppjの補足説明	335
C.3.8	CopyData.ppjの補足説明	335
C.3.9	Listview.ppjの補足説明	336
C.3.10	PowerFORM.ppjの補足説明	336
C.3.11	PowerSORT.ppjの補足説明	336
C.3.12	TreeView.ppjの補足説明	336
付録D	こんなことがしたい - ノウハウ集	339
付録E	困ったときの対処方法 - Q & A集	351
用語集		361
索引		369

第1部 入門編

本部では、PowerCOBOLをはじめてお使いになる方を対象に、概要、アプリケーション作成の流れ、および開発時に利用するウィンドウについて説明します。

第1章 PowerCOBOLの概要

本章では、PowerCOBOLの特長、およびPowerCOBOLが作成するイベント駆動型プログラムと従来のCOBOLで作成した手続き型プログラムとの違いについて説明します。

1.1 PowerCOBOLとは

PowerCOBOLとは、COBOLプログラマがCOBOLの知識を利用して、Windows で動作するアプリケーションをビジュアルに作成するための開発環境を提供し、作成したアプリケーションを実行するためのシステムです。

Windows で動作するための特有なプログラム構造や、Windows のAPI(Application Programming Interface)の知識がなくても、PowerCOBOLを利用することにより、GUI(Graphical User Interface)アプリケーションを作成できます。

PowerCOBOLの特長を以下に示します。

ビジュアルなウィンドウ作成

アプリケーションで利用するウィンドウは、コントロール(部品)をフォーム(ウィンドウ)に配置するだけで簡単に作成できます。

また、フォームを見ながら、マウスを使ってコントロールの文字のフォント、色、位置や大きさを簡単に変更することができます。

イベントごとにプログラミング

Windows のアプリケーションは、[イベント駆動型のプログラムスタイル](#)(p7) になっています。このプログラム構造は、Windows から通知される各種のイベントに対応した処理を行ったあと、Windows に戻ります。イベントには、「コントロールのクリック」や「選択項目の変更」などがあります。

Windows からのイベントは、すべてPowerCOBOLが制御します。ウィンドウとして表示されるフォームを設計し、イベントに対応した手続きを記述するだけでアプリケーションを作成できます。

COBOLでのプログラミング

アプリケーションの動作は、COBOLで記述できます。

プログラミングには、COBOLのすべての機能(中核機能や索引/順/相対ファイルの入出力機能など)を使用できます。また、フォームに配置したコントロールは、通常のデータ名と同様にCOBOLの文で扱えます。したがって、従来のCOBOLの知識や資産をそのまま活用することができます。

一貫した開発支援

ウィンドウとして表示されるフォームの設計から手続きの編集、翻訳、リンク、実行までの操作は、PowerCOBOLだけで行うことができます。

PowerCOBOLを使用すれば、メニューからそれぞれの作業に対応するコマンド(操作を指示するための項目)を選択するだけで、アプリケーションを作成していくことができます。したがって、開発環境の切り替えや、他のツールを学習する必要はありません。

PowerCOBOLを使った開発手順については、「[アプリケーションの作成手順](#)(p12)」を参照してください。

豊富なコントロールを用意

PowerCOBOLでは、以下のコントロールを用意しています。

- Windows の標準的なコントロール (ボタン、スクロールバーなど)
- ビジネス用アプリケーションに必要なコントロール (表、グラフ)
- マルチメディア対応用のコントロール (イメージ、アニメーションなど)
- Windows のコモンコントロール (ツールバー、ツリービューなど)
- データ連携用のコントロール (データベースアクセス、ADOデータソース、DDEなど)

これらのコントロールを利用して、表現力豊かなアプリケーションを作成できます。

PowerCOBOLが用意しているコントロールの一覧は、[「PowerCOBOLが提供するコントロールとオブジェクト」](#)(p303) を参照してください。

他のツールとの容易な連携

COBOLでOLE(Object Linking and Embedding)の機能であるオートメーションや、ActiveXコントロールを利用および作成することができます。

PowerCOBOLのカスタムコントロールの組込み機能を使用して、PowerCOBOLが提供するコントロール以外の、一般に流通するActiveXコントロールも利用することができます。ただし、一般に流通するActiveXコントロールについては、そのActiveXコントロール固有の機能 (インタフェース) をもつものがあります。したがって、システム設計の際、そのActiveXコントロールがPowerCOBOLと組み合わせて正しく動作するか確認してください。

また、PowerCOBOLではActiveXコントロールを作成することもできます。COBOLの機能を活用したActiveXコントロールを作成し、各種コンテナに組み込んだり、Web上で利用したりすることができます。

ActiveXコントロールの作成方法および利用方法については、[「ActiveXコントロールを作成する」](#)(p245) および [「ActiveXコントロールを使ったアプリケーションを作成する」](#)(p260) を、Web上での利用方法については、[「PowerCOBOLで作成したActiveXコントロールをWeb上で利用する」](#)(p275) を参照してください。

クライアント・サーバ型アプリケーションの作成

COBOLのSQL文やPowerCOBOLの提供するDBアクセスコントロール、ADOデータソースコントロールを利用して、各種データベース (たとえば、Microsoft SQL Server(TM) など) へアクセスすることにより、クライアント・サーバ型のアプリケーションを作成できます。

DBアクセスコントロールを使ったアプリケーションの作成方法については、[「DBアクセスコントロールを利用してデータベースと連携する」](#)(p277) を、ADOデータソースコントロールを使ったアプリケーションの作成方法については、[「ADOデータソースコントロールを利用してデータベースと連携する」](#)(p287) を参照してください。

1.2 イベント駆動型プログラムとは

アプリケーションを作成するためのプログラムスタイルには、手続き型プログラムスタイルとイベント駆動型プログラムスタイルの2つのスタイルがあります。

従来のプログラムスタイル

従来のメインフレームやオフコンで使用していたバッチ型アプリケーションのプログラムスタイルは、手続き型プログラムスタイルです。プログラムの実行は、手続き部の先頭から始まり、記述したCOBOLの文単位に実行されます。手続き型プログラムスタイルは、バッチ型アプリケーションだけでなくクライアントアプリケーションの開発でも利用することができるので、従来のプログラム資産を流用することが可能です。

COBOLを使って、ウィンドウへアクセスするプログラムを作成するには、ファイルを扱うようにWRITE文を使ってデータをウィンドウに出力したり、READ文を使ってウィンドウから入力したデータを受け取ったりします。このように手続き型プログラムでは、WRITE文 / READ文、DISPLAY文 / ACCEPT文の実行順序は、実行する手続き自身が制御します。また、スクリーン機能を使ったプログラムでは、DISPLAY文が実行されるとスクリーンにデータを表示し、ACCEPT文が実行されるとスクリーンからデータを読み込みます。COBOLを使ったプログラミング方法の詳細は、『NetCOBOL 使用手引書』を参照してください。

```

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.      DENPYOU1.
000400 ENVIRONMENT      DIVISION.
      :
000700      SELECT 伝票画面 ASSIGN TO GS-DISPFIL
000800      SYMBOLIC DESTINATION IS "DSP"
000900      FORMAT          IS 伝票
      :
001400      SELECT 商品ファイル ASSIGN TO SYOUIHIN
001500      FILE STATUS IS ファイル状態
001600      ORGANIZATION IS INDEXED
      :
002000 DATA          DIVISION.
002100 FILE              SECTION.
002200 FD 商品ファイル.
002300 COPY SYOUIHIN1.
002400 FD 伝票画面.
002500 COPY DENPYOU01 OF XMDLIB.
002700 WORKING-STORAGE SECTION.
      :
006900 PROCEDURE          DIVISION.
010300 OPEN INPUT 商品ファイル.
011900 OPEN I-O 伝票画面.          表示ファイルオープン
      :
012500 WRITE DENPYOU1.          画面出力
012600 READ 伝票画面.          画面データ入力
012700 IF END-KY THEN
012800 GO TO 入力終了.
      :
025000 入力終了.
025100 CLOSE 伝票 画面商品ファイル. クローズ
025200 EXIT PROGRAM.

```

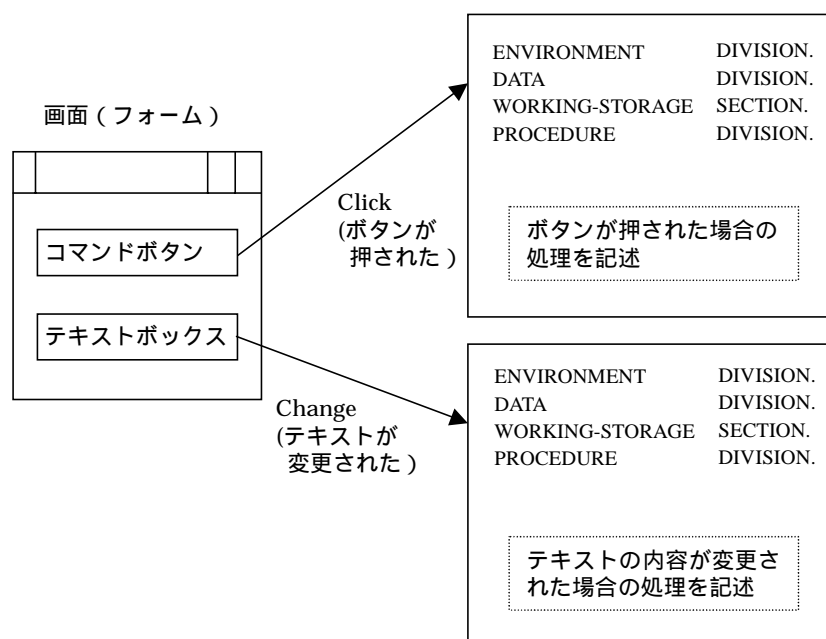
イベント駆動型のプログラムスタイル

イベント駆動型のプログラムスタイルは、ある事象を単位として、その事象が発生した場合の動作を手続きとして記述していく方法です。事象には、「ボタンが押された（クリックされた）」、「テキスト（入力データ）が変更された」といったものがあり、これらをイベントと呼んでいます。これらのイベントに対応する手続きを記述していく方法を、イベント駆動型プログラムスタイルといいます。

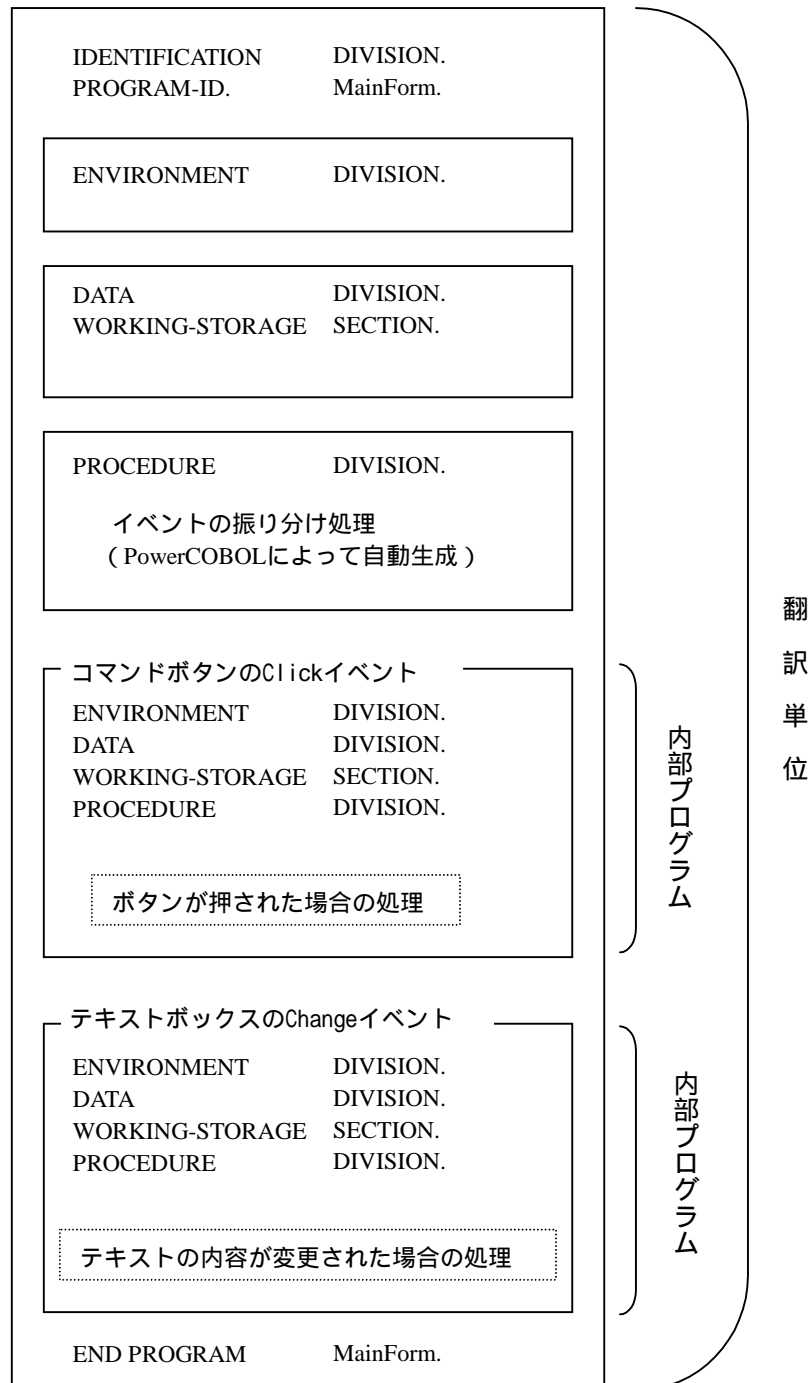
PowerCOBOLのプログラムスタイル

PowerCOBOLのプログラムスタイルは、イベント駆動型プログラムスタイルです。PowerCOBOLでは、イベントごとに実行される手続きをCOBOLで記述していきます。

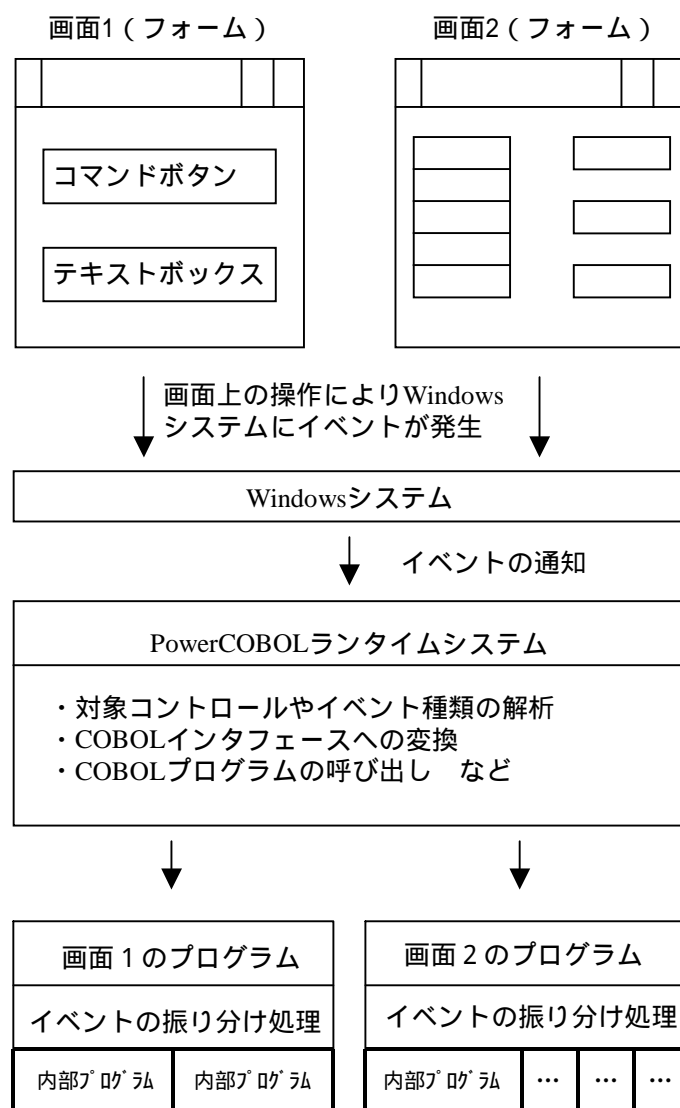
たとえば、「ボタンが押された」場合には、"Click"というイベントが発生します。また、「テキストが変更された」場合には、"Change"というイベントが発生します。これらのイベントごとに、イベントが発生したときの動作をCOBOLで記述していきます。



PowerCOBOLでは、1つのウィンドウが1つの翻訳単位 (外部プログラム) となります。この外部プログラムは、翻訳時にPowerCOBOLによって自動的に生成されます。記述した各イベントに対応するCOBOLの手続きは、その外部プログラムに含まれる内部プログラムとして展開されます。外部プログラムは、以下のようなCOBOLソースとして生成されます。



プログラムが実行されると、PowerCOBOLの実行システム(ランタイムシステム)は、Windows からのイベントを受け取り、そのイベントに対応する内部プログラムを呼び出すための制御をします。



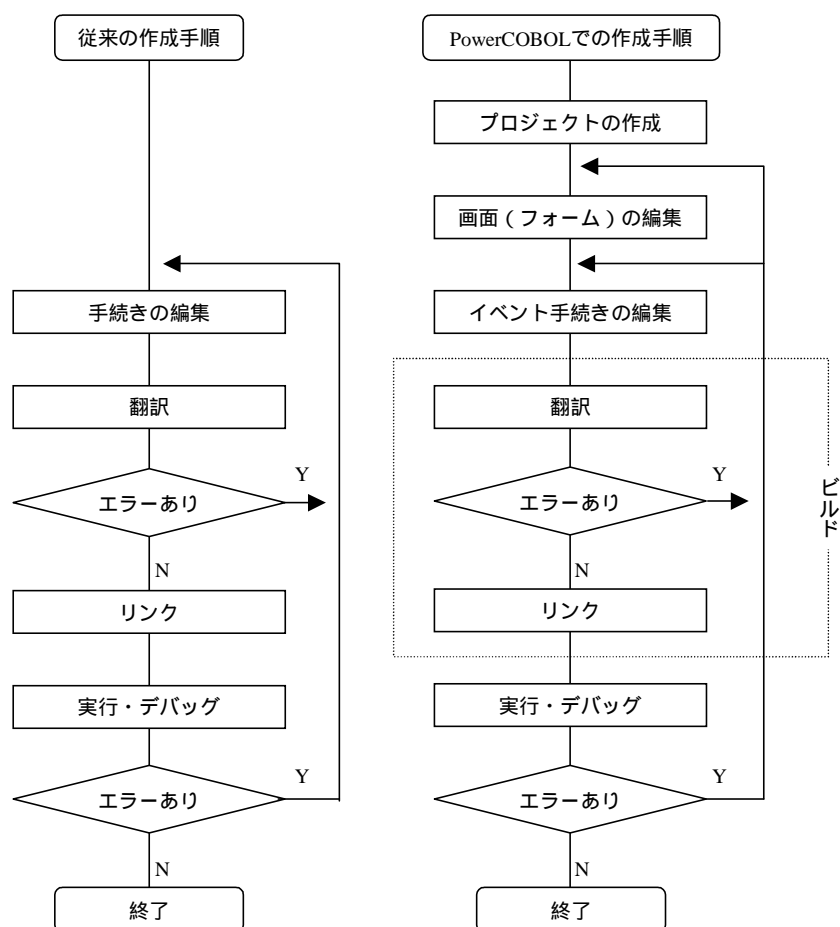
第2章 アプリケーション作成の流れ

本章では、PowerCOBOLでアプリケーションを作成する手順、およびアプリケーションの構成について説明します。

2.1 アプリケーションの作成手順

PowerCOBOLを使ってアプリケーションを作成する手順は、従来のメインフレームやオフコンで使用していたバッチ型アプリケーションを開発する手順と大きな違いはありません。従来のアプリケーションとの違いは、Windows の画面に表示されるウィンドウ（フォーム）の設計が含まれていることです。したがって、ウィンドウの設計を除き、手続きの記述から翻訳～実行までの手順は変わりません。

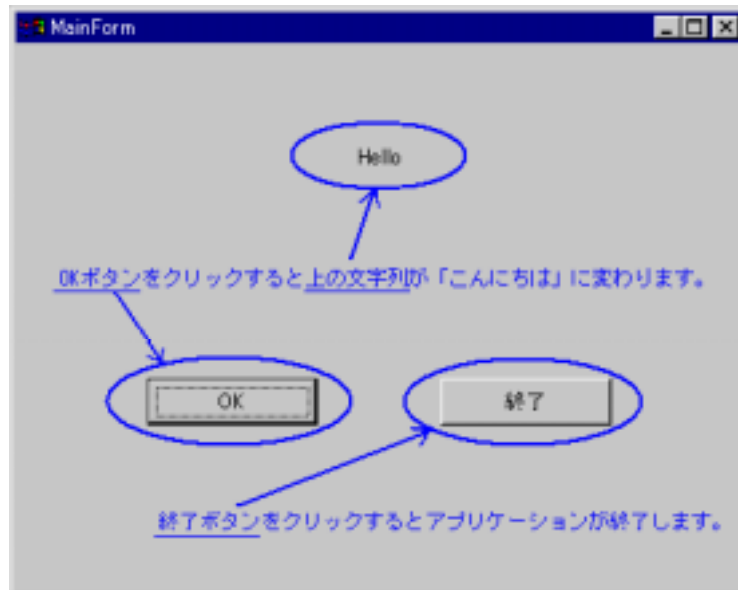
以下に、従来のアプリケーション作成手順とPowerCOBOLでの作成手順をフローチャートで示します。



次節以降、作成手順の概要について、実際にサンプルプログラムを作成しながら説明していきます。このサンプルプログラムは、以下の機能を持ちます。

OKボタン(コマンドボタンコントロール)をクリックすると、文字列表示領域(スタティックテキストコントロール)の文字列「Hello」が、「こんにちは」に変更されます。

終了ボタン(コマンドボタンコントロール)をクリックすると、アプリケーションが終了します。



このサンプルプログラムは、以下の手順で作成していきます。

1. [プロジェクトの作成](#) (p15)
PowerCOBOLを起動します。
新規にプロジェクト(1つのアプリケーションの単位)を作成します。
2. [フォームの編集](#) (p18)
フォーム編集用のウィンドウを表示します。
フォームにコントロールを配置します。
コントロールのプロパティ(属性)を設定します。
3. [イベント手続きの編集](#) (p24)
手続きを編集するためのウィンドウを表示します。
OKボタンがクリックされたときの手続きを記述します。
終了ボタンがクリックされたときの手続きを記述します。
4. [実行可能プログラムの作成\(ビルド\)と実行](#) (p28)
プロジェクトを保存します。
モジュールをビルドして実行可能プログラムを作成します。
エラーがあった場合は、エラー箇所を修正します。
作成した実行可能プログラムを実行します。
PowerCOBOLを終了します。



注意

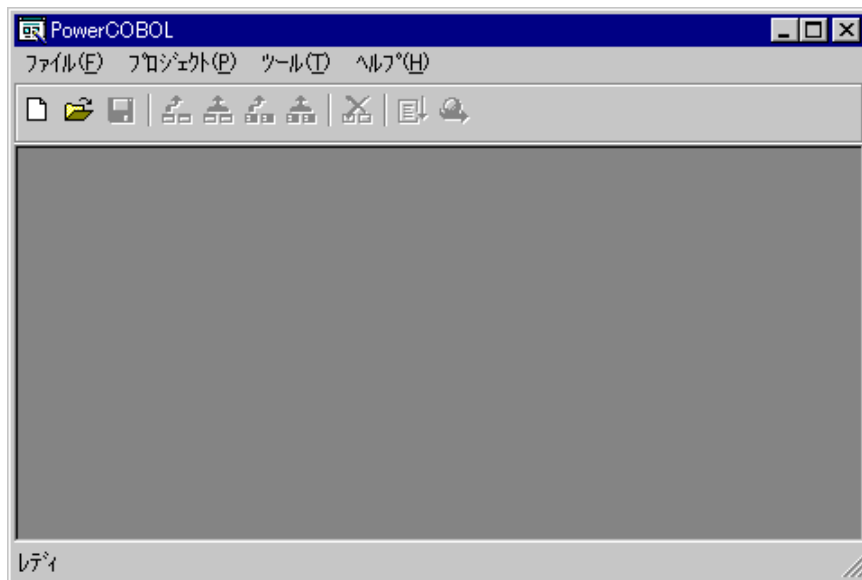
実際にアプリケーションを作成する場合には、プログラムのデバッグが必要になります。デバッグについては、「[アプリケーションをデバッグしよう](#) (p141) 」を参照してください。

2.2 プロジェクトの作成

PowerCOBOLでは、1つのプロジェクトが1つのアプリケーションを作成したり管理したりする単位となります。また、開発資産として保存するファイルの単位となります。

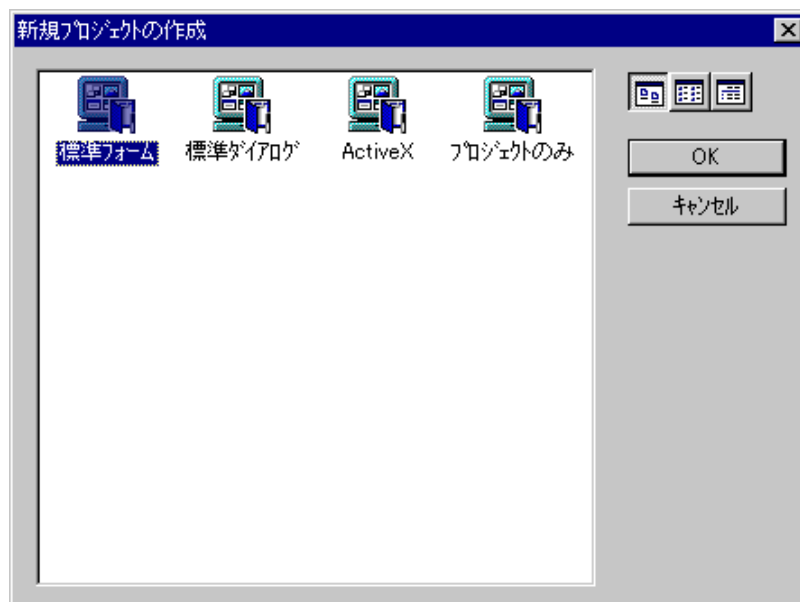
PowerCOBOLを起動する

Windows のスタートメニューから、[プログラム - NetCOBOL V7.0 - PowerCOBOL] メニューの [PowerCOBOL V7.0] コマンドを選択します。PowerCOBOLのスタートアップウィンドウ（スタートアップウィンドウ）が表示されたあと、アプリケーションを開発するためのウィンドウが表示されます。このウィンドウを[プロジェクトウィンドウ](#)（ p34）といいます。プロジェクトウィンドウのタイトルには、「PowerCOBOL」と表示されています。



新規にプロジェクトを作成する

〔ファイル〕メニューの〔新規プロジェクトの作成〕コマンドを選択します。どのような形式のプロジェクトを作成するかを選択するためのダイアログボックスが表示されます。ダイアログボックスのタイトルには、「新規プロジェクトの作成」と表示されています。



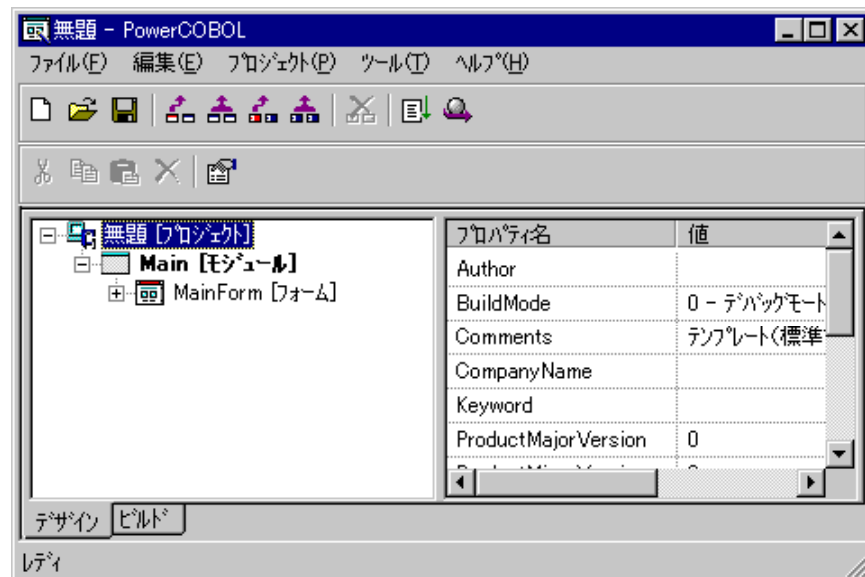
プロジェクトの形式を選択する

[新規プロジェクトの作成] ダイアログボックスの中から、[標準フォーム] を選択し、OKボタンをクリックします。「無題」というプロジェクトが作成され、プロジェクトウィンドウに表示されます。プロジェクトウィンドウのタイトルには、「無題 - PowerCOBOL」と表示されています。

[標準フォーム] 以外の形式については、『リファレンス』を参照してください。

プロジェクトの構成要素として作成されたモジュールは、実行ファイル(EXEファイルやDLLファイル)を作成する単位となります。プロジェクトおよびモジュールについては、「[プロジェクトの構成](#) (p31)」および「[プロジェクトを作成する](#) (p51)」を参照してください。

また、モジュールの構成要素として作成されたフォームは、オブジェクトファイル(Objファイル)を作成する単位となります。



プロジェクトの内容を確認するには

プロジェクトウィンドウには、作成中のプロジェクトに含まれるフォーム(ウィンドウ)や手続きの構成などが表示されます。プロジェクトの構成要素は、+または-で表示された部分をクリックすることにより、その構成要素を展開または閉じた状態で表示できます。

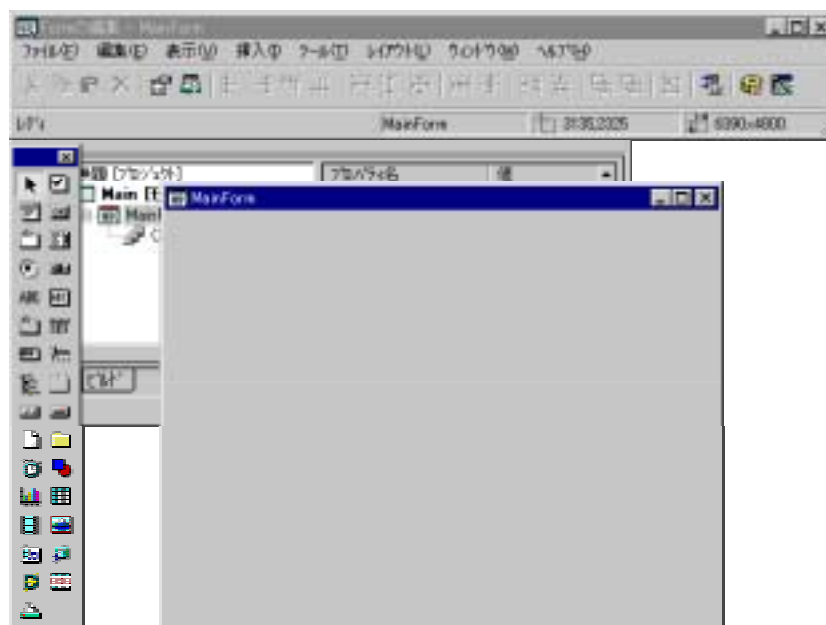
また、プロジェクト内のすべての構成要素を表示する場合には、プロジェクトウィンドウの最上位の要素(上記の図では、[無題 [プロジェクト]])の部分を選択し、マウスの右ボタンをクリックして表示されたポップアップメニューから、[すべて展開] コマンドを選択します。

2.3 フォームの編集

PowerCOBOLで設計するウィンドウを、フォームといいます。PowerCOBOLではフォームを使うことにより、ウィンドウをもつビジュアルなアプリケーションを作成できます。フォームは、モジュールの構成要素であり、COBOLコンパイラでのプログラムの翻訳単位となります。

フォームを開く

プロジェクトウィンドウの左側のウィンドウにある [MainForm [フォーム]] を選択し、マウスの右ボタンをクリックすると、ポップアップメニューが表示されます。そのポップアップメニューの先頭にある [開く] コマンドを選択すると、フォームを編集するための編集環境が表示されます。



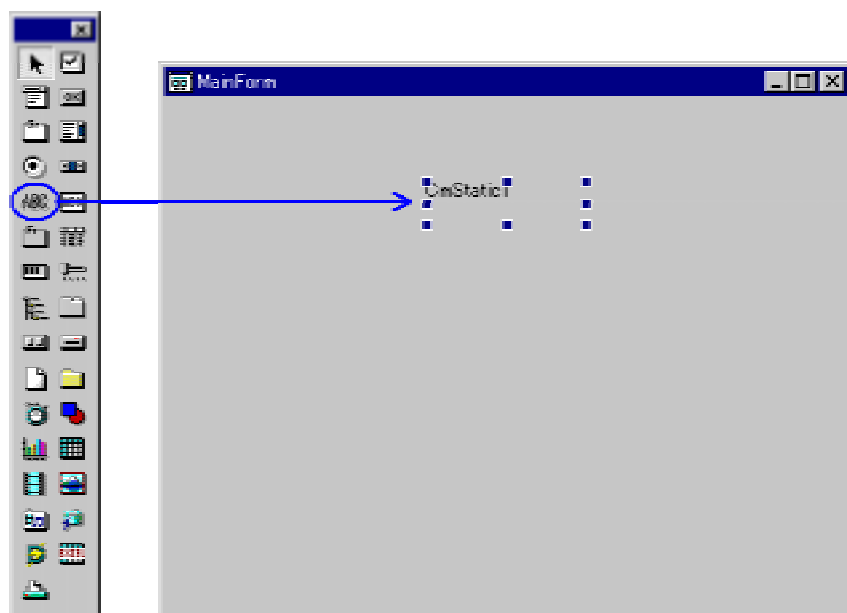
中央にある「MainForm」というタイトルをもつウィンドウがフォームです。その上にある「Formの編集-MainForm」というタイトルが表示されているウィンドウを[フォーム編集ウィンドウ](#) (p37) といいます。フォーム編集ウィンドウには、フォームを編集するためのメニュー、ツールバーやステータスバーが配置されています。

また、左側のタイトルをもたない縦長のウィンドウには、フォームに配置して利用するためのコントロール (部品) が並んでいます。これを[ツールボックス](#) (p38) といいます。

フォームにスタティックテキストコントロールを配置する

スタティックテキスト (StaticText) コントロールとは、フォーム上の任意の位置に文字列を表示するための部品です。コントロールは、以下の手順でフォームに配置できます。

1. ツールボックス上でスタティックテキストコントロールのボタンをクリックします。
2. マウスポインタをフォーム上に移動します。このとき、マウスポインタは十字型に変化します。
3. フォーム上の任意の位置でもう1度クリックします。クリックした位置にスタティックテキストが配置され、8個の四角点で囲まれた状態になります。この状態を選択状態といいます。
4. コントロールのサイズを変更する場合は、四角点のどれかをドラッグします。
5. コントロールの位置を変更する場合は、四角点で囲まれた部分をドラッグします。

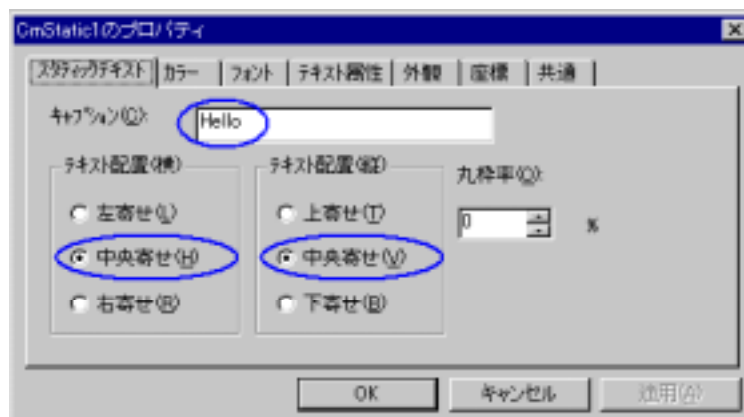


スタティックテキストコントロールのプロパティを設定する

スタティックテキストコントロールを配置したばかりの状態では、コントロール上に "CmStatic1" という文字列が表示されています。スタティックテキストコントロールのプロパティを変更することにより、表示されている文字列を変更したり、表示する位置を調整したりできます。

プロパティは、以下の手順で変更します。

1. スタティックテキストコントロールをクリックし、選択状態にします。
2. マウスの右ボタンをクリックし、ポップアップメニューから [プロパティ] コマンドを選択します。
文字列を設定するためのダイアログボックスが表示されます。
3. [キャプション] を "CmStatic1" から "Hello" に変更します。
4. [テキスト配置(横)] を中央寄せにします。
5. [テキスト配置(縦)] を中央寄せにします。



6. [共通] タブをクリックします。
7. [名前] を "CmStatic1" から "ST-TEXT" に変更します。ここで設定した名前は、COBOLの手続き中でデータ項目名として使用できます。



8. OKボタンをクリックします。



ここで [名前] に設定した "ST-TEXT" は、コントロールの名前となります。コントロールの名前は、手続き中でデータ名と同様に扱うことができます。また、この名前には、日本語（全角文字）を使用することもできます。ただし、半角文字と全角文字を組み合わせることはできません。したがって、"ST-テキスト" といった名前を指定することはできません。

コントロールの命名規則については、「[プロジェクト構成要素の命名規則](#) (p93)」を参照してください。

コントロール名をつける場合、コントロールの種類ごとに名前の先頭にプレフィクスを入れておくと、手続きが読みやすくなります。

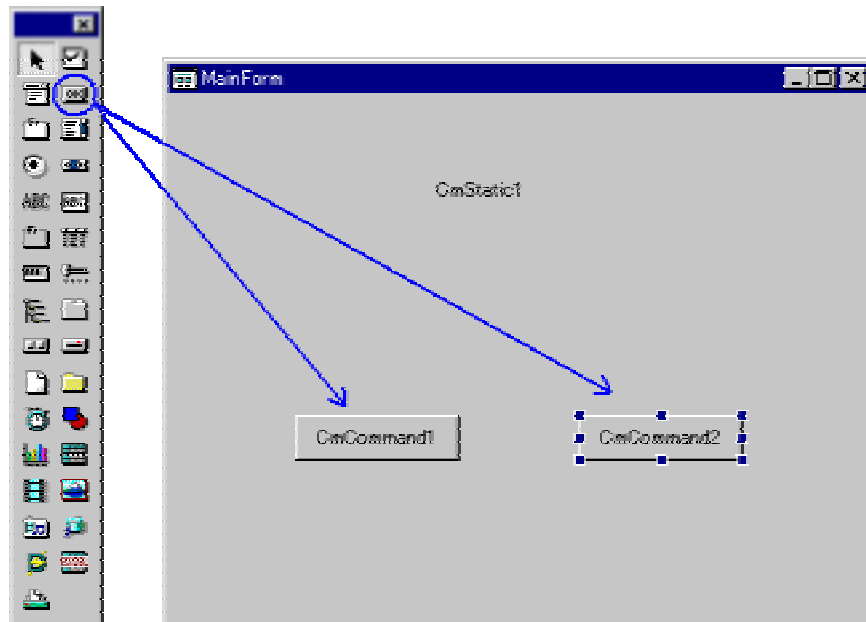
たとえば、ここでは "ST-TEXT" と名付けましたが、"表示用テキスト - あいさつ" や "ラベル - テキスト 1" などの名前が考えられます。



設定できるプロパティの内容は、コントロールの種類ごとに異なります。各コントロールで設定できるプロパティの詳細は、『リファレンス』を参照してください。

フォームにコマンドボタンコントロールを配置する

コマンドボタン (CommandButton) コントロールとは、OKやキャンセルなど、Windows で一般的に使われているボタンをフォーム上で利用するための部品です。コマンドボタンコントロールも、スタティックテキストコントロールを配置したときと同様の手順でフォームに配置することができます。このサンプルプログラムでは、2つのコマンドボタンコントロールをフォームに配置します。



コマンドボタンコントロールのプロパティを設定する

コマンドボタンを配置したばかりの状態では、コントロール上にはそれぞれ、"CmCommand1"および"CmCommand2"という文字列が表示されています。これらの文字列も、スタティックテキストコントロールの場合と同様の手順で変更できます。

1. "CmCommand1"と表示されているコマンドボタンコントロールをクリックし、選択状態にします。
2. マウスの右ボタンをクリックし、ポップアップメニューから[プロパティ] コマンドを選択します。
文字列を設定するためのダイアログボックスが表示されます。
3. [キャプション] を"CmCommand1"から"OK"に変更します。

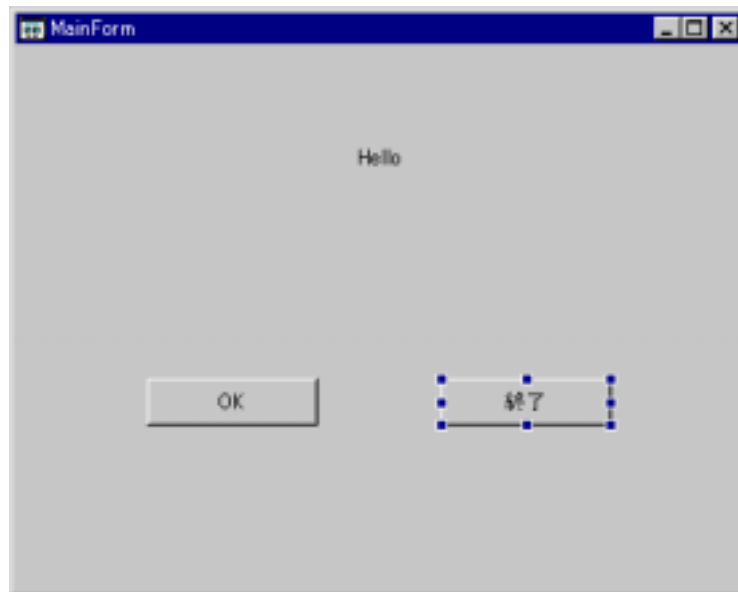


4. [共通] タブをクリックします。
5. [名前] を"CmCommand1"から"BT-OK"に変更します。



6. OKボタンをクリックします。
7. 同様の方法で、もう1つのコマンドボタンコントロールに表示されている文字列を"CmCommand2"から"終了"に、名前を"CmCommand2"から"BT-EXIT"に変更します。

この時点で、以下のようなフォームが作成できていることを確認します。



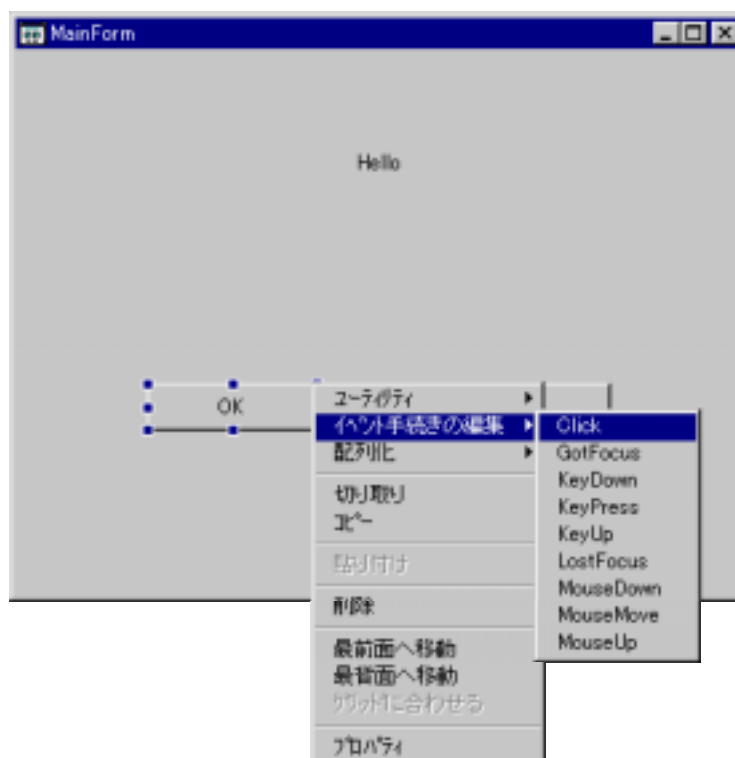
2.4 イベント手続きの編集

PowerCOBOLでは、イベントの発生によって実行される手続きを、イベント手続きとして記述します。本サンプルでは、OKボタンをクリックしたとき（Clickというイベントが発生したとき）の手続き、および終了ボタンをクリックしたときの手続きを記述します。

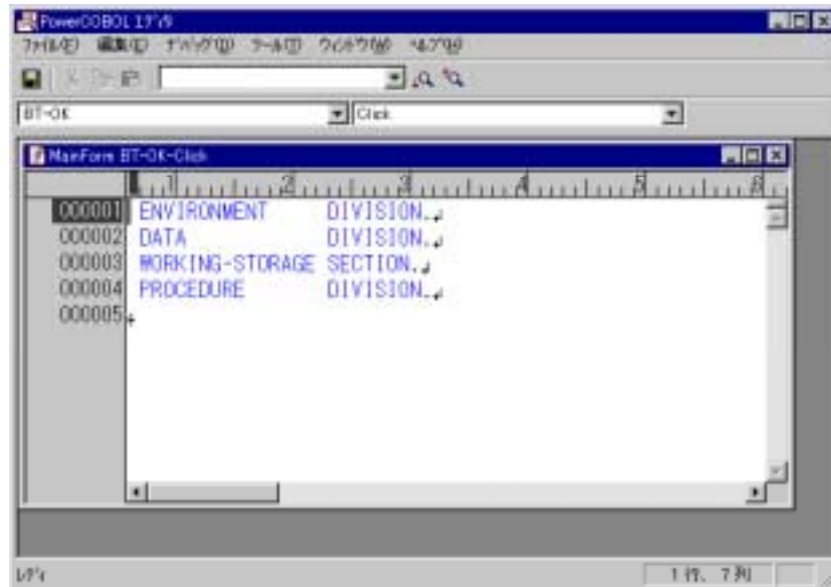
イベント手続きの編集ウィンドウを開く

イベント手続きを記述するための編集ウィンドウを表示します。編集ウィンドウは、以下の手順で表示できます。

1. フォーム上のOKボタンをクリックし、選択状態にします。
2. マウスの右ボタンをクリックし、ポップアップメニューの[イベント手続きの編集] サブメニューから [Click] コマンドを選択します。

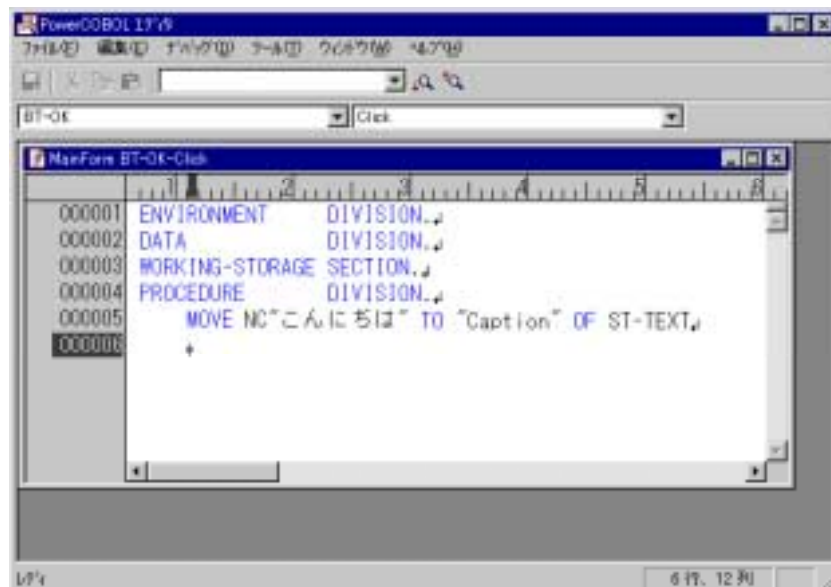


イベント手続きを編集するためのウィンドウが、表示されます。ここで、OKボタンをクリックしたときの手続きを入力します。



OKボタンをクリックしたときの手続き

OKボタンをクリックしたときは、「スタティックテキストコントロールに表示されている文字列を変更する」という手続きを実行します。この手続きは以下のようになります。



ST-TEXTは、「[フォームの編集](#) (p18) 」で設定したスタティックテキストコントロールの名前です。"Caption"は、スタティックテキストコントロールに表示する文字列を示すプロパティです。この手続きで、文字列"こんにちは"をス

タテックテキストコントロールの表示文字列として設定することができます。



フォームやコントロールは、それぞれプロパティをもっています。たとえば、表示する文字列（キャプション）に対応するプロパティは"Caption"、コントロールの背景色に対応するプロパティは"BackColor"といったものです。ここでは、表示する文字列を変更するために"Caption"プロパティに値を転記しました。

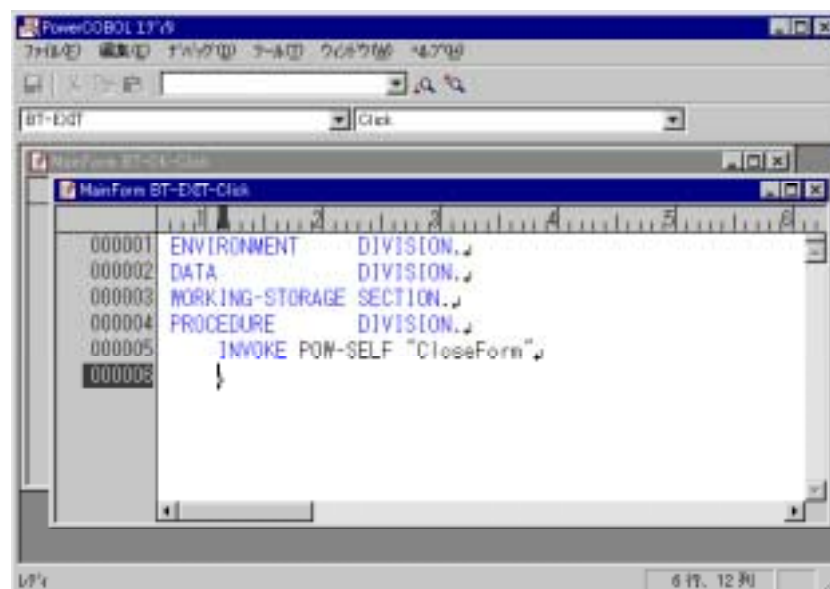
このように、手続き上でプロパティに値を設定したり、また、すでにプロパティに設定されている値を参照したりすることにより、フォーム上のコントロールの状態や動作を変更できます。

プロパティへのアクセス方法については、「[プロパティへのアクセス方法](#) (p171)」を参照してください。また、フォームや各コントロールで利用できるプロパティの詳細は、『リファレンス』を参照してください。このアプリケーションを実行し、OKボタンをクリックすると、WindowsからPowerCOBOLの実行システム（ランタイムシステム）にイベントが通知されます。PowerCOBOLのランタイムシステムは、その通知をもとに、記述されたイベント手続きを実行させます。

終了ボタンをクリックしたときの手続き

OKボタンの場合と同様の手順で、終了ボタンの"Click"イベントを編集するウィンドウを表示します。

終了ボタンをクリックしたときは、「このフォーム自身を閉じて、アプリケーションを終了する」という手続きを実行します。この手続きは以下のようになります。



POW-SELFは、現在の編集対象であるフォーム自身を示しています。また、"CloseForm"は、フォームを閉じるためのメソッドです。このメソッドを、メソッドの動作対象であるフォームに対して、INVOKE文で呼び出すことにより、フォームを閉じることができます。



メソッドは、ある動作を実行するためのサブルーチンのようなもので、フォームやコントロールごとに異なったメソッドが用意されています。たとえば、フォームを閉じるメソッドとして"CloseForm"、リストボックスに項目を追加するメソッドとして"AddString"などがあります。メソッドの呼び出し方法については、「[メソッドの呼び出し方法](#)(p180) 」を参照してください。また、フォームや各コントロールで利用できるメソッドの詳細は、『リファレンス』を参照してください。

2.5 実行可能プログラムの作成（ビルド）と実行

実行可能プログラムは、モジュールごとに作成されます。モジュールをビルドすることにより、実行可能プログラムを作成することができます。ビルドは、更新されたフォームや手続きなどを対象にプログラムを翻訳、リンクします。

プロジェクトを保存する

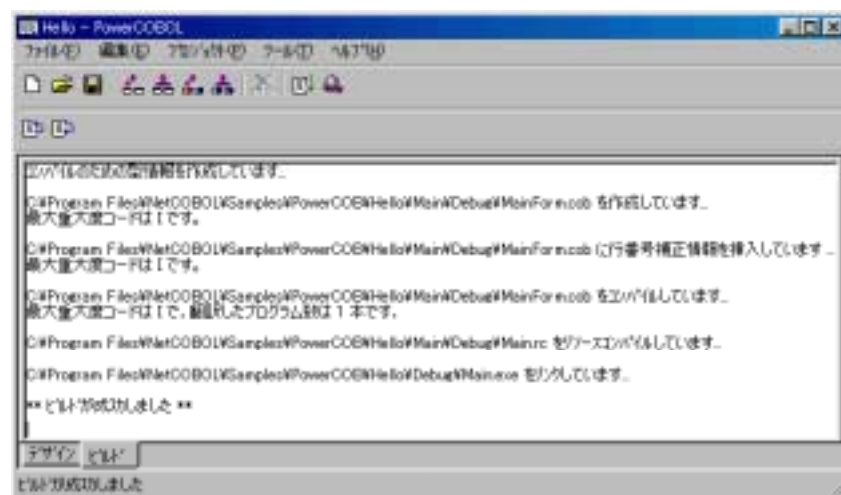
実行可能プログラムを作成するには、あらかじめそのプロジェクトが保存されている必要があります。以下の手順で、プロジェクトを保存します。

1. プロジェクトウィンドウの[ファイル]メニューの[名前を付けてプロジェクトの保存]コマンドを選択します。
2. [ファイル名を付けて保存]ダイアログボックスを使って、任意のフォルダへプロジェクトを保存します。このサンプルプログラムの例では、"`C:\Program Files\NetCOBOL\Samples\PowerCOB\Hello\Hello.ppj`" に保存しています。

モジュールをビルドして実行可能プログラムを作成する

プロジェクトウィンドウの左側のウィンドウにある[Main [モジュール]]を選択し、マウスの右ボタンをクリックして、表示されたポップアップメニューから[ビルド]コマンドを選択します。

プロジェクトウィンドウが以下のように切り替わり、ビルドが成功すると元の状態に戻ります。

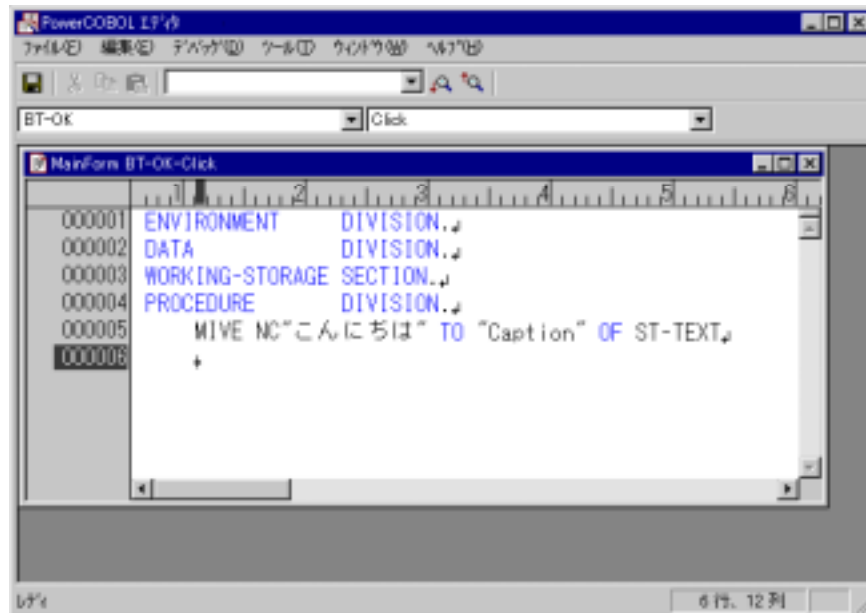


ビルドの結果を確認する場合は、プロジェクトウィンドウの下部にある[ビルド]タブ（しおりのような部分）をクリックしてください。

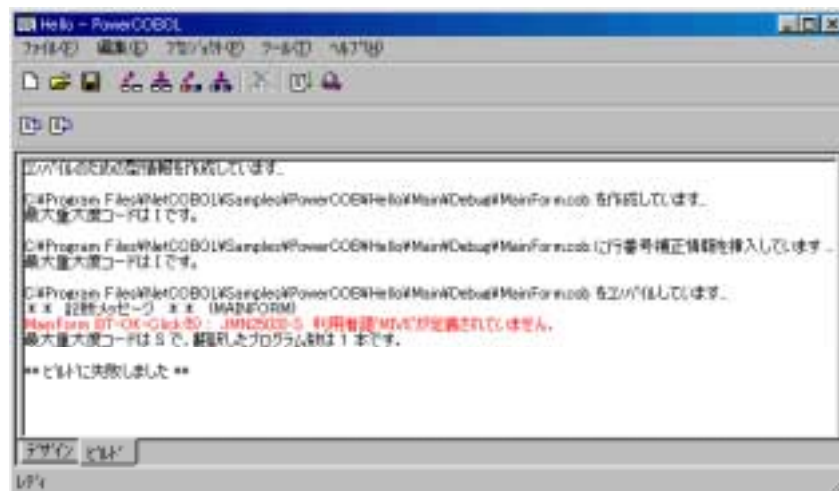
エラーを修正する

エラーがあった場合には、エラー箇所を修正します。

たとえば、OKボタンの"Click"イベントに、以下のような誤った手続き（MOVEをMOVEと記述）を記述したとします。



このモジュールをビルドすると、「ビルドに失敗しました」というメッセージボックスが表示されるので、OKボタンをクリックしてください。エラーがあった場合には、プロジェクトウィンドウは元の状態に自動的に戻らず、以下のような診断メッセージが表示されます。



このような場合は、診断メッセージが表示されている行の任意の位置をダブルクリックします。エラーがあるイベント手続きが表示され、該当する行が表示されるので、すぐに手続きを修正できます。

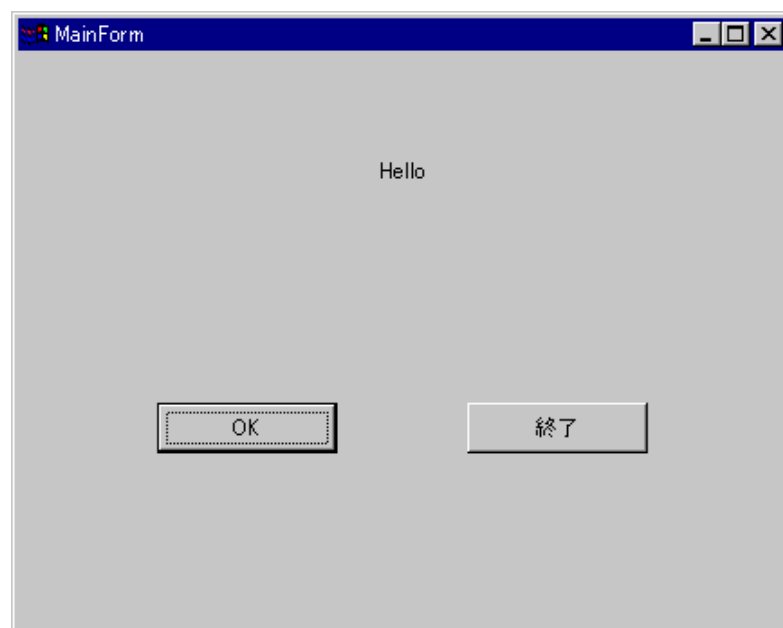
手続きを修正したら [プロジェクト] メニューの [ビルド] コマンド選択し、再度モジュールをビルドします。

このとき、「ビルドまたはコンパイルのためにプロジェクトを保存しますか?」というメッセージが表示されるので、OKボタンをクリックします。

プログラムを実行する

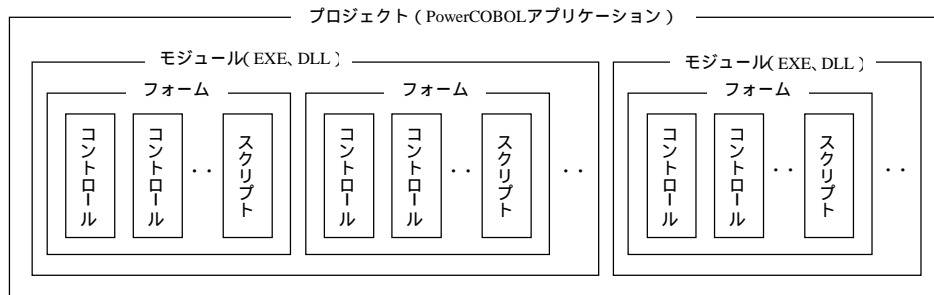
プロジェクトウィンドウの左側のウィンドウにある [Main [モジュール]] を選択し、マウスの右ボタンをクリックして、表示されたポップアップメニューから [実行] コマンドを選択します。

実行すると、以下のようなウィンドウが表示されます。OKボタンおよび終了ボタンが正しく動作することを確認してください。



2.6 プロジェクトの構成

最後に、プロジェクトの構成についてまとめます。
1つのアプリケーションは、1つのプロジェクトによって開発できました。
PowerCOBOLで作成するプロジェクトは、以下のように構成されています。



プロジェクト

プロジェクトとは、1つのアプリケーションを作成したり管理したりする単位であり、開発資産となるファイルの単位です。プロジェクトは、1つまたは複数のモジュールで構成されています。

プロジェクトは複合ファイルとして保存され、1つのファイルの中で、ツリー構造で管理されています。(プロジェクトウィンドウの左側に表示されているツリー構造と同じ構造の複合ファイルです。)

したがって、1つのアプリケーションの開発資産として、1つのプロジェクトファイルを保存しておくだけでよいということになります。

ただし、外部ファイルを利用している場合は、その外部ファイルも開発資産として必要です。外部ファイルについては、「[外部ファイルを使う](#) (p96)」を参照してください。

モジュール

モジュールとは、実行可能ファイル(EXEファイルやDLLファイル)を作成する単位となります。モジュールは、1つまたは複数のフォームで構成されます。また、必要に応じて、COBOLファイル、オブジェクトファイル、ライブラリファイルおよびソースファイルなど、外部ファイルへのリンク情報(ファイルへのパス情報)をもちます。外部ファイルについては、「[外部ファイルを使う](#) (p96)」を参照してください。

フォーム

フォームとは、グラフィカルなユーザインタフェースを実現するためのウィンドウです。フォームは、モジュールの構成要素であり、COBOLコンパイラでのプログラムの翻訳単位となります。フォームは、1つまたは複数のコントロール、およびそれらのイベント手続きであるスクリプトなどで構成されています。

MEMO

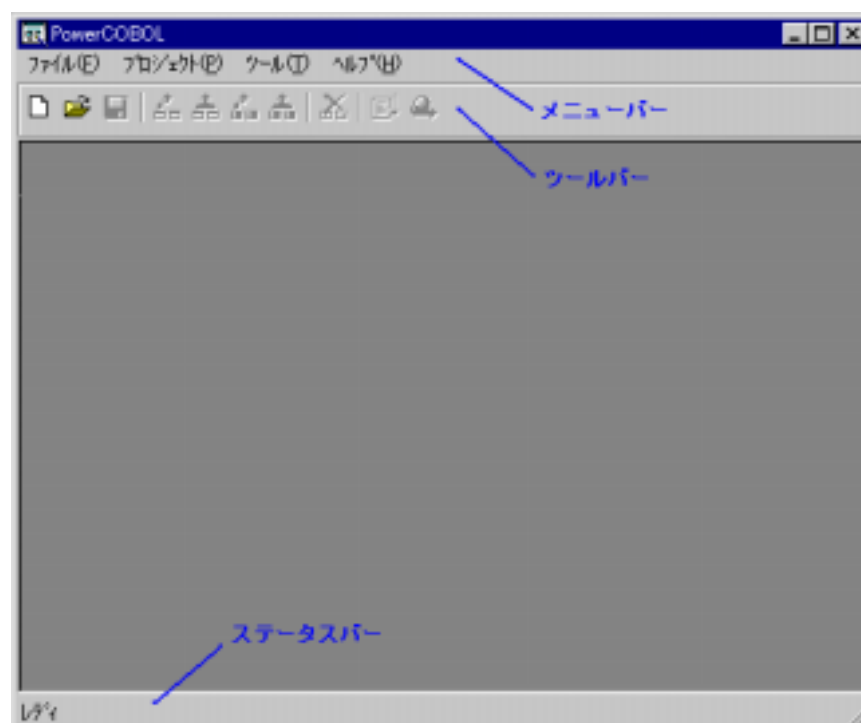
第3章 PowerCOBOLの基本操作

本章では、PowerCOBOLでアプリケーションを作成するために使用する各種ウィンドウの名称、および基本的な操作方法などについて説明します。

3.1 ウィンドウの名称

3.1.1 プロジェクトウィンドウ

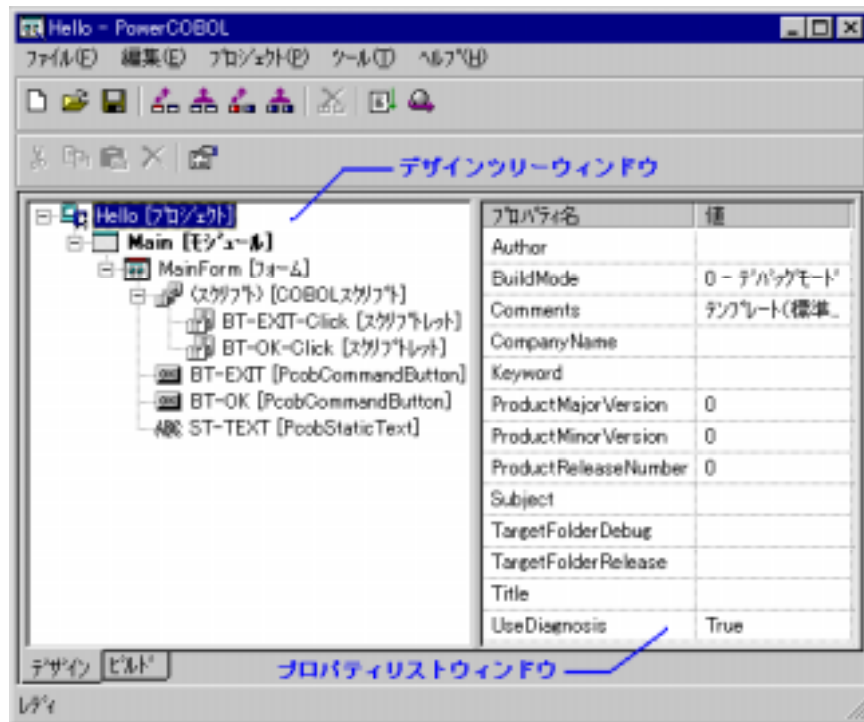
プロジェクトウィンドウは、アプリケーションを開発するために使用されるウィンドウで、PowerCOBOLを起動した場合に表示されます。プロジェクトウィンドウは、メニューバーをもち、さらにそのメニュー項目の中から使用頻度が高い項目と対応づけられたボタンを、メニューバーの下ツールバーにもちます。また、ウィンドウ最下部のステータスバーには、メニュー項目、ツールバーのボタン説明などの情報が表示されます。



プロジェクトウィンドウは、フォームや手続きを編集するためのデザインビュー、ビルド時に使用するビルドビューおよびデバッグ時に使用するデバッグビューをもちます。プロジェクトウィンドウを使った各種機能の操作方法については、「[開発環境編](#) (p47) 」を参照してください。

デザインビュー

デザインビューは、プロジェクトウィンドウの左下にある「デザイン」タブを選択することにより表示できます。デザインビューは、フォーム、コントロールおよび手続きなどを編集する場合の起点となります。



デザインビューは、左側にプロジェクトの構成を示すデザインツリーウィンドウをもち、右側に各構成要素の属性を示すプロパティリストウィンドウをもちます。デザインビューの中央にある分割バーを左右に移動することにより、2つのウィンドウサイズを変更できます。

デザインツリーウィンドウ

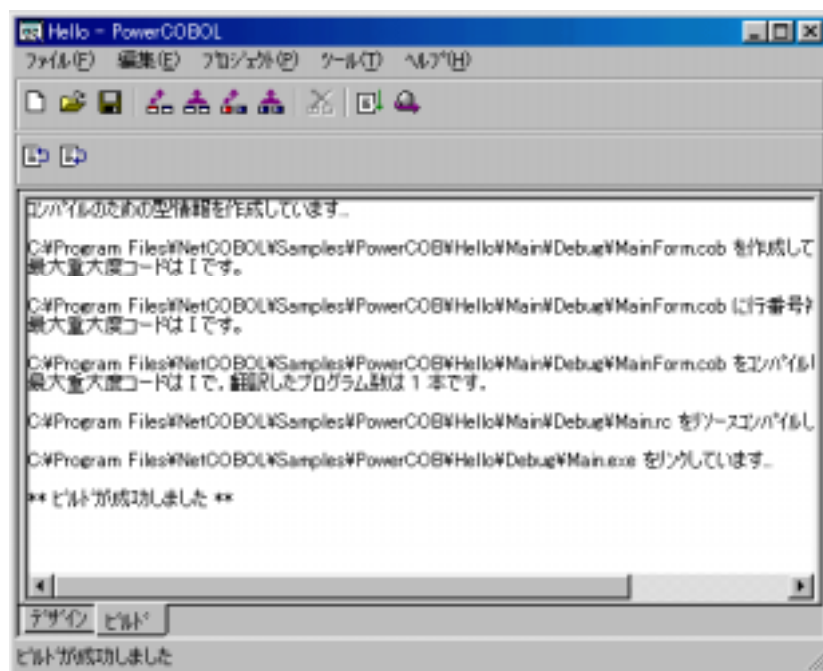
デザインツリーウィンドウには、プロジェクトの構成要素が階層的に表示されます。デザインツリーウィンドウの構成要素を選択し、ポップアップメニューからコマンドを選択することにより、構成要素の編集（削除、コピーおよび移動）、プロパティの設定、名前の変更などができます。

プロパティリストウィンドウ

プロパティリストウィンドウには、デザインツリーウィンドウで選択されている構成要素がもつプロパティ名の一覧と、その値が表示されます。プロパティリストウィンドウの値の列に、直接入力することにより、プロパティ設定ダイアログボックスを使用せず、プロパティを変更することもできます。ただし、プロパティによっては、実行時だけ設定や参照ができるものがあります。これらのプロパティは、このプロパティリストウィンドウには表示されません。また、設計時に参照だけしかできないプロパティは、プロパティリストウィンドウに表示されますが、値を入力することはできません。プロパティへのアクセス方法については、「[プロパティへのアクセス方法](#) (p171)」を参照してください。また、各プロパティについての詳細は、『リファレンス』を参照してください。

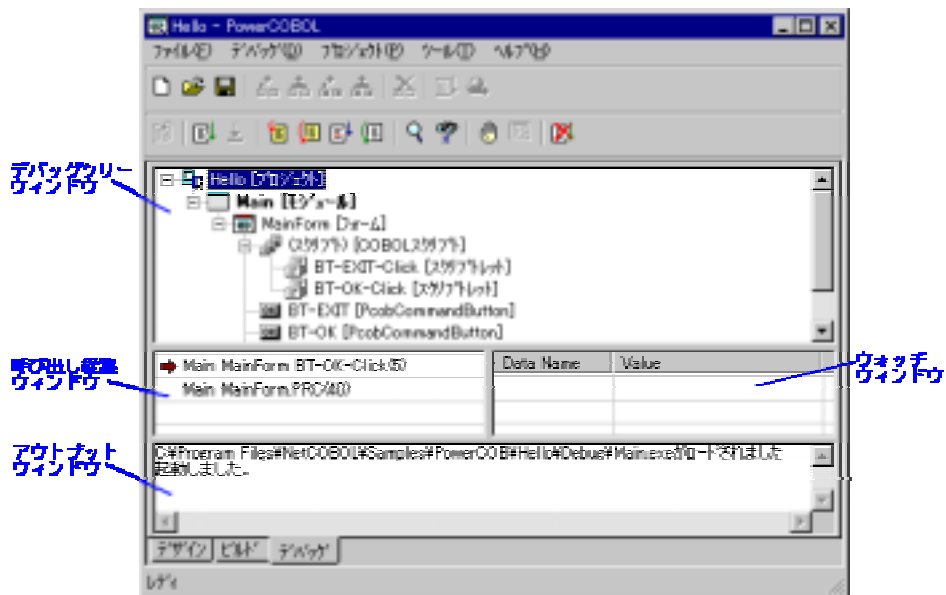
ビルドビュー

ビルドビューは、プロジェクトウィンドウの左下にある[ビルド]タブを選択することにより表示できます。ビルドビューは、ビルド時の進行状況を表示したり、エラーがあった場合のメッセージを表示したりします。



デバッグビュー

デバッグビューは、デバッグ時にだけ、プロジェクトウィンドウの左下にある[デバッグ]タブを選択することにより表示できます。デバッグビューは、デバッグに必要な情報を表示したり設定したりするための、デバッグツリーウィンドウ、呼び出し経路ウィンドウ、ウォッチウィンドウおよびアウトプットウィンドウをもちます。デバッグビューでの操作方法の詳細は、「[アプリケーションをデバッグしよう](#)」(p141)」を参照してください。



デバッグツリーウィンドウ

デバッグツリーウィンドウは、デバッグするプログラムのイベント手続きを表示するために使用します。また、デバッグ対象のモジュールは、ツリー上でボールド（太字）表示されます。

呼び出し経路ウィンドウ

呼び出し経路ウィンドウは、プログラムの現在の中断位置までの呼び出し経路（コールスタック）を表示します。

ウォッチウィンドウ

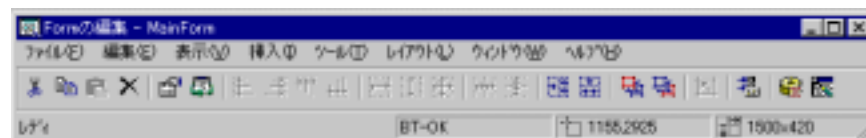
ウォッチウィンドウは、データの値の情報を表示します。

アウトプットウィンドウ

アウトプットウィンドウは、デバッグの状態やエラーなどを表示します。

3.1.2 フォーム編集ウィンドウ

フォーム編集ウィンドウは、フォームを編集する場合に利用します。フォーム編集ウィンドウは、メニューバーをもち、さらにそのメニュー項目の中から使用頻度が高い項目と対応づけられたボタンを、メニューバーの下ツールバーにもちます。また、その下のステータスバーには、メニュー項目、ツールバーのボタンの説明およびフォームやコントロールの名前、位置、大きさなどの情報が表示されます。



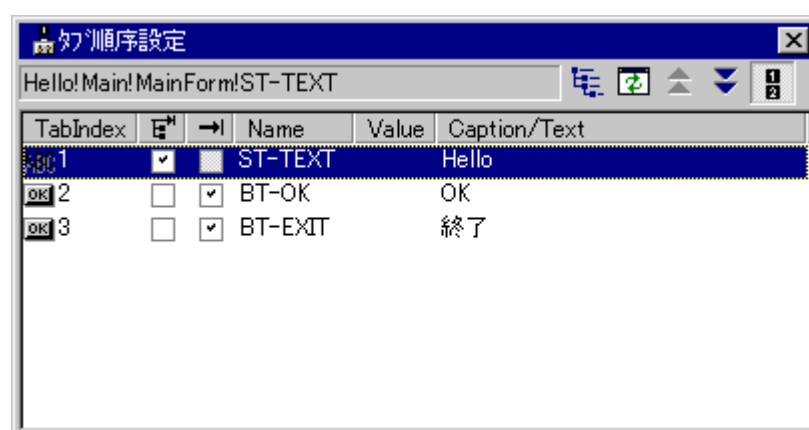
ツールボックス

ツールボックスは、フォームにコントロールを配置する場合に利用します。ツールボックスには、フォームに配置することができるコントロールに対応するボタンの一覧が表示されます。



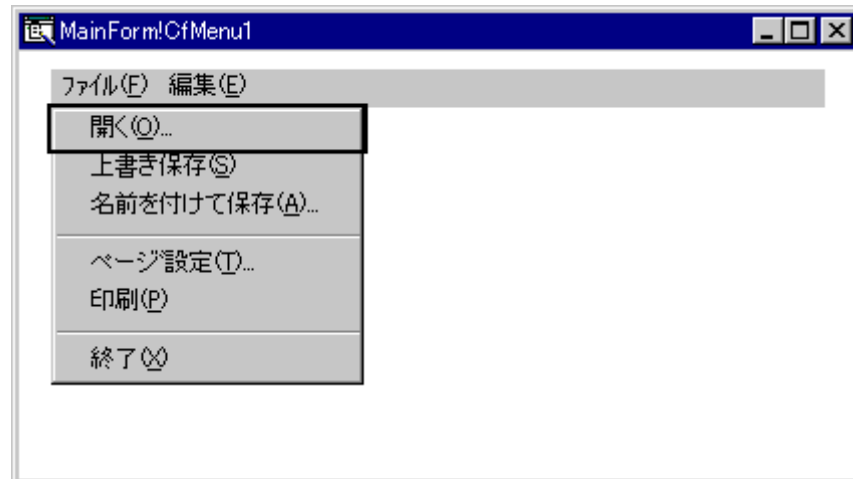
タブ順序設定ウィンドウ

タブ順序設定ウィンドウは、フォームに配置したコントロールをグループ分けしたり、[Tab]キーによって移動するフォーカスの順番を設定したりする場合に利用します。操作方法の詳細は「[タブ順序とタブグループを設定する](#) (p102) 」を参照してください。



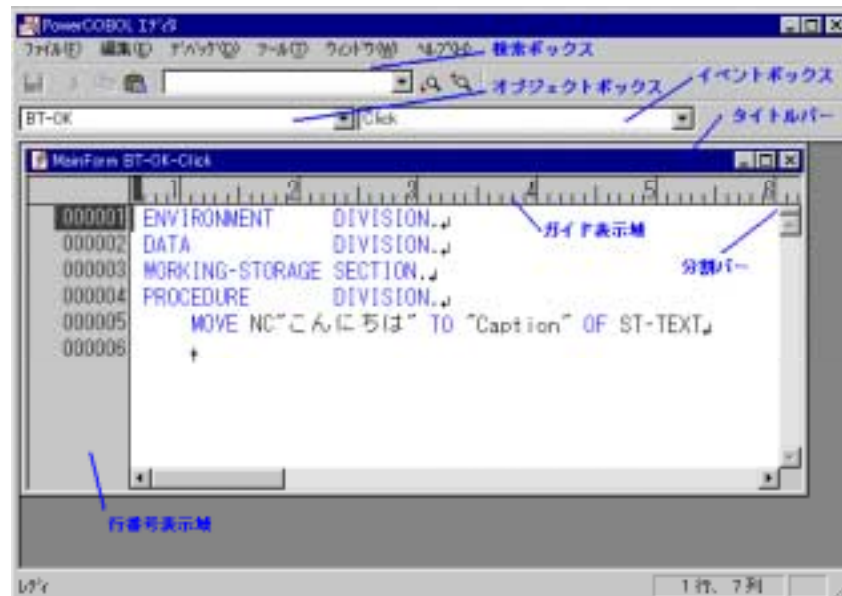
メニュー編集ウィンドウ

メニュー編集ウィンドウは、フォームのメニューを作成する場合に利用します。メニュー編集ウィンドウでは、メニューバーおよびポップアップメニューを作成することができます。操作方法の詳細は「[メニューを作成する](#) (p66)」を参照してください。



3.1.3 手続き編集ウィンドウ

手続き編集ウィンドウは、COBOLのプログラムやイベント手続きを編集するウィンドウです。手続き編集ウィンドウは、ツールバー、ステータスバーをもち、その中にイベント手続きごとに複数のウィンドウをもつことができます。操作方法の詳細は「[手続きを編集する](#) (p72)」を参照してください。



手続き編集ウィンドウの各部分の名称とその概要を以下に示します。

検索ボックス

検索ボックスに文字列を記述し、右側にあるボタンをクリックすることにより、文字列を検索できます。ただし、このボタンでは、単語単位の検索や大文字と小文字を区別した検索はできません。

オブジェクトボックスとイベントボックス

現在編集中のイベント手続きが属するフォームと、そのフォームがもつコントロールの一覧が表示されます。選択を変更すると、選択されたフォームまたはコントロールがもつイベント手続きの一覧が、イベントボックスに表示されます。オブジェクトボックスとイベントボックスを組み合わせることで、プロジェクトウィンドウやフォーム編集環境に戻らずに、特定のイベント手続きを開くことができます。

タイトルバー

イベント手続きのコントロール名およびイベント名が表示されます。

行番号表示域

行番号が表示されます。

ガイド表示域

カラムの目盛りが表示されます。

分割バー

分割バーをドラッグして移動することにより、ウィンドウが上下に分割され、イベント手続き内の2カ所を同時に編集することができます。分割を解除する場合は、分割バーをウィンドウの上限に移動するか、分割バーをダブルクリックしてください。

3.1.4 プロパティ設定ダイアログボックス

ダイアログボックスを開くと、いくつかの関連する機能や種類ごとにページが分類されている場合があります。これらの各ページをプロパティページといい、プロパティページをもつダイアログボックスをプロパティ設定ダイアログボックスといいます。

また、プロパティページの上部にあるしおりのような部分をタブといい、タブをクリックすることによりページを切り替えることができます。

プロパティ設定ダイアログボックスは、プロジェクト、モジュール、フォームやコントロールなどの、プロジェクト構成要素の属性、開発環境のオプションおよびデバッグ方法など、各種情報の設定をするために使用します。設定できる内容は、プロパティの設定対象により異なります。

プロパティ設定ダイアログボックスで設定または参照する項目の説明については、『リファレンス』を参照してください。



プロパティ設定ダイアログボックスの下部にあるボタンの使用方法を、以下に示します。

OKボタン

設定した内容を反映し、プロパティ設定ダイアログボックスを閉じます。

キャンセルボタン

設定した内容を反映しないで、プロパティ設定ダイアログボックスを閉じます。

適用ボタン

プロパティ設定ダイアログボックスを閉じずに、設定した内容を反映します。

ヘルプボタン

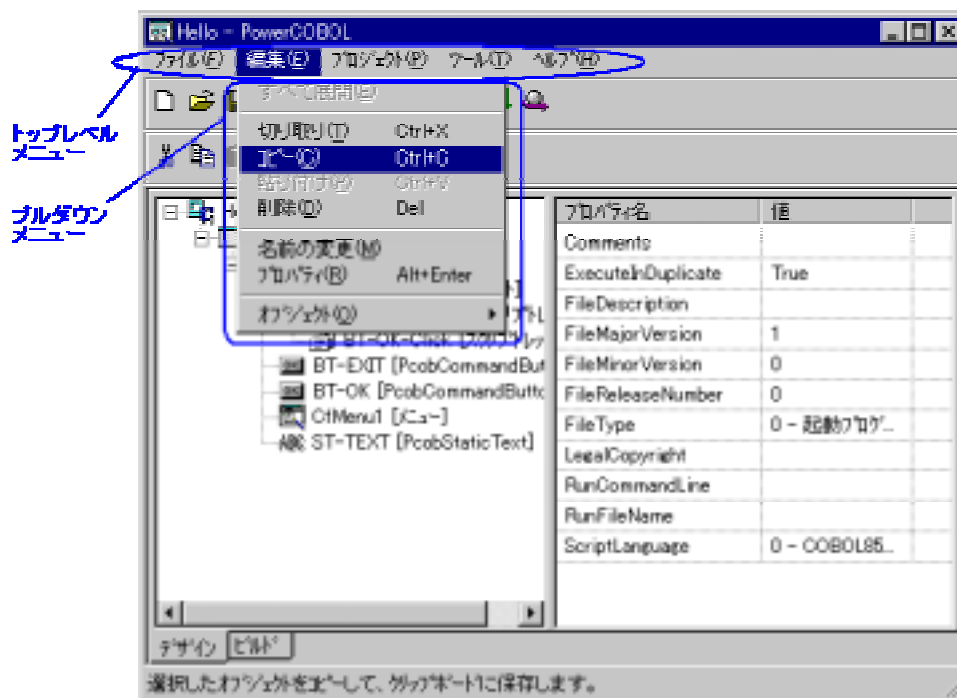
開いているプロパティページに対応する『リファレンス』中の説明を表示します。

3.2 その他の基本操作

本節では、PowerCOBOLでアプリケーションを作成する場合の、基本となるGUIの名称や、その操作方法について説明します。

3.2.1 メニューバー

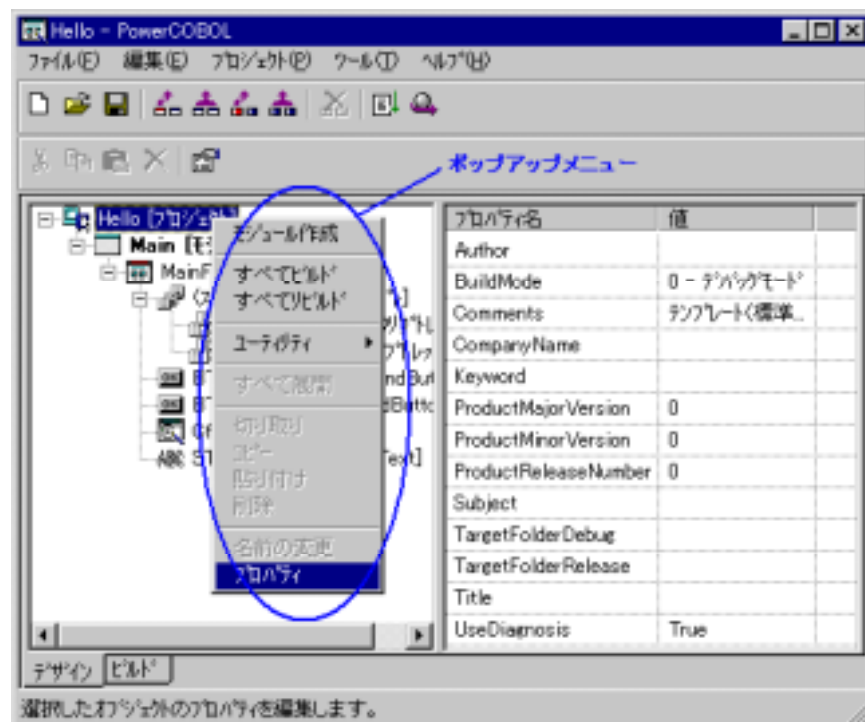
メニューバーは、ウィンドウの上部（タイトルの下）に、関連する機能ごとに分類され配置されています。メニューバーに配置された最上位階層にあるメニュー項目を、トップレベルメニューといいます。
トップレベルメニューからメニューを1つクリックすると、そのメニューのプルダウンメニューが表示されます。ウィンドウ中の現在の選択対象に対して、プルダウンメニュー中のコマンドを選択できます。
プルダウンメニュー中で選択できるメニュー項目は、選択対象によって変化します。たとえば、手続き編集ウィンドウでは、文字列が選択状態（反転表示）でなければ、[編集]メニューの[切り取り]や[コピー]コマンドを実行することはできません。



3.2.2 ポップアップメニュー

フォームや手続きを編集時にマウスの右ボタンをクリックすると、いくつかのメニューを配したポップアップメニューが表示されます。選択できるコマンド

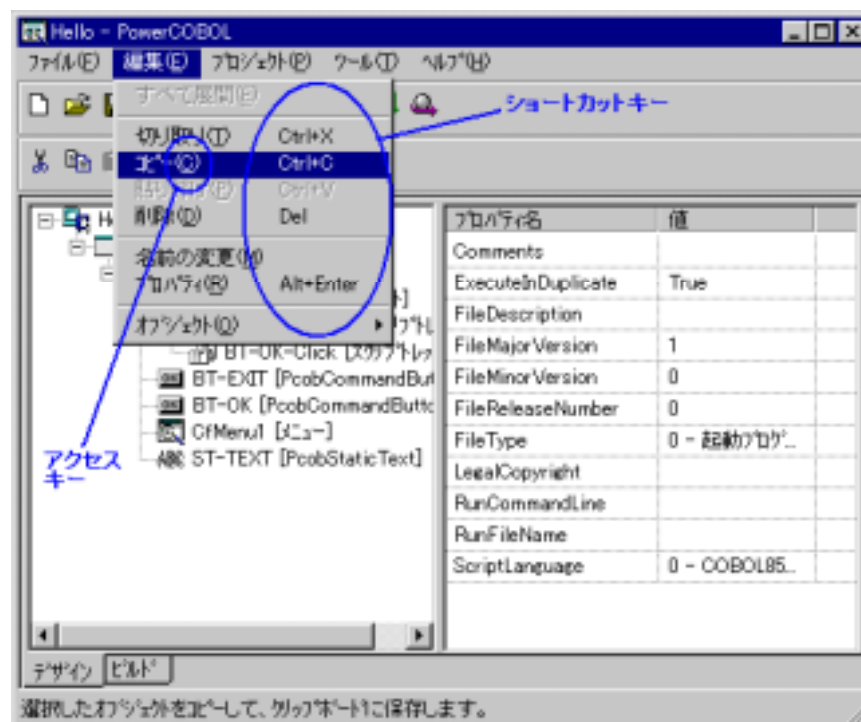
は、現在選択されている対象や、マウスポインタの位置によって変わります。



3.2.3 キーボード操作によるメニューコマンドの実行

キーボード操作によりメニューコマンドを実行するには、以下の2つの方法があります。

- ショートカットキーを使う方法
- アクセスキーを使う方法



ショートカットキーによる操作

メニューのプルダウンメニューを表示すると、メニューコマンドの右側に、キー（たとえば、[Ctrl+V]）が表記されている項目があります。これらのキーをショートカットキーといいます。ショートカットキーは、メニューを表示しないでコマンドを実行するためのキーです。

たとえば、手続き編集ウィンドウ上で文字列を選択している場合、その文字列をコピーするには、[編集]メニューの[コピー]コマンドを選択しますが、[Ctrl+C]キー（[Ctrl]キーと英字の[C]キーを同時に押す）でも同じことができます。

アクセスキーによる操作

メニューコマンドの文字列中には、下線が引かれた英字があります。これらのキーをアクセスキーといいます。アクセスキーは、[Alt]キーと同時に押すことにより、メニューバーのトップレベルメニューおよびその中のメニューコマンドを選択することができるキーです。

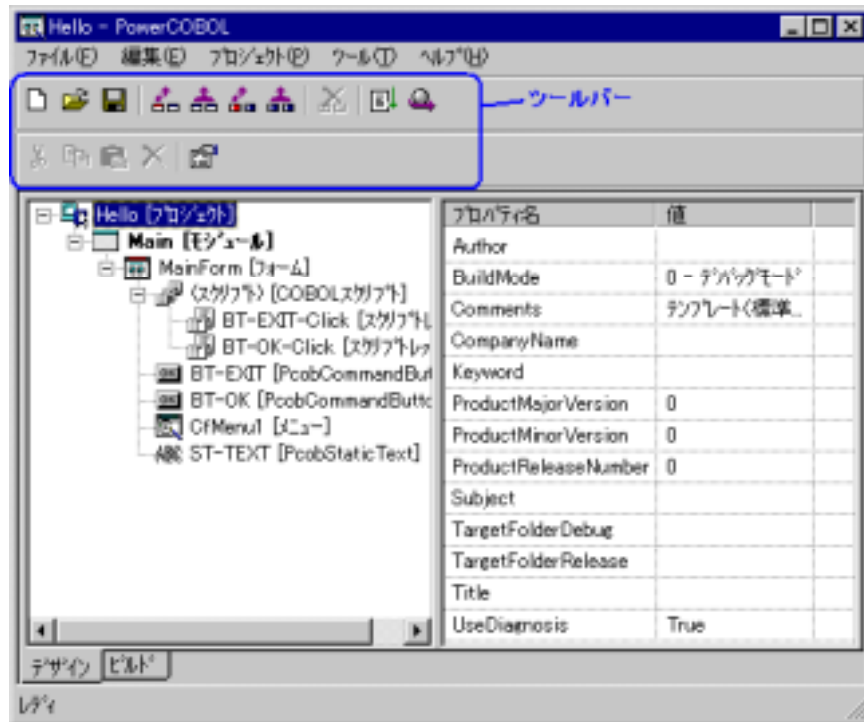
たとえば、手続き編集ウィンドウ上で文字列を選択している場合、[Alt]キーを押しながら、[E]キーと[C]キーを順に押すと、選択している文字列をコピーできます。

3.2.4 ツールバー

メニューバーの下に配置されたボタンの集まりをツールバーといいます。ツールバーには、メニューコマンドの中から使用頻度が高いものを選んで、ボタン

として配置されています。マウスポインタをボタン上に移動すると、各ボタンの意味がツールチップとして表示されます。

ツールバーは、[ツール] メニューの [オプション] コマンドで表示されるダイアログボックスで、[ツールバーを表示] のチェック状態を切り替えることにより、表示したり非表示にしたりできます。



3.2.5 キーボードによるプロパティ設定ダイアログボックスの操作

プロパティページ内の項目の操作

プロパティページ内の項目は、[Tab] キー、[Shift+Tab] キーまたは矢印キーでフォーカス（点線の枠）を移動したり設定を変更したりできます。

また、各項目につけられたアクセスキー（下線がついた英字）を使って、移動や設定ができます。

プロパティページの切り替え

プロパティページは、[Tab] キーまたは[Shift+Tab] キーで、タブ部分にフォーカスを移動し、矢印キーを使って切り替えることができます。

MEMO

第2部 開発環境編

本部では、PowerCOBOLを使ってアプリケーションを作成する方を対象に、PowerCOBOL開発環境がもつ機能や、その操作方法について説明します。

作成するサンプルプログラムは、以下の機能を持ちます。

入力日付が表示されます。

表に個数を入力すると、小計が表示されます。

入力内容を印刷できます。

すでに保存済のファイルを開き、更新

7-11(E) 7-11(E)

・購入商品の入力画面

ファイル(F) 編集(E)

顧客名 富士商店 2003年04月01日

	コード	商品名	単価	個数	小計	備考
1	00011	ブドウ	¥800	5	¥4,800	
2	00006	ナシ	¥240	12	¥2,880	
3	00009	モモ	¥220	12	¥2,640	
4	00017	甘露きかん	¥240	9	¥2,160	
5						
6						
7						
8						
9						
10						

このサンプルプログラムは、以下の手順で作成します。

1. [プロジェクトの作成](#) (p51)
新規にプロジェクトを作成します。
プロジェクトのプロパティを設定します。
2. [フォームの編集](#) (p55)
フォーム編集ウィンドウを表示します。
フォームのプロパティを設定します。
フォームにコントロールを配置します。
コントロールのプロパティを設定します。
コントロールの位置やサイズを調整します。
コントロールの色を変更します。
コントロールの文字のフォントを変更します。
3. [メニューの作成](#) (p66)
メニュー編集ウィンドウを表示します。
メニュー項目を編集します。
メニューのプロパティを設定します。
4. [手続きの編集](#) (p72)
手続き編集ウィンドウを表示します。
イベント手続きを記述します。
5. [ビルドと実行](#) (p88)
プロジェクトを保存します。
モジュールをビルドして実行可能プログラムを作成します。
エラーがあった場合は、エラー箇所を修正します。
作成した実行可能プログラムを実行します。
PowerCOBOLを終了します。

本章でこれから作成していくサンプルプログラムは、"Table¥Table1.ppj"に格納されています。必要に応じて参照してください。

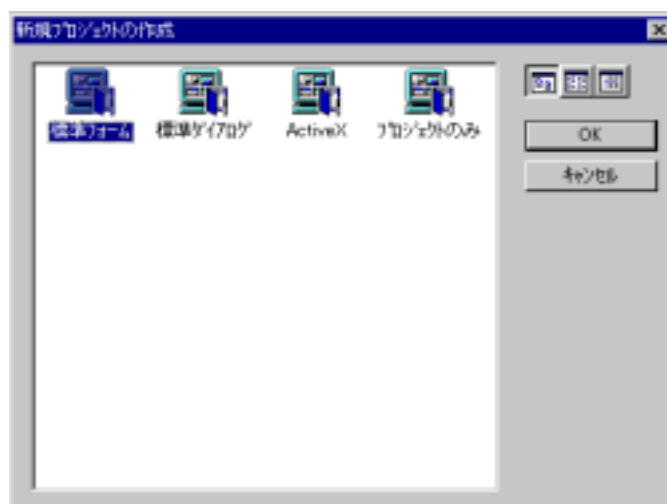
4.1 プロジェクトを作成する

本節では、プロジェクトの作成方法、編集方法、およびプロジェクトとモジュールのプロパティの設定方法について説明します。

4.1.1 新規にプロジェクトを作成する

新規にプロジェクトを作成するには、[ファイル] メニューの [新規プロジェクトの作成] コマンドを選択し、[新規プロジェクトの作成] ダイアログボックスから、これから作成するプロジェクトの形式を選択します。本章で作成するサンプルプログラムでは、[標準フォーム] を選択してください。

[新規プロジェクトの作成] ダイアログボックスで選択できるプロジェクトの形式の詳細は、『リファレンス』を参照してください。



ツールバーの [新規標準フォームの作成] ボタンをクリックすることにより、[標準フォーム] と同じ形式のプロジェクトを作成できます。この場合、[新規プロジェクトの作成] ダイアログボックスは表示されません。

プロジェクトを保存する場合は、[ファイル] メニューの [プロジェクトの上書き保存] または [名前を付けてプロジェクトの保存] コマンドを選択してください。保存したプロジェクトは、[既存プロジェクトを開く] コマンドで開けます。また、開いた日付が新しい順に、[ファイル] メニューの編集履歴 ([PowerCOBOLの終了] コマンドの上の部分) に追加されていくので、[ファイルを開く] ダイアログボックスを経由しないで、プロジェクトを開くこともできます。

4.1.2 プロジェクトのプロパティを設定する

プロジェクトのプロパティは、プロパティ設定ダイアログボックスまたはプロパティリストウィンドウで値を入力することにより設定できます。以下の手順で、プロジェクトのプロパティを設定します。

1. プロジェクトウィンドウのデザインツリーウィンドウで"無題 [プロジェクト]"を選択します。
2. ポップアップメニューの [プロパティ] コマンドを選択します。
3. 以下のように、[概要] タブの情報を設定します。

4. OKボタンをクリックします。



ここで設定したプロジェクトのプロパティは、アプリケーションの動作には影響ありませんが、将来の保守に備えて設定しておくことをお勧めします。また、個々の設定項目およびプロパティリストウィンドウのプロパティ名との対応については、『リファレンス』を参照してください。

4.1.3 プロジェクトの構成要素を編集する

PowerCOBOLでは、デザインツリーウィンドウを使って、プロジェクトを構成する各要素を作成、削除、コピー、または移動できます。たとえば、以下のような操作ができます。

プロジェクト内に複数のモジュールを作成できます。
不要になった手続きやコントロールを削除できます。
モジュール内のフォームを、別のモジュールにコピーしたり移動したり
できます。

プロジェクトの構成要素を作成する

デザインツリーウィンドウでは、以下の構成要素を作成することができます。

モジュール
フォーム
メニュー

これらの構成要素は、以下のようにして作成します。

モジュール

1. デザインツリーウィンドウのプロジェクトを選択します。
2. ポップアップメニューまたは[編集 - オブジェクト]メニューの[モジュール作成]コマンドを選択します。

フォーム

1. デザインツリーウィンドウのモジュールを選択します。
2. ポップアップメニューまたは[編集 - オブジェクト]メニューの[フォーム作成]コマンドを選択します。

メニュー

1. デザインツリーウィンドウのフォームを選択します。
2. ポップアップメニューまたは[編集 - オブジェクト]メニューの[メニュー編集]サブメニューから[メニュー作成]コマンドを選択します。

プロジェクトの構成要素を削除する

デザインツリーウィンドウでは、ポップアップメニューまたは[編集]メニューの[削除]コマンドを使って、構成要素を削除できます。

プロジェクトの構成要素をコピーまたは移動する

デザインツリーウィンドウでは、以下の手順により、構成要素をコピーまたは移動できます

1. デザインツリーウィンドウで、コピーまたは移動する構成要素を選択します。
2. 以下のどちらかのコマンドを選択します。
ポップアップメニューまたは[編集]メニューの[コピー]コマンド
ポップアップメニューまたは[編集]メニューの[切り取り]コマンド
3. デザインツリーウィンドウで、コピーまたは移動先の構成要素を選択します。たとえば、フォームをコピーする場合は、そのフォームがコピーされるモジュールを選択します。
4. ポップアップメニューまたは[編集]メニューの[貼り付け]コマンドを選択します。



ドラッグアンドドロップにより、構成要素をコピーすることもできます。その場合、コピーする構成要素をドラッグし、コピー先の構成要素上でドロップしてください。ドラッグ中にキャンセルする場合は、[Esc]キーを押してください。



注意

構成要素をコピーまたは移動する場合、以下のような注意事項があります。

移動先に同名の構成要素があった場合、名前は自動的に変更されます。

コントロールをコピーまたは移動しても、そのコントロールに対応するイベント手続き（スクリプト）はコピーまたは移動されません。したがって、コントロールを別のフォームに移動した場合、イベント手続きが失われます。

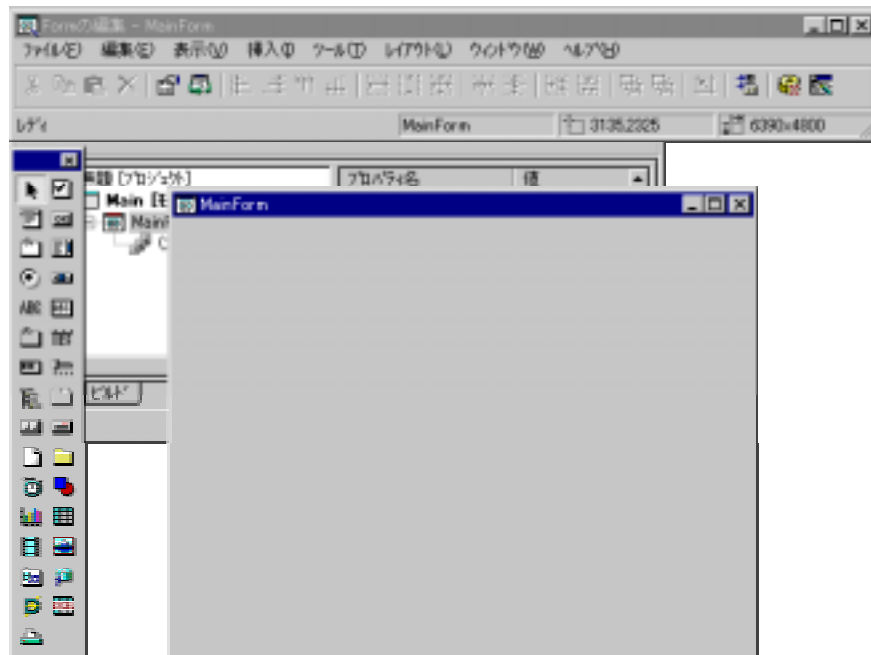
イベント手続き（デザインツリーウィンドウの"（スクリプト）[COBOL スクリプト]"）のコピーまたは移動はできません。

4.2 フォームを編集する

本節では、フォームの編集方法、コントロールの配置方法およびフォームとコントロールのプロパティの設定方法について説明します。

4.2.1 フォーム編集ウィンドウを表示する

デザインツリーウィンドウの"MainForm [フォーム]"を選択状態にし、ポップアップメニューの [開く] コマンドを選択します。



4.2.2 フォームのプロパティを設定する

フォームを開いたばかりの状態では、フォームのキャプション(タイトル)は、"MainForm"と表示されています。フォームのプロパティを変更することにより、表示されている文字列を変更できます。プロパティは、以下の手順で変更します。

1. フォーム上でマウスの右ボタンをクリックし、ポップアップメニューを表示します。
2. ポップアップメニューから、[プロパティ] コマンドを選択します。
[MainFormのプロパティ] ダイアログボックスが表示されます。
3. [キャプション] を "MainForm" から "購入商品の入力画面" に変更します。



4. OKボタンをクリックします。



フォーム、および以降で説明するコントロールやメニューのプロパティ設定ダイアログボックスは、プロジェクトウィンドウのデザインツリーウィンドウで設定対象を選択し、ポップアップメニューの[プロパティ]コマンドを選択することによって表示することもできます。必要に応じて使い分けてください。

4.2.3 フォームにコントロールを配置する

コントロールは、ツールボックスのボタンまたは[挿入]メニューの[コントロールの挿入]コマンドを選択し、フォーム上の任意の位置でクリックすることにより、配置することができます。

このサンプルプログラムでは、以下のコントロールを配置して利用します。

スタティックテキスト(StaticText)コントロール

CmStatic1: 「顧客名」という文字列を表示します。

CmStatic2: 日付を表示します。

CmStatic3: 合計金額を表示します。

テキストボックス(TextBox)コントロール

CmText1: 顧客名を入力します。

コマンドボタン(CommandButton)コントロール

CmCommand1: 合計金額を計算する場合にクリックします。

表(Table)コントロール

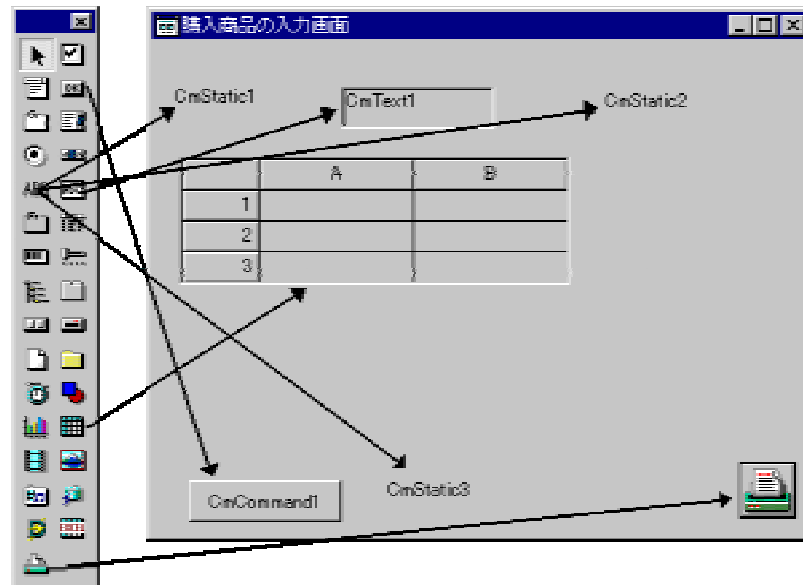
CmTable1: 購入製品や個数を表示および入力します。

印刷(Print)コントロール

CmPrint1: 購入製品の一覧表を印刷するために使用します。

これらのコントロールを以下のように配置します。このとき、各コントロールの位置はプロパティを設定したあとに調整するため、正確でなくてもかまいま

せん。また、表コントロールは、プロパティ設定時の表示が確認しやすくなるよう、サイズを大きくしておいてください。



コントロールは、フォーム編集ウィンドウの[編集]メニューまたはポップアップメニューから、以下のコマンドを選択して、コピーや削除ができます。

- [切り取り]： 選択中のコントロールをクリップボードにコピーし、フォームから削除します。
- [コピー]： 選択中のコントロールをクリップボードにコピーします。
- [貼り付け]： クリップボードにコピーしたコントロールを、フォームの左上端に配置します。
- [削除]： 選択中のコントロールをフォームから削除します。



[Ctrl]キーを押したまま、コントロールをドラッグすることにより、フォーム上の任意の位置へ複写することもできます。この操作は、上記の[コピー]と[貼り付け]をした場合の動作と同様であり、同じスタイルをもつコントロールを複製する場合に利用します。ただし、コントロールのイベント手続きは、コピーされません。

4.2.4 コントロールのプロパティを設定する

各コントロールのプロパティを、以下のように設定します。各コントロールのプロパティ設定ダイアログボックスの詳細は、『リファレンス』を参照してく

ださい。

CmStatic1

表示する文字列と名前を設定します。

1. [キャプション] を "CmStatic1" から "顧客名" に変更します。
2. [テキスト配置(縦)] を中央寄せにします。
3. [共通] タブをクリックします。
4. [名前] を "CmStatic1" から "ST-CUSTOMER" に変更します。
5. OKボタンをクリックします。



このコントロールは、後述の手続き中で使用することはありません。したがって、手続きの読みやすさのために名前を変更する必要はありませんが、デザインツリーウィンドウでの視認性や、タブ順序設定ウィンドウで[タブ順序とタブグループを設定する](#) (p102) 場合の見やすさを考慮し、名前を変更しています。

CmStatic2

日付が表示されるよう、プロパティを設定します。

1. [キャプション] に設定されている "CmStatic2" を削除します。
2. [テキスト配置(横)] を中央寄せにします。
3. [テキスト配置(縦)] も中央寄せにします。
4. [テキスト属性] タブをクリックします。
5. [テキスト種別] を "0 - 標準" から "2 - 日付属性" に変更します。
6. [現在日付使用] がチェックされていることを確認します。
7. [共通] タブをクリックします。
8. [名前] を "CmStatic2" から "ST-DATE" に変更します。
9. OKボタンをクリックします。



このコントロールは、日付を表示するために使用します。PowerCOBOLでは、テキスト種別を変更することで、表示される文字列の形式をあらかじめ指定しておくことができます。また、現在日付を使用することで、このアプリケーションが実行された日付を自動的に表示させることができます。したがって、テキスト種別を変更する以前に、日付の形式と矛盾する文字列 "CmStatic2" を削除しています。

CmStatic3

合計金額が表示できるよう、プロパティを設定します。

1. [キャプション] に設定されている "CmStatic3" を削除します。
2. [テキスト配置(横)] を中央寄せにします。
3. [テキスト配置(縦)] も中央寄せにします。
4. [テキスト属性] タブをクリックします。

5. [テキスト種別] を "0 - 標準" から "1 - COBOL PICTURE属性" に変更します。
6. [PICTURE文字列] が "###,###,##9" であることを確認します。
7. [外観] タブをクリックします。
8. [枠] を "0 - なし" から "1 - 実線" に変更します。
9. [共通] タブをクリックします。
10. [名前] を "CmStatic3" から "ST-TOTAL" に変更します。
11. OKボタンをクリックします。



ワンポイント

このコントロールは、金額を表示するために使用します。PowerCOBOLでは、テキスト種別を "1 - COBOL PICTURE属性" にすることで、表示される文字列の形式をあらかじめ指定しておくことができます。この属性を指定しておくことにより、設定した数値を自動的に変換し、指定した形式で表示させることができます。この場合も、日付属性を指定した場合と同様に、表示文字列に矛盾が生じないよう、文字列 "CmStatic3" を削除しています。

CmText1

顧客名を入力するために使用します。サンプルプログラムでは、顧客名の最大長が50バイトになるようプロパティを設定します。

1. [テキスト] に設定されている "CmText1" を削除します。
2. [テキスト属性] タブをクリックします。
3. [テキスト種別] を "0 - 標準" から "1 - COBOL PICTURE属性" に変更します。
4. [PICTURE文字列] を "###,###,##9" から "X(50)" に変更します。
5. [共通] タブをクリックします。
6. [名前] を "CmText1" から "TX-CUSTOMER" に変更します。
7. OKボタンをクリックします。

CmCommand1

ボタンに表示する文字列と名前を設定します。

1. [キャプション] を "CmCommand1" から "合計" に変更します。
2. [共通] タブをクリックします。
3. [名前] を "CmCommand1" から "BT-TOTAL" に変更します。
4. OKボタンをクリックします。



注意

コマンドボタンコントロールのキャプションには、複数行の文字列を設定できます。したがって、"合計" と入力した直後に [Enter] キーを押すと、次の行に入力位置が移るため、文字列 "合計" は、消えてしまったように見えますが、カーソルで移動すれば、見ることはできます。この場合、1行におさまるよう不要な部分を削除してください。

CmPrint1

印刷コントロールのプロパティでは、用紙サイズ、印刷方向や余白の大きさなどを指定できます。このサンプルプログラムでは、すべて初期値を使用するので、印刷コントロールのプロパティは設定しません。

CmTable1

購入する商品の一覧表となるよう、プロパティを設定します。表コントロールのプロパティは、列ごとに設定できます。このサンプルプログラムでは、商品の「コード」、「名前」、「単価」、「個数」、「小計」および「備考」の6個の情報を表示するため、6列分のプロパティを以下のように設定します。また、一覧表に入力できる商品の件数は最大20件とします。

1. [行数] を3から20に変更します。
2. [列数] を2から6に変更します。
3. [列スタイル] タブをクリックします。
4. [列リスト] から"A"を選択し、以下のように1列めのプロパティを設定します。
 - [列見出し] を"A"から"コード"に変更します。
 - [IMEモード] を"8 - 半角英数"に変更します。
5. [列リスト] から"B"を選択し、以下のように2列めのプロパティを設定します。
 - [列見出し] を"B"から"商品名"に変更します。
 - [配置] を"4 - 左寄せ 中央寄せ"に変更します。
 - [編集可能] のチェックをはずします。
6. [列リスト] から"C"を選択し、以下のように3列めのプロパティを設定します。
 - [列見出し] を"C"から"単価"に変更します。
 - [編集可能] のチェックをはずします。
7. [列リスト] から"D"を選択し、以下のように4列めのプロパティを設定します。
 - [列見出し] を"D"から"個数"に変更します。
 - [IMEモード] を"8 - 半角英数"に変更します。
8. [列リスト] から"E"を選択し、以下のように5列めのプロパティを設定します。
 - [列見出し] を"E"から"小計"に変更します。
 - [編集可能] のチェックをはずします。
9. [列リスト] から"F"を選択し、以下のように6列めのプロパティを設定します。
 - [列見出し] を"F"から"備考"に変更します。
 - [配置] を"4 - 左寄せ 中央寄せ"に変更します。
 - [IMEモード] を"4 - 全角ひらがな"に変更します。
10. [テキスト属性] タブをクリックします。
11. [プロパティ名] から"TableColumn(1).RenderText"を選択し、以下のように1列めのテキスト属性を設定します。
 - [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。

- [PICTURE文字列] を"9(5)"に変更します。これにより、コードは5桁の数値となります。
12. [プロパティ名] から"TableColumn(2).RenderText"を選択し、以下のよう
 2列めのテキスト属性を設定します。
 [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。
 [PICTURE文字列] を"N(10)"に変更します。これにより、商品名は10文字以下の全角文字となります。
13. [プロパティ名] から"TableColumn(3).RenderText"を選択し、以下の
 3列めのテキスト属性を設定します。
 [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。
 [PICTURE文字列] を"¥¥¥,¥¥9"に変更します。これにより、単価の最大値は¥99,999となります。
14. [プロパティ名] から"TableColumn(4).RenderText"を選択し、以下の
 4列めのテキスト属性を設定します。
 [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。
 [PICTURE文字列] を"Z,ZZ9"に変更します。これにより、個数の最大値は9999となります。
15. [プロパティ名] から"TableColumn(5).RenderText"を選択し、以下の
 5列めのテキスト属性を設定します。
 [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。
 [PICTURE文字列] は、そのまま使用します。
16. [プロパティ名] から"TableColumn(6).RenderText"を選択し、以下の
 6列めのテキスト属性を設定します。
 [テキスト種別] を"1 - COBOL PICTURE属性"に変更します。
 [PICTURE文字列] を"N(20)"に変更します。
17. 再度 [列スタイル] タブをクリックします。
18. [列リスト] から"行の見出し"を選択し、[幅] を350に変更します。ここで、ダイアログボックスの適用ボタンをクリックし、フォーム上の表の1列めの幅が、表示内容に合っているかどうかを確認します。表示内容と比較し、幅が大きいまたは小さい場合は、再度[幅] の値を変更し、調整します。
19. [列リスト] から"行の見出し"以降の各列の [幅] も同様の方法で変更してください。設定する値の目安は、1列めから順に、600、1500、750、600、1000、2850程度です。
20. [共通] タブをクリックします。
21. [名前] を"CmTable1"から"TBL-PURCHASE"に変更します。
22. OKボタンをクリックします。



[IMEモード] を設定している列は、実行時にキーボードからの入力が可能な列です。また、キーボードからの入力が不要な列では、[編集可能] のチェックをはずし、入力できないように設定しています。

この時点で、以下のようなフォームが作成できていることを確認します。

	コード	商品名	単価	個数	小計	備考
1	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
2	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
3	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
4	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
5	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
6	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
7	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN

4.2.5 コントロールの位置とサイズを調整する

フォームおよびコントロールの位置とサイズが以下になるよう、調整してください。

	コード	商品名	単価	個数	小計	備考
1	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
2	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
3	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
4	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
5	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
6	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
7	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
8	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
9	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN
10	99999	NNNNNNNNNN	¥¥¥,¥¥¥	2,229	¥¥¥,¥¥¥,¥¥¥	NNNN

コントロールは、ドラッグ操作によりフォーム上の任意の位置へ移動することができます。移動中にキャンセルする場合は、[Esc]キーを押します。また、コントロールを選択し、周囲に表示された四角点をドラッグすることにより、サイズを変更できます。

4.2.6 コントロールの色を変更する

フォームやコントロールの色もプロパティの一種です。これにより、色についてもプロパティ設定ダイアログボックスを使って設定できます。

ここでは、日付を表示するスタティックテキストコントロール(ST-DATE)の色

を変更します。

1. ST-DATEのプロパティ設定ダイアログボックスを表示します。
2. [カラー] タブをクリックします。
3. [プロパティ名] が "BackColor" になっていることを確認します。[プロパティ名] では以下の項目を選択できます。

BackColor: 背景色を指定する場合に選択します。

ForeColor: 文字の色を指定する場合に選択します。

HighlightColor: 文字を強調表示するための文字背景色を指定する場合に選択します。

4. [カラー種別] を "標準色" に変更します。[カラー種別] では以下の項目を選択できます。

標準色: 標準の16色から色を指定する場合に選択します。

システム色: システムが使用している色を指定する場合に選択します。

ユーザ定義色: 新しく作成した色を指定する場合に選択します。

5. [カラーリスト] から [白] を選択します。
6. 同様に、[プロパティ名] で "ForeColor"、[カラー種別] で "標準色" を選択し、[カラーリスト] から [赤] を選択します。
7. OKボタンをクリックします。



[カラー種別] で "ユーザ定義色" を選択して利用できる色は、[色の作成] ダイアログボックスで作成できます。[色の作成] ダイアログボックスは、[カラー] タブの中にある [カスタムカラー] ボタンをクリックすると表示されます。



[プロパティ名]で"ForeColor"を選択し、色を指定しても、[テキスト属性]タブの[テキスト種別]が"1 - COBOL PICTURE属性"の場合、文字の色は淡色表示のまま変わりません。指定した色は、実行時にだけ使用されます。

[カラー種別]で"システム色"を選択した場合、コントロールの色は、アプリケーションを実行するWindows の環境によって、異なります。

4.2.7 コントロールの文字のフォントを変更する

コントロールに表示する文字のフォントもプロパティの一種です。したがって、フォントについてもプロパティ設定ダイアログボックスを使って設定できます。ここでは、顧客名を入力するテキストボックスコントロール(TX-CUSTOMER)のフォントを変更します。

1. TX-CUSTOMERのプロパティ設定ダイアログボックスを表示します。
2. [フォント]タブをクリックします。
3. [フォント]が"MS Pゴシック"になっていることを確認します。
4. [サイズ]を10に変更します。
5. OKボタンをクリックします。

コード	商品名	単価	数量	小計	備考
1	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
2	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
3	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
4	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
5	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
6	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
7	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
8	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
9	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX
10	XXXXXXXXXXXXXXXXXXXX	1000	1.229	1000.000	XXXXXXXXXXXXXXXXXXXX



コントロールの色やフォントなど、フォームがもっているプロパティと同じ内容のプロパティの初期値は、フォームのプロパティと同じ値になります。たとえば、フォームの背景色(BackColor)を黄色にした場合、以降に配置されるコントロールの背景色の初期値は、黄色になります。



注意

[フォント] の選択肢には、Windows にインストールされているすべてのフォントが表示されます。個別にインストールしたフォントを指定し、アプリケーションを作成した場合、指定したフォントがインストールされていない Windows 上で、そのアプリケーションを実行すると、指定したものと異なったフォントで表示される場合があります。

4.2.8 フォーム編集ウィンドウを閉じる

フォームの編集が終了したら、フォーム編集ウィンドウを閉じます。フォーム編集ウィンドウは、以下のどれかの方法で閉じることができます。

フォーム編集ウィンドウの [ファイル] メニューから [閉じる] コマンドを選択します。

フォーム編集ウィンドウの [ファイル] メニューから [終了して戻る] コマンドを選択します。

フォーム編集ウィンドウの [閉じる] ボタン (右上の x ボタン) をクリックします。

編集中のフォームの [閉じる] ボタン (右上の x ボタン) をクリックします。



[終了して戻る] コマンドは、以下の場合に、フォーム編集ウィンドウ、すべてのフォーム、およびメニュー編集ウィンドウを閉じ、プロジェクトウィンドウに戻るときに使用します。

複数のフォームを同時に開いている

複数のメニュー編集ウィンドウを開いている

複数のフォームとメニュー編集ウィンドウを混在して開いている

[閉じる] コマンドは、以下の場合に使用します。

編集中の1つのフォーム (フォーカス持っているフォーム) を閉じる

メニュー編集ウィンドウだけを閉じる

4.3 メニューを作成する

本節では、フォームにメニューを作成し、編集する方法について説明します。



本章では、メニュー編集ウィンドウを使って、フォームの上部に表示されるメニューバーを作成します。

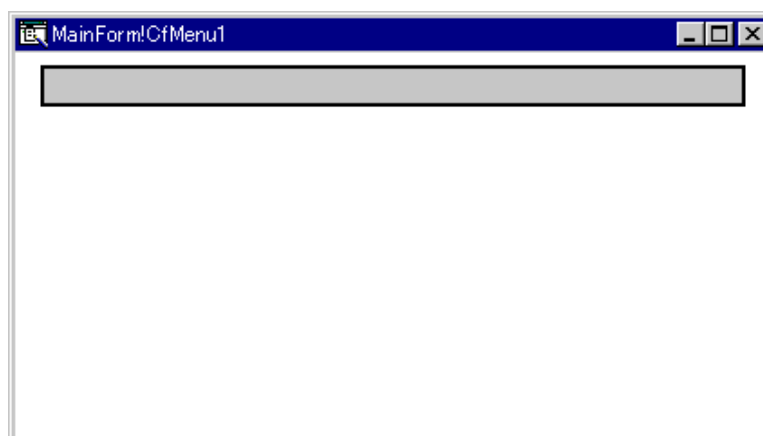
PowerCOBOLでは、マウスの右ボタンをクリックして表示されるポップアップメニューも作成できます。ポップアップメニューの作成方法については、「[ポップアップメニューを使ったアプリケーションを作成する](#) (p219)」を参照してください。

4.3.1 メニュー編集ウィンドウを開く

メニュー編集ウィンドウは、以下の手順で表示できます。このとき、フォーム編集ウィンドウも表示されます。

1. デザインツリーウィンドウの"MainForm [フォーム]"を選択します。
2. ポップアップメニューの[メニュー編集]サブメニューから[メニュー作成]コマンドを選択します。

以下のようなメニュー編集ウィンドウが表示されます。





メニュー編集ウィンドウは、以下のどちらかの方法でも表示できます。

フォーム編集ウィンドウの[ツール]メニューの[メニュー編集]コマンドを選択し、[メニュー編集]ダイアログボックスで、[新規作成]を選択します。

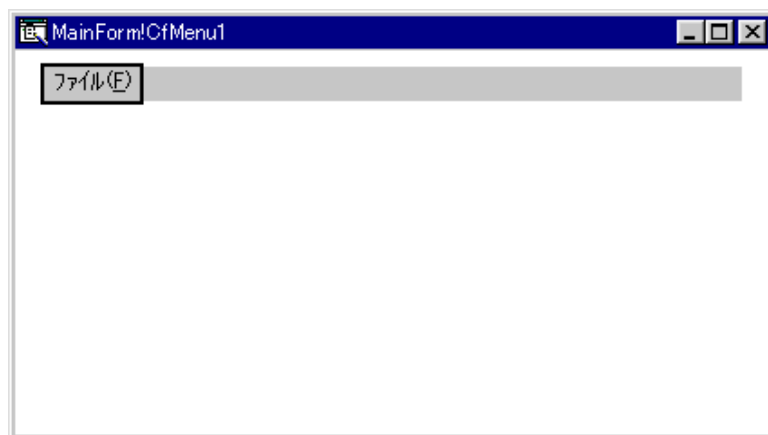
フォーム上のコントロールが配置されていない任意の位置を右クリックし、ポップアップメニューの[メニュー編集]サブメニューから[メニュー作成]コマンドを選択します。

すでに作成済のメニューを編集する場合は、[メニュー編集]サブメニューの[メニュー作成]の下方にメニュー名の一覧が表示されるので、その中から編集対象のメニューを選択してください。また、不要になったメニューを削除する場合は、プロジェクトウィンドウのデザインツリーウィンドウを使用してください。

4.3.2 メニュー項目を追加する

メニューを作成しただけでは、メニューの中にはメニュー項目は存在しないので、順次追加していきます。以下の手順で、基点となるメニュー項目を作成します。

1. メニュー編集ウィンドウのポップアップメニューから[追加]コマンドを選択します。
2. メニューのプロパティ設定ダイアログボックスの[キャプション]に"ファイル(&F)"と入力します。
3. [名前]を"MN-FILE"に変更します。
4. OKボタンをクリックします。
“ファイル”メニューが追加されます。



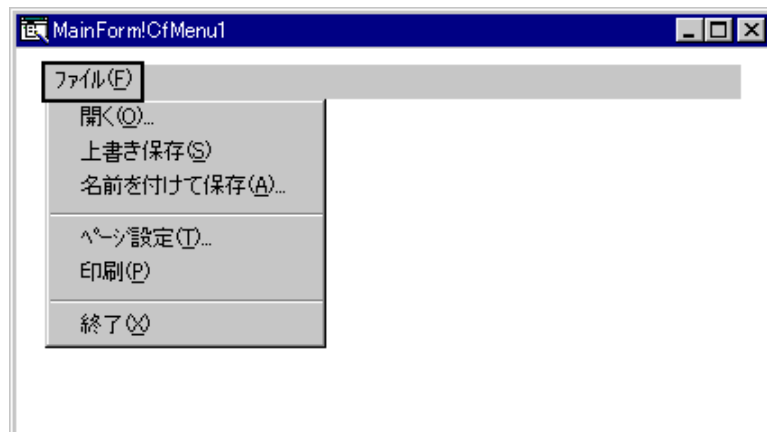
メニューは、選択状態であるメニュー項目のポップアップメニューに表示される、以下のコマンドを使って作成します。

- [追加] : 選択状態のメニュー項目と並列（兄弟）関係となるメニュー項目を、直後に追加します。
- [挿入] : 選択状態のメニュー項目と並列（兄弟）関係となるメニュー項目を、直前に追加します。
- [子メニュー作成] : 選択状態のメニュー項目の子メニューを作成します。

続けて、以下の手順で、このサンプルプログラムにメニューを追加していきます。

1. "ファイル"メニューに、子メニュー"開く"を追加します。
メニュー編集ウィンドウ上の"ファイル"を選択します。
ポップアップメニューの[子メニュー作成]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"開く(&O)..."と入力します。
[名前]を"MN-OPEN"に変更します。
OKボタンをクリックします。
2. "開く"の下に、"上書き保存"を追加します。
メニュー編集ウィンドウ上の"開く..."を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"上書き保存(&S)"と入力します。
[名前]を"MN-SAVE"に変更します。
OKボタンをクリックします。
3. 同様に、"名前を付けて保存"を追加します。
メニュー編集ウィンドウ上の"上書き保存"を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"名前を付けて保存(&A)..."と入力します。
[名前]を"MN-SAVEAS"に変更します。
OKボタンをクリックします。
4. セパレータをはさんで、"ページ設定"を追加します。
メニュー編集ウィンドウ上の"名前を付けて保存..."を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[セパレータ]をチェック状態にします。
OKボタンをクリックします。
作成したセパレータ(横棒で表示されている部分)を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"ページ設定(&T)..."と入力します。

- [名前] を "MN-PAGESETUP" に変更します。
OKボタンをクリックします。
5. さらに、"印刷"を追加します。
メニュー編集ウィンドウ上の"ページ設定..."を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"印刷(&P)"と入力します。
[名前] を "MN-PRINT"に変更します。
OKボタンをクリックします。
6. セパレータをはさんで、"終了"を追加します。
メニュー編集ウィンドウ上の"印刷"を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[セパレータ]
をチェック状態にします。
OKボタンをクリックします。
作成したセパレータ(横棒で表示されている部分)を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"終了(&X)"と入力します。
[名前] を "MN-EXIT"に変更します。
OKボタンをクリックします。



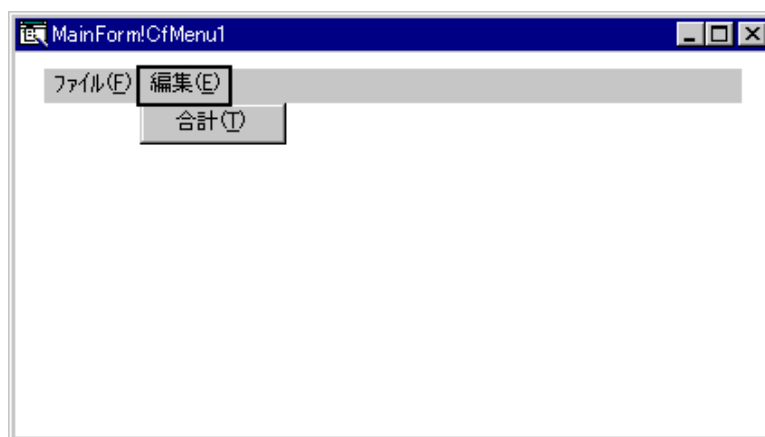
7. "ファイル"メニューの右側に、"編集"メニューを追加します。
メニュー編集ウィンドウ上の"ファイル"を選択します。
ポップアップメニューの[追加]コマンドを選択します。
メニューのプロパティ設定ダイアログボックスの[キャプション]に"編集(&E)"と入力します。
[名前] を "MN-EDIT"に変更します。
OKボタンをクリックします。
8. "編集"メニューの下に、"合計"を追加します。
メニュー編集ウィンドウ上の"編集"を選択します。

ポップアップメニューの[子メニュー作成]コマンドを選択します。

メニューのプロパティ設定ダイアログボックスの[キャプション]に"合計(&T)"と入力します。

[名前]を"MN-TOTAL"に変更します。

OKボタンをクリックします。



"&F"や"&0"のように、"&"に続けて1文字の英字あるいは数値を指定することにより、その指定した英字あるいは数値をアクセスキーとして利用することができます。

"ファイル"や"編集"のように、メニューの最上位にある部分をトップレベルメニューといいます。

4.3.3 メニュー編集ウィンドウでの編集操作

メニュー編集ウィンドウでは、ポップアップメニューの以下のコマンドを使って、メニュー項目を削除したり、コピーや移動したりすることもできます。

[切り取り] : メニュー項目をクリップボードにコピーし、メニュー編集ウィンドウから削除します。

[コピー] : メニュー項目をクリップボードにコピーします。

[貼り付け] : クリップボードにコピーしたメニュー項目を、選択状態のメニュー項目の直前に挿入します。ただし、フォームの直下のメニュー（サンプルプログラムでは"CfMenu1"）を選択している場合は、トップレベルメニューの最後に追加されます。

[削除] : メニュー項目を削除します。

4.3.4 メニュー編集ウィンドウを閉じる

メニューの編集が終了したら、メニュー編集ウィンドウを閉じます。メニュー編集ウィンドウは、以下のどれかの方法で閉じることができます。

フォーム編集ウィンドウの [ファイル] メニューから [閉じる] コマンドを選択します。

フォーム編集ウィンドウの [ファイル] メニューから [終了して戻る] コマンドを選択します。

フォーム編集ウィンドウの [閉じる] ボタン (右上の × ボタン) をクリックします。

編集中のメニュー編集ウィンドウの [閉じる] ボタン (右上の × ボタン) をクリックします。

4.4 手続きを編集する

本節では、イベント手続きの編集方法について説明します。本節では、前節で作成したフォームに対し、以下のように手続きを記述していきます。

1. [ファイル]メニューの[開く]を選択したときの手続きを記述します。
開くファイルを選択します。
表コントロールをクリアします。
ファイルに格納されているデータを表コントロールに読み込みます。
2. [ファイル]メニューの[上書き保存]を選択したときの手続きを記述します。
表コントロールから行単位にデータを取得し、ファイルに書き込みます。
ファイル名が決まっていない場合は、ファイルに名前を付けて保存します。
3. [ファイル]メニューの[名前を付けて保存]を選択したときの手続きを記述します。
保存するファイル名を取得します。
表コントロールから行単位にデータを取得し、ファイルに書き込みます。
4. [ファイル]メニューの[ページ設定]を選択したときの手続きを記述します。
印刷時のページレイアウトを設定するためのダイアログボックスを表示します。
5. [ファイル]メニューの[印刷]を選択したときの手続きを記述します。
表コントロールに入力されたデータを印刷します。
6. [ファイル]メニューの[終了]を選択したときの手続きを記述します。
フォームを閉じて、アプリケーションを終了します。
7. [編集]メニューの[合計]を選択、またはフォーム上の[合計]ボタンをクリックしたときの手続きを記述します。
表コントロールの小計の列をすべて加算します。
合計金額を表示します。
8. 表コントロールに商品コードが入力されたときの手続きを記述します。
商品コードをもとに、商品名と単価を求めます。
商品名と単価を表示します。
9. 表コントロールに個数が入力されたときの手続きを記述します。
商品コードをもとに、単価を求めます。
単価と個数をもとに、小計を計算します。
小計を表示します。

4.4.1 手続き編集ウィンドウを表示する

手続きを編集する前に、手続きを編集するためのウィンドウを表示する方法について説明します。

手続きを編集する対象には、以下のものがあり、それぞれ手続き編集ウィンドウの表示方法が異なります。

- フォームの環境部
- フォームのデータ部
- フォームの手続き部
- フォームのイベント手続き
- コントロールのイベント手続き
- メニュー項目のイベント手続き

フォームの環境部

フォームの環境部には、そのフォームに含まれるすべての手続きに対して有効な環境を定義します。フォームの環境部を編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上でフォームを選択します。
2. ポップアップメニューの[環境部の編集]サブメニューから、記述する対象となる段落名を選択します。

また、編集中のフォーム上で、コントロールが配置されていない部分を右クリックし、ポップアップメニューから表示することもできます。

フォームのデータ部

フォームのデータ部には、そのフォームに含まれるすべての手続きに対して有効なデータを定義します。フォームのデータ部を編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上でフォームを選択します。
2. ポップアップメニューの[データ部の編集]サブメニューから、記述する対象となる節名を選択します。

また、編集中のフォーム上で、コントロールが配置されていない部分を右クリックし、ポップアップメニューから表示することもできます。

フォームの手続き部

フォームの手続き部には、そのフォームに含まれるすべてのイベント手続きで利用できる共通内部プログラムを記述します。共通内部プログラムを編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上でフォームを選択します。
2. ポップアップメニューの[手続き部の編集]サブメニューから[新規作成]コマンドを選択します。

また、編集中のフォーム上で、コントロールが配置されていない部分を右クリックし、ポップアップメニューから表示することもできます。

この操作により、キャプションに"FjCobCmpScrN" (Nは正数値) と表示された手続き編集ウィンドウが表示されます。さらに以下の操作を行い、共通内部プログラム名を設定します。

1. デザインツリーウィンドウの"(スクリプト)[COBOLスクリプト]"の構成要素である"FjCobCmpScrN"を選択します。
2. ポップアップメニューの[名前の変更]コマンドを選択します。
3. "FjCobCmpScrN"部分が編集可能になるので、共通内部プログラム名を入力します。
4. [Enter]キーを押します。

たとえば、ファイル名を取得するための共通内部プログラムを作成する場合は、"FjCobCmpScrN"を"GET-FILE-NAME"というような名前に変更し、共通内部プログラム名とします。



上記のように作成した共通内部プログラムは、自動的にCOMMON属性が付けられます。フォームの共通内部プログラム、環境部およびデータ部については、「[フォームの手続き](#) (p167)」を参照してください。

すでに作成済の共通内部プログラムを開く場合は、ポップアップメニューの[手続き部の編集]サブメニューから対象となる共通内部プログラム名を選択してください。

フォームのイベント手続き

フォームのイベント手続きを編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上でフォームを選択します。
2. ポップアップメニューの[イベント手続きの編集]サブメニューから、記述する対象となるイベント名を選択します。

また、編集中のフォーム上で、コントロールが配置されていない部分を右クリックし、ポップアップメニューから表示することもできます。

たとえば、フォームが開かれたとき(起動されたとき)の手続きを記述する場合は、ポップアップメニューの[イベント手続きの編集]サブメニューから[Opened]コマンドを選択します。

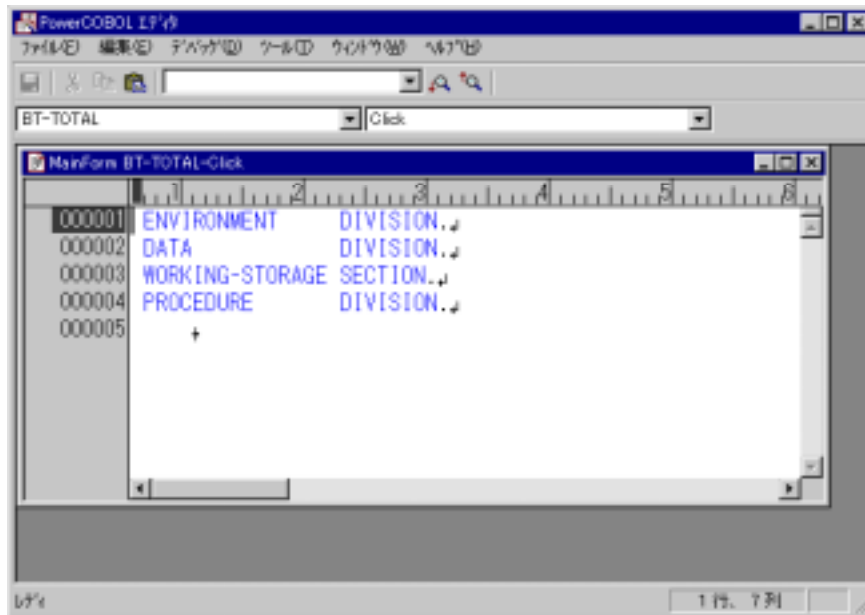
コントロールのイベント手続き

コントロールのイベント手続きを編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上で編集対象となるコントロールを選択します。
2. ポップアップメニューの[イベント手続きの編集]サブメニューから、記述する対象となるイベント名を選択します。

また、編集中のフォーム上で対象となるコントロールを右クリックし、ポップアップメニューから表示することもできます。

たとえば、フォーム上の合計ボタンがクリックされたときの手続きを記述する場合は、ポップアップメニューの[イベント手続きの編集]サブメニューから[Click]コマンドを選択します。



メニュー項目のイベント手続き

メニュー項目のイベント手続きを編集するためのウィンドウは、以下の方法で表示できます。

1. デザインツリーウィンドウ上で編集対象となるメニュー項目を選択します。
2. ポップアップメニューの [イベント手続きの編集] サブメニューから、記述する対象となるイベント名を選択します。

また、メニュー編集ウィンドウ上で、対象となるメニュー項目を選択し、ポップアップメニューから表示することもできます。

たとえば、[編集(E)] メニューの [合計(T)] が選択されたときの手続きを記述する場合は、ポップアップメニューの [イベント手続きの編集] サブメニューから [Click] コマンドを選択します。



デザインツリーウィンドウ上で、フォーム、コントロールまたはメニュー項目を選択し、[オブジェクト - イベント手続きの編集] メニューからイベント名を選択することにより、手続き編集ウィンドウを表示することもできます。

すでに作成済のイベント手続きには、イベント名の先頭に "*" (アスタリスク) が付けられます。作成された手続きの一覧は、デザインツリーウィンドウの "(スクリプト)" [COBOL スクリプト] を展開することにより確認できます。

また、イベント手続きの削除は、デザインツリーウィンドウ上で行ってください。



注意

手続き編集ウィンドウの[ファイル]メニューにある[上書き更新]コマンド、および手続き編集ウィンドウを閉じるときに表示される保存確認メッセージでの保存操作は、プロジェクトの構成要素として保存されることを意味しています。したがって、これらの操作で手続きを保存しても、プロジェクト自身を保存しなければ作成または更新した手続きは保存されません。

4.4.2 手続き編集ウィンドウでの編集操作

手続き編集ウィンドウでは、[編集]メニューまたはポップアップメニューの以下のコマンドを使って、文字列を削除、コピー、または移動することができます。

- [元に戻す]: 直前の編集操作を取り消します。
- [切り取り]: 選択中の文字列をクリップボードにコピーし、手続きから削除します。
- [コピー]: 選択中の文字列をクリップボードにコピーします。
- [貼り付け]: クリップボードにコピーした文字列を、手続き中のカレットの位置に挿入します。文字列が選択されていた場合は、選択中の文字列と置き換わります。
- [削除]: 選択中の文字列を手続きから削除します。

カレットは、キーによって以下のように移動します。

- [Home]: 行の先頭に移動します。
- [Ctrl+Home]: 手続きの先頭に移動します。
- [End]: 行の末尾に移動します。
- [Ctrl+End]: 手続きの末尾に移動します。
- [PageUp]: 1ページ上に移動します。
- [PageDown]: 1ページ下に移動します。
- []: 1行上に移動します。
- []: 1文字右に移動します。
- []: 1行下に移動します。
- []: 1文字左に移動します。

4.4.3 手続きを記述する

手続きの記述方法について説明します。

フォームの環境部

フォームの環境部を記述します。このサンプルプログラムでは、ファイルを利用するために、入出力節 (INPUT-OUTPUT SECTION) のファイル管理段落 (FILE-CONTROL) を記述します。

FILE-CONTROL

- * 入力したデータを扱います。
SELECT 売上ファイル ASSIGN TO FILE-NAME.
- * 商品コードをキーとして、商品名および単価をもちます。
SELECT 商品ファイル ASSIGN TO "..¥PRODUCTS.TBL"
ORGANIZATION IS RELATIVE
RELATIVE KEY IS 商品 - キー
ACCESS MODE IS RANDOM
FILE STATUS IS FS.



注意

"PRODUCTS.TBL"は、サンプルプログラムと同じ"Table"フォルダに格納されています。サンプルプログラムを別のフォルダに作成した場合は、コピーしてお使いください。

フォームのデータ部

フォームのデータ部を記述します。サンプルプログラムでは、以下の節を記述します。これらの節で定義するデータは、フォーム内で共通に利用できるよう、GLOBAL句を付けてください。

ファイル節(FILE SECTION)
作業場所節(WORKING-STORAGE SECTION)

FILE

```
FD 売上ファイル GLOBAL.
01 売上レコード.
   02 売上 - 商品コード PIC X(8).
   02 売上 - 商品名     PIC N(10).
   02 売上 - 単価       PIC 9(5).
   02 売上 - 個数       PIC 9(4).
   02 売上 - 小計       PIC 9(8).
   02 売上 - 備考       PIC N(20).
01 顧客レコード GLOBAL.
   02 顧客 - 顧客名     PIC X(50).
FD 商品ファイル GLOBAL.
01 商品レコード.
   02 商品 - 名         PIC N(10).
   02 商品 - 単価       PIC 9(5).
```

WORKING-STORAGE

- * 編集中のファイルの名前
- * ファイルを開いた場合や名前を付けて保存した場合に設定されます。
01 FILE-NAME PIC X(256) VALUE SPACE GLOBAL.
- * 商品コードに対応した商品名と単価を求めるために使用します。
01 商品 - キー PIC 9(5) BINARY GLOBAL.
01 FS PIC XX GLOBAL.

共通内部プログラム

イベント手続きを記述する前に、各イベント手続きから共通に利用できる内部プログラムを作成しておきます。サンプルプログラムでは、以下の共通内部プログラムを作成します。

- | | |
|---------------|--------------------------------------|
| GET-FILE-NAME | ファイル名を取得するプログラムです。 |
| LOAD-DATA | ファイルに保存されたデータを表コントロール上にロードするプログラムです。 |
| SAVE-DATA | 表コントロール上に入力されたデータをファイルに保存するプログラムです。 |
| GET-TOTAL | 表コントロール上の小計から合計金額を計算するプログラムです。 |

GET-FILE-NAME

ファイル名を取得する場合は、フォームの"GetFileName"メソッドを利用します。このメソッドは、ダイアログボックスを表示してファイル名を取得するためのメソッドです。"POW-SELF"は、この手続きをもつフォームを表しています。メソッドの詳細については、『リファレンス』を参照してください。

- ```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-TITLE PIC X(22).
LINKAGE SECTION.
01 GET-STYLE PIC S9(4) COMP-5.
01 BUTTON-STATE PIC S9(4) COMP-5.
PROCEDURE DIVISION USING GET-STYLE BUTTON-STATE.
* ファイルを開くのか保存するのかを判別します。
 IF GET-STYLE = POW-CDOPEN THEN
 MOVE "ファイルを開く" TO WK-TITLE
 ELSE
 MOVE "ファイル名を付けて保存" TO WK-TITLE
 END-IF
* ファイル名を取得します。
* BUTTON-STATEには、クリックされたボタンがOKボタンか
* キャンセルボタンかの情報が設定されます。
```

( 続く )

( 続き )

```

ADD POW-CDNOCHANGEDIR TO GET-STYLE
INVOKE POW-SELF "GetFileName" USING
 FILE-NAME
 WK-TITLE
 "データファイル(*.DAT)|*.DAT|ALL(*.*)|*.*"
GET-STYLE
RETURNING BUTTON-STATE

```

**LOAD-DATA**

表コントロールの行数を設定または参照する場合は、"RowCount"プロパティを、テキストボックスコントロールの文字列を設定または参照する場合は、"Text"プロパティを利用します。また、表中のセルの内容は、表コントロールがもつ "TableCells" オブジェクトの "Text" プロパティを利用して参照することができます。オブジェクトへのアクセス方法については、「[オブジェクトを使ったアプリケーションを作成する](#) ( p197 )」を、また、オブジェクトおよびプロパティの詳細については、『リファレンス』を参照してください。

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ROW-COUNT PIC S9(4) COMP-5.
PROCEDURE DIVISION.

* フォームを無効にしマウスポインタ(マウスカーソル)を砂時計にします。
MOVE 11 TO "MousePointer" OF POW-SELF
MOVE POW-FALSE TO "Enabled" OF POW-SELF

* 表の内容をクリアします。
INVOKE TBL-PURCHASE "ClearTable"

* ファイルをオープンし、顧客名を読み込みます。
OPEN INPUT 売上ファイル
READ 売上ファイル
END-READ
MOVE 顧客 - 顧客名 TO "Text" OF TX-CUSTOMER
MOVE 1 TO ROW-COUNT
.
LOOP.
* 商品の一覧を読み込みます。
READ 売上ファイル
 AT END GO TO END-LOOP
END-READ
MOVE 売上 - 商品コード TO
 "Text" OF "TableCells"(ROW-COUNT 1) OF TBL-PURCHASE
MOVE 売上 - 商品名 TO
 "Text" OF "TableCells"(ROW-COUNT 2) OF TBL-PURCHASE

```

( 続く )

( 続き )

```

MOVE 売上 - 単価 TO
 "Text" OF "TableCells"(ROW-COUNT 3) OF TBL-PURCHASE
MOVE 売上 - 個数 TO
 "Text" OF "TableCells"(ROW-COUNT 4) OF TBL-PURCHASE
MOVE 売上 - 小計 TO
 "Text" OF "TableCells"(ROW-COUNT 5) OF TBL-PURCHASE
MOVE 売上 - 備考 TO
 "Text" OF "TableCells"(ROW-COUNT 6) OF TBL-PURCHASE
ADD 1 TO ROW-COUNT
GO TO LOOP
.
END-LOOP.
CLOSE 売上ファイル
* フォームを有効にし、マウスカーソルを元に戻します。
 MOVE POW-TRUE TO "Enabled" OF POW-SELF
 MOVE 0 TO "MousePointer" OF POW-SELF
.

```



ファイルからデータを読み込んだりデータを書き込んだりするなど、処理に時間がかかる場合は、手続きの先頭でフォームへの入力を禁止し、マウスポインタ(マウスカーソル)の形を変更しておいて、手続きの最後で元に戻すことをお勧めします。

フォームへの入力を禁止する(無効な状態にする)には、"Enabled"プロパティを利用します。フォーム上のマウスポインタを変更するには、"MousePointer"プロパティを利用します。

#### SAVE-DATA

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ROW-COUNT PIC S9(4) COMP-5.
01 TABLE-ROW PIC S9(4) COMP-5.
PROCEDURE DIVISION.
* フォームを無効にしマウスポインタ(マウスカーソル)を砂時計にします。
 MOVE 11 TO "MousePointer" OF POW-SELF
 MOVE POW-FALSE TO "Enabled" OF POW-SELF
* 最終行を求めます。
 MOVE 0 TO TABLE-ROW.
 MOVE "RowCount" OF TBL-PURCHASE TO ROW-COUNT
 PERFORM ROW-COUNT TIMES

```

( 続く )

( 続き )

```

ADD 1 TO TABLE-ROW
IF "Text" OF "TableCells"(TABLE-ROW 2) OF TBL-PURCHASE = SPACE
THEN
 SUBTRACT 1 FROM TABLE-ROW
EXIT PERFORM
END-IF
END-PERFORM
* 表からデータを取り出し、順に書き出していきます。
OPEN OUTPUT 売上ファイル
* レコードを書き出します。
MOVE "Text" OF TX-CUSTOMER TO 顧客 - 顧客名
WRITE 売上レコード
MOVE 1 TO ROW-COUNT
PERFORM WITH TEST BEFORE UNTIL ROW-COUNT > TABLE-ROW
 MOVE "Text" OF "TableCells"(ROW-COUNT 1) OF TBL-PURCHASE
 TO 売上 - 商品コード
 MOVE "Text" OF "TableCells"(ROW-COUNT 2) OF TBL-PURCHASE
 TO 売上 - 商品名
 MOVE "Text" OF "TableCells"(ROW-COUNT 3) OF TBL-PURCHASE
 TO 売上 - 単価
 MOVE "Text" OF "TableCells"(ROW-COUNT 4) OF TBL-PURCHASE
 TO 売上 - 個数
 MOVE "Text" OF "TableCells"(ROW-COUNT 5) OF TBL-PURCHASE
 TO 売上 - 小計
 MOVE "Text" OF "TableCells"(ROW-COUNT 6) OF TBL-PURCHASE
 TO 売上 - 備考
 WRITE 売上レコード
 ADD 1 TO ROW-COUNT
END-PERFORM
CLOSE 売上ファイル
* フォームを有効にし、マウスカーソルを元に戻します。
MOVE POW-TRUE TO "Enabled" OF POW-SELF
MOVE 0 TO "MousePointer" OF POW-SELF

```

**GET-TOTAL**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 合計金額 PIC S9(8) COMP-5.
01 ROW-COUNT PIC S9(4) COMP-5.
01 TABLE-ROW PIC S9(4) COMP-5.
PROCEDURE DIVISION.

```

( 続く )

( 続き )

```
* 小計の合計を計算し、表示します。
MOVE ZERO TO 合計金額
MOVE 1 TO TABLE-ROW.
MOVE "RowCount" OF TBL-PURCHASE TO ROW-COUNT
PERFORM ROW-COUNT TIMES
 IF "Text" OF "TableCells"(TABLE-ROW 2) OF TBL-PURCHASE = SPACE
 THEN
 EXIT PERFORM
 END-IF
 ADD "Text" OF "TableCells"(TABLE-ROW 5) OF TBL-PURCHASE
 TO 合計金額
 ADD 1 TO TABLE-ROW
END-PERFORM
MOVE 合計金額 TO "Caption" OF ST-TOTAL
```

### イベント手続き

サンプルプログラムでは、以下のイベント手続きを記述します。

|                     |                                           |
|---------------------|-------------------------------------------|
| MN-OPEN-Click       | [ 開く(O)... ] を選択した場合に実行する手続きを記述します。       |
| MN-SAVE-Click       | [ 上書き保存(S) ] を選択した場合に実行する手続きを記述します。       |
| MN-SAVEAS-Click     | [ 名前を付けて保存(A)... ] を選択した場合に実行する手続きを記述します。 |
| MN-PAGESETUP-Click  | [ ページ設定(T)... ] を選択した場合に実行する手続きを記述します。    |
| MN-PRINT-Click      | [ 印刷(P) ] を選択した場合に実行する手続きを記述します。          |
| MN-EXIT-Click       | [ 終了(X) ] を選択した場合に実行する手続きを記述します。          |
| MN-TOTAL-Click      | [ 合計(T) ] を選択した場合に実行する手続きを記述します。          |
| TBL-PURCHASE-Return | 表中の、1つのセルの入力が完了した場合に実行する手続きを記述します。        |
| BT-TOTAL-Click      | 合計ボタンをクリックした場合に実行する手続きを記述します。             |

### MN-OPEN-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

( 続く )



( 続き )

```

01 GET-STYLE PIC S9(4) COMP-5 VALUE POW-CDOPEN.
01 BUTTON-STATE PIC S9(4) COMP-5.
PROCEDURE DIVISION.
* 開くファイル名を取得します。
 CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE
* キャンセルボタンがクリックされた場合は、
* ファイルを開かないで処理を終了します。
 IF BUTTON-STATE = POW-FALSE THEN
 EXIT PROGRAM
 END-IF
* ファイル中のデータを表に表示します。
 CALL "LOAD-DATA"

```

**MN-SAVE-Click**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 GET-STYLE PIC S9(4) COMP-5 VALUE POW-CDSAVE.
01 BUTTON-STATE PIC S9(4) COMP-5.
PROCEDURE DIVISION.
* ファイル名が決まっていない場合、ファイル名を取得します。
 IF FILE-NAME = SPACE THEN
 CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE
* キャンセルボタンがクリックされた場合は、
* ファイルに保存しないで処理を終了します。
 IF BUTTON-STATE = POW-FALSE THEN
 EXIT PROGRAM
 END-IF
END-IF
* 表のデータをファイルに保存します。
 CALL "SAVE-DATA"

```

**MN-SAVEAS-Click**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 GET-STYLE PIC S9(4) COMP-5 VALUE POW-CDSAVE.
01 BUTTON-STATE PIC S9(4) COMP-5.
PROCEDURE DIVISION.
* 保存するファイル名を取得します。
 CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE
* キャンセルボタンがクリックされた場合は、
* ファイルに保存しないで処理を終了します。

```

( 続く )

( 続き )

```
IF BUTTON-STATE = POW-FALSE THEN
 EXIT PROGRAM
END-IF
* 表のデータをファイルに保存します。
CALL "SAVE-DATA"
```

#### **MN-PAGESETUP-Click**

ページを設定する場合は、印刷コントロールの"SetPage"メソッドを利用します。メソッドの詳細については、『リファレンス』を参照してください。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 INVOKE CmPrint1 "SetPage"
```

#### **MN-PRINT-Click**

フォームを印刷する場合は、印刷コントロールの"PrintForm"メソッドを利用します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 INVOKE CmPrint1 "PrintForm"
```

#### **MN-EXIT-Click**

フォームを閉じてアプリケーションを終了する場合は、フォームの"CloseForm"メソッドを利用します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 INVOKE POW-SELF "CloseForm"
```

#### **MN-TOTAL-Click**

合計を計算する共通内部プログラムを呼び出します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 CALL "GET-TOTAL"
```

#### **TBL-PURCHASE-Return**

このイベント手続きは、表の1つのセルに対して入力完了した場合に呼び出されます。ただし、1列め、4列めおよび6列めのそれぞれの入力内容により処理が異なるので、どの列のセルの入力が完了したかは、"Column"プロパティを利用して判定し、それぞれ以下のような処理をします。

1列めが入力されたら、内容をチェックし、商品名と単価を表示し、個数の欄に選択位置を移動します。

4列めが入力されたら、単価と個数から小計を計算し、備考の欄に選択位置を移動します。

6列めが入力されたら、次行の商品コードの欄に選択位置を移動します。

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 W-CODE PIC S9(8) COMP-5.

01 W-PRICE PIC S9(8) COMP-5.

01 W-NUMBER PIC S9(8) COMP-5.

01 ROW-POS PIC S9(4) COMP-5.

PROCEDURE DIVISION.

- \* 表の1つのセルの入力が完了した場合に呼び出されます。  
 MOVE "Row" OF TBL-PURCHASE TO ROW-POS  
 EVALUATE "Column" OF TBL-PURCHASE
- \* 1列めの場合、商品コードの入力完了です。  
 WHEN 1
- \* 入力された品番が無効なら、エラーを表示して、処理を中止します。  
 MOVE "Text" OF "TableCells"(ROW-POS 1) OF TBL-PURCHASE  
   TO W-CODE  
 IF W-CODE = ZERO THEN  
   PERFORM コード入力エラー  
 END-IF
- \* 入力された品番から、製品名と単価を求めます。  
 MOVE W-CODE TO 商品 - キー  
 OPEN INPUT 商品ファイル  
 READ 商品ファイル  
   INVALID KEY  
   CLOSE 商品ファイル  
   PERFORM コード入力エラー  
 END-READ  
 CLOSE 商品ファイル
- \* 製品名と単価を表示します。  
 MOVE 商品 - 名 TO  
   "Text" OF "TableCells"(ROW-POS 2) OF TBL-PURCHASE  
 MOVE 商品 - 単価 TO  
   "Text" OF "TableCells"(ROW-POS 3) OF TBL-PURCHASE
- \* 表の選択位置を個数の欄に移動します。  
 INVOKE TBL-PURCHASE "SelectCell" USING ROW-POS 4
- \* 4列めの場合、個数の入力完了です。  
 WHEN 4
- \* 入力された個数が無効なら、エラー表示して処理を中止します。  
 (続く)

( 続き )

```
IF "Text" OF "TableCells"(ROW-POS 4) OF TBL-PURCHASE = 0
THEN
 PERFORM 個数入力エラー
END-IF
* 単価と個数から小計を計算します。
MOVE "Text" OF "TableCells"(ROW-POS 3) OF TBL-PURCHASE
 TO W-PRICE
MOVE "Text" OF "TableCells"(ROW-POS 4) OF TBL-PURCHASE
 TO W-NUMBER
COMPUTE "Text" OF "TableCells"(ROW-POS 5) OF TBL-PURCHASE =
 W-PRICE * W-NUMBER
* 表の選択位置を備考の欄に移動します。
INVOKE TBL-PURCHASE "SelectCell" USING ROW-POS 6
* 6列めの場合、備考の入力完了です。
WHEN 6
* 表コントロールの入力域がすべて埋まっていないことを確認して、
* 表の選択位置を次行の商品コードの欄に移動します。
IF ROW-POS > "RowCount" OF TBL-PURCHASE THEN
 ADD 1 TO ROW-POS
 INVOKE TBL-PURCHASE "SelectCell" USING ROW-POS 1
END-IF
END-EVALUATE
EXIT PROGRAM
.
コード入力エラー.
* メッセージを表示します。
INVOKE POW-SELF "DisplayMessage" USING
 "品番が正しくありません。"
* 製品名と単価をクリアします。
MOVE SPACE TO "Text" OF "TableCells"(ROW-POS 2) OF TBL-PURCHASE
MOVE SPACE TO "Text" OF "TableCells"(ROW-POS 3) OF TBL-PURCHASE
* 再入力箇所にフォーカスを設定します。
INVOKE TBL-PURCHASE "SetFocus"
INVOKE TBL-PURCHASE "SelectCell" USING ROW-POS 1
EXIT PROGRAM
.
個数入力エラー.
* メッセージを表示します。
INVOKE POW-SELF "DisplayMessage" USING
 "個数を入力してください。"
* 小計をクリアします。
MOVE SPACE TO "Text" OF "TableCells"(ROW-POS 5) OF TBL-PURCHASE
```

( 続く )

( 続き )

```
* 再入力セルにフォーカスを設定します。
 INVOKE TBL-PURCHASE "SetFocus"
 INVOKE TBL-PURCHASE "SelectCell" USING ROW-POS 4
 EXIT PROGRAM
.
```

#### **BT-TOTAL-Click**

"MN-TOTAL-Click"と同じ処理です。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 CALL "GET-TOTAL"
```

## 4.5 ビルド・実行する

本節では、これまでに作成したフォームや手続きを使って、アプリケーションを作成する方法、および作成したアプリケーションを実行する方法について説明します。

### 4.5.1 ビルドとは

実行可能プログラムは、モジュールを単位として作成されます。モジュールに含まれるフォームの更新状態を自動的に判定し、必要な箇所だけ翻訳、リンクすることをビルドと呼びます。

たとえば、イベント手続きを更新した場合は、翻訳とリンクが実行されますが、フォームやコントロールのプロパティだけを更新した場合は、リンクだけが実行されます。

また、更新状態にかかわらず、モジュールのすべての構成要素を翻訳、リンクすることをリビルドと呼びます。リビルドは、モジュール内で使用しているファイル更新日時と、実行可能プログラムの作成日時を比較しただけでは、更新状態の判定が不十分になってしまうような場合に利用します。

たとえば、以下のような場合、正しくビルドできないので、リビルドする必要があります。

1. アプリケーションで利用するためのビットマップファイルを用意します。
2. ビットマップファイルを別のフォルダにバックアップしておきます。
3. ビットマップファイルを更新します。
4. ビットマップファイルをモジュールに追加します。
5. フォームにイメージコントロールを配置し、追加したビットマップファイルをイメージとして採用します。
6. ビルドして実行可能プログラムを作成します。
7. バックアップしてあったビットマップファイルを元のフォルダに戻します。
8. 再度、ビルドします。

ビットマップファイルの最終更新日時が古いのでビルドされません。ビルドでは、実行可能プログラムの作成日時とモジュールに含まれるビットマップファイルの更新日時を比較して翻訳やリンクの判定をします。しかし、翻訳やリンクの対象となるのは、実行可能プログラムの作成日時以降に更新されたファイルや手続きだけであるため、ビルドだけではビットマップファイルを以前のものに戻したことが反映されません。

このような場合には、リビルドが必要になります。

### 4.5.2 プロジェクトを保存する

更新したフォームや手続きをビルドするには、まずプロジェクトを保存する必要があります。プロジェクトを作成し、最初にプロジェクトを保存する場合(プロジェクトの名前が確定していない場合) 以下の操作により保存できます。

1. プロジェクトウィンドウの[ファイル]メニューから[名前を付けてプロジェクトの保存]コマンドを選択します。
2. [ファイル名を付けて保存]ダイアログボックスで、保存するフォルダおよびファイル名を指定します。
3. OKボタンをクリックします。



2回目以降は、[ファイル]メニューの[プロジェクトの上書き保存]コマンドで保存できます。

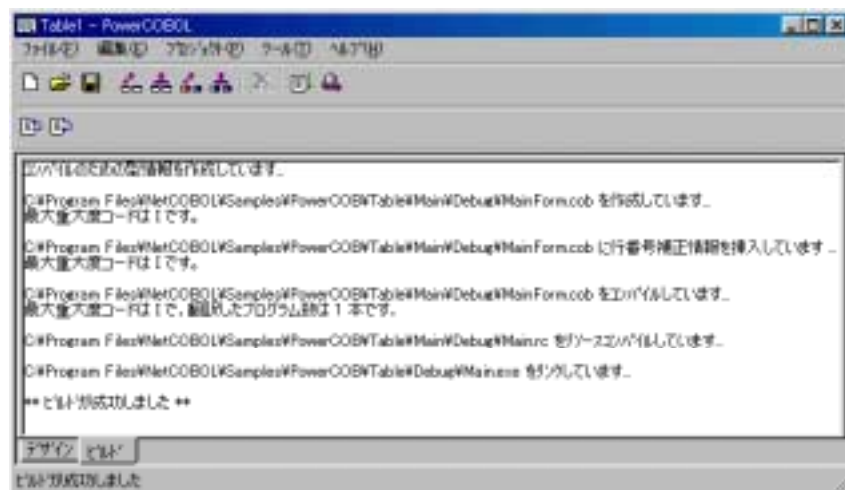
最初にプロジェクトを保存する場合、[ファイル]メニューの[プロジェクトの上書き保存]を選択しても、[ファイル名を付けて保存]ダイアログボックスが表示されます。

### 4.5.3 ビルドする

前節までに作成したフォームと手続きをビルドします。以下の操作により、ビルドできます。

1. デザインツリーウィンドウのMain [モジュール]を選択します。
2. ポップアップメニューの[ビルド]コマンドを選択します。

ビルドが始まると、プロジェクトウィンドウがビルドビューに切り替わり、ビルドの進行状況が表示されます。





以下の方法でも、ビルドすることができます。

デザインツリーウィンドウでモジュールを選択し、[プロジェクト]メニューの[ビルド]コマンドを選択します。

プロジェクトウィンドウのツールバーから、ビルド用のボタンをクリックします。

プロジェクトが複数のモジュールをもつ場合は、以下のどちらかの方法で、すべてのモジュールをビルドできます。

[プロジェクト]メニューの[すべてビルド]コマンドを選択します。

デザインツリーウィンドウでプロジェクトを選択し、ポップアップメニューの[すべてビルド]コマンドを選択します。

プログラムのリンクが不要な場合は、以下のように操作してフォームの翻訳だけを行うことができます。

1) デザインツリーウィンドウのMainForm [ フォーム ] を選択します。

2) ポップアップメニューの[コンパイル]コマンドを選択します。

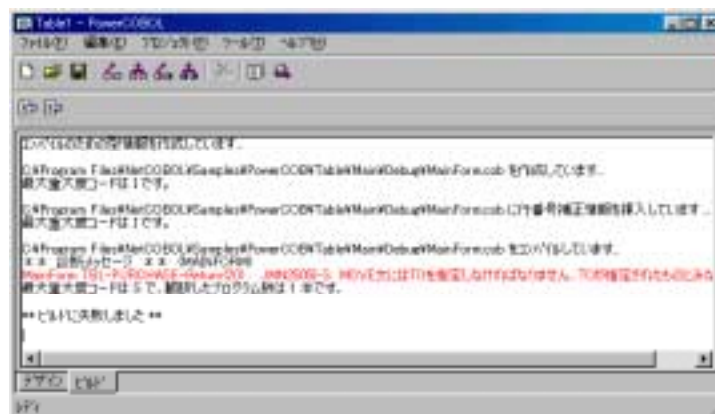


注意

[プロジェクト]メニューの[ビルド]コマンドおよびツールバーのビルドボタンによるビルドは、[デフォルトモジュール](#)( p119)を対象としています。デフォルトモジュール以外のモジュールをビルドする場合は、ポップアップメニューの[ビルド]コマンドを使ってください。

### 4.5.4 エラーがあったら

翻訳やリンクの結果は、ビルドビューに表示されます。たとえば、表コントロール(TBL-PURCHASE)のReturnイベント手続きにエラーがあった場合、以下のように表示されます。





エラーを修正するには、ビルドビューに表示されている診断メッセージのどれかをダブルクリックしてください。この操作により、エラーの対象となったイベントが手続き編集ウィンドウに表示され、該当する行にカレットが移動します。ここで、エラーの原因となった手続きを修正してください。手続きを修正したら、再度ビルドしてください。



ワンポイント

上記の例は、MOVE文の送り先の直前にT0が抜けているために、エラーが発生しています。

### 4.5.5 実行する

できあがったアプリケーションを実行します。

サンプルプログラムを実行するには、手続き中で商品コードから商品名および単価を求めるために使用する"商品ファイル" (PRODUCTS.TBL)が必要です。"PRODUCTS.TBL"は、サンプルプログラムと同じ"Table"フォルダに格納されています。サンプルプログラムを別のフォルダに作成した場合は、コピーしてお使いください。また、誤ってこのファイルを削除してしまった場合は、同じフォルダにある"CreateProductsTable.ppj"をビルド/実行し、ファイルを作成してください。

以下の操作により、アプリケーションを実行することができます。

1. デザインツリーウィンドウのMain [ モジュール ] を選択します。
2. ポップアップメニューの [ 実行 ] コマンドを選択します。

実行が始まりフォームが表示されたら、本章の先頭で述べた機能が実際に動作するか確認してください。



ワンポイント

以下の方法でも、アプリケーションを実行することができます。

プロジェクトウィンドウの [ プロジェクト ] メニューから [ 実行 ] コマンドを選択します。

プロジェクトウィンドウのツールバーから、実行用のボタンをクリックします。



注意

サンプルプログラムでは、エラー系の入力チェックを十分に実施していません。実際に運用するアプリケーションを作成する場合には、エラーチェックを強化する必要があります。たとえば、保存する場合には、以下のようなエラーチェックが必要になります。

- 顧客名が入力されているか
- 表に商品が1つ以上入力されているか

## 4.6 アプリケーション開発時のオプション

PowerCOBOLでアプリケーションを開発する場合のオプションには、以下のものがあります。

- |              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| プロジェクトオプション: | プロジェクトウィンドウの表示方法に関するオプション、およびプロジェクトファイルのバックアップやテンプレートを格納するフォルダなどを指定できます。 |
| ビルドオプション:    | ビルド時の動作に関するオプションを指定できます。                                                 |
| デザインオプション:   | デザインツリーウィンドウの動作に関するオプションを指定できます。                                         |
| フォームオプション:   | フォーム編集時のグリッドに関するオプションを指定できます。                                            |
| エディタオプション:   | 手続き編集ウィンドウの表示方法および動作に関するオプションを指定できます。                                    |
| デバッグオプション:   | デバッグ時の動作に関するオプションを指定できます。                                                |

これらのオプションは、オプションのプロパティ設定ダイアログボックスで設定できます。オプションのプロパティ設定ダイアログボックスは、プロジェクトウィンドウの[ツール]メニューから[オプション]コマンドを選択して表示することができます。

各オプションの詳細については、『リファレンス』を参照してください。

## 4.7 プロジェクト構成要素の命名規則

PowerCOBOLのプロジェクトの構成要素は、以下のように命名することができます。

### 4.7.1 モジュール名

モジュール名は、ファイル名として使用できる名前を付けることができます。ただし、以下の名前は使用できません。

- 先頭の文字が'\$'で始まる名前
- 感嘆符(!)を含む名前
- 同じプロジェクト内の他のモジュールと同じ名前

### 4.7.2 外部ファイル名

以下の条件を満たす名前をもつファイルを、外部ファイルとしてモジュールに追加することができます。外部ファイルについては、「[プロジェクトの便利な機能](#)( p96)」を参照してください。

- 先頭の文字が英字で始まる名前
  - 半角英数字、ハイフン(-)またはアンダースコア(\_)から構成されている名前
  - 255文字以内の名前
- ただし、以下の名前は使用することができません。
- 同じモジュール内のフォームと同じ名前
  - 同じモジュールに追加された他の外部ファイルと同じ名前

### 4.7.3 フォーム名

フォーム名は、COBOLの利用者語の規則の範囲で命名することができます。COBOLの利用者語として使用できる名前については、『COBOL 文法書』を参照してください。ただし、以下の名前は使用できません。

- 日本語を含む名前
- プログラム中で使用しているデータ名と同じ名前
- 同じモジュール内の他のフォームと同じ名前
- 同じモジュールに追加された外部ファイルと同じ名前
- フォーム内のコントロールと同じ名前
- フォーム内のその他の構成要素(カスタムプロパティなど)と同じ名前
- フォームのプロパティ名と同じ名前
- フォームのメソッド名と同じ名前
- 先頭の文字が"POW-"または"POWER-"で始まる名前

#### 4.7.4 コントロール名

コントロール名（配列化されたコントロールを含む）は、COBOLの利用者語の規則の範囲で命名することができます。COBOLの利用者語として使用できる名前については、『COBOL 文法書』を参照してください。ただし、以下の名前は使用できません。

- プログラム中で使用しているデータ名と同じ名前
- 同じフォーム内の他のコントロールと同じ名前（ただし、配列化されているコントロールは、すべて同じ名前になります）
- フォーム内のその他の構成要素（カスタムプロパティなど）と同じ名前
- フォームのプロパティ名と同じ名前
- フォームのメソッド名と同じ名前
- 先頭の文字が"POW-"または"POWER-"で始まる名前

#### 4.7.5 フォームのその他の構成要素の名前

フォームのその他の構成要素には、カスタムプロパティ、カスタムメソッド、カスタムイベントがあります。これらの名前は、コントロール名と同様、COBOLの利用者語の規則の範囲で命名できます。COBOLの利用者語として使用できる名前については、『COBOL 文法書』を参照してください。ただし、以下の名前は使用することができません。

- プログラム中で使用しているデータ名と同じ名前
- 同じフォーム内の他の構成要素（コントロールなど）と同じ名前
- フォームのプロパティ名と同じ名前
- フォームのメソッド名と同じ名前
- 先頭の文字が"POW-"または"POWER-"で始まる名前

---

## 第5章      PowerCOBOLを使いこなそう

---

本章では、より効率的にアプリケーションを作成できるよう、PowerCOBOLの開発環境がもっている便利な機能について説明します。

---

## 5.1 プロジェクトの便利な機能

本節では、プロジェクトを編集する場合に活用できる、便利な機能について説明します。

### 5.1.1 外部ファイルを使う

PowerCOBOLのモジュールには、フォームの他に、実行可能プログラムを作成するためのファイルを追加できます。これらのファイルは、PowerCOBOL以外の環境で作成したファイルであるため、外部ファイルと呼びます。

#### 外部ファイルとは

外部ファイルとは、以下のファイルです。外部ファイルとしてモジュールに追加することにより、ビルド時に翻訳やリンクの対象となります。

#### COBOLファイル

COBOL手続きが記述されているCOBOLのソースファイルです。拡張子は、COB、CBLまたはCOBOLです。ただし、COBOLファイルのプログラム形式は、行番号付きの可変長形式にしてください。ビルド時には、翻訳の対象となります。

#### オブジェクトファイル

COBOLコンパイラによって作成されたオブジェクトファイルです。拡張子は、OBJです。ビルド時には、リンクの対象となります。

#### ライブラリファイル

リンクによって作成されたライブラリファイルです。拡張子はLIBです。ビルド時には、リンクの対象となります。

#### リソースファイル

以下の各形式で作成されたイメージ用のファイルです。ビルド時には、リンクの対象となります。

リソースファイルは、フォームやコントロールでリソースを使用する場合に必要です。ビットマップやアイコンなどのイメージファイルを追加したときにデザインツリーウィンドウに表示される名前がリソース名になります(プロパティリストウィンドウのNameと対応しています)。イメージをリソースファイルとして追加することで、イメージは実行可能プログラムに組み込まれます。したがって、実行可能プログラムのサイズは、その分大きくなります。しかし、実行時には、ファイルの有無やパスの設定などを気にすることなく、利用できます。

| ファイルの種類         | 拡張子 |
|-----------------|-----|
| ビットマップファイル      | BMP |
| アイコンファイル        | ICO |
| ポインタファイル        | CUR |
| アニメーションポインタファイル | ANI |
| イメージリストファイル     | BMP |



イメージリストとは、1つのビットマップの中に、等幅のイメージを横に複数並べたものです。各イメージの幅は、イメージリストのプロパティで設定できます。イメージリストは、ツールバーコントロールやツリービューコントロールなどで、複数のビットマップを利用するような場合に必要となります。使用例については、「[ツールバーを使ったアプリケーションを作成する](#) ( p226 )」を参照してください。



イメージリストとして利用するファイルは、16色ビットマップとして作成してください。16色を超えている場合、実行時に自動的に減色（マッピング）されます。

### 外部ファイルを追加するには

外部ファイルは、以下の手順でモジュールに追加することができます。

1. デザインツリーウィンドウで、外部ファイルを追加するモジュールを選択します。
2. ポップアップメニューの [ ファイルの挿入 ] コマンドを選択します。
3. 表示された [ ファイルの挿入 ] ダイアログボックスで、追加する外部ファイルを選択します。

### COBOLファイルを編集する

モジュールに追加したCOBOLファイルは、PowerCOBOLの手続き編集ウィンドウを使って編集できます。手続き編集ウィンドウは、COBOLファイルを選択し、ポップアップメニューの [ 編集 ] コマンドで表示できます。

### リソースファイルの使用例

リソースファイルは、以下のように利用することができます。

#### フォームのアイコンの設定例

PowerCOBOLが提供しているサンプルプログラム "Icon¥Icon.ppj" を参照してください。このサンプルプログラムでは、以下の操作により、フォームのアイコンを指定しています。

1. 3つのアイコンファイルをそれぞれ "bomb"、"clock"、"color" というリソース名で、モジュールに追加します。
2. フォームのプロパティを開き、リソースタブを選択します。
3. [ プロパティ名 ] から、"IconName" を選択します。
4. [ リソース名 ] から、モジュールに追加したアイコンファイルのリソース名を選択します。

この指定により、このアプリケーションを起動したときのフォームのアイコンは、"bomb" という名前で追加したアイコンが使用されます。

また、オプションボタンコントロールのClickイベントで、フォームのIconNameプロパティに、アイコンファイルのリソース名を転記することにより、実行時

にアイコンを変更することができるようになっています。



注意

COBOLの実行環境情報によるアイコンリソースの指定 (@IconDLL、@IconName) で、フォームのアイコンを設定することはできません。

### アニメーションコントロールのイメージの設定例

PowerCOBOLが提供しているサンプルプログラム"Animation¥Animation.ppj"を参照してください。このサンプルプログラムでは、以下の操作により、アニメーションで使用するイメージを指定しています。

1. モジュールに、8つのビットマップファイルをそれぞれ"Bitmap1" ~ "Bitmap8"というリソース名で追加します。
2. アニメーションコントロールのプロパティを開きます。
3. スタイル中の [ リソース ] をチェック状態にします。
4. [ フレームリスト ] の右側にある挿入ボタン ( + ボタン ) をクリックします。
5. [ フレームリスト ] 中に、ビットマップファイルのリソース名"Bitmap1"を記述します。
6. 同様の操作により、"Bitmap2" ~ "Bitmap8"を [ フレームリスト ] に追加します。

この指定により、アニメーションを再生したとき、8つのビットマップを順に表示できるようになります。

## 5.1.2 テンプレートを追加する

PowerCOBOLで新規にプロジェクトを作成する場合、[ 標準フォーム ] [ 標準ダイアログ ] および [ ActiveX ] という3つのテンプレートが用意されています。これらの他に、あらかじめ作成しておいたプロジェクトファイルをテンプレートとして追加できます。

以下の操作により、作成したプロジェクトファイルをテンプレートとして使用できます。

1. プロジェクトウィンドウの [ ツール ] メニューから [ オプション ] コマンドを選択します。
2. オプションのプロパティ設定ダイアログボックスで、[ プロジェクト ] タブの [ テンプレートフォルダ ] を指定します。
3. OKボタンをクリックします。
4. Windows のエクスプローラなどで、テンプレートとして作成したプロジェクトファイルを、[ テンプレートフォルダ ] で指定したフォルダへコピーまたは移動します。

プロジェクトのオプションについては、『リファレンス』を参照してください。



### 5.1.3 Unicodeを利用する

PowerCOBOLでは、実行時のコード系をUnicodeで扱うアプリケーションを作成することができます。実行時のコード系は、以下の操作により変更できます。

1. デザインツリーウィンドウのプロジェクトを選択します。
2. ポップアップメニューの[プロパティ]コマンドを選択します。
3. プロジェクトのプロパティ設定ダイアログボックスで、[ビルド]タブを選択します。
4. [ランタイムコードセット]から"1 - UCS2"を選択します。
5. OKボタンをクリックします。

ランタイムコードセットは、COBOLコンパイラのRCS（実行時コード系の指定）オプションに相当し、実行時のコード系をシフトJISコード(0 - SJIS)にするか、Unicode(1 - UCS2)にするかを指定することができます。

Unicodeについての詳細は、『NetCOBOL 使用手引書』を参照してください。



ランタイムコードセットをUnicode(1 - UCS2)にした場合、Windows 95、Windows 98およびWindows Meでビルドおよび実行することはできません。

ランタイムコードセットがUnicode(1 - UCS2)である実行可能プログラムと、シフトJISコード(0 - SJIS)である実行可能プログラムを混在させ、1つのアプリケーションとして実行させることはできません。

ランタイムコードセットをUnicode(1 - UCS2)にした場合、V3.0以前に使用していたアイテムの属性名およびCALL文によるメソッドの呼び出しを使用することはできません。使用した場合、ビルド時にエラーとなります。

ランタイムコードセットをUnicode(1 - UCS2)にする場合には、「[プロパティへのアクセス方法](#) ( p171) 」および「[メソッドの呼び出し方法](#) ( p180) 」に記載されている記述形式を使用してください。

### 5.1.4 ユーティリティを利用する

PowerCOBOLでは、以下のユーティリティを提供しています。

検索ユーティリティ

置換ユーティリティ

ユーティリティは、ポップアップメニューの[ユーティリティ]サブメニューから対象となるコマンドを選択して利用します。

#### 検索ユーティリティ

検索ユーティリティは、プロジェクト、モジュールまたはフォーム単位で手続き中の文字列を検索できます。検索ユーティリティについては、『リファレンス』を参照してください。

**検索**

検索の種類(Q) テキスト検索

検索の対象(T) C:\Program Files\POCOB97\SAM

☒ サブオブジェクトも対象にする(S)

☐ サブオブジェクトのみを対象にする(O)

検索の方法(M)

☐ 対話的に検索する

☒ 一気に検索する

検索する文字列(E) レコード

☐ 大文字と小文字を区別する(E)

| 場所                        | テキスト                    |
|---------------------------|-------------------------|
| !Main/MainForm/Staff010.. | 000022* レコードを書き出します。    |
| !Main/MainForm/Staff010.. | 000024 WRITE 売上レコード     |
| !Main/MainForm/Staff010.. | 000033 WRITE 売上レコード     |
| !Main/MainForm/Staff010.. | 000002 01 売上レコード        |
| !Main/MainForm/Staff010.. | 000009 01 顧客レコード GLOBAL |
| !Main/MainForm/Staff010.. | 000013 01 商品レコード        |

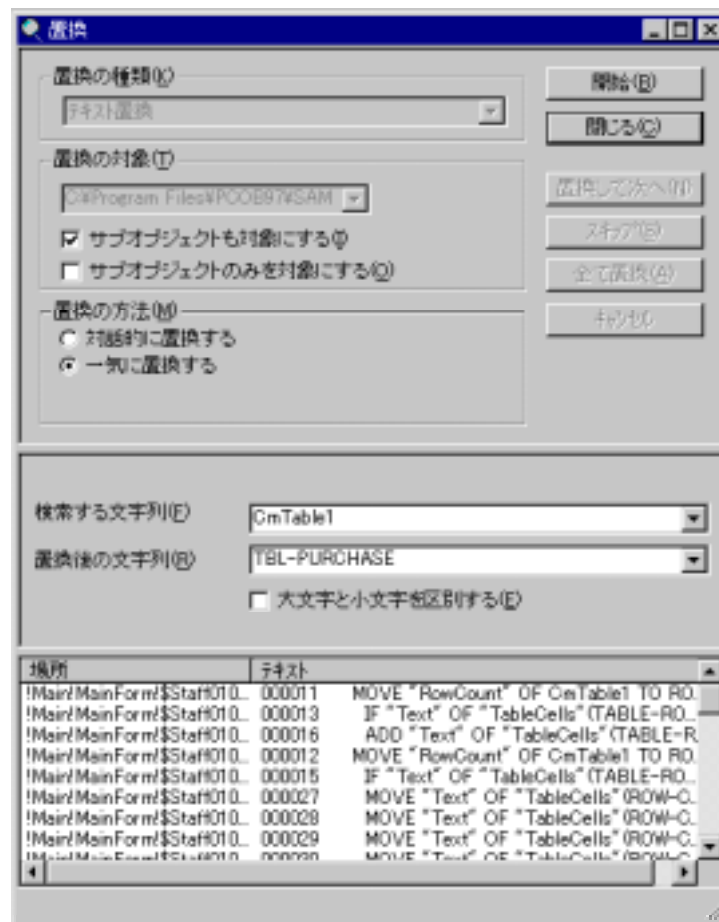


注意

検索後に手続きを変更した場合、検索結果と実際の文字列の位置が一致しない場合があります。

## 置換ユーティリティ

置換ユーティリティは、プロジェクト、モジュールまたはフォーム単位で手続き中の文字列を置換できます。置換ユーティリティについては、『リファレンス』を参照してください。



注意

置換後に手続きを変更した場合、置換結果と実際の文字列の位置が一致しない場合があります。

## 5.2 フォーム編集時の便利な機能

本節では、フォームを編集する場合に利用できる便利な機能について説明します。

### 5.2.1 グリッドを利用する

グリッドは、フォーム上に基準線を表示し、コントロールの位置やサイズの体裁を整えやすくするための機能です。グリッドを利用することにより、以下のことができます。

グリッドを表示することにより、コントロールの位置やサイズを調整しやすくします。グリッドの基準線は、点線で表示されます。

コントロールをグリッドに合わせて移動できます。移動する場合の単位は、グリッドの横幅および縦幅に依存します。

コントロールのサイズをグリッドに合わせて変更できます。大きさの単位は、グリッドの横幅および縦幅に依存します。



グリッドは、オプションのプロパティ設定ダイアログボックスの [ フォーム ] タブまたは [ グリッドの設定 ] ダイアログボックスで設定することができます。詳細については、『リファレンス』を参照してください。

グリッドに合わせられていないコントロールを、あとからグリッドに合わせる場合には、コントロールを選択し、ポップアップメニューまたは [ レイアウト ] メニューの [ グリッドに合わせる ] コマンドを選択してください。


### 5.2.2 タブ順序とタブグループを設定する

タブ順序とタブグループを編集することにより、実行時に、[Tab] キーの操作によるコントロール間のフォーカス移動の順番や、矢印キーの操作でフォーカスが移動できるタブグループを設定できます。

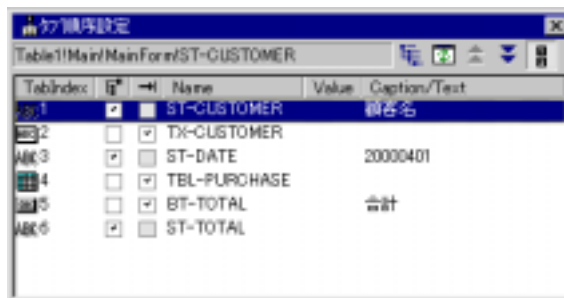
タブ順序やタブグループを編集するには、コントロールのTabIndexプロパティやTabGroupプロパティを直接編集する方法と、タブ順序設定ウィンドウを使って編集する方法があります。

タブ順序設定ウィンドウは、フォーム編集ウィンドウの[レイアウト]メニューから[タブ順序設定]コマンドを選択することにより表示できます。

また、タブ順序設定ウィンドウでは、フォーム上に配置されているコントロールの一覧をタブインデックス順で表示するか、タブグループ別に表示するかを



切り替えることができます。表示の切り替えは、 ボタンを使います。

### タブインデックス順で表示する



タブインデックス順で表示した場合、タブインデックスが昇順になるようコントロールの一覧が表示されます。タブインデックスの順は、以下のどれかの方法で変更することができます。

#### 前後のコントロールを入れ替える（その1）

1. タブ順序設定ウィンドウ上で、タブインデックス順を変更するコントロールを選択します。
2. 1つ前に移動する場合は、 ボタンを、1つ後ろに移動する場合は、 ボタンをクリックします。

#### 前後のコントロールを入れ替える（その2）

1. タブ順序設定ウィンドウ上で、タブインデックス順を変更するコントロールを選択します。
2. 1つ前に移動する場合は、ポップアップメニューの[上に移動]コマンドを、1つ後ろに移動する場合は、ポップアップメニューの[下に移動]コマンドを選択します。

#### ドラッグ&ドロップでコントロールを移動する

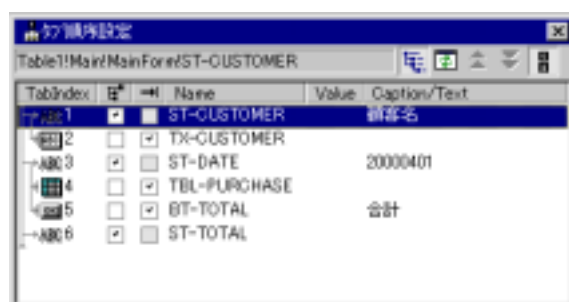
1. タブ順序設定ウィンドウ上で、タブインデックス順を変更するコントロールを、マウスの左ボタンを押して選択します。
2. マウスの左ボタンを押したまま、移動したいタブインデックスの値が表示されているコントロールの位置まで、マウスポインタを移動します（ドラッグ）。
3. マウスの左ボタンを離します（ドロップ）。




タブインデックスの順番を変更すると、それに合わせて各コントロールのTabIndexプロパティの値も変更されます。


イメージコントロールやタイマコントロールなど、TabIndexプロパティをもたないコントロールは、タブ順序設定ウィンドウの一覧には表示されません。また、TabIndexプロパティが変更されても描画順序（重なっている場合の前後関係）は変更されません。描画順序の変更については、「[コントロールの描画順序を変更する](#)（p105）」を参照してください。

## タブグループ別に表示する



タブグループ別に表示した場合、グループボックスコントロールやフレームコントロールなど、子コントロールを配置できるコントロールが、ツリー形式で表示されます。タブグループは、以下の方法で変更できます。

1. タブ順序設定ウィンドウ上で、タブグループの先頭になるコントロールを選択します。
2. で示されるカラムのチェックボックスをチェック状態にします。

また、タブストップの位置を設定することもできます。タブストップを設定すると、実行時に[Tab]または[Shift+Tab]キーによってフォーカスを移動できます。タブストップは、で示されるカラムのチェックボックスをチェック状態にすることで設定できます。




タブグループおよびタブストップを変更すると、それに合わせて各コントロールのTabGroupプロパティおよびTabStopプロパティの値も変更されます。

## タブ順序設定ウィンドウのその他の機能


タブ順序設定ウィンドウには、その他に以下の機能があります。

### 一覧の更新

タブ順序設定ウィンドウを表示中に、フォームにコントロールを配置したり、プロパティ設定ダイアログボックスなどにより、タブインデックスなどの情報を変更したりしても、すぐにはタブ順序設定ウィンドウに反映されません。そ

の場合、ボタンをクリックして、表示されているコントロールの一覧を更新できます。

### タブインデックスラベルの表示

タブインデックスなどの情報が設定しやすいよう、ボタンをクリックすることにより、編集中のフォーム上にタブインデックスの値を表示することができます。  
また、再度、このボタンをクリックすると、表示を取り消すことができます。



### 5.2.3 コントロールの描画順序を変更する

フォーム上の複数のコントロールが重なっている場合、以下のようにして、コントロールの描画順序（どのコントロールを上または下に表示するか）を変更することができます。

1. 描画順序を変更するコントロールを選択します。
2. ポップアップメニューから、上に表示する場合は[最前面に移動]コマンドを、下に表示する場合は[最背面に移動]コマンドを選択します。

最前面に移動した場合は、フォーム上のコントロールの中で、最も上に描画されるようになります。最背面に移動した場合、最も下に描画されるようになります。

### 5.2.4 コントロールをまとめて編集する

PowerCOBOLでは、フォーム上の複数のコントロールをまとめて、以下の編集を行うことができます。

- プロパティの設定
- コピー、切り取り、貼り付け、削除

移動

位置揃え、サイズ合わせなどのレイアウトの変更

配列化

コントロールをまとめて編集する場合も、単体のコントロールを編集する場合と同様に、「選択 操作」という手順で進めていきます。まず複数のコントロールを選択し、それらに対して操作方法を決めるという手順です。

### コントロールを複数選択するには

以下のどちらかの操作により、複数のコントロールを選択できます。

ドラッグによる範囲指定による選択

クリックによる選択

#### ドラッグによる範囲指定による選択

1. マウスポインタをフォーム上の任意の位置に移動します。
2. マウスの左ボタンをドラッグしながら、まとめて選択したいコントロール群を囲むように矩形（長方形の枠）を作ります。
3. 矩形が適当な大きさになったところで、ドラッグを終了します。

この操作により、矩形に含まれるコントロールをまとめて選択することができます。コントロールのすべてが矩形に含まれていなくても、一部が含まれていれば、選択されたことになります。

#### クリックによる選択

1. まとめて選択したいコントロールのうち、1つをクリックし、選択します。
2. [Shift]キーを押しながら、他の選択したいコントロールをクリックします。
3. 同じように、選択したいコントロールに対し、[Shift]キーを押しながらクリックする操作を繰り返します。

また、以下のどちらかの操作により、選択を解除することができます。

選択されていないコントロールまたはフォームをクリックすると、選択中のすべてのコントロールの選択状態が解除されます。

選択されているコントロールを[Shift]キーを押しながらクリックすると、そのコントロールの選択が解除されます。



注意

---

コントロールをまとめて選択する場合は、同一のコンテナに配置されたコントロールが対象となります。したがって、フォーム上に直接配置されたコントロールと、グループボックスコントロール上に配置されたコントロールを、同時に選択することはできません。

---

### 複数のコントロールに対するプロパティの設定

複数のコントロールに対し、まとめてプロパティを設定する場合、コントロールの種類などにより、設定できるプロパティと設定できないプロパティがあります。

たとえば、選択したコントロールがテキストボックスコントロールとスタティ



ックテキストコントロールの場合、お互いに共通してもっているカラーやフォント、テキスト属性など、多くのプロパティを設定することができます。しかし、テキストボックスと印刷コントロールを選択した場合、それぞれ異なった機能をもつコントロールのため、設定できるプロパティはほとんどありません。

### 複数のコントロールに対するレイアウトの変更

複数のコントロールを選択した場合、それらのコントロールに対し、以下のようなレイアウト調整ができます。

#### 位置を揃える

選択した複数のコントロールに対し、それらのコントロールの左端、右端、上端および下端の位置を揃えることができます。この機能は、[ レイアウト - 位置を揃える ]メニューから対応するコマンドを選択することにより利用することができます。

たとえば、前章で作成した商品の合計金額を計算するサンプルプログラムで、スタティックテキストコントロール "ST-CUSTOMER" と表コントロール "TBL-PURCHASE"を選択し、[ レイアウト - 位置を揃える ]メニューの[ 左端 ]コマンドを選択すると、左端をきれいに揃えることができます。

また、スタティックテキストコントロール"ST-CUSTOMER"、テキストボックスコントロール"TX-CUSTOMER"およびスタティックテキストコントロール"ST-DATE"を選択し、[ レイアウト - 位置を揃える ]メニューの[ 上端 ]コマンドを選択すると、上端をきれいに揃えることができます。

#### サイズを合わせる

選択した複数のコントロールに対し、それらのコントロールの幅と高さを合わせることができます。この機能は、[ レイアウト - サイズを合わせる ]メニューから対応するコマンドを選択することにより利用できます。

#### スペースを均等化する

選択した複数のコントロールに対し、それらのコントロール間のスペースを均等な間隔にして配置することができます。この機能は、[ レイアウト - サイズを合わせる ]メニューから[ 横方向 ]または[ 縦方向 ]コマンドを選択することにより利用できます。

#### 中央へ配置する

選択した複数のコントロールをコンテナ(フォームやグループボックスコントロール)の中央に配置することができます。この機能は、[ レイアウト - サイズを合わせる ]メニューから[ 左右方向 ]または[ 上下方向 ]コマンドを選択することにより利用できます。

## 5.2.5 コントロールを配列化して利用する

PowerCOBOLでは、フォーム上の複数のコントロールを配列化して利用できます。配列化することにより、複数のコントロールは同じ名前をもつことになり、インデックス値によって識別されるようになります。また、配列化された複数のコントロールは、イベント手続きを共有できます。

配列化を利用したアプリケーションの作成例については、「[配列化したコントロールを使ったアプリケーションを作成する](#) ( p194 )」を参照してください。



---

配列化が可能なコントロールは、同じコンテナに配置された、同じ種類のコントロールだけです。  
配列化すると、それまでに記述されていたイベント手続きは失われます。

---

### コントロールを配列化するには

コントロールは以下のどちらかの方法で配列化することができます。

- ポップアップメニューを使って配列化する
- 複数のコントロールをまとめて配列化する

#### ポップアップメニューを使って配列化する

1. 配列化の先頭となるコントロールを選択します。
2. ポップアップメニューの [ 配列化 ] メニューから、[ 新規作成 ] コマンドを選択します。
3. 配列に追加する他のコントロールを選択します。
4. ポップアップメニューの [ 配列化 ] サブメニューから、配列の先頭に設定したコントロールの名前を選択します。
5. 同様に、その他のコントロールを配列に追加していきます。



---

配列の先頭となるコントロールは、プロパティ設定ダイアログボックスの [ 共通 ] タブにある [ 配列化 ] をチェックすることにより、設定することもできます。また、[ インデックス値 ] を変更することによって、配列の順番を変更することもできます。

---

#### 複数のコントロールをまとめて配列化する

1. 配列化する複数のコントロールをまとめて選択します。
2. ポップアップメニューの [ 配列化 ] メニューから [ 新規作成 ] コマンドを選択します。



---

複数のコントロールをまとめて選択する方法は、「[コントロールをまとめて編集する](#) ( p105 )」を参照してください。

すでに配列化したコントロールに対して、複数のコントロールをまとめて追加する場合は、ポップアップメニューの [ 配列化 ] サブメニューから追加先のコントロール名を選択してください。

---

### コントロールの配列化を解除するには

配列化は、以下のどちらかの方法で解除することができます。

配列化を解除し、別の名前のコントロールとして再利用する  
コントロール自体を削除する

#### 別の名前のコントロールとして再利用する

1. 配列化を解除するコントロールを選択します。
2. ポップアップメニューの[ 配列化 ]メニューから[ このコントロールを配列解除 ] コマンドを選択します。

配列化を解除すると、他のコントロールのインデックス値は自動的に詰められます。



プロパティ設定ダイアログボックスの[ 共通 ]タブにある[ 配列化 ]のチェックをはずすことにより、配列化を解除することもできます。

#### コントロールを削除する

1. 配列化を解除するコントロールを選択します。
2. ポップアップメニューの[ 削除 ]コマンドを選択します。

配列化したコントロールを削除した場合も、他のコントロールのインデックス値は自動的に詰められます。

## 5.2.6 フォームをプレビューする

フォームをプレビューすることで、設計中のフォームを実行時のイメージで表示することができます。

### プレビューするには

プレビューは、フォーム編集ウィンドウの[ レイアウト ]メニューから[ プレビュー ]コマンドを選択することにより、表示することができます。プレビュー中は、このコマンドにチェックマークが付きます。



プレビューは、プロジェクトウィンドウのデザインツリーウィンドウでフォームを選択し、ポップアップメニューの[ プレビュー ]コマンドでも表示することができます。

### プレビューを終了するには

プレビューは、以下のどれかの方法で終了することができます。

#### プレビューしたフォームを直接閉じる

プレビューしたフォームのシステムメニューから[ 閉じる ]コマンドを選択するか、タイトルバーの[ 閉じる ]ボタンをクリックします。

### フォーム編集ウィンドウから閉じる

フォーム編集ウィンドウの[レイアウト]メニューから[プレビュー]コマンドを選択します。プレビューを終了すると、このコマンドのチェックマークが表示されなくなります。

### キー操作で閉じる

プレビュー中のフォームに対して、[Enter]キーまたは[Esc]キーを押します。なお、[Enter]キーを処理するようなコントロール（たとえばテキストボックスコントロール）が配置されている場合、終了できない場合があります。

## 5.2.7 フォームを印刷する

設計中のフォームを印刷することができます。フォームは、フォーム編集ウィンドウの[ファイル]メニューから[印刷]コマンドを選択することにより、印刷できます。

印刷する用紙や余白の大きさなどを設定する場合は、[ファイル]メニューの[ページの設定]コマンドで表示されるダイアログボックスを使用してください。



---

プロジェクトウィンドウのデザインツリーウィンドウでフォームを選択し、ポップアップメニューの[印刷]コマンドを使って印刷することもできます。

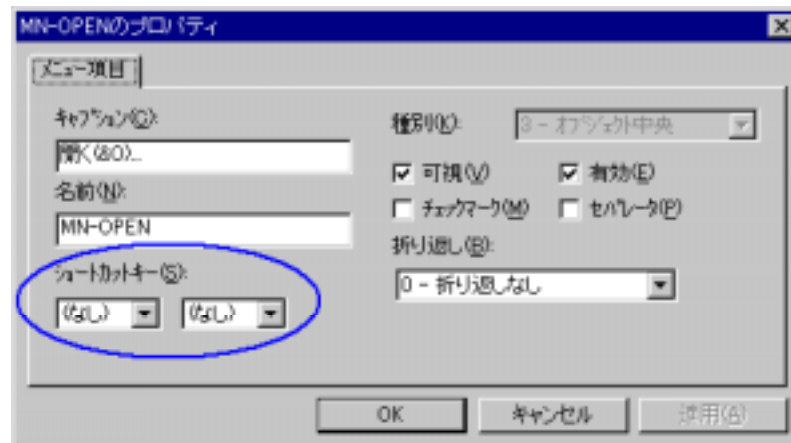
---

## 5.3 メニューの拡張機能

本節では、メニュー編集ウィンドウで利用できる、メニューの拡張機能について説明します。

### 5.3.1 メニュー項目にショートカットキーを割り当てる

メニュー編集ウィンドウで作成するメニュー項目には、ショートカットキーを割り当てることができます。ショートカットキーは、実行時にメニュー項目の右端に表示され、メニュー項目を選択しなくても、キー操作だけで選択を確定させることができるキーです。



ショートカットキーを割り当てるには、メニュー項目のプロパティ設定ダイアログボックスで、[ショートカットキー]を選択します。指定できるショートカットキーを以下に示します。

- [Ctrl+A]から[Ctrl+Z]までの、[Ctrl]キーとアルファベットを組み合わせたキー
- [Del]キー
- [F1]から[F24]までのファンクションキー
- [Ctrl+F1]から[Ctrl+F24]までの、[Ctrl]キーとファンクションキーを組み合わせたキー
- [Shift+F1]から[Shift+F24]までの、[Shift]キーとファンクションキーを組み合わせたキー



注意

ショートカットキーは、トップレベルのメニュー項目（メニューバーに表示されている項目）に割り当ててはできません。

コマンドボタンコントロールにファンクションキーを割り当てている場合、キー操作による優先順位は以下のようになります。

メニュー項目の[種別]がオブジェクトメニューの場合、メニュー項目での設定が優先されます。

メニュー項目の[種別]がフレームメニューの場合、コマンドボタンコントロールでの設定が優先されます。

ファンクションキーは、使用するWindowsの種類、および使用している機種によって、利用できるキーや操作が異なる場合があります。ファンクションキー（とくに[F13以降]）を指定する場合には、アプリケーションを実行するWindowsおよび機種を使って、動作を確認してください。

### 5.3.2 ポップアップメニュー形式で編集する

PowerCOBOLでは、メニュー編集ウィンドウで作成したメニューをポップアップメニューとして利用することもできます。ポップアップメニューとして利用する場合は、実際にポップアップされる形式で編集できるよう、メニューの編集形式を変更できます。ポップアップメニュー形式で編集する場合は、[ポップアップメニュー表示]コマンドを選択します。ポップアップメニュー形式で編集する場合でも、その他の編集操作の方法に変更はありません。



作成したメニューを、実行時にポップアップメニューとして表示するには、フォームやコントロールの"MouseUp"イベントで"PopupMenu"メソッドを呼び出します。ポップアップメニューを利用する方法は、「[ポップアップメニューを使ったアプリケーションを作成する](#)（p219）」を参照してください。

アプリケーションを実行したときに表示されるメニューバーは、フォームのプロパティ設定ダイアログボックスの[スタイル]タブにある[メニューバー名]で指定されたメニューです。したがって、メニュー編集ウィンドウのポップアップメニュー形式で表示していても、[メニューバー名]に名前を設定することで、メニューバーとして使用することもできます。



注意

ポップアップメニュー形式で編集する場合、メニュー項目にショートカットキーを割り当ててはできません。

## 5.4 手続き編集時の便利な機能

本節では、手続きを編集する場合に利用できる便利な機能について説明します。

### 5.4.1 コントロール名を挿入する

フォーム上に配置したコントロールの名前を、簡単に手続き中に挿入できます。コントロール名は、以下のどちらかの操作で挿入することができます。

#### ドラッグアンドドロップによる挿入

1. 手続き編集ウィンドウ上で、コントロール名を挿入する位置にcaretを移動します。
2. フォーム上に配置したコントロールをドラッグし、手続き編集ウィンドウ上でドロップします。

#### コマンドによる挿入

1. フォームまたはプロジェクトウィンドウのデザインツリー上で、コントロールを選択します。
2. [編集]メニューの[コピー]コマンドを選択します。
3. 手続き編集ウィンドウ上で、コントロール名を挿入する位置にcaretを移動します。
4. [編集]メニューの[貼り付け]コマンドを選択します。



手続き編集ウィンドウ上で、文字列が選択された状態でコントロール名を挿入した場合、選択中の文字列がコントロール名に置き換えられます。

### 5.4.2 メソッドやプロパティを挿入する

メソッドやプロパティは、コントロールごとに数多く存在します。この機能を利用することで、各コントロールで使用できるメソッドやプロパティを、簡単に挿入できます。各メソッドおよびプロパティの詳細は、『リファレンス』を参照してください。

メソッドやプロパティは、以下の操作で挿入できます。

1. 手続き編集ウィンドウ上で、コントロール名に対応する文字列を選択状態にします。
2. ポップアップメニューの[メソッドの挿入]または[プロパティの挿入]サブメニューを選択します。
3. コントロールに対応するメソッドまたはプロパティの一覧がメニューに表示されるので、挿入するメソッドまたはプロパティを選択します。



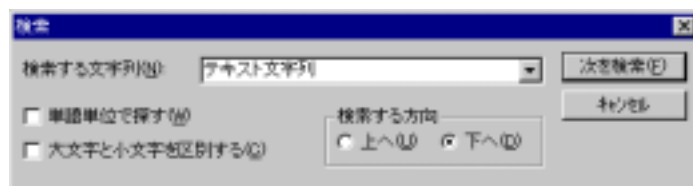
オブジェクト（表コントロール中の1つのセルに対応する"TableCells"オブジェクトなど）のメソッドやプロパティを挿入する場合は、コントロール名とオブジェクトを指すプロパティすべてを選択します。

たとえば、「[アプリケーションを作成しよう](#)（ p49）」で作成した商品の合計金額を計算するサンプルプログラムでは、まず、イベント手続き "TBL-PURCHASE-Return"中の、「"TableCells"(ROW-POS 1) OF TBL-PURCHASE」を選択し、ポップアップメニューの [ プロパティの挿入 ] サブメニューから [ Text ] コマンドを選択します。この結果、以下のようにプロパティが挿入されます。

"Text" OF "TableCells"(ROW-POS 1) OF TBL-PURCHASE

### 5.4.3 文字列を検索・置換する

[ 編集 ] メニューの [ 検索 ] または [ 置換 ] コマンドで表示されたダイアログボックスを使い、文字列を検索または置換できます。







また、プロジェクトのユーティリティ機能を使って、プロジェクト、モジュールまたはフォーム単位で、文字列を検索または置換することもできます。検索ユーティリティおよび置換ユーティリティの詳細は、『リファレンス』を参照してください。

#### 5.4.4 指定行へジャンプする

[編集]メニューの[ジャンプ]コマンドで表示されたダイアログボックスを使って、任意の行を表示し、カレットを移動できます。



#### 5.4.5 注記行を設定する

複数の行を一括して注記行に変更することができます。注記行は[編集]メニューから[注記行の設定]コマンドを選択することで設定することができます。このとき、PowerCOBOLは、標識領域の文字を '\*' (アスタリスク) に変更します。注記行として設定される対象は、以下のようになります。

- 文字列が選択されている場合、その選択文字列を含むすべての行
- 文字列が選択されていない場合、カレットがある行

注記行を解除する場合は、[編集]メニューから[注記行の解除]コマンドを選択します。この操作により、PowerCOBOLは標識領域の文字を空白に変更します。

#### 5.4.6 インデントを利用する

インデントは、[Enter]キーで手続きを改行したとき、[Enter]キーを入力した行の先頭にある空白およびタブを、次の行の先頭へ自動的に入力する機能です。この機能は、複数行の先頭文字の位置をそろえたい場合などに利用します。インデントを使用するかどうかは、オプションのプロパティ設定ダイアログボックスの[エディタ]タブで指定できます。オプションについての詳細は、『リファレンス』を参照してください。

## 5.4.7 文字の色を変更する

手続き編集ウィンドウでは、文字や背景の色を変更することができます。変更は、[ ツール ] メニューの [ 色 ] コマンドで表示されるダイアログボックスを使用して行います。



色の設定ができる項目を、以下に示します。

プログラムテキスト: プログラム中の文字色と手続き編集ウィンドウの背景色を指定します。ただし予約語、注釈、改行、タブおよび全角空白を除きます。

予約語: COBOLの予約語の文字色を指定します。

注釈: COBOLの注記行および行内注記部分の文字色を指定します。

改行、タブと全角空白: 特殊文字 (改行、タブおよび全角空白) の色を指定します。



ワンポイント

[ リセット ] ボタンをクリックすると、[ 設定項目 ] に対応する色を、すべてインストール時の設定に戻します。



注意

手続きで使用する色の情報は、NetCOBOLシリーズの各ツールで共有しています。したがって、以下のどれかのコンポーネントで色を変更すると、その他のコンポーネントにも変更が反映されます。

PowerCOBOL標準の手続き編集ウィンドウ  
PowerGEM Plus エディタ (COBOLエディタ)  
COBOLデバッガ

### 5.4.8 文字のフォントを変更する

手続き編集ウィンドウでは、文字のフォントやサイズを変更できます。変更は、[ツール]メニューの[フォント]コマンドで表示されるダイアログボックスを使用して行うことができます。



### 5.4.9 手続きを印刷する

手続き編集ウィンドウでは、記述した手続きを印刷することができます。印刷は、[ファイル]メニューの[印刷]コマンドで行います。

印刷する用紙や余白の大きさなどを設定する場合は、[ファイル]メニューの[ページ設定]コマンドで表示されるダイアログボックスを使用します。

また、プロジェクトウィンドウでスクリプト(デザインツリーウィンドウの"(スクリプト)[COBOLスクリプト]")を選択し、ポップアップメニューの[手続きの印刷]コマンドを使って、フォームに含まれるすべての手続きを印刷することもできます。

### 5.4.10 外部COBOLファイルを編集する

外部ファイルとしてモジュールに登録したCOBOLのソースファイルも、手続き編集ウィンドウで編集できます。編集する場合は、プロジェクトウィンドウのデザインツリーウィンドウで編集するCOBOLファイルを選択し、ポップアップメニューの[編集]コマンドを選択します。



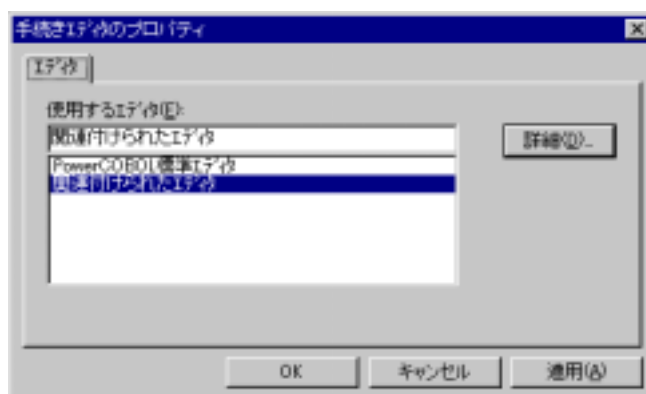
注意

外部COBOLソースファイルを編集する場合、コントロール名の挿入およびメソッドやプロパティ名を挿入する機能を利用することはできません。

### 5.4.11 手続き編集用のエディタを変更する

PowerCOBOLでは、手続きを編集するためのエディタを変更することができます。エディタは、以下の手順で変更します。

1. プロジェクトファイルが開かれている場合は、プロジェクトファイルを閉じます。
2. プロジェクトウィンドウの[ツール]メニューから[カスタマイズ]コマンドを選択します。
3. 手続きエディタのプロパティ設定ダイアログボックスで、使用するエディタを選択します。



以下のエディタが使用できます。

#### PowerCOBOL標準エディタ

PowerCOBOLが標準で提供している、手続き編集ウィンドウを使用します。

#### 関連付けられたエディタ

Windows で、拡張子".COB"に関連付けられているエディタを使用します。拡張子の関連付けについては、Windows のマニュアルを参照してください。また、関連付けられたエディタを使用して手続きを開く場合に、イベント手続きに一連行番号を付加するかどうかを指定することができます。指定は[詳細]ボタンを使って行います。関連付けられたエディタが、COBOLの一連行番号を扱うことができるCOBOL専用のエディタである場合、チェック状態にすることをお勧めします。



拡張子".COB"に関連付けられているエディタがない場合、「メモ帳」(NOTEPAD.EXE)をエディタとして使用します。  
関連付けられているエディタを使用する場合、検索ユーティリティおよび置換ユーティリティを使用することはできません。

## 5.5 実行可能プログラム作成時および実行時の便利な機能

本節では、実行可能プログラムを作成および実行する場合に利用できる、便利な機能について説明します。

### 5.5.1 デフォルトモジュールを設定する

デフォルトモジュールとは、[ プロジェクト ] メニューで以下のコマンドの操作対象になるモジュールです。

- [ ビルド ]
- [ リビルド ]
- [ 実行 ]
- [ デバッグ ]

デフォルトモジュールを設定することにより、プロジェクトウィンドウのデザインツリーウィンドウ上で、モジュールが選択されていない場合でも、ビルドや実行を行うことができます。ただし、モジュールが1つしかない場合は、そのモジュールがデフォルトモジュールとして扱われます。デフォルトモジュールは、[ プロジェクト ] メニューの [ デフォルトモジュールの設定 ] コマンドで表示されるダイアログボックスで選択できます。デフォルトモジュールは、デザインツリーウィンドウ上に太字（ボールド）で表示されます。



注意

デフォルトモジュール以外のモジュールをビルド、リビルド、実行またはデバッグする場合は、デザインツリーウィンドウ上で対象となるモジュールを選択し、ポップアップメニューからコマンドを選択してください。

### 5.5.2 DLLを作成するには

モジュールをDLLとして作成する場合は、以下の手順で実行可能プログラムの種類を設定します。

1. デザインツリーウィンドウで、DLLにするモジュールを選択します。
2. ポップアップメニューの [ プロパティ ] コマンドを選択します。
3. [ 一般 ] タブの [ 種類 ] で、"1 - ダイナミックリンクライブラリ (DLL)" を選択します。

### 5.5.3 ビルドモードを使い分ける

PowerCOBOLでは、「デバッグモード」および「リリースモード」の2つのビルドモードを使用できます。開発状況に応じて使い分けてください。ビルドモード

は、以下の手順で切り替えることができます。

1. デザインツリーウィンドウで、プロジェクトを選択します。
2. ポップアップメニューの [ プロパティ ] コマンドを選択します。
3. [ ビルド ] タブの [ ビルドモード ] で、どちらかのモードを選択します。

### デバッグモード

デバッグモードでモジュールをビルドした場合、デバッグに必要なファイルおよびデバッグ情報が付加された実行可能プログラムが生成されます。PowerCOBOLでアプリケーションをデバッグする場合は、このモードを使用してください。

### リリースモード

リリースモードでモジュールをビルドした場合、デバッグに必要なファイルは生成されません。また、実行可能プログラムにもデバッグ情報は付加されません。したがって、PowerCOBOLでアプリケーションをデバッグすることはできません。このモードは、実際に運用するアプリケーションを作成する場合に使用してください。

## 5.5.4 ビルド時に作成されるファイル

ビルドした場合、以下のようなフォルダおよびファイルが生成されます。

アプリケーション実行時に必要なファイル

アプリケーション開発時に必要なファイル

### アプリケーション実行時に必要なファイル

アプリケーションを実行する場合に必要なファイルは、プロジェクトファイルが保存されているフォルダ配下の、以下のフォルダに生成されます。これらのフォルダをターゲットフォルダといいます。

デバッグモードの場合は、"Debug"

リリースモードの場合は、"Release"

生成されるファイルを以下に示します。

| ファイル名      | 説明                                 |
|------------|------------------------------------|
| モジュール名.EXE | 実行可能プログラムファイル(*1)                  |
| モジュール名.DLL | ダイナミックリンクライブラリ(*2)                 |
| モジュール名.EXP | エクスポートされる関数やデータの情報が格納されているファイル(*3) |
| モジュール名.PLI | 行情報ファイル(*4)                        |
| モジュール名.PDB | COBOLデバッグ情報ファイル(*4)                |
| フォーム名.SVD  | COBOLデバッグ情報ファイル(*5)                |

\*1: 種類が"起動プログラム(EXE)"の場合だけ生成

\*2: 種類が"ダイナミックリンクライブラリ(DLL)"の場合だけ生成

\*3: リンカ(LINK.EXE)が生成

\*4: デバッグモードまたは[診断機能を利用する](#)( p134)場合だけ生成

\*5: リリースモードの場合だけ生成



プロジェクトのプロパティ設定ダイアログボックスで、[ビルド]タブの[ターゲットフォルダ]を指定することにより、これらのファイルの出力先を変更することもできます。



プロジェクトファイルが保存されているフォルダ配下に、ターゲットフォルダと同じ名前のファイルを置かないでください。ターゲットフォルダの生成ができないため、正しくビルドできません。  
また、ターゲットフォルダ内のファイルを書き込み禁止属性にしている場合、ファイルの置き換えができないため、正しくビルドできません。

#### アプリケーション開発時に必要なファイル

ビルドやデバッグなど、アプリケーションを開発する場合に必要なファイルは、プロジェクトファイルが保存されているフォルダ配下の、以下のフォルダに生成されます。これらのフォルダを作業用フォルダといいます。

デバッグモードの場合："モジュール名¥Debug"

リリースモードの場合："モジュール名¥Release"

生成されるファイルを以下に示します。

| ファイル名      | 説明                     |
|------------|------------------------|
| モジュール名.DEF | モジュール定義ファイル            |
| モジュール名.PMI | モジュール情報ファイル            |
| モジュール名.RC  | リソースファイル               |
| モジュール名.RES | .RCファイルのコンパイル結果        |
| モジュール名.TLB | タイプライブラリ               |
| フォーム名.PRC  | イベント手続きファイル            |
| フォーム名.COB  | .PRCから生成したCOBOLソースファイル |
| フォーム名.OBJ  | .COBファイルのコンパイル結果       |
| フォーム名.SLI  | 行番号対応情報ファイル            |
| フォーム名.SVD  | COBOLデバッグ情報ファイル(*1)    |
| ~BUILD.TLB | タイプライブラリ               |

\*1: デバッグモードの場合だけ生成



プロジェクトファイルが保存されているフォルダ配下に、作業用フォルダと同じ名前のファイルを置かないでください。作業用フォルダの生成ができないため、正しくビルドできません。  
また、作業用フォルダ内のファイルを書き込み禁止属性にしている場合、ファイルの置き換えができないため、正しくビルドできません。

### 5.5.5 NetCOBOLのオブジェクト指向プログラミング機能を利用する

PowerCOBOLでは、NetCOBOLのオブジェクト指向プログラミング機能を利用できます。この機能を利用する場合は、モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブにある [ スクリプト言語 ] を "1 - 00COBOL言語仕様" に変更してください。

NetCOBOLのオブジェクト指向プログラミング機能については、『NetCOBOL 使用手引書』を参照してください。



00COBOL言語仕様は、COBOL85言語仕様とは異なるため、モジュール内に手続きが存在する場合には使用する言語仕様を変更することはできません。

### 5.5.6 アプリケーションの多重起動を制御する

アプリケーションを運用する場合、その運用形態により、アプリケーションが多重起動されるのを制御したい場合があります。このような場合は、起動プログラム(EXE)となるモジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブにある [ 多重起動 ] から [ 単一起動(メッセージ) ] または [ 単一起動(アクティベート) ] を選択してください。[ 単一起動(メッセージ) ] を選択した場合、すでに実行されているアプリケーションと同一のアプリケーションを起動すると、実行できないことを示すメッセージが表示されます。[ 単一起動(アクティベート) ] を選択した場合は、すでに実行されているアプリケーションにフォーカスに移り、アクティブな状態になります。

ただし、PowerCOBOLは、同一のフォルダにある実行ファイルに対してだけ、多重起動を制御します。したがって、以下のような場合には、異なるアプリケーションとしてみなされます。

ローカルパスで起動されたアプリケーションとネットワークパスで起動されたアプリケーション

ファイルのコピーなどにより、別のフォルダ(パス)で起動されたアプリケーション

また、多重起動の設定は、起動プログラム(EXE)に対してだけ有効です。ダイナミックリンクライブラリ(DLL)への設定はできません。





多重起動の制御は、ファイルへの多重アクセスを禁止（排他制御）するために利用するものではありません。たとえば、PowerCOBOLで多重起動を制御しても、他のアプリケーションからのファイルアクセスを制御することはできません。ファイルにアクセスする場合には、COBOLのLOCK MODE句、OPEN文でのLOCK指定などを使って、ファイルレベルで排他制御をしてください。ファイルの排他制御についての詳細は、『COBOL 文法書』を参照してください。

### 5.5.7 ビルド用のオプションを設定する

ビルドに使用されるオプションには、以下のオプションがあります。

翻訳オプション  
 プリコンパイラの設定  
 登録集ファイルの設定  
 登録集名の設定  
 リンクオプション  
 リソースオプション

#### 翻訳オプション

翻訳オプションは、スクリプト（デザインツリーウィンドウの(スクリプト) [ COBOLスクリプト ]）のプロパティ設定ダイアログボックスで設定します。翻訳オプションは、フォーム単位で指定できます。また、モジュールへ外部ファイルとして登録されたCOBOLソースファイルは、登録したファイルごとに、プロパティ設定ダイアログボックスから設定できます。



PowerCOBOLでは、以下の翻訳オプションを指定できます。

複数の翻訳オプションを指定する場合は、空白を含めずコンマで区切ります。

|          |         |             |       |       |
|----------|---------|-------------|-------|-------|
| ALPHAL   | FILELIB | OPTIMIZE    | SDS   | TRUNC |
| CURRENCY | FORMEXT | QUOTE/APOST | SSIN  | ZWB   |
| EQUALS   | FORMLIB | REP         | SSOUT |       |
| FILEEXT  | NCW     | REPIN       | STD1  |       |

翻訳オプションの詳細は、『NetCOBOL 使用手引書』を参照してください。

なお、以下の環境変数でも翻訳オプションを指定できます。

COB\_OPTIONS (コマンドラインオプション)

FILELIB (ファイル定義体格納フォルダ)

FFD\_SUFFIX (ファイル定義体拡張子)

FORMLIB (画面帳票定義体格納フォルダ)

SMED\_SUFFIX (画面帳票定義体拡張子)

COB\_REPIN (リポジトリ格納フォルダ)

指定した翻訳オプションと環境変数が重複した場合、優先順位は、以下のようになります。

1. [ 翻訳 ] タブで指定したオプション
2. 環境変数で指定したオプション

指定されている環境変数を確認する場合には、プロジェクトウィンドウの[ ツール ] メニューから、[ 環境変数表示 ] コマンドを選択します。

環境変数についての詳細は、『NetCOBOL 使用手引書』を参照してください。



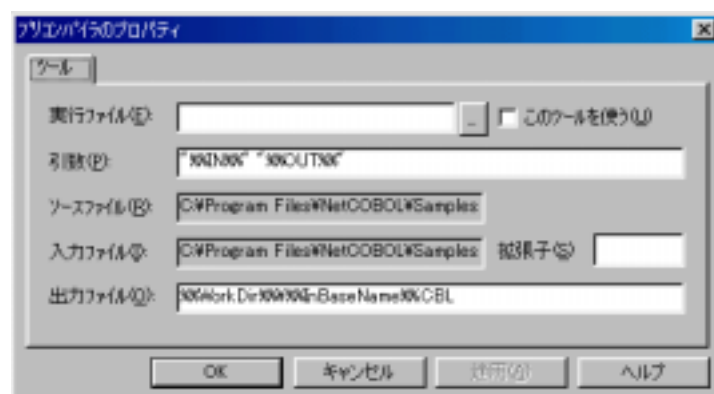
注意

PowerCOBOLでは、ALPHALオプションは指定できますが、NOALPHALオプションは指定できません。英字の小文字と大文字を区別して扱う場合には、ALPHAL(WORD)オプションを使用してください。

## プリコンパイラの設定

PowerCOBOLのイベント手続き、または外部ファイルに登録したCOBOLソース内にSQL文を記述した場合、COBOLコンパイラで翻訳する前にプリコンパイルする必要があります。

PowerCOBOLでは、翻訳の前に呼び出すプリコンパイラを指定することができます。プリコンパイラは、スクリプトのプロパティ設定ダイアログボックスの[ 設定 ] ボタンをクリックして表示されるダイアログボックスで、設定することができます。



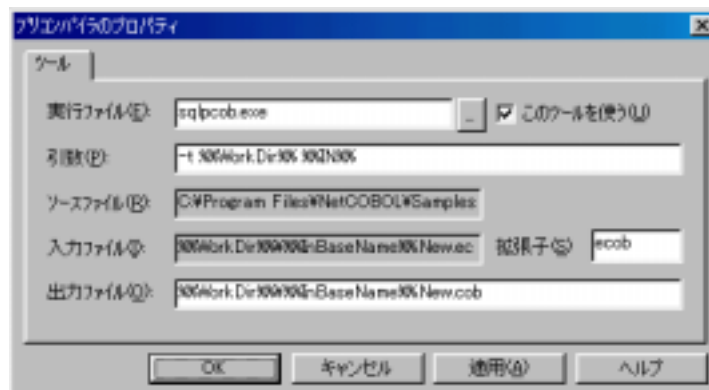
引数および出力ファイルは、以下のマクロ文字列を使って指定します。

| マクロ文字列         | 意味                                                 |
|----------------|----------------------------------------------------|
| %%IN%%         | プリコンパイラの入力ファイル名、つまり[ 入力ファイル ]<br>に表示されたファイル名を示します。 |
| %%InBaseName%% | ソースファイルの拡張子を除いたファイル名を示します。                         |
| %%InDir%%      | ソースファイルのパス名を示します。                                  |
| %%InSuffix%%   | ソースファイルの拡張子を示します。                                  |
| %%OUT%%        | プリコンパイラの実出力ファイル名、つまり[ 出力ファイル ]<br>で指定したファイル名を示します。 |
| %%OutDir%%     | 出力ファイルのパス名を示します。                                   |
| %%ProjectDir%% | プロジェクトファイル(.PPJ)のパス名を示します。                         |
| %%WorkDir%%    | アプリケーション開発時に必要なファイルが格納される<br>作業用フォルダのパス名を示します。     |

各設定項目についての詳細は、『リファレンス』を参照してください。

### SymfoWARE EsqI-COBOLを使用する場合の設定例

PowerCOBOLのイベント手続きに対して、『SymfoWARE Programmer's kit for Windows NT V1.1L10』の『EsqI-COBOL』を使用する場合の設定例を、以下に示します。

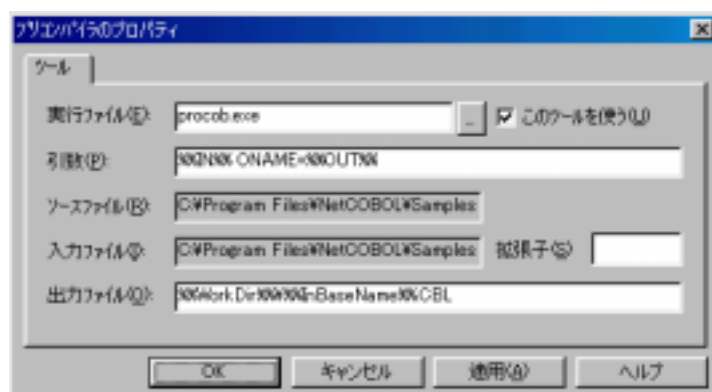


注意

『EsqI-COBOL』では、入力ファイルの拡張子に".COB"を使用することができません。したがって、必ず[ 拡張子 ]を指定する必要があります。また、出力ファイル名は、必ず入力ファイル名の拡張子が".COB"に変換されたものになるため、[ 出力ファイル ]に指定する拡張子は".COB"にしてください。この例では、ソースファイル名と重ならないようにするため、ファイル名に"New"を追加しています。

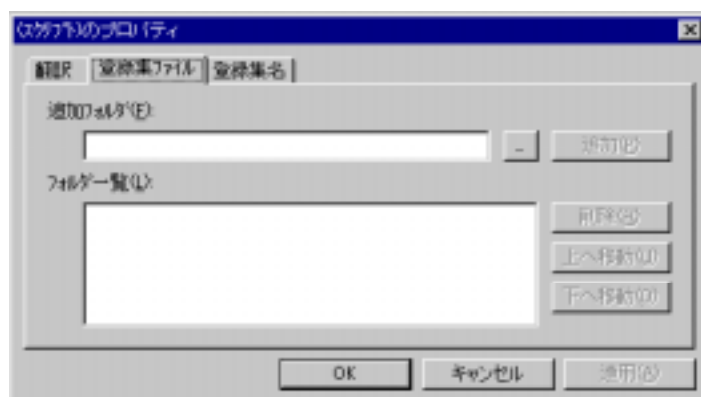
### Oracle Pro\*COBOLを使用する場合の設定例

外部ファイルとして登録したCOBOLソースファイルに対して、Oracle の『Oracle Pro\*COBOL for Windows 95/NT V1.8』を使用する場合の設定例を、以下に示します。



### 登録集ファイルのフォルダの設定

登録集ファイルのフォルダは、[登録集ファイル]タブで設定できます。



なお、以下の環境変数でも登録集ファイルのフォルダを指定できます。

COB\_COBCOPY (登録集格納フォルダ)

登録集ファイルのフォルダの検索順序は、以下のようになります。

1. [登録集ファイル]タブで指定したフォルダ
2. 環境変数で指定したフォルダ

指定されている環境変数を確認する場合には、プロジェクトウィンドウの[ツール]メニューから、[環境変数表示]コマンドを選択します。

環境変数についての詳細は、『NetCOBOL 使用手引書』を参照してください。

### 登録集名の設定

登録集名は、[登録集名]タブで設定できます。



なお、以下の環境変数でも登録集名を指定できます。

COB\_登録集名（登録集名格納フォルダ）

登録集名格納フォルダの検索順序は、以下のようになります。

1. [登録集名] タブで指定したフォルダ
2. 環境変数で指定したフォルダ

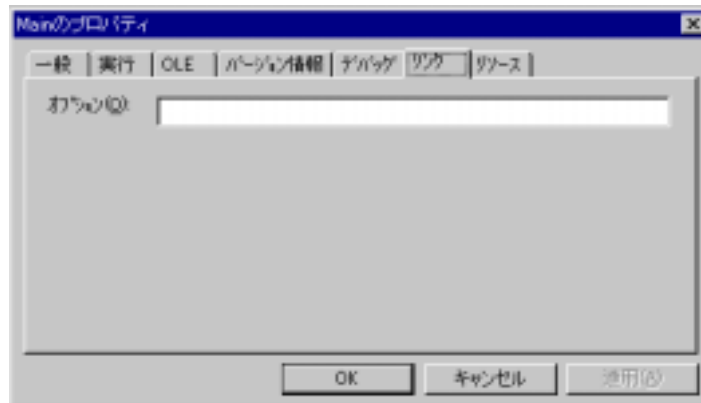
指定されている環境変数を確認する場合には、プロジェクトウィンドウの[ツール]メニューから、[環境変数表示] コマンドを選択してください。

環境変数についての詳細は、『NetCOBOL 使用手引書』を参照してください。

### リンクオプション

リンクオプションは、"LINK.EXE"に渡されるオプションです。

リンクオプションは、モジュールのプロパティ設定ダイアログボックスの[リンク]タブで設定することができます。リンクオプションの詳細については、『NetCOBOL 使用手引書』を参照してください。



### リソースオプション

リソースオプションは、"RC.EXE"に渡されるオプションです。

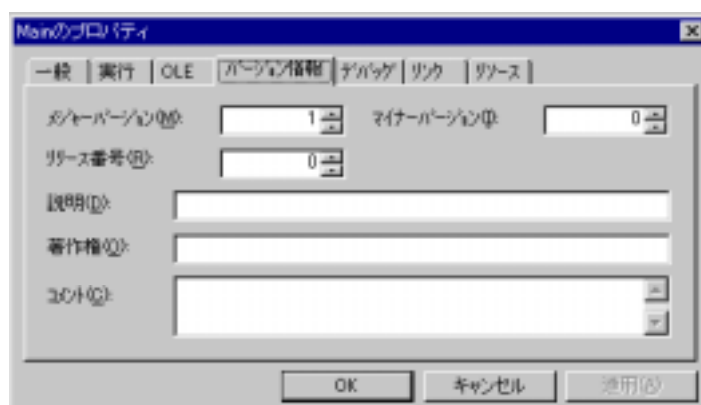
リソースオプションは、モジュールのプロパティ設定ダイアログボックスの[リソース]タブで設定できます。



### 5.5.8 実行可能プログラムのバージョンを設定する

PowerCOBOLで作成する実行可能プログラムには、バージョン情報を付けることができます。バージョン情報とは、Windows のエクスプローラなどで実行可能プログラムのプロパティを参照した場合、[バージョン情報] タブに示される情報です。

バージョン情報は、モジュールのプロパティ設定ダイアログボックスの[バージョン情報] タブで設定できます。



各設定項目についての詳細は、『リファレンス』を参照してください。

### 5.5.9 バッチモードでビルドする

PowerCOBOLでは、プロジェクトファイルに含まれるすべてのモジュールを、バッチモードでビルドまたはリビルドすることができます。(以降、バッチビルドと呼びます)

バッチビルドは、Windows のコマンドプロンプト (DOSプロンプト) で実行します。

## バッチビルドコマンドの指定形式

バッチビルドするためのコマンドは、以下の形式で指定します。

```
PowerCOB { /build | /rebuild }
 [/Debug | /Release]
 [/cbi : " 翻訳オプションファイル名 "]
 " プロジェクトファイル名 "
```

ビルドまたはリビルドした結果は、プロジェクトファイルと同じフォルダに、"プロジェクト名.BLG" というファイル名で出力されます。

### /build | /rebuild

プロジェクトファイルに含まれるすべてのモジュールをビルドする場合は、/buildを、リビルドする場合は、/rebuildを指定します。

### /Debug | /Release

デバッグモードでバッチビルドする場合は、/Debugを、リリースモードでバッチビルドする場合は、/Releaseを指定します。

省略した場合、プロジェクトのプロパティ設定ダイアログボックスで指定されているビルドモードになります。

### /cbi : "翻訳オプションファイル名"

プロジェクトファイルに指定してある翻訳オプション(登録集フォルダおよび登録集名も含む)を無効にし、翻訳オプションファイルに指定した翻訳オプションを有効にします。このオプションが指定されていない場合は、外部COBOLファイルおよびスクリプトのプロパティ設定ダイアログボックスで指定されている翻訳オプションが有効になります。

### "プロジェクトファイル名"

ビルドまたはリビルドするプロジェクトファイル名です。ただし、PowerCOBOLのV3.0以前のバージョンで作成されたプロジェクトファイル(拡張子がPRJ)は、指定できません。

## 復帰値

バッチビルドの復帰値は、以下のとおりです。

ビルドに成功 : 0

ビルドに失敗 : 1

なお、Iレベル、Wレベルのエラーがある場合、またはすべて更新されている状態でビルドしたときにもビルドは成功します。バッチビルド完了後、ビルド結果を確認してください。



ワンポイント

---

バッチビルドのコマンド指定では、大文字と小文字の区別はありません。

---

## バッチビルドの使用例

バッチビルドの使用例を以下に示します。

1つのプロジェクトファイルを、プロジェクトファイルに指定されているとおりにビルドします。

1つのプロジェクトファイルを、翻訳オプションファイルを指定し、デバッグモードでリビルドします。

複数のプロジェクトファイルをリビルドし、ビルド結果を1つのファイルに出力します。

### バッチビルドの使用例1

PowerCOBOLで作成したプロジェクトファイル"C:¥PROJ1¥PROJ1.PPJ"を、バッチビルドする場合、以下のように指定します。

```
powercob /build "c:¥proj1¥proj1.ppj"
```

バッチビルドの結果 ("C:¥PROJ1¥PROJ1.BLG") は、以下のようになります。

```
コンパイルのための型情報を作成しています...
c:¥proj1¥Main¥Debug¥MainForm.cob を作成しています...
最大重大度コードは 1 です。
c:¥proj1¥Main¥Debug¥MainForm.cob をコンパイルしています...
最大重大度コードは 1 で、翻訳したプログラム数は 1 本です。
c:¥proj1¥Main¥Debug¥Main.rc をリソースコンパイルしています...
c:¥proj1¥Debug¥Main.exe をリンクしています...
** ビルドが成功しました **
```

### バッチビルドの使用例2

PowerCOBOLで作成したプロジェクトファイル"C:¥PROJ2¥PROJ2.PPJ"を、翻訳オプションファイル"C:¥PROJ2¥PROJ2.CBI"を有効にして、デバッグモードでリビルドする場合、以下のように指定します。

```
powercob /rebuild /debug /cbi:"c:¥proj2¥proj2.cbi" "c:¥proj2¥proj2.ppj"
```

エラーがあった場合、バッチビルドの結果 ("C:¥PROJ2¥PROJ2.BLG") は、以下のようになります。

```
コンパイルのための型情報を作成しています...
c:¥proj2¥sub1¥Debug¥sub1form.cob を作成しています...
最大重大度コードは 1 です。
c:¥proj2¥sub1¥Debug¥sub1form.cobをコンパイルしています...
** 診断メッセージ ** (SUB1FORM)
sub1form sub1form-Opened(5):JMN2503I-S 利用者語'data1'が定義されていません。
sub1form sub1form-Opened(5):JMN2557I-S DISPLAY文の書き方が不完全です。
最大重大度コードは S で、翻訳したプログラム数は 1 本です。
** ビルドに失敗しました **
```

### バッチビルドの使用例3

複数のプロジェクトファイルを一括してリビルドし、ビルドの結果を1つのファイルに出力する場合は、以下のようなバッチファイル"ALLBUILD.BAT"を作成し、実行します。



```

ECHO OFF
ECHO ##### C:¥PROJ1¥PROJ1.PPJ ##### >> %2
ECHO POWERCOB /%1 "C:¥PROJ1¥PROJ1.PPJ"
START /WAIT POWERCOB /%1 "C:¥PROJ1¥PROJ1.PPJ"
IF ERRORLEVEL 1 ECHO !!! %1 Error !!!
TYPE C:¥PROJ1¥PROJ1.BLG >> %2
ECHO ##### C:¥PROJ2¥PROJ2.PPJ ##### >> %2
ECHO POWERCOB /%1 "C:¥PROJ2¥PROJ2.PPJ"
START /WAIT POWERCOB /%1 "C:¥PROJ2¥PROJ2.PPJ"
IF ERRORLEVEL 1 ECHO !!! %1 Error !!!
TYPE C:¥PROJ2¥PROJ2.BLG >> %2
ECHO ##### C:¥PROJ3¥PROJ3.PPJ ##### >> %2
ECHO POWERCOB /%1 "C:¥PROJ3¥PROJ3.PPJ"
START /WAIT POWERCOB /%1 "C:¥PROJ3¥PROJ3.PPJ"
IF ERRORLEVEL 1 ECHO !!! %1 Error !!!
TYPE C:¥PROJ3¥PROJ3.BLG >> %2
.
.
.
:END

```

実行するコマンドは以下のようになります。

```
allbuild rebuild c:¥temp¥allbuild.blg
```

実行が終了したら、バッチビルド結果 ( "C:¥TEMP¥ALLBUILD.BLG" ) を確認してください。



別のフォルダに移動したプロジェクトファイルをバッチビルドすると、「ビルドまたはコンパイルのために、プロジェクトを保存しますか？」というメッセージボックスが表示されます。この場合、メッセージボックスに応答しないと処理が進まないため、バッチ的なビルドができなくなります。メッセージボックスを表示したくない場合、オプションのプロパティ設定ダイアログボックスの [ ビルド ] タブで、[ ビルド時に自動的に保存する ] をチェック状態にしておきます。

なお、自動的に保存したくない場合は、チェックをはずすか、プロジェクトファイルの属性を読み取り専用にしてください。

### 翻訳オプションファイルの作成方法

翻訳オプションファイルを作成するには、WINCOBコマンドを使用します。WINCOBコマンドは、COBOLが提供しているCOBOLソースを翻訳するコマンドです。このWINCOBコマンドを以下の指定形式で起動すると、[ 翻訳オプション ] ダイアログボックスが表示されます。ここで翻訳オプションを指定してください。

指定形式: WINCOB -i オプションファイルのパス名

使用例 : WINCOB -ic:¥proj2¥proj2.cbi

WINCOBコマンドについては、『NetCOBOL 使用手引書』を参照してください。



/cbi オプションを使用して登録集ファイルを変更する場合、翻訳オプション LIB を使用してください。

### 5.5.10 アプリケーションの動作環境を設定する

作成したアプリケーションを実行する場合、以下の動作環境を設定できます。

DLL を作成した場合は、その DLL を呼び出す起動ファイルの設定

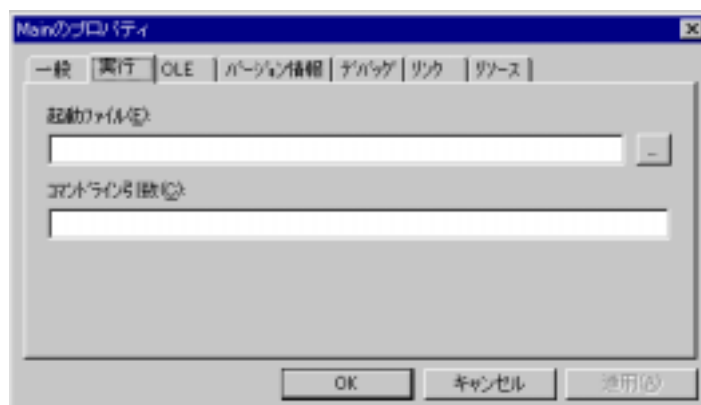
実行可能プログラムに渡すコマンドライン引数の設定

エラーメッセージを表示するかどうかの設定

NetCOBOL の実行環境の設定

#### 起動ファイルを設定する

作成した実行可能プログラムが DLL の場合、そのままでは実行できません。DLL を呼び出す起動ファイルの指定が必要です。起動ファイルは、モジュールのプロパティ設定ダイアログボックスの [ 実行 ] タブで設定することができます。ただし、この指定は、PowerCOBOL の開発環境上で実行またはデバッグする場合にだけ有効です。



#### コマンドライン引数を設定する

作成した実行可能プログラムにコマンドライン引数を渡す場合、コマンドライン引数をモジュールのプロパティ設定ダイアログボックスの [ 実行 ] タブで設定できます。ただし、この指定は、PowerCOBOL の開発環境上で実行またはデバッグする場合だけ有効です。

#### エラーメッセージの表示・非表示を切り替える

アプリケーション実行時に、フォームやコントロールのプロパティへのアクセスやメソッドの呼び出しでエラーがあった場合、PowerCOBOL のランタイムシステムは、メッセージボックスを使ってエラーを表示します。このメッセージボックスは、以下のコマンドを実行することにより抑止できます。この設定は、

PowerCOBOLがインストールされたWindows 上で、PowerCOBOLで作成したすべてのアプリケーションに対して有効となります。

#### エラーメッセージを表示する場合

F5DDSTEV /ERRMSGBOX:SHOW

#### エラーメッセージを表示しない場合

F5DDSTEV /ERRMSGBOX:HIDE



注意

"F5DDSTEV.EXE"は、PowerCOBOLをインストールしたフォルダに格納されています。このコマンドは、Windows のコマンドプロンプト(MS-DOSプロンプト)で実行してください。

#### NetCOBOLの実行環境を設定する

NetCOBOLの実行環境は、[ プロジェクト ]メニューの[ 実行環境設定 ]コマンドを選択することにより、[ 実行環境設定ツール ]を使って、設定できます。NetCOBOLの実行環境の詳細については、『NetCOBOL 使用手引書』を参照してください。NetCOBOLの実行環境は、ターゲットフォルダに"COBOL85.CBR"という名前で保存されます。

#### その他の動作環境を設定する

アプリケーションを実行する場合には、その他に、以下の環境が正しく設定されているか確認してください。

##### PATH

異なるフォルダに存在するファイル、たとえばDLLなどを使用しているアプリケーションを実行する場合、そのファイルが参照できるよう正しくPATH環境が設定されているか確認してください。

### 5.5.11 インストーラを作成する

作成したアプリケーションを他のシステムにインストールして利用する場合、以下のようにしてインストーラを作成します。

1. プロジェクトウィンドウの[ ファイル ]メニューから[ インストーラの作成 ]コマンドを選択します。  
「インストーラを作成しました」というメッセージが表示されます。
2. OKボタンをクリックします。

インストーラを作成すると、ターゲットフォルダに以下のファイルが作成されます。

SETUP.EXE : インストーラ  
SETUP.INF : インストーラが参照する情報ファイル  
F5DDSTEV.EXE : インストール先のシステムで動作環境を設定するためのファイル

アプリケーションを他のシステムにインストールする場合は、上記のファイルを含め、ターゲットフォルダ内のすべてのファイルをコピーし、"SETUP.EXE"を実行してください。

たとえば、ターゲットフォルダ内のすべてのファイルをCD-Rに書き込んでおけば、インストーラ付きのアプリケーション用CD-Rとして利用できます。



インストーラを作成する前に、ビルドモードをリリースモードにしておくと、インストールされるファイルのサイズを小さくできます。

インストールしたアプリケーションは、Windows のコントロールパネルから「アプリケーションの追加と削除」を使ってアンインストールできます。



インストールしたアプリケーションを他のシステムで実行するには、NetCOBOLおよびPowerCOBOLのランタイムシステムが必要です。インストーラを実行する前に、これらのランタイムシステムをインストールしてください。

### 5.5.12 診断機能を利用する

診断機能は、アプリケーションエラーが発生した場合や実行時メッセージが表示されたときに、プログラムを診断し、診断情報をレポートファイルに出力します。レポートファイル名は、アプリケーションエラーが発生した場合に表示されるメッセージの中に表示されます。

PowerCOBOLの診断機能は、COBOLの診断機能をもとに構築されています。診断機能の詳細については、『NetCOBOL 使用手引書』を参照してください。

以下に、診断機能を利用するために必要な環境と、PowerCOBOL固有の診断結果の見かたについて説明します。

#### 診断機能を利用するには

PowerCOBOL固有の診断機能の対象となるのは、モジュールに登録された外部COBOLファイルの手続きおよびフォームのスク립トにある手続きです。モジュールに登録されたオブジェクトファイルやライブラリファイルに対しては、COBOLの診断機能に準じた情報が出力されます。

PowerCOBOL固有の診断機能を利用する場合、以下のファイルが必要です。

各ファイルが生成されるフォルダについては、「[ビルド時に作成されるファイル](#)」( p120) を参照してください。

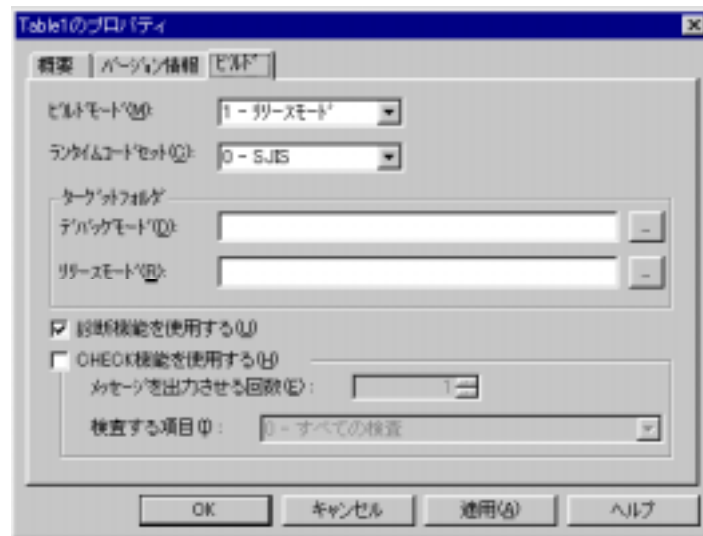
| ファイル名      | 意味                                                  |
|------------|-----------------------------------------------------|
| フォーム名.SVD  | COBOLのデバッグ情報ファイルです。COBOLの診断機能を利用するために必要です。          |
| モジュール名.PLI | PowerCOBOLの行情報ファイルです。PowerCOBOL固有の診断情報を出力するために必要です。 |

PowerCOBOL固有の診断機能を利用するには、PowerCOBOLの行情報ファイル"モジュール名.PLI"が必要です。このファイルは、ターゲットフォルダに生成されます。このファイルを削除すると、PowerCOBOL固有の診断情報は出力されず、COBOLの診断機能に準じた情報だけが出力されます。

ビルドモードにより、診断機能を利用するための設定方法が異なります。

### リリースモードの場合

ビルドモードがリリースモードの場合、プロジェクトのプロパティ設定ダイアログボックスの[ビルド]タブで、[診断機能を使用する]をチェック状態にしてビルドすると、診断機能を利用することができます。



リリースモードで作成されるデバッグ情報ファイル"フォーム名.SVD"は、翻訳オプション"OPTIMIZE"を指定することによってサイズを小さくできます。



リリースモードで作成した実行可能プログラムを、他のフォルダに複写または移動して実行する場合、実行可能プログラムと同じフォルダにあるデバッグ情報ファイル(\*.SVDおよび\*.PLI)も複写または移動してください。デバッグ情報ファイルを複写または移動しなかった場合、言語イメージの診断情報、およびPowerCOBOL固有の診断情報は出力されません。

### デバッグモードの場合

ビルドモードがデバッグモードの場合、プロジェクトのプロパティ設定ダイアログボックスの[ビルド]タブの[診断機能を使用する]のチェック状態に関係なく、診断機能を利用できます。



---

複数のプロジェクトから構成されるプログラムで診断機能を利用する場合には、最初に起動されるモジュールのプロパティ設定ダイアログボックスで[デバッグ]タブにある[デバッグ情報ファイル格納フォルダ]に、他のプロジェクトで作成したモジュールのデバッグ情報ファイル"モジュール名¥Debug¥フォーム名.SVD"が格納されているフォルダを設定してください。

---



---

デバッグモードで作成した実行可能プログラムを、PowerCOBOLの開発環境の外で実行させる場合や、他のフォルダに複写または移動して実行させる場合、デバッグ情報ファイルが格納されているフォルダを参照できないため、言語イメージの診断情報およびPowerCOBOL固有の診断情報は出力されません。

実行環境変数"@GOPT"でエラー検出時の処理実行回数が指定されている場合、PowerCOBOLで指定する[メッセージを出力させる回数]より、実行環境変数"@GOPT"での指定が優先されます。

---

## PowerCOBOL固有部分の診断結果の見かた

診断機能のレポートに出力されるPowerCOBOL固有の部分の見かたを説明します。以下に、診断結果の出力例を示します。

```

:
《問題箇所》
スロット ID : 000000AC
レジスタ : EAX=00000000 EBX=0012FAA4 ECX=00000000 EDX=00000000 ESI=0012FA68
 : EDI=0012FB8C EIP=00401377 ESP=0012FA08 EBP=0012FB54 EFL=00000283
 : CS=001B SS=0023 DS=0023 ES=0023 FS=0038 GS=0000
スタックコミット : 00004000 (トップ:00130000, ヘッス:0012C000)
命令 : アドレス +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f
 : 00401367 00 0F BF 05 42 60 40 00 99 0F BF 0D 3A 60 40 00
フォールト : ->00401377 F7 F9 66 89 45 DA 0F BF 45 DA B1 04 B5 00 8D 3D

モジュールファイル : C:\Samples\Release\Main.exe
セクション相対位置 : .text+00000377
EXPORT相対位置 : MAINFORM+00000343
シンボル相対位置 : MAINFORM+00000377
翻訳情報 : ShiftJIS, シングルスロット, NOOPTIMIZE
PowerCOBOLプロジェクトファイル : C:\Samples\Abend01.ppj
モジュール : Main
フォーム : MainForm
スクリプト : MainForm-Click
行位置 : 8

<呼出経路>
[1]-----
モジュールファイル : C:\Samples\Release\Main.exe
セクション相対位置 : .text+00000F8
EXPORT相対位置 : MAINFORM+000001C4
シンボル相対位置 : MAINFORM+000001F8
翻訳情報 : ShiftJIS, シングルスロット, NOOPTIMIZE
PowerCOBOLプロジェクトファイル : C:\Samples\Abend01.ppj
モジュール : Main
フォーム : MainForm
行位置 : 35
[2]-----
:

```

### **PowerCOBOLプロジェクトファイル**

異常が検出されたPowerCOBOLのプロジェクトファイル名です。診断された異常が、このプロジェクトで作成された実行可能プログラム内で発生していることを示しています。PowerCOBOLの行情報ファイルが見つからない場合は、不明(Unknown)と表示されます。

### **モジュール**

異常が検出されたモジュールの名前です。このモジュールに含まれる手続きの中に異常があります。

### **フォーム**

異常が検出されたフォームの名前です。このフォームに含まれるスクリプトの中に異常があります。

### **スクリプトレット**

異常が検出された手続きの名前です。この手続きの中に異常があります。

### **COBOLファイル**

PowerCOBOLのモジュールに登録された外部COBOLファイル内で異常が検出された場合は、そのオブジェクト名と、ファイル名が出力されます。

### **行位置**

異常が検出された手続き中での行番号、またはCOBOLファイル中でのファイル内相対行番号です。



### 中断位置・情報ファイルと出力される情報との関係

異常が検出され中断された位置の情報は、その中断位置と情報ファイルにより、以下の表のようになります。

| 情報名                            | 中断位置    |             |            | 情報ファイル |          |            | 備考           |
|--------------------------------|---------|-------------|------------|--------|----------|------------|--------------|
|                                | イベント手続き | 外部COBOLファイル | フォームの外部手続き | 行情報なし  | デバッグ情報なし | 行・デバッグ情報なし |              |
| モジュールファイル                      |         |             |            |        |          |            |              |
| セクション相対位置                      |         |             |            |        |          |            |              |
| EXPORT相対位置                     |         |             |            |        |          |            |              |
| シンボル相対位置                       |         |             |            |        | *2       | *2         |              |
| PowerCOBOL<br>プロジェクトファイル<br>あり |         |             |            | *1     | -        | -          | PowerCOBOL固有 |
| モジュール                          |         |             |            | -      | -        | -          | PowerCOBOL固有 |
| フォーム                           |         | -           |            | -      | -        | -          | PowerCOBOL固有 |
| スクリプトレット                       |         | -           | -          | -      | -        | -          | PowerCOBOL固有 |
| COBOLファイル                      | -       |             | -          | -      | -        | -          | PowerCOBOL固有 |
| 行位置                            |         |             |            | -      | -        | -          | PowerCOBOL固有 |
| ソースファイル                        | -       | -           | -          |        | -        | -          |              |
| 外部プログラム<br>/クラス                | -       |             | -          |        | -        | -          |              |
| 内部プログラム<br>/メソッド               | -       |             | -          |        | -        | -          |              |
| 文位置                            | -       | -           | -          |        | -        | -          |              |

\*1: PowerCOBOL行情報ファイルが見つからない場合には、PowerCOBOLプロジェクトファイルは不明(Unknown)となります。

\*2: LINKのオプションの指定(/DEBUG, /DEBUGTYPE: {COFF|BOTH})により、出力されます。

### 5.5.13 CHECK機能を利用する

CHECK機能は、[診断機能](#) ( p134 )とともに利用することで、実行時にエラーが発生した手続きの位置を求めることができます。

PowerCOBOLのCHECK機能は、COBOLのCHECK機能をもとに構築されています。CHECK機能の詳細については、『NetCOBOL 使用手引書』を参照してください。

#### CHECK機能を利用するには

プロジェクトのプロパティ設定ダイアログボックスの[ビルド]タブで、[CHECK機能を使用する]をチェック状態にしてビルドすると、実行時にCHECK機能を利用することができます。

CHECK機能を利用する場合の、各設定項目についての詳細は、『リファレンス』を参照してください。

アプリケーションを実行し、CHECK機能による検査でエラーが発生すると、CHECK機能によるメッセージおよび診断機能によるメッセージが表示されます。診断機能によるメッセージ内に表示されたレポートファイル内を参照することにより、エラーが発生した手続きの位置を求めることができます。



CHECK機能は、以下の2つの状況で利用すると効果的です。

プログラミングが完了したあと、デバッグモードでのテスト期間中にCHECK機能を利用します。CHECK機能を有効にしておくことで、より多くのトラブルの早期検出に役立ちます。テスト終了後は、リリースモードに変更し、CHECK機能を解除します。

運用中にトラブルが発生した場合は、CHECK機能を有効にして作成した実行可能プログラムと置き換えます。実行可能プログラムを置き換えることにより、トラブルの原因を検出できる場合があります。



CHECK機能を利用すると、データ内容の検査など、プログラムで記述した以外の処理が追加されます。そのため、プログラムのサイズが大きくなり、実行速度も遅くなります。プログラムのデバッグが終了したら、[CHECK機能を使用する]のチェックをはずして、プロジェクト全体をビルドしなおすことをお勧めします。

メッセージ出力回数を指定することにより、エラー検出後も実行を継続できます。ただし、エラー検出後の動作は保証されないため、通常は、1回めにメッセージが出力されるように指定してください。

---

## 第6章      アプリケーションをデバッグしよう

---

PowerCOBOLで作成したアプリケーションが正しく動作しない場合、PowerCOBOLの対話型デバッガを使ってプログラムをデバッグすることにより、動作およびデータを検証し、プログラムの誤りを検出できます。  
本章では、PowerCOBOLの対話型デバッガについて説明します。

---

## 6.1 デバッグの概要

PowerCOBOLでは、デバッグモードでビルドしたプロジェクトをデバッグできます。デバッグモードについては、「[ビルドモードを使い分ける](#) ( p119) 」を参照してください。

PowerCOBOLのデバッガでは、以下の機能を利用することができます。

- 中断点の設定、解除および一覧表示
- プログラムの実行と実行条件の設定
- データの参照と監視
- 中断位置までの呼び出し経路の表示

### 6.1.1 デバッグできる範囲

PowerCOBOLでは、以下のプログラムをデバッグできます。

- フォームのスク립トがもつイベント手続き
- モジュールに登録した外部COBOLファイル中のプログラム
- COBOLのプロジェクトマネージャでデバッグ用に作成されたCOBOLプログラム (EXE、DLL)



以下のファイルやプログラムは、デバッグできません。

- モジュールに登録したオブジェクトファイル
- モジュールに登録したライブラリファイル
- #INCLUDE文で取り込まれるプログラム
- COPY文で取り込まれるプログラム

### 6.1.2 デバッグの進めかた

実際にサンプルプログラムを使ってデバッグの進めかたについて説明します。サンプルプログラムでは、以下のようにデバッグを進めていきます。

1. デバッグするプロジェクトを開く
2. デバッグを開始する
3. 中断点を設定する
4. 実行する
5. 中断位置でデータを参照する
6. 中断位置までの呼び出し経路を確認する
7. デバッグを終了する

#### デバッグするプロジェクトを開く

「[アプリケーションを作成しよう](#) ( p49) 」で作成した、商品の合計金額を計算するサンプルプログラム "Table1.ppj" を表示します。同様のサンプルプログラムは、

"Table¥Table1.ppj"に格納されています。

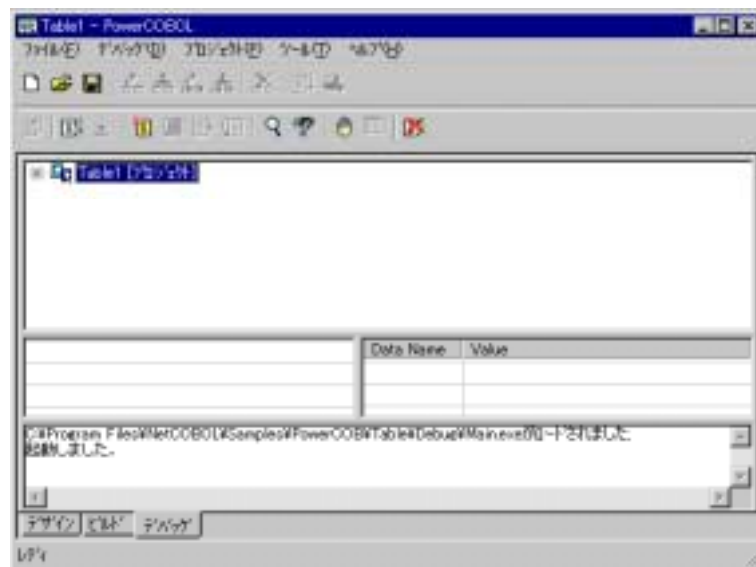
本節では、サンプルプログラム中の [ 名前を付けて保存 ] コマンドが選択されたときの処理を検証します。

### デバッグを開始する

以下の手順で、デバッグする実行可能プログラムを作成し、デバッガを起動します。

1. プロジェクトのプロパティ設定ダイアログボックスで、[ ビルド ] タブの [ ビルド ] モードがデバッグモードになっていることを確認します。
2. リリースモードであった場合、デバッグモードに変更します。
3. [ プロジェクト ] メニューの [ ビルド ] コマンドを選択し、デバッグする実行可能プログラムを作成します。ただし、すでに作成済であれば、この操作は不要です。
4. [ プロジェクト ] メニューの [ デバッグ ] コマンドを選択し、デバッガを起動します。

プロジェクトウィンドウがデバッグビューに切り替わり、実行可能プログラムがロードされデバッガが起動されたことを示すメッセージが、アウトプットウィンドウに表示されます。

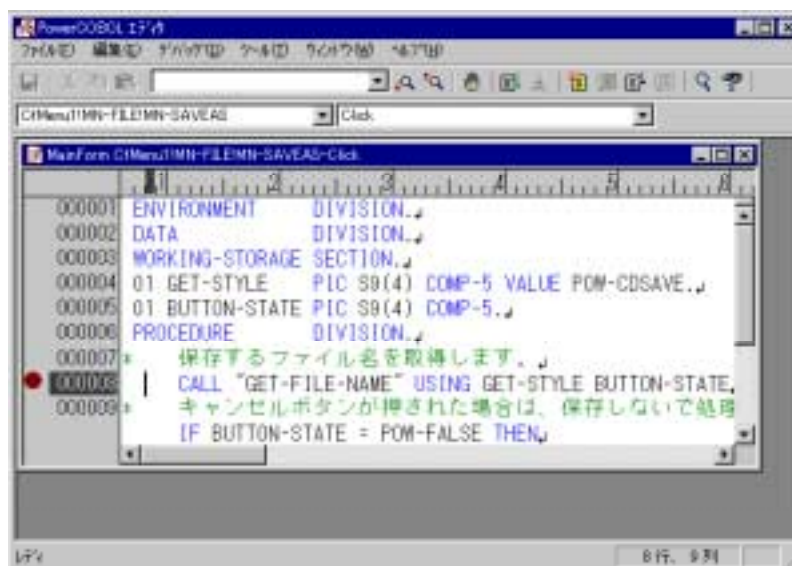


### 中断点を設定する

以下の手順で、表コントロールのセルを入力したときに発生するReturnイベントに中断点を設定します。

1. デバッグツリーウィンドウのメニュー "CfMenu1-MN-FILE" 中にある "MN-SAVEAS" を選択します。
2. ポップアップメニューの [ Click ] コマンドを選択します。
3. 8行めの 「CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE」と記述されている行に、カレットを移動します。
4. ポップアップメニューの [ 中断点の設定 ] コマンドを選択します。
5. 手続き編集ウィンドウの行番号領域左側に、中断点を示す赤い丸が表示

されていることを確認します。



中断点の設定方法に関する詳細は、「[中断点を設定する](#) ( p148 )」を参照してください。



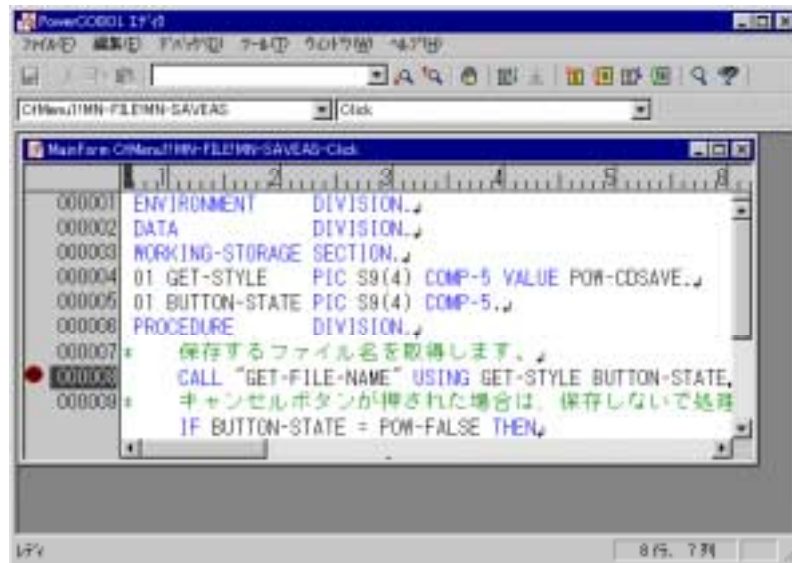
デバッグツリーウィンドウのスクリプトから

"CfMenu1!MN-FILE!MN-SAVEAS-Click"を選択して、ポップアップメニューから手続きを開くこともできます。また、手続きの入口や出口に中断点を設定する場合は、手続きを開かずに、ポップアップメニューの [ 入口に中断点を設定 ] または [ 出口に中断点を設定 ] コマンドで設定できます。

### 実行する

以下の手順で、プログラムを実行し、中断点で中断させます。

1. [ デバッグ ] メニューの [ 実行 ] コマンドを選択します。
2. アプリケーションが起動されたら、顧客名を入力し、表にいくつかの商品コードおよび個数を入力します。
3. アプリケーションの [ ファイル ] メニューから [ 名前を付けて保存 ] コマンドを選択します。
4. 手続き編集ウィンドウをクリックします。
5. イベント手続き "CfMenu1!MN-FILE!MN-SAVEAS-Click" の8行めの行番号領域に、右向きの矢印が表示されていることを確認します。



また、以下の手順で1ステップずつ、処理を進めることができます。

1. [ デバッグ ] メニューの [ ステップイン ] コマンドを選択します。
2. フォームの共通内部手続き "GET-FILE-NAME" が表示され、中断位置を示す右向きの矢印が、その中の10行め「 IF GET-STYLE = POW-CDOPEN THEN 」に移動したことを確認します。
3. [ デバッグ ] メニューの [ ステップイン ] コマンドを3回選択し、中断位置を「 INVOKE POW-SELF "GetFileName" USING 」と記述された19行めまで進めます。

実行に関する詳細は、「[プログラムを実行する](#) ( p152 )」を参照してください。



注意

アプリケーションを実行し、中断点に達しても、手続き編集ウィンドウは自動的にウィンドウの前に出てこない場合があります。その場合は、手続き編集ウィンドウをクリックし、前に表示されるようにしてください。

### データを参照する

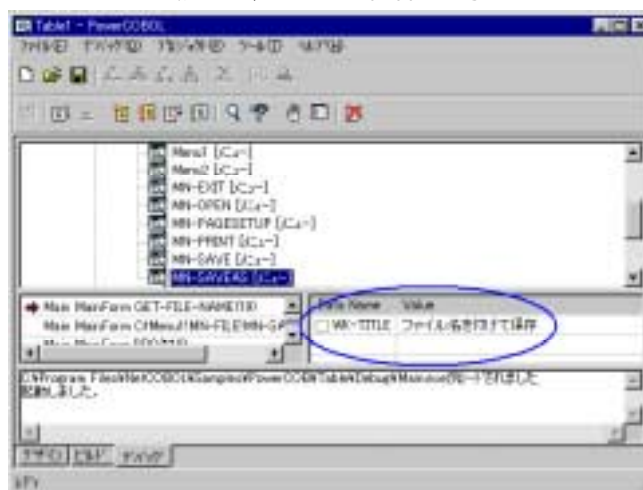
以下の手順で、13行めの手続き「 MOVE "ファイル名を付けて保存" TO WK-TITLE 」によって転記されたデータの内容を参照することができます。

1. 19行めに中断している状態で、マウスポインタを21行めの文字列 "WK-TITLE" の上に移動します。
2. マウスポインタ右下付近に、"WK-TITLE = ファイル名を付けて保存" と表示されるのを確認してください。



データの内容は、[ デバッグ ] メニューの [ ウォッチ ] コマンドで表示されるダイアログボックスを使用して参照することもできます。この場合は、以下の手順で参照します。

1. [ デバッグ ] メニューの [ ウォッチ ] コマンドを選択します。
2. [ ウォッチの設定 ] ダイアログボックスの [ データ名 ] に、"WK-TITLE" と入力します。
3. プロジェクトウィンドウで、デバッグビューのウォッチウィンドウに "WK-TITLE" が追加され、データの内容が表示されることを確認します。



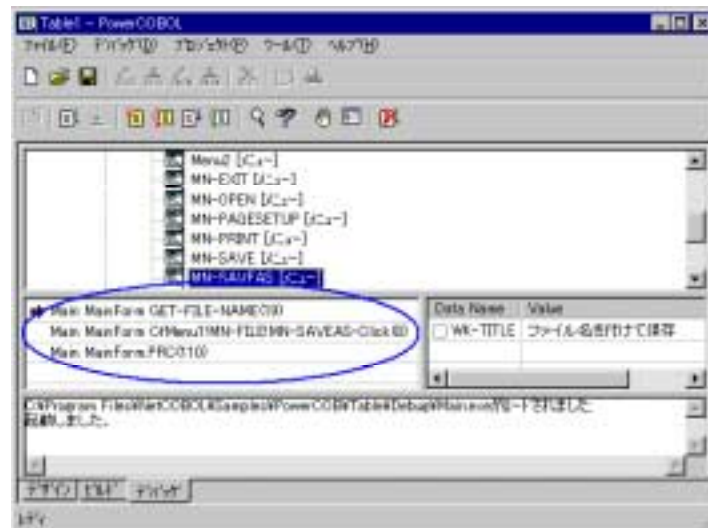
データの参照に関する詳細は、「[データを参照する](#) ( p154 )」を参照してください。



### 呼び出し経路を確認する

呼び出し経路ウィンドウを参照することにより、現在中断している手続きが、どこから呼び出されたものなのか確認することができます。

このサンプルプログラムで、上記のデータを参照している時点で呼び出し経路ウィンドウを参照すると、現在中断している手続き"GET-FILE-NAME"は、イベント手続き"CfMenu1!MN-FILE!MN-SAVEAS-Click"から呼び出されていることが確認できます。



呼び出し経路に関する詳細は、「[呼び出し経路を確認する](#) ( p158 )」を参照してください。

### デバッグを終了する

デバッグを終了する場合は、[ デバッグ ] メニューの [ デバッグの終了 ] コマンドを選択してください。



デバッグを最初からやりなおす場合は、[ デバッグ ] メニューの [ 再デバッグ ] コマンドを選択します。このとき、設定した中断点およびウォッチウィンドウに設定されたデータは、再デバッグ前の状態のままになります。

ただし、再デバッグはプログラムが終了した状態から開始しなければなりません。再デバッグは、プログラムをいったん終了してから行ってください。

## 6.2 中断点を設定する

本節では、中断点を設定 / 解除する方法および中断点の一覧を表示する方法について説明します。

### 6.2.1 中断点を設定するには

中断点は、以下のどちらかの操作により設定することができます。

手続き編集ウィンドウのポップアップメニューから

[ 中断点の設定 ] ダイアログボックスから

#### 手続き編集ウィンドウのポップアップメニューから

1. 中断点を設定したい手続きを、手続き編集ウィンドウで表示します。
2. 中断点を設定したい行にカレットを移動します。
3. ポップアップメニューの [ 中断点の設定 ] コマンドを選択します。

同様に、ポップアップメニューの [ 中断点の解除 ] コマンドを選択することにより、中断点を解除することができます。

#### [ 中断点の設定 ] ダイアログボックスから

1. プロジェクトウィンドウまたは手続き編集ウィンドウの [ デバッグ ] メニューから [ 中断点の設定 ] コマンドを選択します。
2. [ イベント選択 ] タブを開きます。
3. ツリー表示されたウィンドウから、中断したい手続きを選択します。
4. 中断したい手続きを選択したら、[ 確定 ] ボタンをクリックします。
5. [ 位置指定 ] タブを開きます。
6. 手続きの入口または出口で中断したい場合は、それぞれ [ 入口 ] または [ 出口 ] のどちらかを選択します。
7. 任意の行で中断したい場合は、[ 行番号 ] を選択し、中断する行番号を入力します。
8. [ オプション ] タブを開きます。
9. 必要に応じて、中断する条件式または中断までの到達回数を指定します。





[ 中断点の設定 ] ダイアログボックスは、手続き編集ウィンドウでのカレット位置やデバッグツリーウィンドウでの選択項目の状態により、最初に表示されるタブが異なります。

たとえば、手続き編集ウィンドウが表示され、手続き中の任意の行にカレットがある状態では、その手続きと行がデフォルトとして設定されますので、[ オプション ] タブが最初に表示されます。

オプションの条件式が評価できない場合、偽 (FALSE) として処理されます。また、条件式と到達回数の両方を指定した場合、どちらか一方が真 (TRUE) になった場合に中断されます。

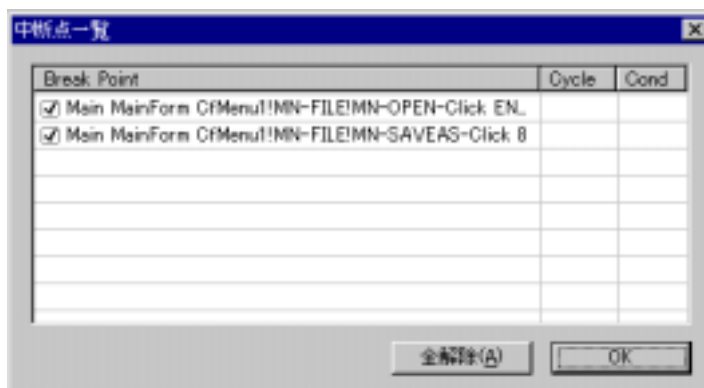
モジュールに登録した外部COBOLファイル中のソースに中断点を設定する場合、[ プログラム選択 ] タブを使ってプログラム名を選択できます。ここで選択できるのは、すでにWindows 上にロードされているプログラムだけです。ロードされていないプログラムを指定する場合は、直接 [ 外部プログラム・内部プログラム ] にプログラム名を入力してください。ただし、この場合、中断点の設定は、実際にプログラムがロードされるまで延期されます。



[ プログラム選択 ] タブを使ってプログラム名を選択する場合、外部プログラム名としてフォーム名を選択すると、内部プログラム名として "POW-SCRIPTLETn" (nは正数値) が表示されます。これらは、そのフォームに含まれるイベント手続きに対応したCOBOLの内部プログラム名です。

### 6.2.2 中断点の一覧を表示する

中断点は、プロジェクトウィンドウのデバッグビューまたは手続き編集ウィンドウの [ デバッグ ] メニューから [ 中断点一覧 ] コマンドを選択することにより、一覧表示できます。



中断点の一覧を表示することにより、以下の操作ができます。

- 中断点の有効 / 無効化
- 中断点の表示
- 中断点のオプションの設定
- 中断点の解除
- すべての中断点の解除

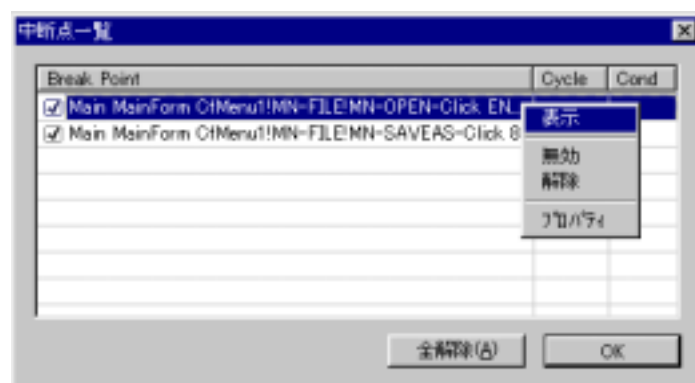
### 中断点の有効 / 無効化

左端部分のチェック状態を切り替えることにより、中断点を有効にするか無効にするかを切り替えることができます。

チェック状態の場合、中断点で実行が停止します。チェックをはずした場合、中断点一覧には表示されますが、その中断点では停止しません。

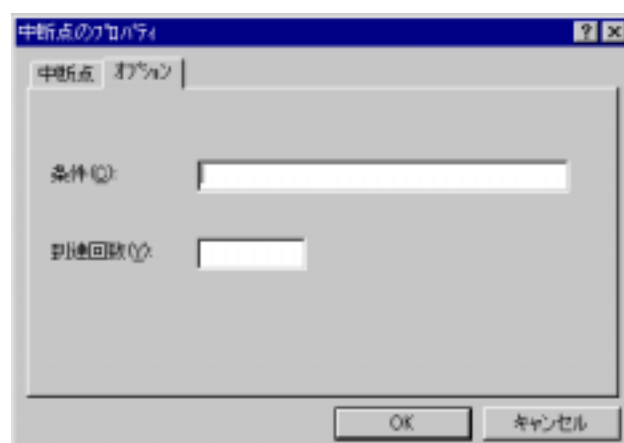
### 中断点の表示

中断点を選択し、ポップアップメニューの[表示]コマンドを選択すると、その中断点がある手続きが表示され、カレットを位置付けることができます。



### 中断点のオプションの設定

中断点を選択し、ポップアップメニューの[プロパティ]コマンドにより表示されるダイアログボックスで、中断の条件式や到達回数を設定することができます。



**中断点の解除**

中断点を選択し、ポップアップメニューの「解除」コマンドを選択すると、中断点を解除することができます。解除された中断点は、一覧から削除されます。ただし、中断点の設定が延期されている場合、その中断点を解除することはできません。この場合、中断点を解除するためには、次の「すべての中断点の解除」の操作が必要です。

**すべての中断点の解除**

「全解除」ボタンをクリックすることにより、すべての中断点を解除することができます。

## 6.3 実行する

本節では、中断点までの実行方法や、任意の位置での実行の中断方法について説明します。

### 6.3.1 実行するには

PowerCOBOLのデバッグでは、以下の実行方法を使用することができます。

- 中断点まで実行する
- 1ステップだけ実行する
- カレット位置まで実行する
- 手続きの出口まで実行する



注意

PowerCOBOLのイベント手続きの最後に、出口文(EXIT PROGRAM文など)が記述されていない場合、記述したイベント手続きよりも後方まで実行されることがあります。これは、PowerCOBOLが、記述されたイベント手続きを組み合わせ、必要な手続きを付加して1つのCOBOLプログラムを生成し、実行可能プログラムを作成しているためです。この現象は、イベント手続きの最後に出口文を記述することにより回避することができます。

PowerCOBOLが作成するCOBOLプログラムの構造は、「[イベント駆動型プログラムスタイル](#) ( p7) 」を参照してください。

#### 中断点まで実行する

中断点まで実行する場合は、[ デバッグ ] メニューの [ 実行 ] コマンドを選択します。中断点が設定されていない場合は、プログラムが終了するまで実行します。

#### 1ステップだけ実行する

1ステップだけ実行する場合は、[ デバッグ ] メニューの [ ステップイン ] または [ ステップオーバー ] コマンドを選択します。実行する手続きが、他の手続きを呼び出す文であった場合、ステップインとステップオーバーで動作が異なります。

ステップイン : 他の手続きの先頭で中断します。

ステップオーバー : 他の手続きの中では中断せず、現在中断している手続き中の次の文で中断します。

ただし、ステップオーバーしても、呼び出している他の手続きの中に中断点が設定されている場合は、その位置で中断します。

#### カレット位置まで実行する

カレット位置まで実行する場合は、[ デバッグ ] メニューの [ カレット位置まで実行 ] コマンドを選択します。中断するカレット位置は、最後にフォーカスをもっていた手続き編集ウィンドウのカレットがある行です。

ただし、カレット位置まで到達する前に、中断点が設定されている場合は、その位置で中断します。

### 手続きの出口まで実行する

手続きの出口まで実行する場合は、[ デバッグ ] メニューの [ 手続きの出口まで実行 ] コマンドを選択します。中断する位置は、現在実行中の手続きから抜ける直前の行です。

ただし、手続きの出口まで到達する前に、中断点が設定されている場合は、その位置で中断します。

## 6.3.2 実行を中断させるには

プログラムが論理的なエラーで無限ループした場合などは、以下のどちらかの操作により、実行を中断させることができます。

- メニューのコマンドからの中断
- キーボード操作からの中断

### メニューのコマンドからの中断

プロジェクトウィンドウまたは手続き編集ウィンドウの [ デバッグ ] メニューから、[ 実行の中断 ] コマンドを選択することにより、実行を中断できます。

### キーボード操作からの中断

[Esc] キーを入力することにより、実行を中断できます。

[Esc] キーによる中断は、デバッグ対象のアプリケーションの動作により、中断方法を切り替えることができます。

たとえば、[Esc] キーは、キャンセルボタンを処理するためのキーとして使用されたり、テキストボックスへの入力をキャンセルするためのキーとして使用されたりする場合があります。このような場合、[Esc] キーによる中断方法を切り替えます。

中断方法は、以下の操作により切り替えることができます。

1. プロジェクトウィンドウで [ デバッグ ] メニューの [ オプション ] コマンドを選択します。
2. [ デバッグ ] タブを選択します。
3. [ PowerCOBOL にフォーカスがある場合にだけ ESC キーで中断する ] のチェック状態を切り替えます。
4. OK ボタンをクリックします。

チェックをはずすと、デバッグ対象のアプリケーション、PowerCOBOL のプロジェクトウィンドウおよび手続き編集ウィンドウにフォーカスがある場合、[Esc] キーにより中断できます。

チェック状態にすると、デバッグ対象のアプリケーションにフォーカスがある場合、[Esc] キーによる中断は行われず、PowerCOBOL のプロジェクトウィンドウおよび手続き編集ウィンドウにフォーカスがある場合だけ、[Esc] キーにより中断できます。



**注意**

中断された位置によって、[ デバッグ ] メニューで使用できるコマンドは異なります。

## 6.4 データを参照する

本節では、プログラムで使用しているデータの値を参照する方法について説明します。

PowerCOBOLのデバッグでデータを参照する場合、以下の機能を使用します。

データチップによるデータの現在値の参照

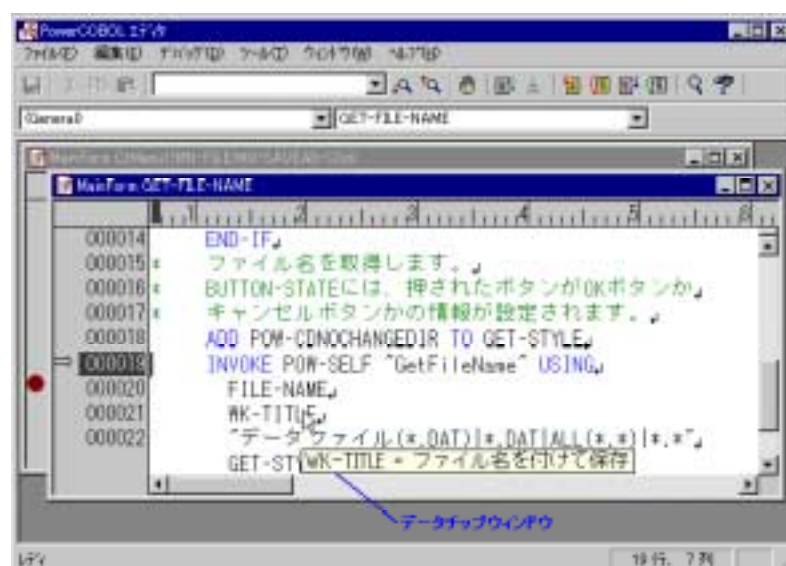
クイックウォッチによるデータの現在値の参照および値の編集

ウォッチによるデータ値の継続的な監視

### 6.4.1 データチップ

プログラムが中断している場合、手続き編集ウィンドウで任意のデータ名の上にマウスポインタを停止させることにより、そのデータが参照可能ならば、その値をデータチップウィンドウに表示できます。

修飾が必要なデータ名の場合は、修飾を含めたデータ名全体を選択状態にしておくことで、値を参照できます。



注意

手続き中に選択文字列がある場合は、その文字列がデータチップの表示対象となります。

表示される値は、データの型に応じて表示されます。16進数での表示はできません。

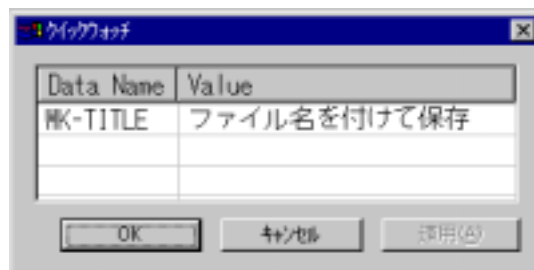
初期化されていないデータや、数値項目を含む集団項目など、文字として表示できない値を含む場合、正しく表示できない場合があります。



## 6.4.2 クイックウォッチ

プログラムが中断している場合、[ クイックウォッチの設定 ] ダイアログボックスにデータ名を指定することにより、データの現在値を参照することができます。[ クイックウォッチの設定 ] ダイアログボックスは、[ デバッグ ] メニューの [ クイックウォッチ ] コマンドで表示することができます。また、表示されたデータ上では、ポップアップメニューからコマンドを選択することにより、以下のことができます。

- [ 編集 ] : データの値を変更します。
- [ ウォッチ ] : データをウォッチウィンドウへ追加します。
- [ 16進数 ] : データの値を16進数で表示します。
- [ 変更時中断 ] : データの値が変更されたときに、実行を中断するかどうかを変更します。
- [ スケール ] : データの値の上方にスケールを挿入します。



指定したデータが集団項目や添字付きデータ項目の場合、データ名の左側の符号をクリックすることにより、細分化して表示できます。

[ クイックウォッチの設定 ] ダイアログボックスでは、中断点を設定する場合と同様に、データが使用されているイベント手続きやプログラムを選択することもできます。

モジュールに登録した外部COBOLファイル中のデータの値を参照する場合は、[ プログラム選択 ] タブを使ってプログラム名を選択することができます。ここで選択できるのは、すでにWindows 上にロードされているプログラムだけです。ロードされていないプログラムを指定する場合は、直接 [ 外部プログラム・内部プログラム ] にプログラム名を入力してください。

[ クイックウォッチの設定 ] ダイアログボックスで最初に表示されるタブは、手続き編集ウィンドウやデバッグツリーウィンドウでの選択項目の状態などにより、異なります。



COBOLプログラムの実行が始まる前に、EXTERNAL属性データの値を表示することはできません。

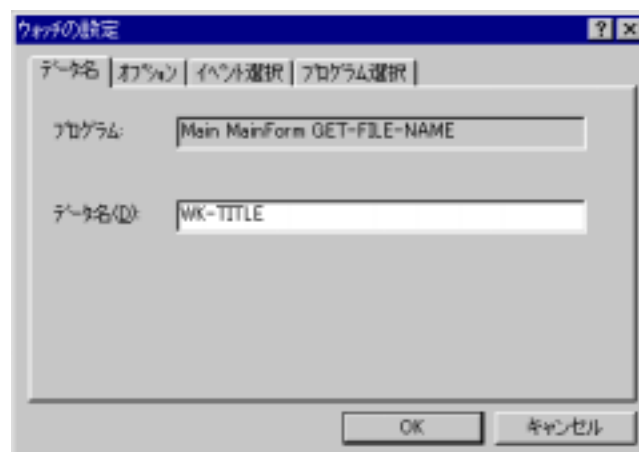
連絡節(LINKAGE SECTION)に記述されたデータの値は、そのプログラムが呼び出され、実行しているあいだだけ表示することができます。

[プログラム選択]タブを使ってプログラム名を選択する場合、外部プログラム名としてフォーム名を選択すると、内部プログラム名として"POW-SCRIPTLETn" (nは正数値)が表示されます。これらは、選択したフォームに含まれるイベント手続きに対応したCOBOLの内部プログラム名です。

ランタイムコードセットをUnicode(1 - UCS2)にして作成したアプリケーションをデバッグ中、Unicode固有の文字(シフトJISコードに対応するコードがない文字)を、データの値として入力する場合には、16進数の形式で指定してください。

### 6.4.3 ウォッチ

[デバッグ]メニューの[ウォッチ]コマンドで表示された[ウォッチの設定]ダイアログボックスにデータ名を指定することにより、プログラムが実行しているあいだ、指定したデータの値を継続して表示できます。ウォッチは、値の変化を監視できるため、データの値が変更された場合に、実行を中断することもできます。



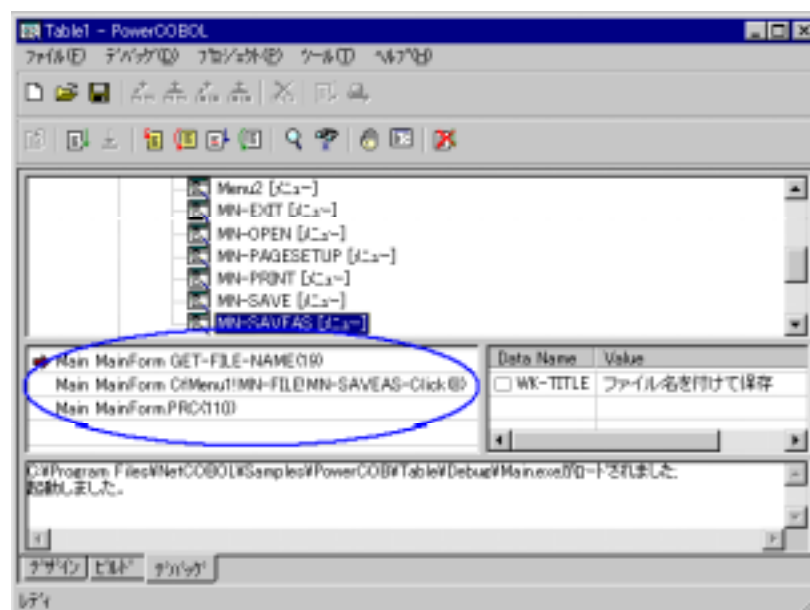
データ名を[ウォッチの設定]ダイアログボックスで指定すると、データ名がウォッチウィンドウに追加されます。

ウォッチウィンドウ上では、ポップアップメニューからコマンドを選択することにより、以下のことができます。

- [ クイックウォッチ ] : データの値をクイックウォッチします。
- [ 16進数 ] : データの値を16進数で表示します。
- [ 変更時中断 ] : データの値が変更されたときに実行を中断するかどうか指定します。
- [ 削除 ] : データのウォッチを解除し、ウォッチウィンドウから削除します。
- [ プロパティ ] : データのプロパティを表示します。

## 6.5 呼び出し経路を確認する

呼び出し経路ウィンドウは、Windows がアプリケーションを呼び出した位置から、現在中断しているプログラムの位置までの、COBOLプログラムの呼び出し経路（コールスタック）を最初に呼び出したプログラムを下にして順に表示します。PowerCOBOLでは、呼び出し経路ウィンドウから、対応する手続きを表示し、現在中断しているプログラムまでの呼び出し方法などを確認することができます。



呼び出し経路ウィンドウには、以下の情報が表示されます。

- |                        |                     |
|------------------------|---------------------|
| フォームに対応する外部プログラ        | モジュール名、フォーム名.PRC（外部 |
| ム（PowerCOBOLによって自動生成）： | プログラム内相対行番号）        |
| イベント手続き                | ：モジュール名 フォーム名 イベント手 |
|                        | 続き名（イベント手続き内相対行番号）  |
| 共通内部プログラム              | ：モジュール名 フォーム名 共通内部プ |
|                        | ログラム名（内部プログラム内相対行   |
|                        | 番号）                 |
| その他（外部COBOLソースなど）      | ：モジュール名 ソースファイル名（ファ |
|                        | イル内相対行番号）           |

呼び出し経路ウィンドウでは、ポップアップメニューの[表示]コマンドを選択することにより、対応する手続きを手続き編集ウィンドウに表示できます。ただし、表示できるのは「[デバッグできる範囲](#)（p142）」で示されているプログラムだけです。

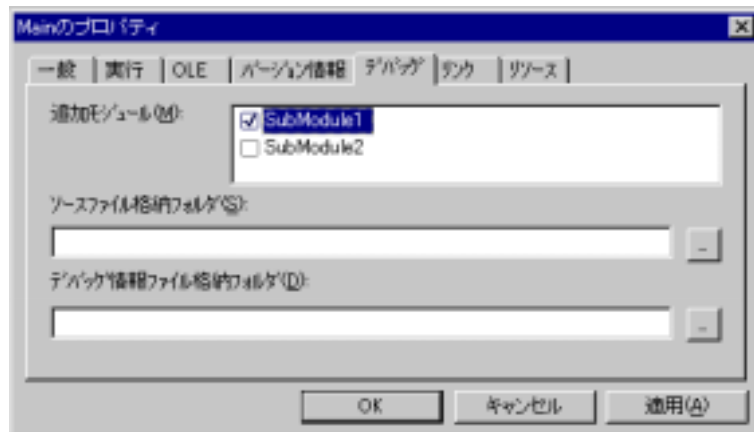
## 6.6 いろいろな形態のアプリケーションをデバッグする

本節では、PowerCOBOLでデバッグできる範囲および複数の実行可能プログラムから構成されるアプリケーションのデバッグ方法について説明します。

### 6.6.1 同一プロジェクト内の複数のモジュールをデバッグする

アプリケーションが複数のモジュールから構成され、それらのモジュールが同一のプロジェクトに属する場合、デバッグを開始する前に、以下の設定をします。

1. デバッグを開始するモジュール（通常は、起動プログラム）のプロパティ設定ダイアログボックスを表示します。
2. [ デバッグ ] タブを選択します。
3. [ 追加モジュール ] のリストから、同時にデバッグするモジュールを選択し、チェック状態にします。



### 6.6.2 複数プロジェクトにまたがる複数のモジュールをデバッグする

PowerCOBOLでは、複数のプロジェクトにまたがるモジュールを、同時にデバッグすることはできません。

複数のプロジェクトにまたがるモジュールは、以下の設定により、同一プロジェクト内の範囲でデバッグできます。

### 起動プログラム(EXE)をもつプロジェクトの場合

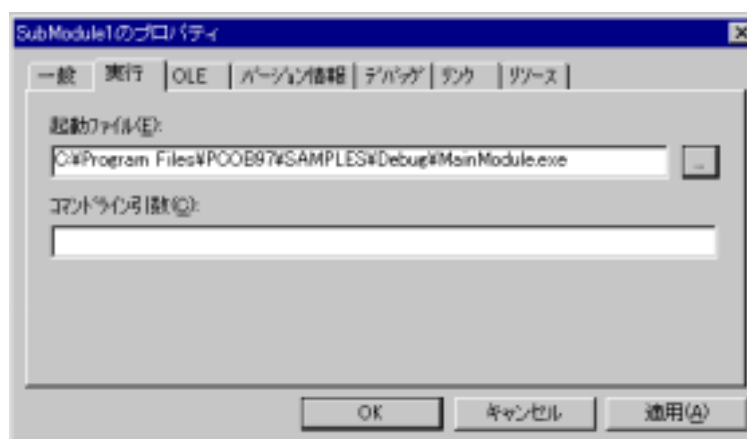
そのままデバッグすることが可能です。同一プロジェクト内にあるDLLをデバッグする場合は、前項の方法を使って、デバッグを開始する前にデバッグ対象モジュールとして追加してください。

別プロジェクト内にあるDLLをデバッグすることはできません。

### 起動プログラム(EXE)をもたないプロジェクトの場合

そのままデバッグすることはできません。デバッグを開始する前に、以下の設定をします。

1. デバッグするモジュール(DLL)のプロパティ設定ダイアログボックスを表示します。
2. [ 実行 ] タブを選択します。
3. [ 起動ファイル ] に、このDLLを呼び出す実行可能プログラムのファイル名を入力します。



このモジュールのデバッグは、このモジュールに含まれるプログラムが呼び出された箇所から開始することができます。デバッグを起動したら、このDLLに含まれるプログラムが呼び出されるように、アプリケーションを操作してください。

## 6.6.3 COBOLプログラムと組み合わせてデバッグする

COBOLプログラムと組み合わせてデバッグする場合は、以下のようにします。

### COBOLプログラムを呼び出している場合

PowerCOBOLで作成したモジュールから、COBOLのプロジェクトマネージャで作成した実行可能プログラム(EXE、DLL)を呼び出している場合、デバッグを開始する前に以下の設定をします。

1. デバッグを開始するモジュール(通常は、起動プログラムまたはデフォルトモジュール)のプロパティ設定ダイアログボックスを表示します。
2. [ デバッグ ] タブを選択します。
3. [ ソースファイル格納フォルダ ] に、COBOLのプロジェクトマネージャ

で作成したプログラムのソースファイルが格納されているフォルダ名を入力します。複数のフォルダを指定する場合は、セミコロン(;)で区切ります。

4. [ デバッグ情報ファイル格納フォルダ ] に、COBOLのプロジェクトマネージャで作成したプログラムのデバッグ情報ファイル(SVD)が格納されているフォルダ名を入力します。複数のフォルダを指定する場合は、セミコロン(;)で区切ります。

#### COBOLプログラムから呼び出されている場合

PowerCOBOLで作成したモジュールが、COBOLのプロジェクトマネージャで作成した実行可能プログラム(EXE)から呼び出されている場合、「[起動プログラム\(EXE\)をもたないプロジェクトの場合](#)」( p160)と同様の方法で、起動ファイル名にCOBOLの実行可能プログラムのファイル名を指定します。



---

COBOLのプロジェクトマネージャで作成されたプログラムのソースは、そのプログラムが実行され、中断されるまで開くことはできません。中断は、以下のどちらかの方法で行います。

[ 中断点の設定 ] ダイアログボックスの [ プログラム選択 ] タブでプログラム名を指定して、中断点を設定します。

プログラムの先頭まで、1ステップずつ実行します。

PowerCOBOLでデバッグできるCOBOLプログラムは、デバッグ用に作成されている必要があります。

---

# ***MEMO***



---

## 第3部 プログラミング編

---

---

本部では、PowerCOBOLを使ってアプリケーションを作成する場合のプログラミング方法、およびサンプルプログラムを作成しながら、各種アプリケーションの作成方法について説明します。

---

---

---

## 第7章      プログラミングの基礎知識

---

本章では、PowerCOBOLでのプログラミングの概要、フォームやコントロールのプロパティへのアクセス方法、およびメソッドの呼び出し方法などについて説明します。

---

## 7.1 コントロールとフォームの手続き

本節では、フォームやコントロールのイベント手続きの使用方法、およびフォームがもつ各種共通宣言について説明します。

### 7.1.1 コントロールの手続き

コントロールがもつ、各イベント手続きは、フォーム内でそれぞれ1つの内部プログラムとして扱われます。したがって、データ部で宣言したデータ項目は、そのイベント手続き内だけで有効となります。また、イベント手続きには、さらに内部プログラムを記述することもできます。

編集するイベントを選択し、手続き編集ウィンドウを開くと、あらかじめ環境部、データ部および手続き部を記述するための定型の手続きが自動的に生成されます。

たとえば、コマンドボタンコントロールのClickイベントを選択すると、以下のような手続きが表示されます。ここに必要な手続きを追加していきます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
```

また、イベントの種類により、引数をもつ場合があります。

たとえば、コマンドボタンコントロールのMouseDownイベントでは、以下のように、マウスのどのボタンをクリックしたか、同時に押されたキーは何か、クリックしたときのマウスポインタの位置はどこであったかという情報が、イベントの引数として渡されます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-BUTTON PIC S9(4) COMP-5.
01 POW-SHIFT PIC S9(4) COMP-5.
01 POW-X PIC S9(9) COMP-5.
01 POW-Y PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW-BUTTON POW-SHIFT POW-X POW-Y.
```

さらに、コントロールが配列化されている場合は、イベントが発生したコントロールのインデックス値を引数として受け取ることができます。

たとえば、コマンドボタンが配列化されている場合、そのMouseDownイベントは、以下ようになります。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

( 続く )

( 続き )

|               |                                                               |
|---------------|---------------------------------------------------------------|
| LINKAGE       | SECTION.                                                      |
| 01 POW-INDEX  | PIC S9(9) COMP-5.                                             |
| 01 POW-BUTTON | PIC S9(4) COMP-5.                                             |
| 01 POW-SHIFT  | PIC S9(4) COMP-5.                                             |
| 01 POW-X      | PIC S9(9) COMP-5.                                             |
| 01 POW-Y      | PIC S9(9) COMP-5.                                             |
| PROCEDURE     | DIVISION USING POW-INDEX POW-BUTTON POW-SHIFT<br>POW-X POW-Y. |

配列化したコントロールの利用方法については、「[配列化したコントロールを使ったアプリケーションを作成する](#) ( p194 )」を参照してください。

また、個々のコントロールがもつイベントについては、『リファレンス』を参照してください。

## 7.1.2 フォームの手続き

フォームもコントロールと同様、イベント手続きをもっています。たとえば、フォームが開かれたときはOpened、フォームが閉じられたときはClosedといったイベントが発生します。フォームは、これらに加え、フォーム内の手続き全体で有効な共通宣言および共通内部プログラムを定義できます。さらに、フォームをActiveXコントロールとして利用するため、新しいイベント（カスタムイベント）も定義できます。ActiveXコントロールとして利用するための方法は、「[ActiveXコントロールを作成する](#) ( p245 )」を参照してください。

以下に、フォームがもつ共通宣言および共通内部プログラムについて説明します。各宣言の記述方法についての詳細は『COBOL 文法書』を参照してください。

### 共通宣言（環境部）

#### SPECIAL-NAMES

構成節 (CONFIGURATION SECTION) の特殊名段落 (SPECIAL-NAMES) を記述します。SPECIAL-NAMESでは、必要となる機能名に対する呼び名や、記号定数などを定義します。そのフォーム中で、機能名に対する呼び名や記号定数を使用しない場合は、記述する必要はありません。

#### REPOSITORY

構成節 (CONFIGURATION SECTION) のリポジトリ段落 (REPOSITORY) を記述します。REPOSITORYでは、この環境部の有効範囲内で使われるクラス名を指定します。フォーム中で、クラス名を使用しない場合は、記述する必要はありません。

#### FILE-CONTROL

入出力節 (INPUT-OUTPUT SECTION) のファイル管理記述項 (FILE-CONTROL) を記述します。

FILE-CONTROLでは、フォーム中で使用するファイルのSELECT句を記述します。フォーム中で、ファイルを扱わない場合は、記述する必要はありません。

## 共通宣言（データ部）

### BASED-STORAGE

基底場所節 (BASED-STORAGE SECTION) を記述します。

BASED-STORAGE SECTIONでは、ポインタデータ項目により、明示的または暗示的に領域のアドレスが決定されるデータ項目を定義します。そのフォーム中で、ポインタデータ項目を使用しない場合は、記述する必要はありません。

### FILE

ファイル節 (FILE SECTION) を記述します。

FILE SECTIONでは、ファイルの物理的な構造、識別名およびファイルに関するレコード名の情報を指定します。そのフォーム中で、ファイルを扱わない場合は、記述する必要はありません。



---

モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "0 - COBOL85言語仕様" である場合、ファイル節で定義するファイルは、各手続きの中でOPEN文、CLOSE文、READ文、WRITE文などを使用するために、GLOBAL句を記述する必要があります。記述しないと、手続き中でそのファイルに対する入出力文が、翻訳エラーとなります。

ただし、SORT文/MERGE文ではGLOBAL句の指定ができませんので、SORT文/MERGE文を使用する場合は、個々のイベント手続きのFILE SECTIONに記述してください。

逆に、モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "1 - OOCOBOL言語仕様" である場合、GLOBAL句を記述する必要はありません。GLOBAL句を記述すると翻訳エラーとなります。

---

### WORKING-STORAGE

作業場所節 (WORKING-STORAGE SECTION) を記述します。

WORKING-STORAGE SECTIONでは、作業用データ項目の宣言をします。そのフォーム中で共通の、作業用データ項目を使用しない場合は、記述する必要はありません。



---

モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "0 - COBOL85言語仕様" である場合、作業場所節で定義するデータ項目は、内部プログラムとなる各手続きの中で使用するために、GLOBAL句を記述する必要があります。記述しないと、そのデータ項目を参照している手続きが、翻訳エラーとなります。

逆に、モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "1 - OOCOBOL言語仕様" である場合、GLOBAL句を記述する必要はありません。GLOBAL句を記述すると翻訳エラーとなります。

---

**CONSTANT**

定数節 (CONSTANT SECTION) を記述します。

CONSTANT SECTIONでは、値が原始プログラムで与えられ、プログラムの実行中に変わることのないデータ項目を記述します。そのフォーム中で共通の、定数用のデータ項目を使用しない場合は、記述する必要はありません。



モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "0 - COBOL85言語仕様" である場合、定数節で定義するデータ項目は、内部プログラムとなる各イベント手続きの中で使用するために、GLOBAL句を記述する必要があります。記述しないと、そのデータ項目を参照している手続きが、翻訳エラーとなります。

逆に、モジュールのプロパティ設定ダイアログボックスで、[ 一般 ] タブの [ スクリプト言語 ] が "1 - OOCOBOL言語仕様" である場合、GLOBAL句を記述する必要はありません。GLOBAL句を記述すると翻訳エラーとなります。

**共通内部プログラム (手続き部)**

フォーム内で利用する共通内部プログラムは、以下のどちらかの方法で作成します。

個々の共通内部プログラムを新規作成する方法

複数の共通内部プログラムをまとめてPROCEDUREに記述する方法

ただし、そのフォーム中で共通の内部プログラムを使用しない場合、作成する必要はありません。

**新規作成する方法**

以下の操作により、共通内部プログラムを新規に作成できます。

通常、共通内部プログラムを作成する場合は、この方法を使用してください。

1. フォームのポップアップメニューの [ 手続き部の編集 ] サブメニューから [ 新規作成 ] コマンドを選択します。
2. 共通内部プログラムを記述します。このとき、見出し部 (IDENTIFICATION DIVISION) の記述は不要です。
3. プロジェクトウィンドウのデザインツリーウィンドウで、スクリプト中にある "FjCobCmpScrN" (Nは正数値) を選択します。
4. ポップアップメニューから [ 名前の変更 ] コマンドを選択し、共通内部プログラム名を入力します。



---

再度、このプログラムを編集する場合は、ポップアップメニューの[ 手続き部の編集 ]サブメニューから共通内部プログラム名を選択してください。

モジュールのプロパティ設定ダイアログボックスで、[ 一般 ]タブの[ スクリプト言語 ]が"0 - COBOL85言語仕様"である場合、この方法で作成したプログラムには、自動的にCOMMON属性が付けられます。

---

#### PROCEDUREに記述する方法

以下のような特殊な内部プログラムを作成する場合は、共通内部プログラムをPROCEDUREに記述します。

ここに記述される内部プログラムは、すべての内部プログラムの先頭に位置づけられます。

#INCLUDE文を利用して、別ファイル中の共通内部プログラムを利用する場合

SQL文のカーソル宣言など、記述された順番に依存する手続きを使用する場合



---

PROCEDUREに共通内部プログラムを記述する場合、各プログラムには見出し部およびプログラムの終わり見出しが必要です。また、モジュールのプロパティ設定ダイアログボックスで、[ 一般 ]タブの[ スクリプト言語 ]が"0 - COBOL85言語仕様"である場合、プログラム名段落には、COMMON属性を付けてください。

---



## 7.2 プロパティへのアクセス方法

フォームやコントロールのプロパティは、フォームを編集するときに設定できますが、手続き中でも参照したり設定したりすることもできます。  
本節では、フォームやコントロールのプロパティを参照および設定するためのプログラミング方法について説明します。

### 7.2.1 プロパティの記述形式

プロパティとは、フォームやコントロールがもつ属性です。プロパティには、コントロールに表示される文字列、コントロールの色、コントロールの位置や大きさなどがあります。

コントロールやフォームがもつプロパティの詳細については、『リファレンス』を参照してください。

プロパティは、以下の形式で記述します。

```
"プロパティ名" [(添字 - 1 . . .)]
[OF "プロパティ名" [(添字 - 1 . . .)]]
[OF コントロール名 [(添字 - 2)]]
OF { コントロール名 [(添字 - 2)] | POW - SELF }
```

#### プロパティ名

コントロールおよびフォームがもつプロパティの名前です。

プロパティ名は、" "で囲まないで記述することもできます。ただし、プロパティ名がCOBOLの予約語と同じである場合は、" "で囲まなければなりません。

また、プロパティ名の大文字と小文字は区別する必要はありません。ただし、オブジェクトのクラス名を利用してプロパティにアクセスする場合は、大文字と小文字を区別して記述する必要があります。オブジェクトのクラス名を利用してプロパティにアクセスする方法については、「[オブジェクトへアクセスするには](#) ( p197 )」を参照してください。

#### 添字-1

複数の要素をもつプロパティの場合、そのプロパティの個々の要素を参照するためのインデックス値を指定します。

インデックス値には、データ名または定数が使用できます。たとえば、表コントロールのセルを特定するためのTableCellsプロパティは、"TableCells" (a b) というように記述します。

#### 添字-2

コントロールが配列化されている場合、個々のコントロールを参照するためのインデックス値を指定します。

インデックス値には、データ名または定数が使用できます。コントロールを配列化した場合のプログラミング方法については、「[配列化したコントロールを使ったアプリケーションを作成する](#) ( p194 )」を参照してください。

### コントロール名

コントロールのプロパティを記述する場合には、そのプロパティをもつコントロールの名前を記述します。

コントロール名とは、コントロールを識別するためにフォーム編集時に設定した名前です。

コントロールの命名規則については、「[プロジェクト構成要素の命名規則](#) ( p93) 」を参照してください。

### POW-SELF

フォームのプロパティを記述する場合には、POW-SELFを使用します。POW-SELFはフォーム自身を表しています。POW-SELFの代わりに、フォーム名を使うこともできます。

フォームの命名規則については、「[プロジェクト構成要素の命名規則](#) ( p93) 」を参照してください。

## 7.2.2 プロパティの記述例

以下に、プロパティの記述例を示します。

### フォームのプロパティ

"プロパティ名" OF POW-SELF

フォームのキャプション ( フォームのタイトルに表示する文字列 ) に "Form-Name1"を設定する場合、以下のように記述します。

MOVE "Form-Name1" TO "Caption" OF POW-SELF.

### コントロールのプロパティ ( 標準 )

"プロパティ名" OF コントロール名

コマンドボタンコントロール (CommandButton1) のキャプション ( コントロール上に表示する文字列 ) に "example1"を設定する場合、以下のように記述します。

MOVE "example1" TO "Caption" OF CommandButton1.

### 配列化されたコントロールのプロパティ

"プロパティ名" OF コントロール名 (配列指定子)

配列化されたオプションボタンコントロール (OptionButton2) の3番めのオプションボタンのキャプションに "example1"を設定する場合、以下のように記述します。

MOVE "example1" TO "Caption" OF OptionButton2(3).

### グループボックスコントロール上に配置されたコントロールのプロパティ

"プロパティ名" OF コントロール名 OF グループボックスコントロール名

グループボックスコントロール (Group1) の上に配置されたチェックボックス (CheckBox3) のキャプションに "example1"を設定する場合、以下のように記述します。

MOVE "example1" TO "Caption" OF CheckBox3 OF Group1.



タブコントロールまたはフレームコントロールの上に配置されたコントロールのプロパティは、標準の形式を使用して記述します。

### 7.2.3 プロパティの参照方法

プロパティの値は、MOVE、ADD、SUBTRACT、COMPUTE、IF、DISPLAYおよびEVALUATE文の7つの文を使用して参照できます。各文の記述形式を以下に示します。

#### MOVE文

プロパティの値を転記して、別のデータまたはプロパティへ取り出すときに使用します。

#### 書きかた

**MOVE プロパティ TO 受け取り側項目**

#### 記述規則

受け取り側項目は、一意名またはプロパティでなければなりません。  
CORRESPONDINGを指定することはできません。  
その他の規則については『COBOL 文法書』を参照してください。

#### 使用例

スタティックテキストコントロール(Static1)のフォントのサイズを、作業場所節で定義した領域(FONT-SIZE)に取り出す場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FONT-SIZE PIC S9(4) COMP-5.
.
PROCEDURE DIVISION.
.
MOVE "Size" OF "Font" OF Static1 TO FONT-SIZE.
```

#### ADD文

数値をもつプロパティの値を、別のデータに加算するときに使用します。

#### 書きかた

**ADD プロパティ TO {一意名 [ ROUNDED ]} ...  
[ ON SIZE ERROR 無条件文 ]  
[ NOT ON SIZE ERROR 無条件文 ]  
[ END - ADD ]**

#### 記述規則

TOの前に、作用対象を複数記述することはできません。  
GIVINGを指定することはできません。

CORRESPONDINGを指定することはできません。  
その他の規則については『COBOL 文法書』を参照してください。

### 使用例

スクロールバーコントロール(Scroll1)の増減値(大)を、作業場所節で定義した数値項目(CURR-VALUE)に足す場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CURR-VALUE PIC S9(4) COMP-5 VALUE IS 0.
.
PROCEDURE DIVISION.
.
 ADD "LargeStep" OF Scroll1 TO CURR-VALUE.
```

### SUBTRACT文

数値をもつプロパティの値を、別のデータから減算するときに使用します。

### 書きかた

```
SUBTRACT プロパティ FROM {一意名[ROUNDED]}...
 [ON SIZE ERROR 無条件文]
 [NOT ON SIZE ERROR 無条件文]
 [END - SUBTRACT]
```

### 記述規則

FROMの前に、作用対象を複数記述することはできません。  
GIVINGを指定することはできません。  
CORRESPONDINGを指定することはできません。  
その他の規則については『COBOL 文法書』を参照してください。

### 使用例

作業場所節で定義した数値項目(CURR-VALUE)から、スクロールバーコントロール(Scroll1)の増減値(大)を引く場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CURR-VALUE PIC S9(4) COMP-5 VALUE IS 0.
.
PROCEDURE DIVISION.
.
 SUBTRACT "LargeStep" OF Scroll1 FROM CURR-VALUE.
```

### COMPUTE文

数値をもつプロパティを使って、演算するときに使用します。

### 書きかた

```
COMPUTE 受け取り側項目 = 算術式 - 1
```

**記述規則**

算術式 - 1 の中にプロパティを記述することができます。  
 受け取り側項目は、一意名またはプロパティでなければなりません。  
 その他の規則については『COBOL 文法書』を参照してください。

**使用例**

スタティックテキストコントロール(Static1)のフォントのサイズを、コマンドボタンコントロール(Command1)のフォントのサイズより2だけ大きくする場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
 COMPUTE "Size" OF "Font" OF Static1
 = "Size" OF "Font" OF Command1 + 2.
```

**IF文**

プロパティの値を使って、比較条件を判定するときに使用します。

**書きかた**

```
IF 比較条件 THEN
 {{文 - 1}}... | NEXT SENTENCE }
 {{ELSE {文 - 2}}... [END - IF]} |
 {ELSE NEXT SENTENCE} |
 {END - IF}}
```

**記述規則**

比較条件の中にプロパティを使用することができます。  
 組み合わせ比較条件を省略することはできません。  
 その他の規則については『COBOL 文法書』を参照してください。

**使用例**

チェックボックスコントロール(Check1)がチェック状態であれば、イメージコントロール(Image1)に"IMAGE-BMP"というリソース名で登録されたイメージを表示する場合、以下のように記述します。

チェック状態でなければ、何も表示しません。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CHECK-ON PIC S9(4) COMP-5 VALUE IS +1.
.
PROCEDURE DIVISION.
.
 IF "Value" OF Check1 = CHECK-ON THEN
 MOVE "IMAGE-BMP" TO "ImageName" OF Image1
 END-IF.
```

## DISPLAY文

少量のデータの内容をハードウェア装置に転送し、表示するときに使用します。

### 書きかた

**DISPLAY** プロパティ ... [**UPON** 呼び名]

### 記述規則

DISPLAY文の記述規則については『COBOL 文法書』を参照してください。

### 使用例

COBOLのコンソールウィンドウに、イメージコントロール(Image1)に表示されているイメージ名を表示する場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
 DISPLAY NC"現在表示しているイメージは "
 "ImageName" OF Image1 NC"です.".
```

## EVALUATE文

プロパティの内容を、複数の条件で評価するときに使用します。

### 書きかた

```
EVALUATE プロパティ
WHEN 選択対象
...
WHEN 選択対象
...
...
END - EVALUATE
```

### 記述規則

選択主体としてプロパティを使用できます。ALSOを指定することはできません。

その他の規則については『COBOL 文法書』を参照してください。

### 使用例

テキストボックスコントロール(Text1)に入力された色名によって、スタティックテキストコントロール(Static1)に表示される文字列を変更する場合、以下のように記述します。

色名として扱えない文字列が入力された場合は、"色無効"と表示します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
 EVALUATE "Text" OF Text1
```

( 続く )

( 続き )

```

WHEN "BLACK"
 MOVE NC"黒色" TO "Caption" OF Static1
WHEN "RED"
 MOVE NC"赤色" TO "Caption" OF Static1
WHEN "YELLOW"
 MOVE NC"黄色" TO "Caption" OF Static1
WHEN OTHER
 MOVE NC"色無効" TO "Caption" OF Static1
END-EVALUATE.

```

#### 7.2.4 プロパティの設定 ( 変更 ) 方法

プロパティの値は、MOVE、ADD、SUBTRACTおよびCOMPUTE文の4つの文を使用して設定 ( 変更 ) できます。各文の記述形式を次に示します。

##### MOVE文

別のデータまたはプロパティから、プロパティへ値を転記して、取り出すときに使用します。

##### 書きかた

**MOVE 送り出し側項目 TO プロパティ**

##### 記述規則

送り出し側項目は、定数、一意名またはプロパティでなければなりません。  
 受け取り側項目を複数指定する場合には、すべての項目がプロパティでなければなりません。  
 CORRESPONDINGを指定することはできません。  
 その他の規則については『COBOL 文法書』を参照してください。

##### 使用例

スタティックテキストコントロール(Static1)のフォントのサイズを設定する場合、以下のように記述します

```

DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
MOVE 16 TO "Size" OF "Font" OF Static1.

```

##### ADD文

数値をもつプロパティの値に、別のデータの値を加算するときに使用します。

##### 書きかた

**ADD { 定数 | 一意名 } TO プロパティ  
 [ END - ADD ]**

### 記述規則

TOの前に、作用対象を複数記述することはできません。  
TOの後に、作用対象を複数記述することはできません。  
ON SIZE ERRORおよびNOT ON SIZE ERRORを指定することはできません。  
GIVINGを指定することはできません。  
CORRESPONDINGを指定することはできません。  
その他の規則については『COBOL 文法書』を参照してください。

### 使用例

スタティックテキストコントロール(Static1)に表示されている文字列のフォントの大きさを、2だけ大きくする場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
ADD 2 TO "Size" OF "Font" OF Static1.
```

## SUBTRACT文

数値をもつプロパティの値から、別のデータの値を減算するときに使用します。

### 書きかた

**SUBTRACT 一意名 FROM プロパティ  
[ END - SUBTRACT ]**

### 記述規則

FROMの前に、作用対象を複数記述することはできません。  
ON SIZE ERRORおよびNOT ON SIZE ERRORを指定することはできません。  
GIVINGを指定することはできません。  
CORRESPONDINGを指定することはできません。  
その他の規則については『COBOL 文法書』を参照してください。

### 使用例

スタティックテキストコントロール(Static1)に表示されている文字列のフォントの大きさを、2だけ小さくする場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
ADD 2 TO "Size" OF "Font" OF Static1.
```

## COMPUTE文

数値をもつプロパティに、演算結果を設定するときに使用します。

### 書きかた

**COMPUTE プロパティ = 算術式 - 1**



**記述規則**

算術式 - 1 の中にもプロパティを書くことができます。  
受け取り側項目は1つのプロパティでなければなりません。  
その他の規則については『COBOL 文法書』を参照してください。

**使用例**

スタティックテキストコントロール(Static1)のフォントのサイズを、コマンドボタンコントロール(Command1)のフォントのサイズより2だけ小さくする場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 .
 COMPUTE "Size" OF "Font" OF Static1
 = "Size" OF "Font" OF Command1 - 2.
```

## 7.3 メソッドの呼び出し方法

メソッドとは、コントロールやオブジェクトに、何らかの動作をさせたり、動作を制御したりするためのサブルーチンのようなものです。  
本節では、メソッドを呼び出すためのプログラミング方法について説明します。

### 7.3.1 メソッドの呼び出し形式

メソッドを呼び出す場合には、以下の形式で手続きを記述します。コントロールやフォームがもつメソッドの詳細については、『リファレンス』を参照してください。

```
INVOKE {[コントロール名[(添字 - 1)] OF }
 {コントロール名[(添字 - 1)] | POW - SELF } |
 プロパティ}
"メソッド名" [USING 引数 - 1 . . .]
 [RETURNING 引数 - 2]
```

#### コントロール名

コントロールのメソッドを呼び出す場合には、そのメソッドをもつコントロールの名前を記述します。コントロール名とは、コントロールを識別するためにフォーム編集時に設定した名前です。

コントロール名については、「[プロジェクト構成要素の命名規則](#) ( p93 )」を参照してください。

#### 添字-1

コントロールが配列化されている場合、個々のコントロールを参照するためのインデックス値を指定します。インデックス値には、データ名または定数を使用できます。

コントロールを配列化した場合のプログラミング方法については、「[配列化したコントロールを使ったアプリケーションを作成する](#) ( p194 )」を参照してください。

#### POW-SELF

フォームのメソッドを呼び出す場合には、POW-SELFを使用します。POW-SELFはフォーム自身を表しています。POW-SELFの代わりに、フォーム名を使うこともできます。

フォーム名については、「[プロジェクト構成要素の命名規則](#) ( p93 )」を参照してください。

#### プロパティ

「[プロパティの記述形式](#) ( p171 )」で記述されたプロパティです。コントロールのプロパティが、そのコントロールがもつオブジェクト ( そのコントロールの構成要素 ) を指している場合に指定します。

**メソッド名**

フォームおよびコントロールがもつメソッドの名前を記述します。メソッド名は大文字と小文字を区別して指定してください。

**引数-1**

メソッドを呼び出す場合の引数を記述します。引数1には、定数およびデータ名を指定できます。ただし、記号定数は、POW-で始まるものだけを使用できません。

**引数-2**

メソッドの復帰値を引数で指定します。引数2には、データ名を指定することができます。

**7.3.2 メソッドの復帰値**

PROGRAM-STATUSを使って、メソッドの呼び出しが成功したかどうかを判定することができます。

正常に終了した場合、0または正数が設定されます。エラーが発生した場合には、負数が設定されます。

ただし、メソッドの呼び出しでプロパティを使用した場合、復帰値を使用することはできません。

**7.3.3 メソッドの呼び出し例**

以下に、メソッドの呼び出し例を示します。

**フォームのメソッド**

```
INVOKE POW-SELF "メソッド名" [USING 引数...]
```

フォームを閉じる場合、以下のように記述します。

```
INVOKE POW-SELF "CloseForm".
```

**コントロールのメソッド（標準）**

```
INVOKE コントロール名 "メソッド名" [USING 引数...]
```

リストボックスコントロール(List1)に、"example1"という項目を追加する場合、以下のように記述します。

```
INVOKE List1 "AddString" USING "example1".
```

**配列化されたコントロールのメソッド**

```
INVOKE コントロール名(配列指定子) "メソッド名" [USING 引数...]
```

配列化されたリストボックスコントロール(List2)の3番めのリストボックスコントロールに、"example1"という項目を追加する場合、以下のように記述します。

```
INVOKE List2(3) "AddString" USING "example1".
```

## グループボックスコントロール上に配置されたコントロールのメソッド

INVOKE コントロール名 OF グループボックスコントロール名 "メソッド名"  
[USING 引数...]

グループボックスコントロール(Group1)上に配置されたリストボックスコントロール(List3)に、"example1"という項目を追加する場合、以下のように記述します。

```
INVOKE List3 OF Group1 "AddString" USING "example1".
```



注意

---

タブコントロールまたはフレームコントロールの上に配置されたコントロールのメソッドは、標準の形式を使用して呼び出します。

---

## プロパティのメソッド

INVOKE プロパティ名 OF コントロール名 "メソッド名" [USING 引数...]  
リストビューコントロール(ListView1)の10番めの項目が、コントロール上の見えない位置にある場合、その10番めの項目が見えるようにリストビューをスクロールさせるには、以下のように記述します。

```
INVOKE "ListItem"(10) OF ListView1 "EnsureVisible".
```

## 7.4 登録集ファイルの利用方法

登録集ファイルを利用する場合、PowerCOBOLでは#INCLUDE文を使います。  
#INCLUDE文を使うことにより、PowerCOBOL固有のプロパティへのアクセスやメソッドの呼び出し処理で、登録集ファイル内で定義されたデータ項目を利用することや、同一の完結文を複数の手続き中で利用することができます。

### #INCLUDE文の書きかた

# I N C L U D E " 登録集ファイル名 "

#### 登録集ファイル名

#INCLUDE文で展開する原文が記述されている、ファイルの名前です。  
ファイルの形式は可変長でなければなりません。

### #INCLUDE文の記述規則

#INCLUDE文は、データ部および環境部の、A領域またはB領域でデータの宣言を取り込むために記述できます。

#INCLUDE文は、手続き部で完結文を取り込むために記述できます。

登録集ファイル内の原文中に、#INCLUDE文を記述できます。

#INCLUDE文の入れ子は、15階層まで利用できます。

#INCLUDE文の入れ子によって、同じ登録集ファイルを再帰的に利用することはできません。

登録集ファイル内の原文は、COBOLの文法に従います。

登録集ファイル内の原文中に、COBOLの原始文操作機能（COPY文）を記述することもできます。ただし、COPY文で取り込まれる登録集ファイル内の原文中には#INCLUDE文を記述できません。

### #INCLUDE文の使用例

登録集ファイル"NUMDATA.COB"を利用する場合、以下のように記述します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
#INCLUDE "NUMDATA.COB".
PROCEDURE DIVISION.
.
.
MOVE "Text" OF "TableCells"(WK-1 WK-2) OF Table1 TO WK-D1.
MOVE WK-D1 TO WK-D2.
```

登録集ファイル"NUMDATA.COB"の内容は、以下のように記述します。

```
000100 01 WK-G.
000200 02 WK-D1 PIC 9(8).
000300 02 WK-D2 PIC 9(8).
000400 01 WK-1 PIC S9(9) COMP-5.
000500 01 WK-2 PIC S9(9) COMP-5.
```



注意

COBOLのCOPY文を使って登録集ファイルを利用することもできます。ただし、その場合、登録集ファイル内で定義されたデータ項目を、プロパティへのアクセスやメソッドの呼び出し処理で使用することはできません。使用した場合、エラーとなります。

たとえば、上記の例で#INCLUDE文をCOPY文に置き換えた場合、2つめの手続きはエラーにはなりませんが、1つめの手続きは表コントロールのプロパティへアクセスしているため、エラーとなります。

これは、フォームを翻訳する場合の、PowerCOBOLの実行論理によるものです。「[イベント駆動型プログラムスタイル](#) ( p7) 」で説明したように、PowerCOBOLは、記述されたイベント手続きを、COBOLコンパイラで翻訳する直前に1つのプログラムにまとめています。同時にプロパティへのアクセスやメソッド呼び出し文を、COBOLコンパイラが認識できる形に変換しています。また、使用しているデータの項類をCOMの型と対応づけています。

#INCLUDE文で指定された登録集ファイルは、この変換処理の最中に展開されます。したがって、登録集ファイル内で定義されたデータ項目を、プロパティへのアクセスやメソッドの呼び出し文の変換時に認識できません。

これに対し、COPY文で指定された登録集ファイルは、COBOL翻訳時に展開されます。したがって、登録集ファイル内のデータ項目を、プロパティへのアクセスやメソッド呼び出し文の変換時に認識できません。

#INCLUDE文で取り込むファイルが#INCLUDE文の記述規則に違反している場合、デバッグおよび診断機能を正しく利用できない場合があります。この場合、#INCLUDE文の記述規則を満たすようにソースプログラムを修正してください。

---

## 7.5 PowerCOBOL固有のデータの取り扱い方法

PowerCOBOLは、COBOLのデータをWindows で利用するため、COBOLの項類をCOMで扱うことができるVT型のデータと対応づけています。

PowerCOBOLは、この対応づけで、プロパティやメソッドの引数で使用するCOMのVT\_BSTR型（文字列）およびVT\_VARIANT型（実行時に型と値を変更することができる型）のデータを扱うことができます。

また、転記の送り出し側または受け取り側で使用する、数字項目や数字編集項目は、VT\_CY型に変換して取り扱っています。

本節では、これらのVT型のデータとCOBOLのデータを、PowerCOBOLがどのように変換し、対応づけているかについて説明します。

### 7.5.1 VT\_BSTR型の変換方法

PowerCOBOLでは、プロパティやメソッドの引数の型がVT\_BSTR型の場合、標準では英数字項目として扱われますが、VT\_BSTR型のプロパティやメソッドの引数を、COBOLの手続き中で、異なる項類のデータと組み合わせて利用する場合、その利用方法により、項類は以下のように変換して扱われます。

MOVE文では、転記元または転記先のデータの項類に変換します。

ADD文、SUBTRACT文、およびCOMPUTE文では、数値項類として変換します。

IF文では、比較対象のデータの項類に合わせて変換します。

たとえば、テキストボックスコントロール(Text1)のTextプロパティはVT\_BSTR型ですが、以下のように記述することで、数値項類に変換され、数値を直接テキストボックスコントロールに表示できます。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NUM PIC S9(4) COMP-5 VALUE IS 0.
PROCEDURE DIVISION.
```

.

```
MOVE NUM TO "Text" OF Text1.
```

また、VT\_BSTRのプロパティやメソッドの引数を、他のプロパティと組み合わせて利用する場合は、項類は変換されず、標準の英数字項目として扱われます。たとえば、コンボボックスコントロール(Combo1)のStyleプロパティ (VT\_I2型) を、テキストボックスコントロール(Text1)のTextプロパティ (VT\_BSTR型) に転記した場合、正しい結果は取得できません。

```
PROCEDURE DIVISION.
```

.

```
MOVE "Style" OF Combo1 TO "Text" OF Text1.
```

このような場合には、作業用のデータを定義し、そのデータへ値をいったん取り出すことにより、正しい結果を得ることができるようになります。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NUM PIC S9(4) COMP-5 VALUE IS 0.
PROCEDURE DIVISION.
.
 MOVE "Style" OF Combo1 TO NUM.
 MOVE NUM TO "Text" OF Text1.
```



注意

EVALUATE文では上記のような変換は行われません。つねに英数字項目として扱われます。

### VT\_BSTR型の空白の取り扱い方法

VT\_BSTR型のプロパティやメソッドの引数に空白を使用した場合、PowerCOBOLでは文字列の末尾にある全角および半角の空白は、すべて取り除かれます。たとえば、以下の手続きを実行すると、スタティックテキストコントロール(Static1)に表示される文字列には、"#A#"ではなく、末尾の空白を取り除いた、"#A"が設定されます。("#"は空白を示します。)

```
MOVE "#A#" TO "Caption" OF Static1.
```

## 7.5.2 VT\_VARIANT型の変換方法

VT\_VARIANT型も、VT\_BSTR型と同様、利用される状況に応じて項類が変換されます。

ただし、VT\_VARIANT型が実際に変換される型は、実行時にならないと決まらないため、手続き中で、正しく項類が変換されない場合があります。変換に失敗した場合には、"メソッドの呼び出しまたはプロパティの設定/参照時の引数に誤りがあります"といったメッセージが表示されます。この場合、VT\_VARIANT型の値を設定しているプログラムを変更して、変換可能な値を設定するか、VT\_BSTR型の変換で説明したように、作業用のデータへ値をいったん取り出すようにしてください。



ワンポイント

PowerCOBOLのフォーム、コントロールおよびオブジェクトは、VT\_VARIANT型のプロパティやメソッドの引数をもちません。PowerCOBOL以外の製品で提供されるActiveXコントロールを利用する場合、VT\_VARIANT型のプロパティやメソッドの引数をもつことがあります。

## 7.5.3 VT\_CY型への変換方法

転記の送り出し側または受け取り側で使用する数字項目または数字編集項目は、VT\_CY型に変換されます。



VT\_CY型は、COBOLの"S9(14)V9(4)"に対応しています。したがって、プロパティの値を直接参照したり、設定したりする場合、整数部は14桁、小数部は4桁までしか扱うことができません。

5桁以上の小数部を扱う場合には、以下のように、転記元または転記先の項類を英数字項目として扱ってください。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-V PIC +99.999999.
01 WK-X REDEFINES WK-V PIC X(10).
01 WK-N PIC 99V999999.
PROCEDURE DIVISION.
```

\* COBOLのデータ項目からTextプロパティへの転記方法

```
MOVE WK-N TO WK-V
MOVE WK-X TO "Text" OF Text1
```

\* TextプロパティからCOBOLのデータ項目への転記方法

```
MOVE "Text" OF Text1 TO WK-X
MOVE WK-V TO WK-N
```

テキストボックスコントロール以外のコントロールなどで扱う場合は、TextプロパティをCaptionプロパティに置き換えてください。

この転記では、テキスト種別が"標準"の場合、コントロールに設定されている数字の文字列によって、WK-Xに設定される桁位置が以下のように、変わってしまいます。( "#"は空白を示します。)

| コントロールの文字列 | WK-Xの値       |
|------------|--------------|
| "12.3"     | "12.3#####"  |
| "1.23456"  | "1.23456###" |

したがって、この数字文字列をWK-Nに設定するためには、WK-Xの内容を解析する必要があります。

また、テキスト種別が"1 - COBOL PICTURE属性"で、PICTURE文字列を数字項目または数字編集項目として設定している場合、コントロールに設定されている数字の文字列は、以下のように変換されWK-Xに設定されます。

符号 整数部桁数分の文字列 [ 小数点 小数部桁数分の文字列 ]

したがって、コントロールにどのような数字文字列が設定されていても、WK-Xに設定される桁位置は固定になります。コントロールのPICTURE文字列が"Z9.99999"の場合、以下ようになります。

| コントロールの文字列 | WK-Xの値       |
|------------|--------------|
| "12.3"     | " +12.30000" |
| "1.23456"  | " +01.23456" |

## 7.6 Unicodeの取り扱い方法

本節では、実行時のコード系をUnicodeにし、プログラムを作成する場合の取り扱い方法について説明します。

実行時のコード系は、ランタイムコードセットを指定することで変更できます。ランタイムコードセットの変更方法については、「[Unicodeを利用する](#) ( p99 )」を参照してください。

PowerCOBOLでは、データとしてだけUnicode文字を取り扱うことができます。したがって、コントロール名、リソース名、イベント手続き名およびイベント手続き中などでは、Unicode文字を使用できません。

PowerCOBOLでは、フォームやコントロールのKeyPressイベントやPreKeyPressイベントの引数として、押されたキーの文字コードが渡されます。PowerCOBOLで提供されているコントロールを利用する範囲では、この文字コードはランタイムコードセットの指定に従って、自動的に切り替えられます。

しかし、サードベンダーが提供しているカスタムコントロールを利用した場合、つねに引数の文字コードをシフトJISコードで渡すイベントもあります。そのような場合、イベントの引数で渡された文字コードをUnicodeへ変換してから、文字を扱う必要があります。

PowerCOBOLでは、文字コードを変換するために、以下の変換ルーチンを提供しています。

シフトJISコードからUnicodeへの変換:POWERACPTOUCS2

UnicodeからシフトJISコードへの変換:POWERUCS2TOACP



これらの変換ルーチンは、Windows の機能を利用して実現しています。したがって、Windows の仕様により、たとえば、Unicode シフトJISコード Unicodeや、シフトJISコード Unicode シフトJISコードのように変換を繰り返した場合、同じコードに戻らない場合があります。

### 7.6.1 シフトJISコードからUnicodeへの変換

シフトJISコードからUnicodeへの変換は、以下の変換ルーチンを呼び出します。

#### 書きかた

```
CALL "POWERACPTOUCS2"
 USING シフトJISコード文字
 Unicode文字
```

#### シフトJISコード文字

```
PIC S9(4) COMP - 5.
```

KeyPressイベントやPreKeyPressイベントの引数として渡されたシフトJISコードの文字を示すデータ項目です。

**Unicode文字**

P I C N .

Unicodeへ変換された文字を取り出すためのデータ項目です。

**復帰値**

復帰値 (PROGRAM-STATUS) は、つねに0です。

**7.6.2 UnicodeからシフトJISコードへの変換**

UnicodeからシフトJISコードへの変換は、以下の変換ルーチンを呼び出します。

**書きかた**

```
CALL "POWERUCS2TOACP"
 USING Unicode文字
 シフトJISコード文字
```

**Unicode文字**

P I C N .

Unicodeの文字を示すデータ項目です。

**シフトJISコード文字**

P I C S 9 ( 4 ) COMP - 5 .

シフトJISコードへ変換された文字を取り出すためのデータ項目です。

**復帰値**

復帰値 (PROGRAM-STATUS) は、つねに0です。

## 7.7 プログラミング上の留意事項

本節では、PowerCOBOLでプログラムを作成する場合の留意事項について説明します。

### 利用者語

PowerCOBOL内で翻訳の対象となる手続き内では、以下の文字で始まる利用者語を使用できません。

POW-

POWER-

また、以下の名前と同じ文字列も使用できません。

フォームまたはコントロールの名前

フォームまたはコントロールのプロパティ名

フォームまたはコントロールのメソッド名

フォームまたはコントロールのイベント名

### PROGRAM-STATUS

プロパティを参照または設定すると、PROGRAM-STATUSの値は変更され、不定の値が設定されます。これは、PowerCOBOLが内部的にランタイムシステムのサブルーチンを呼び出しているためです。したがって、プロパティの参照または設定処理をまたがって、PROGRAM-STATUSを参照することはできません。

### イベント手続きの引数

引数をもつイベント手続きを編集する場合、手続き中の連絡節(LINKAGE SECTION)やUSING指定を削除したり、引数の数を変更したりすると、正しく動作しない場合があります。

これらの記述を削除したり、変更したりしないようにしてください。

### COBOLの表示ファイル機能

COBOLの表示ファイルを使用するプログラムでは、表示ファイルを扱う部分を、共通手続きまたは別翻訳単位として、かならずOPEN文からCLOSE文までを実行してから、呼び出し元に復帰するようにしてください。また、表示ファイルを開いているあいだは、フォーカスを別のウィンドウに移さないようにしてください。

### 手続きの終了方法

PowerCOBOLのアプリケーションでは、COBOLのSTOP RUN文を使用しないでください。STOP RUN文が実行されると、PowerCOBOLの資源がWindows のメモリ上に残ったままプログラムが終了してしまいます。

### 手続きの無限ループ

イベント手続きの中で、再びそのイベントが発生する再帰的な手続きを記述すると、無限ループとなります。

たとえば、テキストボックスコントロールのChangeイベントで、同じコントロールのTextプロパティ(テキスト文字列)に文字列を設定すると、再びChangeイベントが発生します。このとき、終了するための判定がなければ無限ループ

となります。このような場合、アプリケーションは通常、スタック違反で異常終了します。

### イベント駆動型とサブシステムの応答

PowerCOBOLはイベント駆動型のランタイムシステムをもっています。したがって、たとえばプリンタやデータベースなど応答を待たなければならない(応答を待たずに続けて処理すると不具合が発生する)ようなサブシステムの処理では、プログラムの中で応答を待つような手続きを実行してください。

たとえば、コマンドボタンコントロール(Command1)のClickイベント中で、印刷コントロール(Print1)のPrintFormメソッドを記述している場合、コマンドボタンを複数回続けてクリックすると、アプリケーションが異常終了する場合があります。このような場合には、以下のように、PrintFormメソッドを実行する前後でコマンドボタンコントロールのEnabledプロパティの値を切り替えてください。

```
MOVE POW-FALSE TO "Enabled" OF Command1.
INVOKE Print1 "PrintForm".
MOVE POW-TRUE TO "Enabled" OF Command1.
```

### 登録集ファイル

#INCLUDE文による、大量の登録集ファイルの利用は、デバッグや翻訳性能の点でお勧めできません。なるべく最小限にとどめるようにしてください。  
#INCLUDE文の使用方法については、「[登録集ファイルの利用方法](#)( p183)」を参照してください。

### ロングファイル名の扱い

ロングファイル名を扱う場合、ファイル名のパスとして有効な文字数は259文字までです。ただし、上限まで長い名前を使用することはお勧めできません。

### 複数のコントロール間でのイベント発生順序

複数のコントロール間で発生するイベントの順序は、とくに決まっていません。たとえば、フォーム上に2つのテキストボックスコントロール(TextBox1とTextBox2)があり、TextBox1がフォーカスをもっている状態から、TextBox2にフォーカスが移った場合、TextBox1のLostFocusイベントとTextBox2のGotFocusイベントの発生順序は、決まっていません。したがって、アプリケーションを作成する場合は、複数のコントロール間でのイベントの発生順序に依存しないプログラムを作成してください。

# ***MEMO***

---

## 第8章 プログラミングテクニック

---

本章では、いろいろなアプリケーションを作成するためのプログラミングテクニックについて、サンプルプログラムを作成しながら説明します。

---

## 8.1 配列化したコントロールを使ったアプリケーションを作成する

本節では、配列化したコントロールを使ったアプリケーションを、以下のよう  
に作成しながら説明します。

1. フォームにコントロールを配置します。
2. コントロールを配列化します。
3. 配列化したコントロールの手続きを記述します。

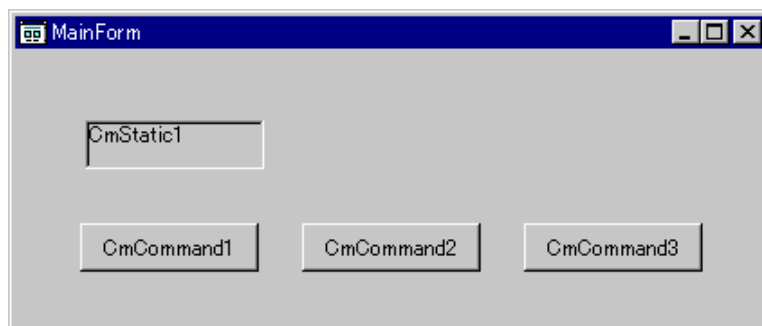


本節でこれから作成していくサンプルプログラムは、"Array¥Array1.ppj"に  
格納されています。必要に応じて参照してください。

### フォームにコントロールを配置する

以下の手順で、フォームにコントロールを配置します。

1. プロジェクトのテンプレートから[ 標準フォーム ]を選択し、新規にプ  
ロジェクトを作成します。
2. フォームを開き、スタティックテキストコントロール1個と、コマンド  
ボタンコントロール3個を、下図のように配置します。



プロジェクトの作成方法は、「[プロジェクトを作成する](#) ( p51 )」を、コント  
ロールの配置方法は、「[フォームを編集する](#) ( p55 )」を参照してください。

### コントロールを配列化する

以下の手順で、コマンドボタンコントロールを配列化します。

1. 左側に配置したコマンドボタンコントロール"CmCommand1"を選択しま  
す。
2. ポップアップメニューの[ 配列化 ]サブメニューから[ 新規作成 ]コマ



- ンドを選択します。
- プロジェクトウィンドウのデザインツリーウィンドウ上で、配列名 "CmCommand1" を "ArrayButton1" に変更します。
  - フォーム編集ウィンドウに戻り、中央に配置したコマンドボタンコントロール "CmCommand2" を選択します。
  - ポップアップメニューの [ 配列化 ] サブメニューから [ ArrayButton1 ] コマンドを選択します。
  - 同様に、右側に配置したコマンドボタンコントロール "CmCommand3" を選択し、ポップアップメニューの [ 配列化 ] サブメニューから [ ArrayButton1 ] コマンドを選択します。
  - プロジェクトウィンドウのデザインツリーウィンドウで、"ArrayButton1 [ 配列 ]" に3個のコマンドボタンコントロールが含まれていることを確認します。



配列化するための、その他の操作方法については、「[コントロールを配列化して利用する](#) ( p107 )」を参照してください。

### 配列化したコントロールの手続きを記述する

複数のコントロールを配列化すると、イベント手続きはそれぞれのコントロールがもつのではなく、同じイベント手続きを共有することになります。つまり、どのコントロールでイベントが発生しても同じイベント手続きが実行されるということです。

配列化しているコントロールのイベント手続きを編集する場合、イベント手続きの引数として、イベントが発生したコントロールのインデックス (POW-INDEX) が追加されます。このデータ項目を参照することにより、配列内のどのコントロールでイベントが発生したか知ることができます。配列内のコントロールがもつインデックスは、プロパティ設定ダイアログボックスの [ 共通 ] タブにある、[ インデックス ] の値です。

このサンプルプログラムでは、以下のように動作する手続きを記述します。

左側 (インデックスが1) のボタンをクリックすると、スタティックテキストコントロールの背景色が赤に変化します。

中央 (インデックスが2) のボタンをクリックすると、スタティックテキストコントロールの背景色が緑に変化します。

右側 (インデックスが3) のボタンをクリックすると、スタティックテキストコントロールの背景色が青に変化します。

手続きは、以下のようになります。

#### ArrayButton1-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-COLOR PIC S9(9) COMP-5.
```

( 続く )

( 続き )

```
LINKAGE SECTION.
01 POW-INDEX PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW- INDEX.
* クリックしたコマンドボタンのインデックスから背景色を決めます。
 EVALUATE POW- INDEX
 WHEN 1
 MOVE POW-COLOR-RED TO W-COLOR
 WHEN 2
 MOVE POW-COLOR-GREEN TO W-COLOR
 WHEN 3
 MOVE POW-COLOR-BLUE TO W-COLOR
 END-EVALUATE
* スタティックテキストの背景色を設定します。
 MOVE W-COLOR TO "BackColor" OF CmStatic1
```



ワンポイント

---

このサンプルプログラムを拡張し、スタティックテキストも配列化したサンプルプログラムが、"Array¥Array2.ppj"に格納されています。

---



注意

---

配列化した場合に引数に追加されるインデックス"POW-INDEX"のデータ項目名を変更することはできませんが、引数自体を削除することはできません。引数を削除した場合、アプリケーションが正常に動作しなくなる場合があります。

---

## 8.2 オブジェクトを使ったアプリケーションを作成する

PowerCOBOLでは、コントロールだけでなく、オブジェクトについても手続き上で扱うことができます。

たとえば、ツリービューコントロールを構成するノードオブジェクト、リストビューコントロールを構成するリストアイテムオブジェクトやツールバーコントロールのボタンオブジェクトなどにアクセスし、それらのプロパティを操作することができます。

また、オブジェクトをイベントの引数として受け取ることもできます。

### 8.2.1 オブジェクトへアクセスするには

オブジェクトへアクセスするには、以下の3つの方法があります。

コントロールと同様にアクセスする方法

オブジェクトのクラス名を利用してアクセスする方法

NetCOBOLの\*COMクラスを利用してアクセスする方法

#### コントロールと同様にアクセスする

フォーム(Form)オブジェクトは、コントロールと同様にアクセスすることができます。たとえば、フォームのキャプションを変更する場合は、以下のように記述できます。

```
MOVE "新キャプション名" TO "Caption" OF POW-SELF
```

この形式では、POW-SELFの代わりにフォーム名を指定することもできます。フォーム以外のオブジェクトは、コントロールの構成要素になっています。コントロールは、これらの構成要素となるオブジェクトを指し示すプロパティをもっています。このプロパティを使って、COBOLの集団項目内のデータにアクセスするように、修飾を用いてオブジェクトにアクセスします。

たとえば、ツリービューコントロールの2番めのルートノード（最上位階層にある項目）に表示されている文字列を参照する場合は、以下のように記述します。

```
MOVE "Text" OF "Root"(2) OF ツリービューコントロール名 TO 転記先
```

また、スタティックテキストコントロールのフォントサイズを20に変更する場合は、以下のように記述します。

```
MOVE 20 TO "Size" OF "Font" OF スタティックテキストコントロール名
```



ワンポイント

各コントロールがもつオブジェクトの種類、およびオブジェクトがもつプロパティやメソッドについての詳細は、『リファレンス』を参照してください。

## オブジェクトのクラス名を利用してアクセスする

オブジェクトに対応するクラス名をUSAGE句で宣言したデータに、オブジェクト自体を格納することにより、そのデータを利用してオブジェクトにアクセスすることができます。

PowerCOBOLでは、以下のクラス名が利用できます。

| オブジェクトの種類         | クラス名          |
|-------------------|---------------|
| カラム(Column)       | POW-CCOLUMN   |
| フォント(Font)        | POW-CFONT     |
| リストアイテム(ListItem) | POW-CLISTITEM |
| ノード(Node)         | POW-CNODE     |
| その他               | POW-COBJECT   |

たとえば、ツリービューコントロールのノードにアクセスする場合、以下のよう記述します。

```
WORKING-STORAGE SECTION.
```

```
01 WK-ROOT-NODE USAGE IS OBJECT REFERENCE POW-CNODE.
```

```
01 WK-CHILD-NODE USAGE IS OBJECT REFERENCE POW-CNODE.
```

```
01 WK-COUNT-NUM PIC S9(9) COMP-5.
```

```
PROCEDURE DIVISION.
```

```
* ツリービューコントロールの2番めのルートノードを取り出します
```

```
MOVE "Root"(2) OF ツリービューコントロール名 TO WK-ROOT-NODE
```

```
* 2番めのルートノードの3番めの子ノードを取り出します
```

```
MOVE "Child"(3) OF WK-ROOT-NODE TO WK-CHILD-NODE
```

```
* 子ノードの表示文字列を変更し、その配下にあるノード数を求めます
```

```
MOVE "新テキスト" TO "Text" OF WK-CHILD-NODE
```

```
MOVE "Count" OF WK-CHILD-NODE TO WK-COUNT-NUM
```



ワンポイント

各コントロールがもつオブジェクトの種類、およびオブジェクトがもつプロパティやメソッドについての詳細は、『リファレンス』を参照してください。また、その他の使用例については、オブジェクトをもつコントロールのサンプルプログラム ("¥TreeView¥TreeView.ppj" や "ListView¥ListView.ppj" など) を参照してください。



注意

オブジェクトのクラス名を利用してプロパティにアクセスする場合は、大文字と小文字を区別してプロパティ名を記述する必要があります。

## NetCOBOLの\*COMクラスを利用してアクセスする

NetCOBOLのCOM連携機能である\*COMクラスを利用して、オブジェクトにアクセスすることができます。

この方法は、サードパーティによって提供されているコントロールのオブジェ

クトにアクセスする場合など、オブジェクトのクラス名がわからないときに使用します。アクセスするための手順を以下に示します。

1. フォーム環境部のリポジトリ段落(REPOSITORY)で\*COMクラスを宣言します。
2. イベント手続きまたはフォームの作業場所節(WORKING-STORAGE)で、オブジェクトを取り出すためのデータを定義します。
3. イベント手続きまたはフォームの作業場所節で、\*COMクラスのオブジェクト(COMオブジェクト)として参照するためのデータを定義します。
4. 手続き中で、処理対象とするオブジェクトを取り出します。
5. "POWERCONVTOCOM"を呼び出し、オブジェクトをCOMオブジェクトに変換します。
6. COMオブジェクトを使って、オブジェクトにアクセスします。
7. 最後に、使用したCOMオブジェクト用のデータにNULLを設定し、クリアします。

"POWERCONVTOCOM"は、オブジェクトをCOMオブジェクトとして扱えるようにするための、PowerCOBOLで用意している変換ルーチンです。また、COMオブジェクトをPowerCOBOLで扱えるようにするための変換ルーチンとして、"POWERCONVFROMCOM"があります。"POWERCONVTOCOM"および"POWERCONVFROMCOM"の記述形式を以下に示します。

#### POWERCONVTOCOMの書きかた

```
CALL "POWERCONVTOCOM"
 USING オブジェクト
 RETURNING COMオブジェクト
```

#### オブジェクト

USAGE IS OBJECT REFERENCE POW-COBJECT  
PowerCOBOLのプロパティやメソッドを使って取り出した、オブジェクトを示すデータ項目です。この変換ルーチンの入力となります。この引数を省略することはできません。データ部の定義では、"USAGE IS"を省略することができます。

#### COMオブジェクト

USAGE IS OBJECT REFERENCE COMのクラス名  
NetCOBOLで扱うことができるCOMオブジェクトを取り出すためのデータ項目です。この変換ルーチンの出力となります。この引数を省略することはできません。データ部の定義では、"USAGE IS"を省略することができます。COMのクラス名は、リポジトリ段落(REPOSITORY)でNetCOBOLの\*COMクラスとして宣言したクラス名です。

#### 復帰値

ありません。PROGRAM-STATUSの値は不定となります。

#### POWERCONVTOCOMの使用例

たとえば、ツリービューコントロール上でノードが選択されている状態で、コマンドボタンコントロール(Btn-Rename)をクリックしたとき、選択中のノードに表示されている文字列を変更するには、以下のような手続きを作成します。

## REPOSITORY

CLASS COM AS "\*COM"

### Btn-Rename-Click

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-TEXT PIC X(100).
* オブジェクトを取り出すための領域
01 PCMNODE USAGE IS OBJECT REFERENCE POW-COBJECT.
* *COMクラスを使ってアクセスするためのノード用のCOMオブジェクト
01 COMNODE USAGE IS OBJECT REFERENCE COM.
PROCEDURE DIVISION.
* 指定された文字列を取り出します。
 MOVE "Text" OF CmText11 TO WK-TEXT
* ツリービューコントロールで選択中のノードオブジェクトを取り出します。
 MOVE "SelNode" OF TreeView TO PCMNODE
* COM連携機能を利用して、NODEオブジェクトにアクセスします。
 IF PCMNODE NOT = 0 THEN
* *COMクラスを使ってアクセスできるように、
* ノードオブジェクトを変換します。
 CALL "POWERCONVTOCOM" USING PCMNODE RETURNING COMNODE
* 指定された文字列を選択中のノードオブジェクトのラベルに設定します。
 INVOKE COMNODE "SET-TEXT" USING WK-TEXT
* 使用したCOMオブジェクトをクリアします。
 SET COMNODE TO NULL
 END-IF

```



この例題では、PowerCOBOLが提供するツリービューコントロールを使っています。したがって、PCMNODEをPOW-CNODE型のオブジェクトとして宣言することもできますが、POWERCONVTOCOMの利用方法を示すために、POW-COBJECTを使っています。

POW-CNODEを使って宣言した場合、COMオブジェクトに変換しないで、ノードオブジェクトを操作できますが、POW-COBJECTを使って宣言した場合、オブジェクトを操作するにはPOWERCONVTOCOMでCOMオブジェクトに変換する必要があります。また、最後にCOMオブジェクトをクリアする必要があります。

"SET-TEXT"メソッドは、NetCOBOLが用意しているメソッドで、対象となるCOMオブジェクトの"TEXT"プロパティに引数の値を設定することを示しています。詳細は、『NetCOBOL 使用手引書』を参照してください。ここで作成したサンプルプログラムは、"ComClass\*AccessToObject.ppj"に格納されています。必要に応じて参照してください。

**POWERCONVFROMCOMの書きかた**

```
CALL "POWERCONVFROMCOM"
 USING COMオブジェクト
 RETURNING オブジェクト
```

**COMオブジェクト**

USAGE IS OBJECT REFERENCE COMのクラス名  
NetCOBOLで扱うことができるCOMオブジェクトを示すデータ項目です。この変換ルーチンの入力となります。この引数を省略することはできません。データ部の定義では、"USAGE IS"を省略することができます。COMのクラス名は、リポジトリ段落(REPOSITORY)でNetCOBOLの\*COMクラスとして宣言したクラス名です。

**オブジェクト**

USAGE IS OBJECT REFERENCE POW-COBJECT  
PowerCOBOLのプロパティやメソッドで利用するために取り出した、COMオブジェクトを示すデータ項目です。この変換ルーチンの出力となります。この引数を省略することはできません。データ部の定義では、"USAGE IS"を省略することができます。

**復帰値**

ありません。PROGRAM-STATUSの値は不定となります。

**POWERCONVFROMCOMの使用例**

たとえば、ADOデータソースコントロールのRecordsetプロパティに、COMオブジェクトとして作成されたRecordsetオブジェクトを設定する場合、以下のような手続きを作成します。

**REPOSITORY**

```
CLASS COM AS "*COM"
```

**MainForm-WORKING-STORAGE**

```
* Recordsetオブジェクトを示すCOMオブジェクト
01 COMRST OBJECT REFERENCE COM.
* オブジェクトを取り出すための領域
01 PCMRST OBJECT REFERENCE POW-COBJECT.
```

**Btn-Connect-Click**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
* 作成したRecordsetオブジェクトをPowerCOBOLのオブジェクト形式に変換します。
CALL "POWERCONVFROMCOM" USING COMRST RETURNING PCMRST
* ADOデータソースコントロールのRecordsetプロパティに設定します。
INVOKE CmADODataSource1 "SETREF-Recordset" USING PCMRST
...
```

## 8.2.2 オブジェクトをイベントの引数として受け取るには

オブジェクトを引数として受け取るイベント手続きを編集する場合、手続きを記述する前に、連絡節の修正が必要です。たとえば、ツリービューコントロールのノードがクリックされた場合に発生するNodeClickイベントでは、次のような手続きが初期値として生成されます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-PCMNODE OBJECT REFERENCE [Class Name].
PROCEDURE DIVISION USING POW-PCMNODE.
```

オブジェクトを受け取る場合、上記の[Class Name]の部分を変更する必要があります。修正してオブジェクトを受け取るには、以下の2つの方法があります。

オブジェクトのクラス名を設定して受け取る方法

NetCOBOLの\*COMクラスを利用して受け取る方法

### オブジェクトのクラス名を設定して受け取る

受け取るオブジェクトに対応するクラス名を、[Class Name]の部分に記述することで、手続き中でオブジェクトにアクセスできます。たとえば、上記のNodeClickイベントでは、ノードオブジェクトを受け取るため、[Class Name]を以下のように変更します。この変更により、手続き中でノードオブジェクト(POW-PCMNODE)にアクセスできるようになります。

#### NodeClickイベント

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-PCMNODE OBJECT REFERENCE POW-CNODE.
PROCEDURE DIVISION USING POW-PCMNODE.
```

利用可能なオブジェクトのクラス名については、「[オブジェクトのクラス名を利用してアクセスする](#) ( p198) 」の表を参照してください。

### NetCOBOLの\*COMクラスを利用して受け取る

NetCOBOLのCOM連携機能である\*COMクラスを利用してオブジェクトを受け取ります。この方法は、サードパーティによって提供されているコントロールのイベントの引数に、オブジェクトが割り当てられている場合など、クラス名がわからないオブジェクトを受け取る場合に利用します。この方法では、以下の手順で、引数で受け取ったオブジェクトを操作できるようにします。

1. フォーム環境部のリポジトリ段落(REPOSITORY)で\*COMクラスを宣言します。
2. 連絡節の[Class Name]をPOW-COBJECTに変更します。



3. イベント手続きまたはフォームの作業場所節で、COMオブジェクトとして参照するためのデータを定義します。
  4. PowerCOBOLで用意している変換ルーチン"POWERCONVTOCOM"を呼び出し、連絡節で受け取ったオブジェクトをCOMオブジェクトに変換します。
  5. COMオブジェクトを使って、オブジェクトにアクセスします。
  6. 最後に、使用したCOMオブジェクトにNULLを設定し、クリアします。
- "POWERCONVTOCOM"の記述形式については、「[NetCOBOLの\\*COMクラスを利用してアクセスする](#) ( p198) 」を参照してください。
- たとえば、NodeClickイベントで、NetCOBOLの\*COMクラスを利用して引数を受け取る場合、以下ようになります。

**REPOSITORY**

```
CLASS COM AS "*COM"
```

**NodeClickイベント**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

* *COMクラスを使ってアクセスするためのノード用のCOMオブジェクト
01 COMNODE OBJECT REFERENCE COM.
LINKAGE SECTION.
01 POW-PCMNODE OBJECT REFERENCE POW-COBJECT.
PROCEDURE DIVISION USING POW-PCMNODE.

* 最初に受け取ったオブジェクトをCOMオブジェクトに変換します。
 CALL "POWERCONVTOCOM" USING POW-PCMNODE RETURNING COMNODE
* COMオブジェクト(COMNODE)を使ってオブジェクトにアクセスします。
 ...
* 最後に、使用したCOMオブジェクトをクリアします。
 SET COMNODE TO NULL
```

## 8.3 複数ウィンドウをもつアプリケーションを作成する

本節では、複数ウィンドウをもつアプリケーションの作成方法について説明します。

PowerCOBOLで複数のウィンドウをもつアプリケーションを作成するには、以下の4つの方法があります。

OpenFormメソッドを使用する方法

CallFormメソッドを使用する方法

ActiveXコントロールを作成して利用する方法

オートメーションサーバにして利用する方法

本節では、最初の2つの方法について説明します。

OpenFormメソッドは、別のフォームをモードレスなウィンドウとして開くときに使用します。CallFormメソッドは、別のフォームをモーダルなウィンドウとして開くときに使用します。

モードレスなウィンドウとは、開いたフォームを閉じなくても、呼び出し元フォームの処理が続行できる形態です。逆に、モーダルなウィンドウとは、開いたフォームが閉じられるまで、呼び出し元フォームの処理が止まっている形態です。通常、ダイアログボックスはモーダルなウィンドウとして作成されます。その他の方法については、「[ActiveXコントロールを使ったアプリケーションを作成する](#) ( p260) 」および「[オートメーションサーバを使ったアプリケーションを作成する](#) ( p270) 」を参照してください。

### 8.3.1 OpenFormメソッドを使用する

OpenFormメソッドを使用することにより、PowerCOBOLで作成した別のフォームを開くことができます。

#### OpenFormメソッドの書きかた

OpenFormメソッドは、以下のように記述します。

```
INVOKE POW - SELF "OpenForm"
 USING "フォーム名"
 ["DLL名"]
 [フォーム識別ID]
RETURNING 復帰値
```

#### POW-SELF

これから子フォームを開こうとしている親フォーム自身を表しています。

POW-SELFの代わりに、親フォーム自身のフォーム名を使うこともできます。

フォーム名については、「[プロジェクト構成要素の命名規則](#) ( p93) 」を参照してください。

### フォーム名

これから開こうとしている子フォームの名前を指定します。

### DLL名

開こうとしている子フォームが別のDLL（モジュール）に含まれている場合、そのDLLファイル名を指定します。

### フォーム識別ID

複数の子フォームを開き、それらのどれかが閉じられた場合に、親フォームのCloseChildイベントで閉じられた子フォームを識別するために使用するIDです。

### 復帰値

正常に子フォームを開くことができた場合は、0が設定されます。失敗した場合、0以外の値が設定されます。

### OpenFormメソッドを使った複数ウィンドウの使用例

本節では、「[アプリケーションを作成しよう](#)（p49）」で作成したアプリケーションに新しくフォームを追加し、フォームの呼び出し方法のサンプルプログラムを作成します。このサンプルプログラムでは、表に入力できる商品が追加できるよう、以下の機能を追加しています。

[ 商品の追加 ] ボタンをクリックすると、商品の情報を追加するための、[ 商品の追加 ] ウィンドウが表示されます。

[ 商品の追加 ] ウィンドウが表示されると、[ 購入商品の入力画面 ] ウィンドウの [ 商品の追加 ] ボタンが無効になります。

[ 商品の追加 ] ウィンドウで [ 商品名 ] と [ 単価 ] を入力し、[ 追加 ] ボタンをクリックすると、商品ファイルに商品の情報が追加されます。追加した商品は、[ 購入商品の入力画面 ] ウィンドウの表で使うことができるようになります。

[ 商品の追加 ] ウィンドウの [ 閉じる ] ボタンをクリックすると、[ 商品の追加 ] ウィンドウが閉じます。

[ 商品の追加 ] ウィンドウが閉じられると、[ 商品の追加 ] ボタンが有効になります。

アプリケーションは、以下の手順で作成します。

1. モジュールに子ウィンドウとなる新しいフォームを追加し、名前を設定します。
2. 追加した子ウィンドウを開くためのボタンを、呼び出し元のフォームに追加します。
3. 呼び出し元のフォームに、フォームを開くための手続きを記述します。
4. コントロールを新しいフォームの適切な位置に配置し、大きさを変更します。
5. 新しいフォームの手続きを記述します。



本節で作成するサンプルプログラムは、「SimpleForm¥Table2.ppj」に格納されています。必要に応じて参照してください。

## モジュールに新しいフォームを追加する

以下の操作により、モジュールに新しいフォームを追加し、名前を設定します。

1. プロジェクトウィンドウのデザインツリーウィンドウで、モジュールを選択します。
2. ポップアップメニューの[フォーム作成]コマンドを選択します。
3. モジュールにCfFormN (Nは正数値) という名前でフォームが追加され、フォーム編集ウィンドウが表示されます。
4. フォームのプロパティ設定ダイアログボックスを開き、[名前]を "AddForm" に変更します。
5. [キャプション]を "商品の追加" に変更します。

## 呼び出し元のフォームを編集する

新しく作成したフォームを呼び出すために、以下の手順でフォームを編集します。

1. MainFormのポップアップメニューで[開く]コマンドを選択し、フォーム編集ウィンドウを表示します。
2. 下図のように、コマンドボタンコントロールをフォームの空いている位置に任意に配置します。
3. 配置したコマンドボタンコントロールのプロパティ設定ダイアログボックスで、[キャプション]を "商品の追加"、[名前]を "BT-ADD" に変更します。



## フォームを開くための手続きを記述する

新しく作成したフォームを開くために、以下の手続きを追加します。

- 開くフォームを識別するためのIDの定義(WORKING-STORAGE)
- [商品の追加] ボタンをクリックしたときの手続き(BT-ADD-Click)
- 開いたフォームが閉じられたときの手続き(MainForm-CloseChild)

### WORKING-STORAGE

フォームの作業場所節(WORKING-STORAGE)に以下の定義を追加します。

- \* 商品追加用フォームのID
- 01 ADD-FORM-ID PIC S9(9) COMP-5 VALUE 1 GLOBAL.

**BT-ADD-Click**

[ 商品の追加 ] ウィンドウを開き、[ 商品の追加 ] ボタンを無効（淡色表示）にします。

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.

* 商品追加用のフォームを開きます。
 INVOKE POW-SELF "OpenForm" USING "AddForm" ADD-FORM-ID
 RETURNING ReturnValue

* フォームが開くことができたなら、ボタンを無効にします。
 IF ReturnValue = 0 THEN
 MOVE POW-FALSE TO "Enabled" OF BT-ADD
 END-IF

```



[ 商品の追加 ] ボタンを無効にするのは、ボタンが重複してクリックされることを防ぐためです。ボタンが重複してクリックされると、複数の [ 商品の追加 ] ウィンドウを表示しようとするため、処理の内容によっては、データに矛盾が生じたり、異常終了したりすることがあります。このサンプルプログラムでは、OpenFormメソッドの第2引数(DLLの名前)を省略しています。DLLとして作成された他のモジュールに含まれるフォームを開く場合は、第2引数にDLL名を指定してください。OpenFormメソッドについての詳細は、『リファレンス』を参照してください。

**MainForm-CloseChild**

CloseChildイベントは、このフォーム中の手続きから開かれたフォームが閉じられた場合に発生します。したがって、BT-ADD-Clickイベントで開いた [ 商品の追加 ] ウィンドウ(AddForm)が閉じられたとき、ここに記述した手続きが実行されます。ここでは、[ 商品の追加 ] ウィンドウが閉じられたら、再び [ 商品の追加 ] ボタンがクリックできるよう、ボタンを有効にしています。

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-COOKIE PIC S9(9) COMP-5.
PROCEDURE DIVISION POW-COOKIE.

* 商品追加用フォームが閉じられたら、ボタンを有効にします。
 EVALUATE POW-COOKIE
 WHEN ADD-FORM-ID
 MOVE POW-TRUE TO "Enabled" OF BT-ADD
 END-EVALUATE

```



このイベントのパラメタ(POW-COOKIE)には、OpenFormメソッドのパラメタで指定した、フォームの識別IDが渡ってきます。このパラメタは、2つ以上のフォームをOpenFormメソッドで開いた場合に、どのフォームが閉じられたかを判定するために使用します。このサンプルプログラムでは、1つのフォームを開いただけなので、とくにこのパラメタを判定する必要はありませんが、例として記述しています。

### 新しいフォームにコントロールを配置する

新しく作成したフォームを開き、以下のコントロールを配置します。

スタティックテキストコントロールを4つ(CmStatic1～CmStatic4)

テキストボックスコントロールを3つ(CmText1～CmText3)

コマンドボタンコントロールを2つ(CmCommand1～CmCommand2)

各コントロールのプロパティ設定ダイアログボックスを使って、以下のようなフォームを作成します。

各コントロールのプロパティ設定ダイアログボックスでの設定内容を、以下に示します。

#### CmStatic1のプロパティ設定

[ スタティックテキスト ] タブの [ キャプション ] を "追加する商品の情報を入力してください。" に変更します。

[ 共通 ] タブの [ 名前 ] を "ST-TITLE" に変更します。

#### CmStatic2のプロパティ設定

[ スタティックテキスト ] タブの [ キャプション ] を "商品コード:" に変更します。

[ 共通 ] タブの [ 名前 ] を "ST-CODE" に変更します。

#### CmStatic3のプロパティ設定

[ スタティックテキスト ] タブの [ キャプション ] を "商品名:" に変更

します。

[ 共通 ] タブの [ 名前 ] を "ST-NAME" に変更します。

#### **CmStatic4のプロパティ設定**

[ スタティックテキスト ] タブの [ キャプション ] を "単価:" に変更します。

[ 共通 ] タブの [ 名前 ] を "ST-PRICE" に変更します。

#### **CmText1のプロパティ設定**

[ テキストボックス ] タブの [ テキスト ] をクリアします。

[ テキスト属性 ] タブの [ テキスト種別 ] を "1 - COBOL PICTURE属性" に変更します。

[ PICTURE文字列 ] を "9(5)" に変更します。

[ 共通 ] タブの [ 名前 ] を "TX-CODE" に変更します。

#### **CmText2のプロパティ設定**

[ テキストボックス ] タブの [ テキスト ] をクリアします。

[ テキスト属性 ] タブの [ テキスト種別 ] を "1 - COBOL PICTURE属性" に変更します。

[ PICTURE文字列 ] を "N(10)" に変更します。

[ 共通 ] タブの [ 名前 ] を "TX-NAME" に変更します。

#### **CmText3のプロパティ設定**

[ テキストボックス ] タブの [ テキスト ] を "0" に変更します。

[ テキスト属性 ] タブの [ テキスト種別 ] を "1 - COBOL PICTURE属性" に変更します。

[ PICTURE文字列 ] を "¥¥¥,¥¥9" に変更します。

[ 共通 ] タブの [ 名前 ] を "TX-PRICE" に変更します。

#### **CmCommand1のプロパティ設定**

[ コマンドボタン ] タブの [ キャプション ] を "追加" に変更します。

[ 共通 ] タブの [ 名前 ] を "BT-ADD" に変更します。

#### **CmCommand2のプロパティ設定**

[ コマンドボタン ] タブの [ キャプション ] を "閉じる" に変更します。

[ 共通 ] タブの [ 名前 ] を "BT-EXIT" に変更します。

### **新しいフォームの手続きを記述する**

新しく作成したフォームに、以下の手続きを記述します。

使用するファイルの宣言 (FILE-CONTROL)

レコードの定義 (FILE)

作業用データの定義 (WORKING-STORAGE)

フォームが開かれたときの手続き (AddForm-Opened)

[ 追加 ] ボタンをクリックしたときの手続き (BT-ADD-Click)

[ 閉じる ] ボタンをクリックしたときの手続き (BT-EXIT-Click)

## FILE-CONTROL

```
* 商品コードをキーとして、商品名および単価をもちます。
SELECT 商品ファイル ASSIGN TO "..¥PRODUCTS.TBL"
 ORGANIZATION IS RELATIVE
 RELATIVE KEY IS 商品 - キー
 ACCESS MODE IS RANDOM
 FILE STATUS IS FS.
```

## FILE

```
FD 商品ファイル GLOBAL.
01 商品レコード.
 02 商品 - 名 PIC N(10).
 02 商品 - 単価 PIC 9(5).
```

## WORKING-STORAGE

```
01 商品 - キー PIC 9(5) BINARY GLOBAL.
01 FS PIC XX GLOBAL.
```

## AddForm-Opened

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-CODE PIC S9(8) COMP-5.
PROCEDURE DIVISION.

* 現在のレコード数を求め、次の商品コードを決定します。
OPEN INPUT 商品ファイル
MOVE 1 TO W-CODE
PERFORM WITH NO LIMIT
 MOVE W-CODE TO 商品 - キー
 READ 商品ファイル
 INVALID KEY
 PERFORM 最終レコード到達
 END-READ
 ADD 1 TO W-CODE
 END-PERFORM
.

最終レコード到達.
 MOVE W-CODE TO "Text" OF TX-CODE
 CLOSE 商品ファイル

* 最初の入力箇所にフォーカスを設定します。
 INVOKE TX-NAME "SetFocus"
 EXIT PROGRAM.
```



**BT-ADD-Click**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

* 入力内容をチェックします。
 IF "Text" OF TX-NAME = SPACE THEN
 INVOKE POW-SELF "DisplayMessage" USING
 "商品名を入力してください。"
 EXIT PROGRAM
 END-IF
 IF "Text" OF TX-PRICE = 0 THEN
 INVOKE POW-SELF "DisplayMessage" USING
 "単価を入力してください。"
 EXIT PROGRAM
 END-IF

* 商品ファイルをオープンします。
 OPEN I-O 商品ファイル

* 入力された商品の情報を商品ファイルに追加します。
 MOVE "Text" OF TX-CODE TO 商品 - キー
 MOVE "Text" OF TX-NAME TO 商品 - 名
 MOVE "Text" OF TX-PRICE TO 商品 - 単価
 WRITE 商品レコード

* 商品ファイルをクローズします。
 CLOSE 商品ファイル

* 商品コードの値を加算します。
 ADD 1 TO "Text" OF TX-CODE

* 入力領域をクリアします。
 MOVE SPACE TO "Text" OF TX-NAME
 MOVE 0 TO "Text" OF TX-PRICE

```

**BT-EXIT-Click**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 INVOKE POW-SELF "CloseForm"

```

**8.3.2 CallFormメソッドを使用する**

CallFormメソッドを使用しても、PowerCOBOLで作成した別のフォームを開くことができます。

**CallFormメソッドの書きかた**

CallFormメソッドは、以下のように記述します。

```
INVOKE POW-SELF "CallForm"
 USING "フォーム名"
 ["DLL名"]
 [フォーム識別ID]
RETURNING 復帰値
```

#### POW-SELF

これから子フォームを開こうとしている親フォーム自身を表しています。  
POW-SELFの代わりに、親フォーム自身のフォーム名を使うこともできます。  
フォーム名については、「[プロジェクト構成要素の命名規則](#) ( p93) 」を参照してください。

#### フォーム名

これから開こうとしている子フォームの名前を指定します。

#### DLL名

開こうとしている子フォームが別のDLL ( モジュール ) に含まれている場合、そのDLLファイル名を指定します。

#### フォーム識別ID

複数の子フォームを開き、それらのどれかが閉じられた場合に、親フォームのCloseChildイベントで閉じられた子フォームを識別するために使用するIDです。

#### 復帰値

子フォームが閉じられるときに、CloseFormメソッドで指定されたResultパラメタの値です。省略またはCloseFormメソッド以外の方法でクローズされた場合は、0が設定されます。それ以外の場合は、-1が設定されます。  
CloseFormメソッドについての詳細は、『リファレンス』を参照してください。

#### CallFormメソッドとOpenFormメソッドの違い

OpenFormメソッドでフォームを開いた場合と異なるのは、CallFormメソッドで開いたフォームは、ダイアログボックスのように、開かれたフォームが閉じられるまで、呼び出し元のフォームの操作ができなくなることです。

たとえば、前項のOpenFormの説明で使ったサンプルプログラムTable2.ppjでは、[ 商品の追加 ]ウィンドウをCallFormメソッドで開いた場合、[ 商品の追加 ]ウィンドウを閉じるまで、[ 購入商品の入力画面 ]ウィンドウの操作ができなくなります。

つまり、OpenFormメソッドを呼び出すと、フォームが開かれていても、続いて次の処理が実行されますが、CallFormメソッドでは、フォームが閉じられるまで、呼び出し元の処理は停止しています。

また、CallFormメソッドは、復帰値を通して、開いたフォームから情報を受け取ることができます。復帰値を利用することにより、たとえば、ダイアログボックス形式でフォームを開いた場合、OKボタンがクリックされたか、キャンセルボタンがクリックされたかを簡単に判定することができます。使用例は、サンプルプログラム"SimpleForm¥ReturnCheck.ppj"で、参照できます。CallFormメソッドについての詳細は、『リファレンス』を参照してください。

### 8.3.3 フォーム間での情報の受け渡し方法

OpenFormメソッドのサンプルプログラム(Table2.ppj)では、商品ファイルを使って、フォーム間で情報を共有しました。また、CallFormメソッドのサンプルプログラム(ReturnCheck.ppj)では、復帰値を使って、情報の受け渡しができます。

これら以外のフォーム間の共有情報を受け渡し方法について説明します。

#### COBOL85言語仕様の場合

モジュールのプロパティ設定ダイアログボックスの[スクリプト言語]が"0 - COBOL85言語仕様"である場合、EXTERNALデータを使って情報を受け渡すことができます。たとえば、10バイトの文字列(DATA1)を共有する場合には、両方のフォームの作業場所節に、以下のようなデータ宣言をして、データを受け渡します。

##### WORKING-STORAGE

```
01 DATA1 PIC X(10) GLOBAL EXTERNAL.
```

#### OOCOBOL言語仕様の場合

モジュールのプロパティ設定ダイアログボックスの[スクリプト言語]が"1 - OOCOBOL言語仕様"である場合、EXTERNALデータを使って情報を受け渡すことはできません。この場合、利用する子フォームをActiveXコントロールとして作成し、カスタムプロパティやカスタムメソッドを使って、情報を受け渡します。ActiveXコントロールの作成方法およびその利用方法については、「[ActiveXコントロールを作成する](#) ( p245) 」および「[ActiveXコントロールを使ったアプリケーションを作成する](#) ( p260) 」を参照してください。



注意

COBOL85言語仕様で作成されたモジュールと、OOCOBOL言語仕様で作成されたモジュールのあいだで、EXTERNALデータを使って情報を受け渡すことはできません。

## 8.4 コレクションオブジェクトを使ったアプリケーションを作成する

コントロールやオブジェクトの集合体を、コレクションオブジェクトといいます。

PowerCOBOLでは、Controlsコレクションオブジェクトを利用することにより、フォーム上に配置されたコントロールを操作することができます。

コントロールの集合体であるControlsコレクションオブジェクト、およびそれを構成するコントロールを操作するには、以下の2つの方法があります。

POWERGETCONTROLユーティリティを利用する方法

NetCOBOLのCOM連携機能を利用する方法

### 8.4.1 POWERGETCONTROLユーティリティを使って操作する

POWERGETCONTROLユーティリティを使って、Controlsコレクションオブジェクトを構成する任意のコントロールを取り出すことができます。

POWERGETCONTROLユーティリティでは、コントロールをNetCOBOLの\*COMクラスに対応するオブジェクトとして取り出すことができます。

#### POWERGETCONTROLの書きかた

```
CALL "POWERGETCONTROL"
 USING POW-SELF コントロールのインデックス
 RETURNING COMオブジェクト
```

#### POW-SELF

POW-SELFはフォーム自身を表しています。POW-SELFの代わりに、フォーム名を使うこともできます。

#### コントロールのインデックス

```
PIC S9(9) COMP - 5
```

Controlsコレクションオブジェクトを構成するコントロールのインデックスを指定します。インデックスは、フォーム上に配置されたすべてのコントロールに、1から描画順序（背面から前面方向に昇順）に従って振られた、一意な値です。

描画順序の設定方法については、「[コントロールの描画順序を変更する](#)（p105）」を参照してください。また、描画順序は、フォームのZOrderメソッドによって実行時に動的に変更することもできます。

#### COMオブジェクト

USAGE IS OBJECT REFERENCE COMのクラス名  
NetCOBOLで扱うことができる、COMオブジェクトを取り出すためのデータ項目

です。この変換ルーチンの出力となります。省略することはできません。COMのクラス名は、リポジトリ段落(REPOSITORY)でNetCOBOLの\*COMクラスとして宣言したクラス名です。

### POWERGETCONTROLの使用例

たとえば、フォーム上のすべてのテキストボックスコントロールに表示、または入力されたテキストをクリアする場合、以下のような手続きを作成します。

#### REPOSITORY

```
CLASS COM AS "*COM"
```

#### ClearButton-Click

```
ENVIRONMENT DIVISION.
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

\* フォーム上に配置されたコントロールの個数

```
01 CONTROLS-COUNT PIC S9(9) COMP-5.
```

\* フォーム上に配置されたコントロールのインデックス

```
01 CONTROL-INDEX PIC S9(9) COMP-5.
```

\* \*COMクラスを使ってコントロールにアクセスするための

\* COMオブジェクト

```
01 COM-CONTROL OBJECT REFERENCE COM.
```

```
01 WK-CLASSPROGID PIC X(80).
```

```
01 EMPTY-TEXT PIC X VALUE SPACE.
```

```
PROCEDURE DIVISION.
```

\* フォーム上のコントロールの個数を取得します。

```
MOVE "Count" OF "Controls" OF POW-SELF TO CONTROLS-COUNT
```

\* フォーム上のコントロールを順次取り出し、テキストボックス

\* コントロールだけを探索し、テキストをクリアします。

```
PERFORM VARYING CONTROL-INDEX FROM 1 BY 1
```

```
UNTIL CONTROL-INDEX > CONTROLS-COUNT
```

\* コントロールに対応するCOMオブジェクトを取り出します。

```
CALL "POWERGETCONTROL" USING POW-SELF CONTROL-INDEX
```

```
RETURNING COM-CONTROL
```

\* 取り出したコントロールのクラス名を求めます。

```
INVOKE COM-CONTROL "GET-CLASSPROGID" RETURNING WK-CLASSPROGID
```

\* テキストボックスコントロールかどうか、クラス名を使って判定し、

\* テキストボックスコントロールであれば、テキストをクリアします。

```
IF WK-CLASSPROGID = "Fujitsu.PcobTextBox.4" THEN
```

```
INVOKE COM-CONTROL "SET-TEXT" USING EMPTY-TEXT
```

```
END-IF
```

\* 取り出したコントロールのCOMオブジェクトを解放します。

```
SET COM-CONTROL TO NULL
```

```
END-PERFORM
```

コントロールの個数は、上記の例のように、Controlsコレクションオブジェク

トのCountプロパティを使って求めることができます。

ここで作成したサンプルプログラムは、"FormControls¥Controls1.ppj"に格納されています。必要に応じて参照してください。

PowerCOBOLが提供している各コントロールのクラス名については、『リファレンス』を参照してください。



ControlsコレクションオブジェクトのItemプロパティを使って、コントロールに対応するCOMオブジェクトを取り出すこともできます。以下にItemプロパティを使った2つの例を示します。

#### Itemプロパティの使用例1

フォーム上の、3番めに描画されるコントロールに対応するCOMオブジェクトは、以下のように取り出すことができます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
* コントロールを取り出すためのデータ
01 WK-CONTROL OBJECT REFERENCE POW-COBJECT.
* コントロールに対応するCOMオブジェクト
01 COM-CONTROL OBJECT REFERENCE COM.
PROCEDURE DIVISION.
* フォーム上の3番めのコントロールを取り出します。
 MOVE "Item"(3) OF "Controls" OF POW-SELF TO WK-CONTROL
* COMオブジェクトに変換します。
 CALL "POWERCONVTOCOM" USING WK-CONTROL RETURNING COM-CONTROL
```

#### Itemプロパティの使用例2

フォーム上の、CmText1という名前のコントロールに対応するCOMオブジェクトは、以下のように取り出すことができます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
* コントロールを取り出すためのデータ
01 CONTROL-NAME PIC X(10) VALUE "CmText1".
* コントロールを取り出すためのデータ
01 WK-CONTROL OBJECT REFERENCE POW-COBJECT.
* コントロールに対応するCOMオブジェクト
01 COM-CONTROL OBJECT REFERENCE COM.
PROCEDURE DIVISION.
* フォーム上のCmText1という名前のコントロールを取り出します。
 MOVE "Item"(CONTROL-NAME) OF "Controls" OF POW-SELF TO
 WK-CONTROL
* COMオブジェクトに変換します。
 CALL "POWERCONVTOCOM" USING WK-CONTROL RETURNING COM-CONTROL
```

### 8.4.2 NetCOBOLのCOM連携機能を使って操作する

POWERGETCONTROLユーティリティを利用せず、NetCOBOLのCOM連携機能だけを利用して、Controlsコレクションオブジェクトを構成する任意のコントロールを取り出し、操作することもできます。

以下に、POWERGETCONTROLユーティリティの説明で作成したサンプルと同様の処理をするプログラムのサンプルを示します。

#### REPOSITORY

```
CLASS COM AS "**COM"
```

#### ClearButton-Click

```
ENVIRONMENT DIVISION.
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

\* \*COMクラスを使ってフォームにアクセスするための

\* COMオブジェクト

```
01 COM-FORM OBJECT REFERENCE COM.
```

\* Controlsコレクションオブジェクト

```
01 CONTROLS-COLLECTION OBJECT REFERENCE COM.
```

\* \*COMクラスを使ってコントロールにアクセスするための

\* COMオブジェクト

```
01 COM-CONTROL OBJECT REFERENCE COM.
```

\* フォーム上に配置されたコントロールの個数

```
01 CONTROLS-COUNT PIC S9(9) COMP-5.
```

\* フォーム上に配置されたコントロールのインデックス

```
01 CONTROL-INDEX PIC S9(9) COMP-5.
```

```
01 WK-CLASSPROGID PIC X(80).
```

```
01 EMPTY-TEXT PIC X VALUE SPACE.
```

```
PROCEDURE DIVISION.
```

\* フォームに対応するCOMオブジェクトを取り出します。

```
CALL "POWERCONVTOCOM" USING POW-SELF RETURNING COM-FORM
```

\* フォームのControlsコレクションオブジェクトを取得します。

```
INVOKE COM-FORM "GET-CONTROLS" RETURNING CONTROLS-COLLECTION
```

\* Controlsコレクションオブジェクトからコントロールの

\* 個数を取得します。

```
INVOKE CONTROLS-COLLECTION "GET-COUNT" RETURNING CONTROLS-COUNT
```

\* フォーム上のコントロールを順次取り出し、テキストボックス

\* コントロールだけを検索し、テキストをクリアします。

```
PERFORM VARYING CONTROL-INDEX FROM 1 BY 1
```

```
UNTIL CONTROL-INDEX > CONTROLS-COUNT
```

\* コントロールに対応するCOMオブジェクトを取り出します。

( 続く )

( 続き )

```
INVOKE CONTROLS-COLLECTION "GET-ITEM" USING CONTROLS-COUNT
RETURNING COM-CONTROL
```

- \* 取り出したコントロールのクラス名を求めます。  
INVOKE COM-CONTROL "GET-CLASSPROGID" RETURNING WK-CLASSPROGID
- \* テキストボックスコントロールかどうか、クラス名を使って判定し、
- \* テキストボックスコントロールであれば、テキストをクリアします。  
IF WK-CLASSPROGID = "Fujitsu.PcobTextBox.4" THEN  
INVOKE COM-CONTROL "SET-TEXT" USING EMPTY-TEXT  
END-IF
- \* Controlsコレクションオブジェクトを解放します。  
SET CONTROLS-COLLECTION TO NULL  
END-PERFORM
- \* 取り出したコントロールのCOMオブジェクトを解放します。  
SET COM-CONTROL TO NULL
- \* フォームのCOMオブジェクトを解放します。  
SET COM-FORM TO NULL

ここで作成したサンプルプログラムのコレクションオブジェクトを操作する部分を、外部COBOLファイルとして作成し、COM連携時のエラーチェック機能を組み込んだサンプルプログラムが、"FormControls¥Controls2.ppj"に格納されています。必要に応じて参照してください。

NetCOBOLのCOM連携機能の詳細については、『NetCOBOL 使用手引書』を参照してください。



## 8.5 ポップアップメニューを使ったアプリケーションを作成する

本節では、ポップアップメニューを使ったアプリケーションの作成方法について説明します。

本節では、「[アプリケーションを作成しよう](#)」( p49) で作成したアプリケーションをもとに、メニューバーからだけでなく、ポップアップメニューによる操作もできるように変更します。

このアプリケーションは、以下の手順で作成します。

1. フォームに新しくメニューを作成します。
2. メニューバーと同様のメニュー項目を作成します。
3. マウスの右ボタンがクリックされたときの手続きを記述します。
4. ポップアップメニューのメニュー項目が選択されたときの手続きを記述します。



本節でこれから作成していくサンプルプログラムは、"Menu¥Table3.ppj"に格納されています。必要に応じて参照してください。

### 8.5.1 フォームに新しくメニューを作成する

以下の手順で、フォームに新しくメニューを作成します。

1. デザインツリーウィンドウ上でフォームを選択します。
2. ポップアップメニューの[メニュー編集]サブメニューから[メニュー作成]コマンドを選択します。
3. メニュー編集ウィンドウ上で、ポップアップメニューの[ポップアップメニュー表示]コマンドを選択します。
4. デザインツリーウィンドウ上で新しく作成されたメニュー"CfMenu2"を選択します。
5. ポップアップメニューの[名前の変更]コマンドを使って、メニュー名を"Popupmenu1"に変更します。

### 8.5.2 メニュー項目を追加する

以下の手順で、メニューバー(CfMenu1)のファイルメニュー(MN-FILE)と同じ構成のメニューを作成します。

1. メニューに"開く"を追加します。  
ポップアップメニューの[追加]コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの[キャプション]に"開く(&O)..."と入力します。

- [ 名前 ] を "PMN-OPEN" に変更します。  
OK ボタンをクリックします。
2. "開く" の下に、"上書き保存" を追加します。  
メニュー編集ウィンドウ上の "開く..." を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ キャプション ] に "上書き保存(&S)" と入力します。  
[ 名前 ] を "PMN-SAVE" に変更します。  
OK ボタンをクリックします。
3. 同様に、"名前を付けて保存" を追加します。  
メニュー編集ウィンドウ上の "上書き保存" を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ キャプション ] に "名前を付けて保存(&A)..." と入力します。  
[ 名前 ] を "PMN-SAVEAS" に変更します。  
OK ボタンをクリックします。
4. セパレータを挟んで、"ページ設定" を追加します。  
メニュー編集ウィンドウ上の "名前を付けて保存..." を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ セパレータ ] をチェック状態にします。  
OK ボタンをクリックします。  
作成したセパレータ( 横棒で表示されている部分 )を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ キャプション ] に "ページ設定(&T)..." と入力します。  
[ 名前 ] を "PMN-PAGESETUP" に変更します。  
OK ボタンをクリックします。
5. さらに、"印刷" を追加します。  
メニュー編集ウィンドウ上の "ページ設定..." を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ キャプション ] に "印刷(&P)" と入力します。  
[ 名前 ] を "PMN-PRINT" に変更します。  
OK ボタンをクリックします。
6. セパレータを挟んで、"終了" を追加します。  
メニュー編集ウィンドウ上の "印刷" を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。  
メニューのプロパティ設定ダイアログボックスの [ セパレータ ] をチェック状態にします。  
OK ボタンをクリックします。  
作成したセパレータ( 横棒で表示されている部分 )を選択します。  
ポップアップメニューの [ 追加 ] コマンドを選択します。

メニューのプロパティ設定ダイアログボックスの [ キャプション ] に "終了(&X)" と入力します。

[ 名前 ] を "PMN-EXIT" に変更します。

OKボタンをクリックします。

上記の作業の結果、以下のようなメニューが作成されます。



### 8.5.3 マウスの右ボタンがクリックされたときの手続きを記述する

マウスの右ボタンがクリックされたときの手続きは、MouseUpイベントに記述します。サンプルプログラムでは、フォーム上で右クリックした場合にポップアップメニューを表示します。ポップアップメニューを表示するための手続きは、以下のようになります。

#### MainForm-MouseUp

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-BUTTON PIC S9(4) COMP-5.
01 POW-SHIFT PIC S9(4) COMP-5.
01 POW-X PIC S9(9) COMP-5.
01 POW-Y PIC S9(9) COMP-5.
PROCEDURE DIVISION.
```

\* マウスの右ボタンの場合、ポップアップメニューを表示します。

```
IF POW-BUTTON = 2 THEN
 INVOKE POW-SELF "PopupMenu" USING "Popupmenu1"
END-IF
```

MouseUpイベントでは、クリックされたボタンの種類、同時に押されたキーおよびマウスポインタの位置が引数で渡ってきます。ボタンの種類を示す POW-BUTTON の値から、ポップアップメニューを表示するかどうかを判定します。ポップアップメニューは、PopupMenuメソッドを使って表示できます。



上記の手続きは、フォーム自身が右クリックされた場合に実行されます。したがって、表コントロールやコマンドボタンコントロールなど、コントロールが配置されている位置で右クリックしても、ポップアップメニューは表示されません。つまり、複数のポップアップメニューを用意しておけば、右クリックしたコントロールごとに、異なったポップアップメニューを表示できます。

複数のポップアップメニューを表示する方法は、サンプルプログラムの "Menu¥PopupMenu.ppj" で参照することができます。

#### 8.5.4 ポップアップメニューの項目が選択されたときの 手続きを記述する

ポップアップメニューの各メニュー項目が選択されたときの手続きは、メニューバーのファイルメニュー内のメニュー項目が選択された場合と同じ手続きを記述します。したがって、"PMN-OPEN-Click" は "MN-OPEN-Click" と、"PMN-SAVE-Click" は "MN-SAVE-Click" と同じ手続きを記述してください。



このサンプルプログラムでは、メニューバーのメニュー項目とポップアップメニューのメニュー項目を同じ内容にしていますが、必要に応じて自由に変更できます。

## 8.6 ステータスバーを使ったアプリケーションを作成する

ステータスバーとは、フォームの下部にあり、操作状態や選択項目などの説明文字列を表示する部分のことです。本節では、ステータスバーを使ったアプリケーションの作成方法について説明します。

本節では、「[アプリケーションを作成しよう](#) ( p49) 」で作成したアプリケーションをもとに、メニューバーからメニュー項目を選択する場合に、その項目の補足説明をステータスバーに表示するように変更します。

このアプリケーションは、以下の手順で作成します。

1. フォームにステータスバーを追加します。
2. メニュー項目選択時の手続きを記述します。
3. メニュー項目の選択が確定( クリック )したときの手続きを追加します。
4. メニュー項目の選択がキャンセルされたときの手続きを追加します。



本節で作成するサンプルプログラムは、"Statusbar¥Table4.ppj" に格納されています。必要に応じて参照してください。

### 8.6.1 フォームにステータスバーを追加する

以下の手順で、フォームにステータスバーを追加します。

1. デザインツリーウィンドウでフォームを選択します。
2. ポップアップメニューの [ プロパティ ] コマンドを選択します。
3. プロパティ設定ダイアログボックスの [ ステータスバー ] タブを開きます。
4. [ ステータスバーの表示 ] をチェック状態にします。
5. OKボタンをクリックします。



## 8.6.2 メニュー項目選択時の手続きを記述する

ステータスバーの文字列は、メニューバーのメニュー項目が選択状態（反転表示）のときに、表示されるようにします。メニュー項目が選択状態になったときの手続きは、各メニュー項目がもつ、Selectイベントに記述します。たとえば、[ 編集 ]メニューの[ 合計 ]コマンド(MN-TOTAL)が選択状態の場合、以下のように手続きを記述します。

### MN-TOTAL-Select

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

 MOVE "商品の合計金額を計算します。" TO "StatusText" OF POW-SELF
[ファイル]メニューの他のメニュー項目に対応するSelectイベントにも、同様の手続きを記述します。設定する文字列には、メニュー項目の意味を示す簡単な説明を入れてください。
```

## 8.6.3 メニュー項目の選択が確定（クリック）したときの手続きを追加する

メニュー項目の選択が確定したら、ステータスバーの文字列をクリアします。クリアしない場合、設定した文字列がそのままステータスバー上に残ってしまいます。ステータスバーの文字列は、各メニュー項目のClickイベントでクリアします。

たとえば、[ 編集 ]メニューの[ 合計 ]コマンド(MN-TOTAL)の選択が確定した場合、Clickイベントの先頭に、以下のように手続きを記述します。

### MN-TOTAL-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

* ステータスバーの文字列をクリアします。
 MOVE SPACE TO "StatusText" OF POW-SELF
 CALL "GET-TOTAL"
[ファイル]メニューの他のメニュー項目に対応するClickイベントの先頭にも、同様の手続きを記述します。
```

#### 8.6.4 メニュー項目の選択がキャンセルされたときの 手続きを記述する

メニュー項目の選択がキャンセルされた場合(何も選択しないでメニューを閉じた場合)や、メニューバーのトップレベルメニュー([ファイル]メニューや[編集]メニュー)が選択状態になった場合、ステータスバーの文字列をクリアします。クリアしない場合、設定した文字列がそのままステータスバー上に残ってしまいます。選択がキャンセルされたときの手続きは、メニューのSelectCloseイベントに、トップレベルメニューが選択状態になったときの手続きは、各トップレベルメニューのSelectイベントに記述します。メニューのSelectCloseイベントの手続きは、以下のように記述します。

##### CfMenu1-SelectClose

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
```

\* ステータスバーの文字列をクリアします。

```
MOVE SPACE TO "StatusText" OF POW-SELF
```

同様の手続きを、MN-EDIT-SelectイベントおよびMN-FILE-Selectイベントにも記述します。

## 8.7 ツールバーを使ったアプリケーションを作成する

ツールバーとは、メニューバーやポップアップメニュー中のメニュー項目に対応するボタンなどを配置し、メニュー項目を選択することなく、いろいろな操作をできるようにするものです。通常、ツールバーはメニューバーの下に配置されます。本節では、ツールバーを使ったアプリケーションの作成方法について説明します。

本節では、「[アプリケーションを作成しよう](#)」( p49) で作成したアプリケーションをもとに、メニューバーからだけでなく、ツールバー上のボタンによる操作もできるように変更します。

このアプリケーションは、以下の手順で作成します。

1. ツールバーのボタン上に表示するイメージを作成します。
2. フォームにツールバーコントロールを配置します。
3. ツールバーコントロールにボタンを追加します。
4. ツールバーのボタンがクリックされたときの手続きを記述します。



本節で作成するサンプルプログラムは、"ToolBar¥Table5.ppj" に格納されています。必要に応じて参照してください。

### 8.7.1 ツールバーのボタン上に表示するイメージを作成する

ツールバーのボタン上に表示するイメージは、イメージリストを使用します。イメージリストは、複数のイメージを等しい間隔で横に並べたビットマップです。Windows では、1つのイメージの幅が16ピクセル、高さが15ピクセルになるよう推奨されています。このサンプルプログラムでは、3つのボタンを作成しますので、幅が48ピクセルのビットマップを作成します。以下の手順で、イメージリストを作成し、モジュールに登録します。

1. Windows のアクセサリ内にある「ペイント」を起動します。
2. [ 変形 ]メニューの[ キャンパスの色とサイズ ]コマンドを選択します。
3. [ キャンパスの色とサイズ ]ダイアログボックスの[ 幅 ]に48、[ 高さ ]に15を設定し、[ 単位 ]を[ ピクセル ]に変更します。
4. OKボタンをクリックします。
5. 開く、保存および印刷に対応するイメージを16ピクセル幅で横に並べて作成します。  
作成例として、Table5.ppj と同じフォルダにあるToolButton.bmpを参照してください。



6. ビットマップを保存します。  
ここでは、ToolButton.bmpというファイル名で保存します。
7. プロジェクトウィンドウのデザインツリーウィンドウでモジュールを選択します。
8. ポップアップメニューの[ ファイルの挿入... ]コマンドを選択します。
9. [ ファイルの挿入 ]ダイアログボックスで、[ ファイルの種類 ]を"イメージリスト(\*.bmp)"に変更します。
10. 先ほど保存したToolButton.bmpを選択します。
11. [ 開く ] ボタンをクリックします。
12. デザインツリーウィンドウ上で追加したイメージリスト名を設定します。  
サンプルプログラムでは、そのままの名前(ImageList1)を使用します。



イメージリストのプロパティ設定ダイアログボックスを参照すると、イメージ幅が16になっていることが確認できます。PowerCOBOLでは、イメージリストのイメージ幅の初期値として16を設定しています。個々のイメージを16ピクセル以外の幅で作成した場合は、このダイアログボックスを使ってイメージ幅を変更してください。



モジュールのイメージリストを追加する場合には、ファイルの種類として"イメージリスト(\*.bmp)"を選択してください。イメージリストのファイル拡張子は"bmp"ですが、"ビットマップファイル(\*.bmp)"として追加すると、イメージリストとして使用できません。

### 8.7.2 フォームにツールバーを配置する

以下の手順で、フォームにツールバーを配置します。

1. フォーム編集ウィンドウを開きます。
2. フォームの上端にツールバーを作成するので、フォームの高さを少し広げます。
3. フォーム上のコントロールすべてをまとめて選択します。コントロールをまとめて選択する方法は、「[コントロールをまとめて編集する \( p105\)](#)」を参照してください。

4. まとめて選択したコントロール全体を下方に少し移動します。
5. ツールボックスから"ツールバーコントロール"を選択し、フォーム上に配置します。



上記のような作業を省くため、ツールバーを使用するフォームを設計する場合は、最初にツールバーコントロールを配置しておくことをお勧めします。

### 8.7.3 ツールバーにボタンを追加する

以下の手順で、ツールバーにボタンを追加していきます。

1. ツールバーコントロールを選択し、プロパティ設定ダイアログボックスを開きます。
2. [ ツールバー ] タブの[ イメージリスト ] から、"ImageList1"を選択します。
3. [ ボタン ] タブを選択します。
4. [ 後ろに挿入 ] ボタンをクリックします。
5. [ イメージインデックス ] に1を設定します。
6. [ 後ろに挿入 ] ボタンをクリックします。
7. [ イメージインデックス ] に2を設定します。
8. [ 後ろに挿入 ] ボタンをクリックします。
9. [ イメージインデックス ] に3を設定します。
10. OKボタンをクリックします。

上記の作業の結果、以下のようなフォームが作成されます。

|    | コード   | 商品名        | 単価    | 個数    | 小計        | 備考                       |
|----|-------|------------|-------|-------|-----------|--------------------------|
| 1  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 2  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 3  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 4  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 5  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 6  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 7  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 8  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 9  | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |
| 10 | 00000 | NNNNNNNNNN | ¥¥.¥¥ | 2,229 | ¥¥,¥¥¥.¥¥ | NNNNNNNNNNNNNNNNNNNNNNNN |



[ ツールバー ] タブの [ イメージリスト ] には、モジュールに追加されているイメージリストに対応するリソース名の一覧が表示されます。  
 [ ボタン ] タブでボタンのプロパティを設定するとき、[ ツールチップ ] にボタンの説明文字列を設定しておく、実行時にマウスポインタをボタン上に移動したとき、各ボタンに対応する説明をツールチップとして表示することができます。

#### 8.7.4 ツールバーのボタンがクリックされたときの続きを記述する

ツールバーのボタンがクリックされたときの手続きは、ツールバーコントロールのButtonClickイベントに記述します。このサンプルプログラムでは、ボタンのインデックスが1から順に、メニュー項目の "MN-OPEN-Click"、"MN-SAVE-Click" および "MN-PRINT-Click" の手続きと同様の手続きを実行します。ButtonClick イベントでは、クリックされたボタンのインデックスが引数に設定されているため、その値を判別して処理を切り分けます。

##### MainForm-MouseUp

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 GET-STYLE PIC S9(4) COMP-5.
01 BUTTON-STATE PIC S9(4) COMP-5.
LINKAGE SECTION.
01 POW-BUTTONINDEX PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW-BUTTONINDEX.
 EVALUATE POW-BUTTONINDEX
 WHEN 1
 * ツールバーの開くボタンがクリックされた場合です。
 MOVE POW-CDOPEN TO GET-STYLE
 CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE
 IF BUTTON-STATE = POW-FALSE THEN
 EXIT PROGRAM
 END-IF
 CALL "LOAD-DATA"
 WHEN 2
 * ツールバーの保存ボタンがクリックされた場合です。
 IF FILE-NAME = SPACE THEN
 MOVE POW-CDSAVE TO GET-STYLE

```

( 続く )

( 続き )

```
CALL "GET-FILE-NAME" USING GET-STYLE BUTTON-STATE
IF BUTTON-STATE = POW-FALSE THEN
 EXIT PROGRAM
END-IF
END-IF
CALL "SAVE-DATA"
WHEN 3
* ツールバーの印刷ボタンがクリックされた場合です。
 INVOKE CmPrint1 "PrintForm"
END-EVALUATE
```



---

ツールバーコントロールには、テキストボックスコントロールやコンボボックスコントロールなど他のコントロールを配置して利用することもできます。コンボボックスコントロールを使用した例は、サンプルプログラム "Toolbar¥Toolbar.ppj" を参照してください。

---

## 8.8 タブコントロールを使ってアプリケーションを作成する

タブコントロールは、タブ（通常、ダイアログボックスの上部にあるタブ）をクリックし、ページを切り替えることで、ウィンドウに多くの項目がある場合でも、それらを分類して表示したり、設定したりできるようにするコントロールです。

本節では、「[アプリケーションを作成しよう](#)（ p49）」で作成したアプリケーションをもとに、「[複数ウィンドウをもつアプリケーションを作成する](#)（ p204）」で作成したアプリケーションと同様の機能をもつアプリケーションを、タブコントロールを使って、作成します。異なる点は、子フォームをモードレスなウィンドウとして呼び出す方法から、モーダルなウィンドウ（ダイアログボックス）として呼び出す方法に変更していることです。

このアプリケーションは、以下の手順で作成します。

1. モジュールにダイアログボックスにする新しいフォームを追加し、名前を設定します。
2. 追加したダイアログボックスを開くためのボタンを、呼び出し元のフォームに追加します。
3. 呼び出し元のフォームに、ダイアログボックスを開くための手続きを記述します。
4. タブコントロールを新しいフォームに配置し、プロパティなどを設定します。
5. その他のコントロールを適切な位置に配置し、プロパティなどを設定します。
6. 新しいフォームの手続きを記述します。

最初の2つの手順は、「[複数ウィンドウをもつアプリケーションを作成する](#)（ p204）」の「モジュールに新しいフォームを追加する」および「呼び出し元のフォームを編集する」を参照してください。操作方法は、すべて同じです。



本節で作成するサンプルプログラムは、"Tab¥Table6.ppj"に格納されています。必要に応じて参照してください。

### 8.8.1 ダイアログボックスを開くための手続きを記述する

新しく作成したフォームを、ダイアログボックスとして開くために、以下の手続きを追加します。

[ 商品の追加 ] ボタンをクリックしたときの手続き (BT-ADD-Click)



ダイアログボックスとして開くので、「開くフォームを識別するためのIDの定義(WORKING-STORAGE)」および「開いたフォームが閉じられたときの手続き(MainForm-CloseChild)」を記述する必要はありません。

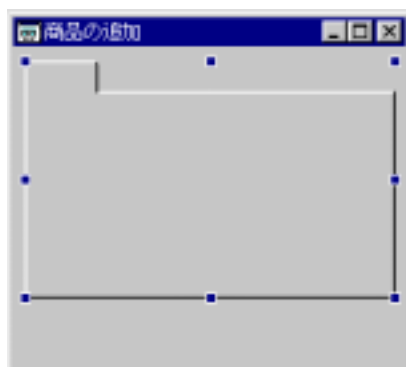
#### BT-ADD-Click

[ 商品の追加 ] ウィンドウを、ダイアログボックスとして開きます。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* 商品追加用のダイアログボックスを開きます。
 INVOKE POW-SELF "CallForm" USING "AddForm"
```

### 8.8.2 新しいフォームにタブコントロールを配置する

新しく作成したフォームを開き、タブコントロールを配置します。このとき、タブコントロールの大きさとフォームの大きさを、以下の図のようになるよう、調節してください。



次に、以下の手順でタブコントロールのプロパティを設定します。この設定により、商品名を入力するページおよび単価を入力するページを作成します。

1. タブコントロールのプロパティ設定ダイアログボックスを開きます。
2. [ ページ数 ] を2にします。
3. [ ページ ] タブの [ カレントページ ] を1にします。
4. [ ページキャプション ] に"商品コード"と入力します。
5. [ ツールチップ文字列 ] に"商品コードの設定"と入力します。
6. [ カレントページ ] を2にします。
7. [ ページキャプション ] に"単価"と入力します。
8. [ ツールチップ文字列 ] に"単価の設定"と入力します。
9. OKボタンをクリックします。



[ ページキャプション ] は、タブコントロールのタブ部分に表示される文字列です。[ ツールチップ文字列 ] は、実行時にマウスポインタをタブ部分に移動したときに表示される説明文字列です。

### 8.8.3 その他のコントロールを配置する

まず、タブコントロールの下に、コマンドボタンコントロールを2つ (CmCommand1 ~ CmCommand2) 配置します。

次に、タブコントロールの [ 商品コード ] ページに、以下のコントロールを配置します。

スタティックテキストコントロールを1つ (CmStatic1)

テキストボックスコントロールを1つ (CmText1)

さらに、タブコントロールの [ 単価 ] ページに、以下のコントロールを配置します。

スタティックテキストコントロールを1つ (CmStatic2)

テキストボックスコントロールを1つ (CmText2)

各コントロールのプロパティ設定ダイアログボックスを使って、以下のようなフォームを作成します。



各コントロールのプロパティ設定ダイアログボックスでの設定内容を、以下に示します。

#### CmStatic1のプロパティ設定

[ 共通 ] タブの [ 名前 ] を "ST-NAME" に変更します。

#### CmText1のプロパティ設定

[ テキストボックス ] タブの [ テキスト ] をクリアします。

[ テキスト属性 ] タブの [ テキスト種別 ] を "1 - COBOL PICTURE属性" に変更します。

[ PICTURE文字列 ] を "N(10)" に変更します。

[ 共通 ] タブの [ 名前 ] を "TX-NAME" に変更します。

#### CmStatic2のプロパティ設定

[ 共通 ] タブの [ 名前 ] を "ST-PRICE" に変更します。

#### CmText2のプロパティ設定

[ テキストボックス ] タブの [ テキスト ] を "0" に変更します。

[ テキスト属性 ] タブの [ テキスト種別 ] を "1 - COBOL PICTURE属性" に変更します。

[ PICTURE文字列 ] を "¥¥¥,¥¥9" に変更します。

[ 共通 ] タブの [ 名前 ] を "TX-PRICE" に変更します。

#### CmCommand1のプロパティ設定

[ コマンドボタン ] タブの [ キャプション ] を "OK" に変更します。

[ 共通 ] タブの [ 名前 ] を "BT-OK" に変更します。

[ ボタンスタイル ] の [ デフォルトボタン ] をチェック状態にします。

#### CmCommand2のプロパティ設定

[ コマンドボタン ] タブの [ キャプション ] を "キャンセル" に変更します。

[ 共通 ] タブの [ 名前 ] を "BT-CANCEL" に変更します。

[ ボタンスタイル ] の [ キャンセルボタン ] をチェック状態にします。

### 8.8.4 新しいフォームの手続きを記述する

新しく作成したフォームに、以下の手続きを記述します。

使用するファイルの宣言 (FILE-CONTROL)

レコードの定義 (FILE)

作業用データの定義 (WORKING-STORAGE)

フォームが開かれたときの手続き (AddForm-Opened)

OKボタンをクリックしたときの手続き (BT-OK-Click)

キャンセルボタンをクリックしたときの手続き (BT-CANCEL-Click)

FILE-CONTROLおよびFILEの手続きは、「[複数ウィンドウをもつアプリケーションを作成する](#) ( p204 )」で記述した内容と同じです。



**WORKING-STORAGE**

```
01 商品 - キー PIC 9(5) BINARY GLOBAL.
01 FS PIC XX GLOBAL.
01 G-CODE PIC S9(8) COMP-5 GLOBAL.
```

**AddForm-Opened**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-LABEL.
02 FILLER PIC N(5) VALUE NC"商品コード".
02 FILLER PIC X(1) VALUE SPACE.
02 W-CODE PIC 9(5).
02 FILLER PIC X(1) VALUE SPACE.
02 FILLER PIC N(5) VALUE NC"に対応する".
02 FILLER PIC X(1) VALUE SPACE.
02 W-TARGET PIC N(3).
02 FILLER PIC X(1) VALUE SPACE.
02 FILLER PIC N(9) VALUE NC"を入力してください".
PROCEDURE DIVISION.
* 現在のレコード数を求め、次の商品コードを決定します。
OPEN INPUT 商品ファイル
MOVE 1 TO W-CODE
PERFORM WITH NO LIMIT
MOVE W-CODE TO 商品 - キー
READ 商品ファイル
INVALID KEY
PERFORM 最終レコード到達
END-READ
ADD 1 TO W-CODE
END-PERFORM
.
最終レコード到達.
CLOSE 商品ファイル
* ページに表示する文字列を設定します。
MOVE G-CODE TO W-CODE
MOVE NC"商品名" TO W-TARGET
MOVE W-LABEL TO "Caption" OF ST-NAME
MOVE NC"単価" TO W-TARGET
MOVE W-LABEL TO "Caption" OF ST-PRICE
* 最初の入力箇所にフォーカスを設定します。
INVOKE TX-NAME "SetFocus"
EXIT PROGRAM.
```

#### BT-OK-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

* 入力内容をチェックします。
 IF "Text" OF TX-NAME = SPACE THEN
 INVOKE POW-SELF "DisplayMessage" USING
 "商品名を入力してください。"
 EXIT PROGRAM
 END-IF
 IF "Text" OF TX-PRICE = 0 THEN
 INVOKE POW-SELF "DisplayMessage" USING
 "単価を入力してください。"
 EXIT PROGRAM
 END-IF

* 商品ファイルをオープンします。
 OPEN I-O 商品ファイル

* 入力された商品の情報を商品ファイルに追加します。
 MOVE G-CODE TO 商品 - キー
 MOVE "Text" OF TX-NAME TO 商品 - 名
 MOVE "Text" OF TX-PRICE TO 商品 - 単価
 WRITE 商品レコード

* 商品ファイルをクローズします。
 CLOSE 商品ファイル

* ダイアログボックスを閉じます。
 INVOKE POW-SELF "CloseForm"
```

#### BT-CANCEL-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.

* 何もしないでダイアログボックスを閉じます。
 INVOKE POW-SELF "CloseForm"
```

---

## 第9章 他のアプリケーションとの連携

---

本章では、PowerCOBOL以外のツールとの連携方法について、サンプルプログラムを使用して説明します。

---

## 9.1 DLLを使ったアプリケーションを作成する

本節では、DLLを使ったアプリケーションの作成方法について説明します。

### 9.1.1 PowerCOBOLで作成したDLLを使用する

PowerCOBOLで作成したDLLは、以下の4つの方法で利用することができます。

OpenFormメソッドを使用する方法

CallFormメソッドを使用する方法

ActiveXコントロールとして利用する方法

オートメーションサーバにして利用する方法

最初の2つの方法については、「[複数ウィンドウをもつアプリケーションを作成する](#) ( p204) 」で説明した方法を使い、各メソッドの第2引数にDLL名を指定することで利用できます。

後者の2つの方法については、「[ActiveXコントロールを使ったアプリケーションを作成する](#) ( p260) 」、および「[オートメーションサーバを使ったアプリケーションを作成する](#) ( p270) 」を参照してください。

### 9.1.2 COBOLで作成したDLLを使用する

COBOLで作成したDLLは、以下のどちらかの方法で利用することができます。

インポートライブラリ（動的リンク構造）による呼び出し方法

"CALL 一意名"（動的プログラム構造）による呼び出し方法

#### インポートライブラリによる呼び出し

この方法は、COBOLのDLLに対応するインポート用のライブラリファイルを、呼び出し元となるモジュールの外部ファイルとして登録しておく方法です。

モジュールへの外部ファイルの登録方法については、「[外部ファイルを使う](#) ( p96) 」を参照してください。

この方法では、以下のように、CALL文を使ってDLL中のサブプログラムを直接呼び出します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
...
CALL "COBOLSUB"
```



実行時には、呼び出すサブプログラムが含まれるDLLファイルが必要です。DLLファイルは、呼び出し元ファイルと同じフォルダまたはPATH環境変数で指定されているフォルダに格納してください。

インポートライブラリによる呼び出しを使用する場合には、処理がそのモジュール内で完結するようにしてください。処理が複数のモジュールに分散していて、複数のDLLから呼び出されるDLLを使用する場合は、次に説明する、「"CALL 一意名"による呼び出し」を使用してください。

### "CALL 一意名"による呼び出し

この方法は、インポートライブラリを使用せず、アプリケーションの実行時にDLLを動的にロードしてサブプログラムを呼び出す方法です。

この方法では、アプリケーションを実行する前に、DLLを動的にロードするためのエントリ情報を、NetCOBOLの実行環境情報に設定しておく必要があります。エントリ情報の設定方法については、『NetCOBOL 使用手引書』を参照してください。

この方法では、以下のように、CALL文を使ってエントリ情報に設定するエントリ名を呼び出します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 エントリ名 PIC X(16) VALUE "COBOLSUB".
PROCEDURE DIVISION.

...
CALL エントリ名
```



この呼び出しでは、上記のように、CALL文で変数を呼び出します。これは、定数を呼び出すCALL文(CALL "COBOLSUB")とは異なり、実行時に呼び出し先が確定する呼び出し方法です。"CALL 一意名"を使った呼び出し方法の詳細については、『NetCOBOL 使用手引書』を参照してください。

## 9.1.3 PowerCOBOLで作成したDLLをCOBOLから使用する

COBOLで作成したプログラムから、PowerCOBOLで作成したDLLを利用するには、以下の2つの方法があります。

POWEROPENFORMユーティリティを使用する方法

PowerCOBOLのフォームをオートメーションサーバとして利用する方法

### POWEROPENFORMユーティリティを使用する

PowerCOBOLでは、COBOLのプログラムからPowerCOBOLで作成したフォームを開

くために、以下の3つのサブルーチンを用意しています。

POWEROPENFORM  
POWEROPENFORMUX  
POWEROPENFORMUN

POWEROPENFORMは、ランタイムコードセットがシフトJISコードで作成されたフォームを開く場合に使用します。また、POWEROPENFORMUXとPOWEROPENFORMUNは、ランタイムコードセットがUnicodeで作成されたフォームを開く場合に使用します。ランタイムコードセットの設定方法については、「[Unicodeを利用する](#) ( p99) 」を参照してください。

POWEROPENFORMUXとPOWEROPENFORMUNは、フォームを含むDLL（モジュール）の格納フォルダ名が日本語を含むかどうかによって、使い分けます。

これらのサブルーチンを総称して、POWEROPENFORMユーティリティと呼びます。このユーティリティは、PowerCOBOLのインストールフォルダにある"F5DDFCB4.DLL"がもっています。したがって、POWEROPENFORMユーティリティを利用するには、このDLLに対応するインポートライブラリ"F5DDFCB4.LIB"を、呼び出し元となるCOBOLのモジュールにインポート結合しておく必要があります。

POWEROPENFORMユーティリティを使ってフォームを開くと、フォームが閉じられるまでCOBOLの実行を中断することができます。つまり、PowerCOBOL上でCallFormメソッドを使ってフォームを開いた場合と同様の動作になります。



注意

実行時には、開こうとするフォームが含まれるPowerCOBOLで作成したDLLファイルが必要です。POWEROPENFORMユーティリティの引数で指定したDLL名が、フルパスで指定されていない場合、以下の順序でDLLファイルを検索します。

- 1) COBOLで作成した呼び出し元の実行可能プログラムが起動またはロードされたフォルダ
- 2) カレントフォルダ
- 3) Windows のシステムフォルダ
- 4) Windows がインストールされているフォルダ
- 5) PATH環境変数で指定されているフォルダ

POWEROPENFORMユーティリティは、COBOLのソースプログラムを翻訳およびリンクして作成したアプリケーションから、PowerCOBOLで作成したフォームを開く場合に使用してください。PowerCOBOLで開発するアプリケーションの手続きからフォームを開く場合は、POWEROPENFORMは使用できません。CallFormメソッドおよびDoModalメソッドを使用してください。

また、COBOL以外の言語で作成したプログラムから呼び出すこともできません。

---

## POWEROPENFORM

POWEROPENFORMは、ランタイムコードセットがシフトJISコードで作成されたフ

---

フォームを開く場合に使用します。

POWEROPENFORMを呼び出すCOBOLプログラムは、RCS(SJIS)オプションを指定して翻訳する必要があります。

#### 書きかた

```
CALL "POWEROPENFORM"
 USING DLL名
 フォーム名
```

#### DLL名

```
PIC X(260).
```

開こうとしているフォームをもつDLL名前です。領域の大きさは、260でなければなりません。

設定できる文字列は、英数字で260文字以下、日本語で130文字以下です。

#### フォーム名

```
PIC X(14).
```

開こうとしている子フォームの名前です。領域の大きさは、14でなければなりません。

設定できる文字列は、英数字で14文字以下、日本語で7文字以下です。

#### 復帰値

正常に終了した場合は0が設定されます。エラーが発生した場合には0以外の値が設定されます。

#### 使用例

たとえば、PowerCOBOLで作成したSubA.DLL内のSubFormという名前のフォームを開く場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DLL名 PIC X(260) VALUE "SubA.DLL".
01 フォーム名 PIC X(14) VALUE "SubForm".
PROCEDURE DIVISION.
...
CALL "POWEROPENFORM" USING DLL名 フォーム名
```

#### POWEROPENFORMUX

POWEROPENFORMUXは、ランタイムコードセットがUnicodeで作成されたフォームを開く場合に使用します。

また、フォームを含むDLLの格納フォルダ名が日本語を含まない場合に使用します。POWEROPENFORMUXを呼び出すCOBOLプログラムは、RCS(UCS2)オプションを指定して翻訳する必要があります。

#### 書きかた

```
CALL "POWEROPENFORMUX"
 USING DLL名
 フォーム名
```

### D L L 名

P I C X ( 3 9 0 ).

開こうとしているフォームをもつDLLの名前です。領域の大きさは、390でなければなりません。

設定できる文字列は、英数字で260文字以下です。

### フォーム名

P I C X ( 2 1 ).

開こうとしている子フォームの名前です。領域の大きさは、21でなければなりません。

設定できる文字列は、英数字で14文字以下です。

### 復帰値

正常に終了した場合は0が設定されます。エラーが発生した場合には0以外の値が設定されます。

### 使用例

たとえば、PowerCOBOLで作成したSubU.DLL内のSubFormという名前のフォームを開く場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 D L L 名 PIC X(390) VALUE "SubU.DLL".
01 フォーム名 PIC X(21) VALUE "SubForm".
PROCEDURE DIVISION.
...
CALL "POWEROPENFORMUX" USING D L L 名 フォーム名
```

### POWEROPENFORMUN

POWEROPENFORMUNは、ランタイムコードセットがUnicodeで作成されたフォームを開く場合に使用します。

また、フォームを含むDLLの格納フォルダ名が日本語を含む場合に使用します。POWEROPENFORMUNを呼び出すCOBOLプログラムは、RCS(UCS2)オプションを指定して翻訳する必要があります。

### 書きかた

```
CALL " POWEROPENFORMUN "
 USING D L L 名
 フォーム名
```

### D L L 名

P I C N ( 2 6 0 ).

開こうとしているフォームをもつDLLの名前です。領域の大きさは、260でなければなりません。

設定できる文字列は、英数字で260文字以下、日本語で130文字以下です。

### フォーム名

P I C N ( 1 4 ).

開こうとしている子フォームの名前です。領域の大きさは、14でなければなり



ません。

設定できる文字列は、英数字で14文字以下、日本語で7文字以下です。

### 復帰値

正常に終了した場合は0が設定されます。エラーが発生した場合には0以外の値が設定されます。

### 使用例

たとえば、PowerCOBOLで作成したSubU.DLL内のSubFormという名前のフォームを開く場合、以下のように記述します。

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 D L L 名 PIC N(260) VALUE "SubU.DLL".
01 フォーム名 PIC X(14) VALUE "SubForm".
PROCEDURE DIVISION.
...
CALL "POWEROPENFORMUN" USING D L L 名 フォーム名
```

### オートメーションサーバとして利用する

NetCOBOLのCOM連携機能(\*COMクラス)を利用することにより、PowerCOBOLのフォームをオートメーションサーバとして利用することができます。オートメーションサーバの利用方法については、『NetCOBOL 使用手引書』を参照してください。また、フォームをオートメーションサーバとして作成する方法については、「[オートメーションサーバを使ったアプリケーションを作成する](#) ( p270)」を参照してください。

## 9.1.4 COBOL以外の言語で作成したプログラムから使用する

PowerCOBOLで作成したDLLを、C++言語やBasicなど、COBOL以外の言語で作成するプログラムから利用するには、フォームをActiveXコントロールとして作成しておく必要があります。ActiveXコントロールとして作成しておくことにより、他言語で作成するプログラムから、ActiveXコントロールのメソッドを呼び出したり、プロパティにアクセスしたりできます。

PowerCOBOLでActiveXコントロールを作成する方法は、「[ActiveXコントロールを作成する](#) ( p245)」を参照してください。他言語のプログラムでActiveXコントロールを利用する方法については、各言語のマニュアルを参照してください。



注意

POWEROPENFORMユーティリティは、COBOL以外の言語から、直接呼び出すことはできません。使用する場合は、COBOLで作成したDLLを介して呼び出してください。

### 9.1.5 PowerCOBOLとCOBOLで作成したDLLが混在する場合

PowerCOBOLで作成したDLLとCOBOLで作成したDLLが複雑に混在している場合、以下のような注意点があります。

COBOL85言語仕様の手続きでは、再帰的な呼び出しを禁止しています。COBOLまたはPowerCOBOLの手続き中で、再帰的に呼び出されるような手続きは記述しないようにしてください。

PowerCOBOLの手続きからCOBOLのプログラムを呼び出すと、PowerCOBOLのランタイムシステムは、その先のNetCOBOLのランタイムシステムを制御することができません。したがって、PowerCOBOLから呼び出されたCOBOLのプログラム内で、PowerCOBOLのメソッドを呼び出したリプロパティへアクセスしたりすることはできません。

PowerCOBOLのDLLがCOBOL85言語仕様で作成されている場合、すでに開かれているフォームと同一のフォームを開くことはできません。

フォームは、処理中のイベント手続きがすべて終了するまで、アンロード可能な状態になりません。たとえば、フォームを閉じたとしても、そのフォームのイベント手続きによって開かれた子フォームが閉じられていなければ、親フォームはWindows のメモリ上にロードされています。子フォームをもつアプリケーションを実行する場合は、子フォームを先に閉じるようにしてください。

オートメーションサーバとして作成されたフォームをNetCOBOLのCOM連携機能(\*COMクラス)を使用して呼び出す場合、フォームのActivateメソッドを使用することはできません。この場合、DoModalメソッドを使用してください。

### 9.1.6 PowerCOBOLで作成したDLLの寿命

PowerCOBOLで作成したDLL中のフォームを開いた場合、そのDLLは動的にロードされ、開かれたフォームがすべて閉じられたときに、アンロード可能状態となります。アンロードされてしまった場合、そのDLL内の変数に設定されているデータは失われます。

## 9.2 ActiveXコントロールを作成する

本節では、サンプルプログラムを作成しながら、ActiveXコントロールを作成する方法について説明していきます。作成するActiveXコントロールは、以下の機能を持ちます。

赤、緑、青の3色から1種類の色を選択できます。

選択されている色を示す、SelectedColorプロパティを持ちます。

選択されている色を変更する場合、SelectColorメソッドを呼び出して変更することができます。

選択されている色が変更されると、OnSelectイベントが発生します。



このサンプルプログラムは、以下の手順で作成します。

1. プロジェクトの作成

ActiveXの形式で新規にプロジェクトを作成します。

モジュールのプロパティを設定します。

2. フォームの編集

フォームのプロパティを設定します。

フォームにフレームコントロールおよびオプションボタンコントロールを配置します。

コントロールのプロパティを設定します。

3. ActiveXコントロール用インタフェースの定義

カスタムプロパティを定義します。

カスタムメソッドを定義します。

カスタムイベントを定義します。

4. 手続きの編集

カスタムメソッドが呼び出されたときの手続きを記述します。

色が変更されたときの手続きを記述します。

5. ツールボックス用ビットマップの定義

ツールボックス用のビットマップを作成します。

ビットマップファイルをモジュールに追加します。

ActiveXコントロールとビットマップを関連づけます。

6. システム (Windows )への登録

作成したActiveXコントロールをシステムに登録します。

インストーラを作成します。



---

本節で作成するサンプルプログラムは、"ActiveX¥SampleActiveX.ppj"に格納されています。必要に応じて参照してください。

---

### 9.2.1 プロジェクトを作成する

本項では、ActiveXコントロールを作成する場合のプロジェクトの作成方法、およびプロジェクトとモジュールのプロパティの設定方法について説明します。

#### 新規にプロジェクトを作成する

[ ファイル ]メニューの[ 新規プロジェクトの作成 ]コマンドを選択すると[ 新規プロジェクトの作成 ]ダイアログボックスが表示されます。ActiveXコントロールを作成する場合、"ActiveX"を選択します。"ActiveX"を選択すると、あらかじめ、"MyActiveX"というモジュール、"Control1"というフォームが作成されたプロジェクトが作成され、モジュールの種類はDLLとなります。また、モジュールのスクリプト言語は、"OOOBCOL言語仕様"となります。

このサンプルプログラムでは、最初に、モジュール名を"PcobSampleActiveX"、フォーム名を"SelectColorControl"に変更してください。



---

作成したActiveXコントロールを、同一プロセス内で複数個利用するかどうかは、フォームのプロパティ設定ダイアログボックスの[ 多重生成 ] (MultipleInstanceプロパティに相当)で選択することができます。複数個利用する場合は、チェック状態(TRUE)にしてください。[ 新規プロジェクトの作成 ]ダイアログボックスで"ActiveX"を選択した場合の初期値は、チェック状態(TRUE)になっています。

---



COBOL言語仕様で手続きを記述する場合には、以下のような注意事項があります。詳細は、『COBOL 文法書』を参照してください。

EXTERNALデータを使用することはできません。

フォーム内で共通に使用するデータにGLOBAL句を付加する必要はありません。GLOBAL句を使用すると翻訳エラーとなります。

フォームのPROCEDUREに共通内部プログラムを記述する場合、プログラム定義ではなく、メソッド定義を書かなければなりません。

フォームの共通内部プログラムは、メソッドとして定義されるため、呼び出す場合にはCALL文ではなく、INVOKE SELF"手続き名"[USING 引数]のように、INVOKE文を使わなければなりません。

イベント手続きのデータ領域は、手続きが呼び出されるたびに初期化され、解放されます。

ActiveXコントロールは、モジュールのスクリプト言語を"COBOL85言語仕様"にした場合でも、モジュールの種類をDLLにすることによって作成できます。ただし、以下のような制限がありますので、ActiveXコントロールを作成する場合は、[ 新規プロジェクトの作成 ] ダイアログボックスで"ActiveX"を選択し、作成していくことをお勧めします。

モジュールのスクリプト言語が"COBOL85言語仕様"の場合、EXTERNALデータやEXTERNALファイルを利用できますが、ActiveXコントロールの利用者が同じ名前のデータやファイルを使用した場合、実行時エラーやデータ破壊などが発生する可能性があります。したがって、不特定多数の利用者に提供するActiveXコントロールを作成する場合は、EXTERNALデータやEXTERNALファイルを使わないようにしてください。

モジュールのスクリプト言語が"COBOL85言語仕様"の場合、フォームのプロパティ設定ダイアログボックスの[ 多重生成 ]をチェックすることはできません。したがって、作成したActiveXコントロールを、同一プロセス内で複数個利用することはできません。

ActiveXコントロールをWeb上で利用する場合は、[ 新規プロジェクトの作成 ] ダイアログボックスで"ActiveX"を選択し、作成してください。

ActiveXコントロールをWeb上で利用する方法については、「[PowerCOBOLで作成したActiveXコントロールをWeb上で利用する](#) ( p275 )」を参照してください。

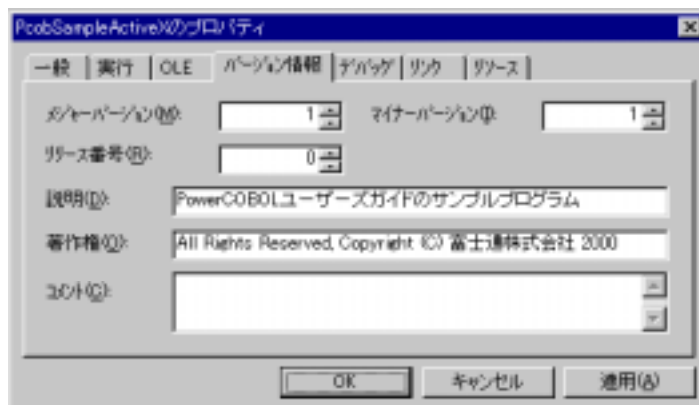
## モジュールのプロパティを設定する

モジュールのプロパティ設定ダイアログボックスの[ バージョン情報 ]タブで、これから作成するActiveXコントロールのバージョンや説明を入力します。ここで設定された情報は、他のアプリケーションでこのActiveXコントロールを利用する場合に参照されます。

このサンプルプログラムでは、以下の情報を設定します。

[ メジャーバージョン ] を1にします。( 初期値のまま )

[ マイナーバージョン ] にも1を設定します。  
[ 説明 ] に "PowerCOBOLユーザズガイドのサンプルプログラム" と記述します。  
必要に応じて、[ 著作権 ] および [ コメント ] を入力します。



## 9.2.2 フォームを編集する

フォーム編集ウィンドウを開き、以下のようにフォームを作成します。

### フォームのプロパティを設定する

以下のように、フォームのプロパティを設定します。

フォームのプロパティ設定ダイアログボックスの [ フォーム ] タブを開きます。

[ キャプション ] に "色の設定" を入力します。

[ 多重生成 ] がチェック状態であることを確認します。未チェックの場合はチェックします。

[ OLE ] タブを開きます。

[ ProgID ] に "PcobSampleActiveX.SelectColorControl" と入力します。

[ メジャーバージョン ] と [ マイナーバージョン ] に 1 を設定します。

[ 説明文字列 ] を "色を選択するコントロール" に変更します。





ProgIDは、作成するActiveXコントロールをシステム内で一意にするための文字列です。ActiveXコントロールを作成する場合、必ず設定しなければなりません。ProgIDは、このサンプルプログラムで指定したように、"モジュール名.フォーム名"という書式で設定することをお勧めします。

説明文字列は、作成するActiveXコントロールを示す簡単な文字列です。この文字列は、作成したActiveXコントロールをツールボックスに追加して再利用する場合に、ツールチップとして表示されます。

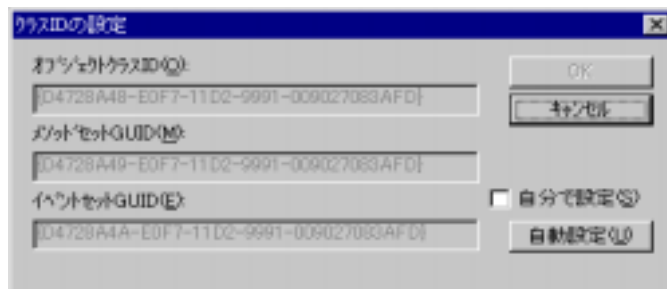


### 重要

クラスIDは、フォームごとに異なった値になっている必要があります。PowerCOBOLでアプリケーションを開発する場合、PowerCOBOLはフォームごとに一意な値を自動的に割り当てています。したがって、通常の開発では、クラスIDを設定する必要はありません。ただし、作成したプロジェクトファイルをWindowsのエクスプローラなどを使って、別のフォルダにコピーした場合、フォームのクラスIDが重なってしまいます。クラスID、ProgIDおよびActiveXコントロール本体は、システム内で関連づけられていますので、フォームのクラスIDが重複していると、システム内で矛盾が生じてしまい正しく動作しません。

このような場合は、以下のようにして、コピー先フォームのクラスIDを割り当てなおしてください。

1. [クラスIDの設定] ボタンをクリックします。
2. 警告メッセージが表示されたら、OKボタンをクリックします。
3. [クラスIDの設定] ダイアログボックスの[自動設定] ボタンをクリックします。
4. OKボタンをクリックします。





注意

---

### 重要

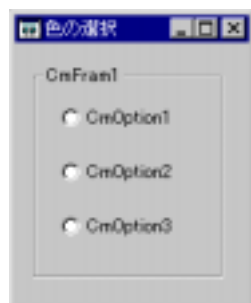
クラスIDやProgIDは、ActiveXコントロールごとに一意になっています。したがって、クラスIDやProgIDが変更されると、たとえ同じ機能をもつコントロールであっても、異なったコントロールとして扱われます。

たとえば、最初に作成したActiveXコントロールを、他のアプリケーションで利用しているとします。そのActiveXコントロールを更新し、[プロジェクトを名前を付けて保存]すると、新しいクラスIDが自動的に割り当てられます。その後、最初に作成したActiveXコントロールを削除してしまうと、最初に作成したActiveXコントロールを使用していた他のアプリケーションは、更新されたActiveXコントロールを認識することができません。たとえ同じProgIDのActiveXコントロールであっても、クラスIDが異なるため、利用することができなくなってしまいます。

---

### フォームにコントロールを配置する

まず、以下の図のように、フォームに、フレームコントロールを1つ、オプションボタンコントロールを3つ配置し、フォームの大きさを調整します。コントロールの配置方法および位置やサイズの調整方法は、「[フォームを編集する](#) ( p55) 」を参照してください。



次に、3つのオプションボタンコントロールを配列化します。コントロールを配列化する方法は、「[コントロールを配列化して利用する](#) ( p107) 」を参照してください。

### コントロールのプロパティを設定する

以下のように、コントロールのプロパティを設定します。

1. フレームコントロール(CmFrame1)の[キャプション]を"色"に変更します。
2. 配列化されたオプションボタンコントロール(CmOption1)の名前を"OP-COLOR"に変更します。
3. 1番めのオプションボタンコントロール(OP-COLOR(1))の[キャプション]を"赤"に変更します。
4. 2番めのオプションボタンコントロール(OP-COLOR(2))の[キャプション]を"緑"に変更します。



5. 3番めのオプションボタンコントロール(OP-COLOR(3))の[ キャプション ]を"青"に変更します。



### 9.2.3 ActiveXコントロール用インタフェースを定義する

ActiveXコントロールを作成する場合、ActiveXコントロールの利用者とのインタフェースとしてカスタムプロパティ、カスタムメソッドおよびカスタムイベントを定義することができます。



**注意**

ActiveXコントロールのインタフェースとして使用するカスタムプロパティ、カスタムメソッドおよびカスタムイベントの名前は、そのActiveXコントロールを利用する言語処理系で許される文字セットの範囲で構成されている必要があります。

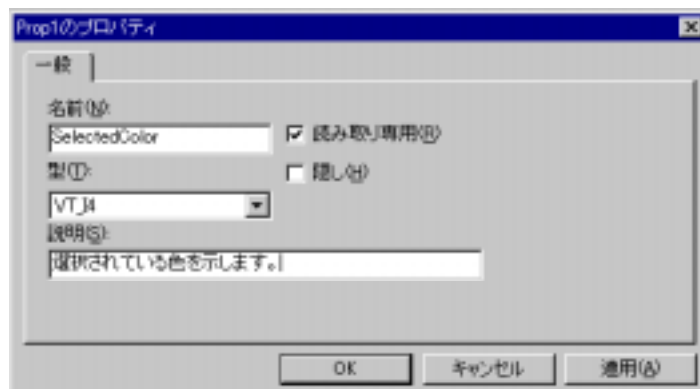
たとえば、"-" (ハイフン) は、COBOLではデータ名の一部として使用できませんが、マイナスの記号として判別される言語では使用できません。

#### カスタムプロパティを定義する

カスタムプロパティとは、ActiveXコントロールの利用者が参照したり、設定したりできる、ActiveXコントロールの各種情報です。

カスタムプロパティは、以下のように定義します。

1. デザインツリーウィンドウでフォームを選択します。
2. ポップアップメニューの[ カスタムプロパティの挿入 ]コマンドを選択します。
3. Prop1 のプロパティ設定ダイアログボックスで[ 名前 ]を "SelectedColor" に変更します。
4. [ 読み取り専用 ] をチェック状態にします。
5. [ 説明 ] に "選択されている色を示します。" と記述します。



6. OKボタンをクリックします。



---

[名前] に設定した文字列が新しいプロパティ名になります。ただし、すでにフォームが持っているプロパティまたはメソッドと同じ名前を設定することはできません。

[隠し] をチェック状態にすると、手続き編集ウィンドウの [プロパティの挿入] でのプロパティ一覧に表示されなくなります。ただし、表示されなくなるだけで、利用することは可能です。

カスタムプロパティの初期値は、プロジェクトウィンドウのデザインツリーウィンドウでフォームを選択し、右側のプロパティリストウィンドウから設定できます。

ただし、カスタムプロパティのプロパティ設定ダイアログボックスで、[読み取り専用] または [隠し] がチェック状態の場合、初期値を設定することができません。初期値を設定する場合は、いったん [読み取り専用] および [隠し] のチェックをはずしてから設定してください。

---

### カスタムメソッドを定義する

カスタムメソッドとは、ActiveXコントロールの利用者が呼び出すことができる手続きです。

カスタムメソッドは、以下のようにして定義します。

1. デザインツリーウィンドウでフォームを選択します。
2. ポップアップメニューの [カスタムメソッドの挿入] コマンドを選択します。
3. Method1 のプロパティ設定ダイアログボックスで [名前] を "SelectColor" に変更します。
4. [型] から "VT\_BOOL" を選択します。
5. [説明] に "選択されている色を変更します。" と記述します。



6. [パラメタ]タブを開きます。
7. [追加]ボタンをクリックします。
8. [名前]を"Color"に変更します。



9. OKボタンをクリックします。



[一般]タブにある[型]は、メソッドの復帰値の型を示します。VT\_BOOLは真偽を示すための型です。このサンプルプログラムでは、色の設定が成功した場合TRUE（真）を返却します。パラメタの色が赤、緑、青のどの色でもない場合は、FALSE（偽）を返却します。  
型とCOBOLの項類との対応については、『リファレンス』を参照してください。

[隠し]をチェック状態にすると、手続き編集ウィンドウの[メソッドの挿入]でのメソッド一覧に表示されなくなります。ただし、表示されなくなるだけで、利用することは可能です。



サンプルプログラムでは、パラメタの値を呼び出し元から呼び出し先に渡すために使用しています。このような場合は、パラメタの種類には"in"を指定します。それに対し、パラメタの値を呼び出し先から呼び出し元へ返すために使用する場合は、"out"を指定してください。また、呼び出し元から呼び出し先に値を渡し、かつ、呼び出し先から呼び出し元に値を返す場合は、"in,out"を指定してください。

種類が正しく設定されていない場合、OLEのVT型データとCOBOLのデータとの間で値を正しく変換できません。

### カスタムイベントを定義する

カスタムイベントとは、ActiveXコントロールの利用者に通知することができるイベントです。利用者は、そのカスタムイベントが発生したときの手続きを記述することができます。

カスタムイベントは、以下のようにして定義します。

1. デザインツリーウィンドウでフォームを選択します。
2. ポップアップメニューの[ カスタムイベントの挿入 ]コマンドを選択します。
3. Event1のプロパティ設定ダイアログボックスで[ 名前 ]を"OnSelect"に変更します。
4. [ 説明 ]に"選択されている色を変更されたときに発生します。"と記述します。



5. [ パラメタ ] タブを開きます。
6. [ 追加 ] ボタンをクリックします。
7. [ 名前 ] を"Color"に変更します。



8. OKボタンをクリックします。



[ 隠し ] をチェック状態にすると、このActiveXコントロールをPowerCOBOLのフォームに配置したとき、イベント手続きの編集対象となるイベントの一覧に表示されなくなります。



サンプルプログラムでは、パラメタの値を呼び出し元から呼び出し先に渡すために使用しています。このような場合は、パラメタの種類には" in "を指定します。それに対し、パラメタの値を呼び出し先から呼び出し元へ返すために使用する場合は、" out "を指定してください。また、呼び出し元から呼び出し先に値を渡し、かつ、呼び出し先から呼び出し元に値を返す場合は、" in, out "を指定してください。種類が正しく設定されていない場合、OLEのVT型データとCOBOLのデータとの間で値を正しく変換できません。

#### 9.2.4 手続きを編集する

手続きは、以下のように記述します。

##### カスタムメソッドが呼び出されたときの手続き

カスタムメソッドが呼び出されたときの手続きは、カスタムメソッド "SelectColor" のInvokedイベントに、以下のように記述します。

##### SelectColor - Invoked

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
```

( 続く )

( 続き )

```

01 POW-COLOR PIC S9(9) COMP-5.
01 RETURN-VALUE PIC S9(4) COMP-5.
PROCEDURE DIVISION USING POW-COLOR RETURN-VALUE.
* 復帰値を初期化します。
 MOVE POW-TRUE TO RETURN-VALUE
* 選択されている色を変更します。
 EVALUATE POW-COLOR
 WHEN POW-COLOR-RED
 MOVE POW-TRUE TO "Value" OF OP-COLOR(1)
 WHEN POW-COLOR-GREEN
 MOVE POW-TRUE TO "Value" OF OP-COLOR(2)
 WHEN POW-COLOR-BLUE
 MOVE POW-TRUE TO "Value" OF OP-COLOR(3)
 WHEN OTHER
* 赤、緑、青以外の場合、復帰値をFALSEにします。
 MOVE POW-FALSE TO RETURN-VALUE
 END-EVALUATE
 MOVE POW-COLOR TO "SelectedColor" OF POW-SELF

```



ワンポイント

連絡節のPOW-COLORは、カスタムメソッドのパラメタに定義したColorを意味し、その項類は、Colorパラメタの型(VT\_I4)に対応しています。また、RETURN-VALUEは、カスタムメソッドの復帰値で、その項類は、カスタムメソッド自身の型(VT\_BOOL)に対応しています。  
型とCOBOLの項類との対応については、『リファレンス』を参照してください。

## カスタムイベントの発生方法

カスタムイベントは、以下の記述形式で発生させることができます。

```

 INVOKE カスタムイベント名 "Invoke"
 USING [パラメタ...]

```

### カスタムイベント名

カスタムイベントの名前です。

### パラメタ

カスタムイベントのパラメタです。指定するパラメタ数およびその型は、カスタムイベントのパラメタの定義と一致していなければなりません。

### 復帰値

ありません。

## 色が変更されたときの手続き

色が変更された場合、オプションボタンコントロールのClickイベントが発生します。したがって、選択中の色を示すカスタムプロパティ"SelectedColor"

は、オプションボタンコントロールのClickイベント内で更新します。また、色が変更されたときに利用者へ通知するカスタムイベント"OnSelect"を、オプションボタンコントロールのClickイベント内で発生させます。

#### OP-COLOR-Click

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WK-COLOR-NEW PIC S9(9) COMP-5.
01 WK-COLOR-OLD PIC S9(9) COMP-5.
LINKAGE SECTION.
01 POW-INDEX PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW-INDEX.
* 選択されている色を保存しておきます。
 MOVE "SelectedColor" OF POW-SELF TO WK-COLOR-OLD
* 新しく選択された色を求めます。
 EVALUATE POW-INDEX
 WHEN 1
 MOVE POW-COLOR-RED TO WK-COLOR-NEW
 WHEN 2
 MOVE POW-COLOR-GREEN TO WK-COLOR-NEW
 WHEN 3
 MOVE POW-COLOR-BLUE TO WK-COLOR-NEW
 END-EVALUATE
* 色が変更されていれば、カスタムイベントを発生させます。
 IF WK-COLOR-NEW NOT = WK-COLOR-OLD THEN
 INVOKE OnSelect "Invoke" USING WK-COLOR-NEW
 END-IF
* カスタムプロパティに色を設定します。
 MOVE WK-COLOR-NEW TO "SelectedColor" OF POW-SELF

```

### 9.2.5 ツールボックス用ビットマップを定義する

PowerCOBOLではActiveXコントロールを作成し、そのActiveXコントロールをツールボックスに追加して表示するためのビットマップを定義することができます。

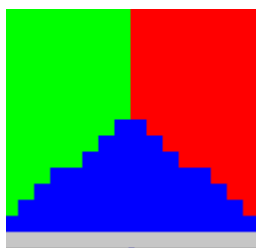
#### ツールボックス用のビットマップを作成する

ツールボックス用のビットマップは、Windows のアクセサリにある「ペイント」などのツールを使って作成します。

ツールボックスに表示できるビットマップは、以下の設定で作成します。

- 幅16ピクセル
- 高さ15ピクセル
- 色16色
- 最下端のラインは、背景色となる色

サンプルプログラムでは、以下のようなビットマップを作成して使用します。



### ビットマップファイルをモジュールに追加する

作成したビットマップファイルをモジュールに追加します。サンプルプログラムでは、リソース名を"Bitmap1"と指定してください。(初期値のまま)ビットマップファイルをモジュールに追加する方法は、「[外部ファイルを使う](#) ( p96) 」を参照してください。

### ActiveXコントロールとビットマップを関連づける

作成したビットマップは、以下のようにしてActiveXコントロールと関連づけます。

1. フォームのプロパティ設定ダイアログボックスを開きます。
2. [ リソース ] タブを開きます。
3. [ プロパティ名 ] から"ToolboxBitmap"を選択します。
4. [ リソース名 ] からモジュールに追加したビットマップファイルに対応するリソース名"Bitmap1"を選択します。

## 9.2.6 システムに登録する

作成したActiveXコントロールを利用するには、ActiveXコントロールをシステム(Windows )に登録しておく必要があります。また、別のシステムで利用する場合には、ActiveXコントロールをシステムに登録するためにインストーラを作成し、あらかじめ別のシステムにインストールしておく必要があります。システムへの登録およびインストーラの作成は、モジュールをビルドし、エラーを取り除いて正しくDLLが作成できたあとに行ってください。

### 作成したActiveXコントロールをシステムに登録する

ActiveXコントロールは、以下の手順でシステムに登録することができます。

1. デザインツリーウィンドウでモジュールを選択します。
2. ポップアップメニューの[ システムに登録 ] コマンドを選択します。
3. システムに登録された内容のメッセージが表示されたら、OKボタンをクリックします。

システムへの登録を解除する場合は、ポップアップメニューの[ システムから解除 ]コマンドを選択してください。ただし、登録を解除した場合、そのActiveXコントロールを利用している他のプロジェクトファイルを編集できなくなります。

### インストーラを作成する

インストーラを作成しておくことにより、PowerCOBOLの開発環境が使用できな



いシステム（ランタイムシステムだけがインストールされているシステム）でも、ActiveXコントロールをシステムに登録して利用することができます。  
また、ActiveXコントロールが不要になった場合は、Windows のコントロールパネルから「アプリケーションの追加と削除」でActiveXコントロールをアンインストールし、システムへの登録を解除できます。インストーラの作成方法については、「[インストーラを作成する](#)（ p133）」を参照してください。



注意

---

Windows NT 4.0、Windows 2000およびWindows XPでは、ActiveXコントロールをシステムに登録するためには、管理者（Administratorsグループ）の権限が必要です。したがって、作成したActiveXコントロールをシステムに登録する場合、および、インストーラを使って他のシステムにインストールする場合には、管理者の権限をもつユーザ名でシステムにログオンしてください。

---

## 9.3 ActiveXコントロールを使ったアプリケーションを作成する

本節では、ActiveXコントロールを使ったアプリケーションの作成方法について説明します。

ActiveXコントロールを利用するには、以下の2つの方法があります。

ActiveXコントロールをフォームに配置し、そのまま利用する方法

ActiveXコントロールを別ウィンドウで表示して、利用する方法

これらの方法は「[ActiveXコントロールを作成する](#) ( p245) 」で作成したActiveXコントロールを利用して説明していきます。

1つめの方法を使って作成するアプリケーションは、以下の機能を持ちます。

フォームの背景色をActiveXコントロール上で選択された色に変更します。

ActiveXコントロール上の色の選択肢を赤に戻すためのボタンを持ちます。

さらに、このアプリケーションに、ActiveXコントロールを別ウィンドウに表示するボタンを追加し、2つめの方法について説明します。

これから作成するアプリケーションを実行すると、以下のようなウィンドウが表示されます。



サンプルプログラムは、以下の手順で作成します。ActiveXコントロールを利用するためのプロジェクトは、標準フォームの形式を使用してください。プロジェクトを作成したら、モジュール名を"UsingActiveX"に変更してください。

### 1. フォームの編集

ActiveXコントロールをツールボックスに追加します。

フォームにActiveXコントロールおよびコマンドボタンコントロールを配置します。

コントロールのプロパティを設定します。

### 2. 手続きの編集

ActiveXコントロールを起動するため手続きを記述します。

ActiveXコントロール上の選択肢が変更されたときのイベント手続きを記述します。

ボタンがクリックされたときのイベント手続きを記述します。

次に、ActiveXコントロールを別ウィンドウで表示するサンプルプログラムは、以下の手順で機能を追加します。

1. フォームの編集

コマンドボタンコントロールを追加します。

2. 手続きの編集

ボタンがクリックされたときのイベント手続きを記述します。



ワンポイント

本節で作成するサンプルプログラムは、"ActiveX¥UsingActiveX.ppj"に格納されています。必要に応じて参照してください。



注意

PowerCOBOLで作成したActiveXコントロールを他のシステムで利用するには、そのシステムにNetCOBOLおよびPowerCOBOLのランタイムシステムがインストールされている必要があります。

### 9.3.1 フォームを編集する

フォーム編集ウィンドウを開き、以下のようにフォームを作成します。

#### ActiveXコントロールをツールボックスに追加する

以下の手順で、ActiveXコントロールをツールボックスに追加し、フォーム上で利用できるようにします。

1. [ ツール ]メニューの[ カスタムコントロール ]コマンドを選択します。
2. [ カスタムコントロール ]ダイアログボックスの[ 利用可能なコントロール ]一覧中の [ PowerCOBOLユーザーズガイドのサンプルプログラム ] をチェック状態にします。
3. OKボタンをクリックします。
4. ツールボックスに、ActiveXコントロールのビットマップが追加されたことを確認します。



---

一覧に表示される"PowerCOBOLユーザーズガイドのサンプルプログラム"は、ActiveXコントロール作成時にモジュールのプロパティで[バージョン情報]タブの"説明"に設定した文字列に対応しています。

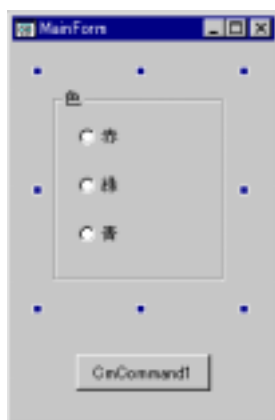
ツールボックスに追加されるビットマップは、ActiveXコントロール作成時にフォームのプロパティで[リソース]タブの"ToolboxBitmap"で指定したリソースに対応しています。

ツールボックスのActiveXコントロールを示すビットマップ上に、マウスポインタを移動したときに表示される文字列"色を選択するコントロール"は、[OLE]タブの"説明文字列"に設定した文字列です。

---

### フォームにコントロールを配置する

以下のように、ActiveXコントロールとコマンドボタンコントロールを1つずつ配置し、フォームの大きさを調整します。コントロールの配置方法および位置やサイズの調整方法は、「[フォームを編集する](#) ( p55) 」を参照してください。



### コントロールのプロパティを設定する

コマンドボタンコントロール(CmCommand1)の[キャプション]を"赤に戻す"に変更します。



### 9.3.2 手続きを編集する

手続きは、以下のように記述します。

#### フォームが最初に関かれたときの手続き

最初に、フォームの背景色の初期値として赤を選択します。

##### MainForm-Opened

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
```

- \* ActiveXコントロールで選択されている色の初期値を設定します。  
 INVOKE SelectColorControl1 "SelectColor" USING POW-COLOR-RED



SelectColorメソッドは、「[ActiveXコントロールを作成する](#) ( p245) 」で、ActiveXコントロールを作成したときに定義したカスタムメソッドです。  
 このように、カスタムメソッドやカスタムプロパティは、ActiveXコントロールを利用する側（コンテナ側）の手続き中で利用することができます。  
 また、カスタムメソッドやカスタムプロパティに加え、フォームの以下のメソッドおよびプロパティを利用できます。

```
Activateメソッド
Deactivateメソッド
DoModalメソッド
Enabledプロパティ
```

メソッドの使用方法については、後述の「[ボタンがクリックされたときの手続きを記述する](#) ( p265) 」を参照してください。

また、メソッドとプロパティの詳細については、『リファレンス』を参照してください。

### 色の選択肢が変更されたときの手続き

ActiveXコントロールの色の選択肢が変更された場合、ActiveXコントロールの OnSelect イベントが発生します。その場合、選択肢に合わせてフォームの背景色を変更します。

#### SelectColorControl1-OnSelect

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-COLOR PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW-COLOR.
```

- \* ActiveXでの選択肢が変更されたらフォームの背景色を変更します。  
MOVE POW-COLOR TO "BackColor" OF POW-SELF

### ボタンがクリックされたときの手続き

[ 赤に戻す ] ボタンがクリックされた場合、ActiveXコントロール上の選択肢を赤に変更し、フォームの背景色を赤に戻します。

#### CmCommand1-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
PROCEDURE DIVISION.
```

- \* ActiveXのメソッドを呼び出して、選択されている色を赤に戻します。  
INVOKE SelectColorControl1 "SelectColor" USING POW-COLOR-RED

ここで、モジュールをビルドし、作成したアプリケーションを実行してみてください。配置したActiveXコントロールは、フォーム上にそのまま表示され、オプションボタンの選択により、フォームの背景色を変更できます。

## 9.3.3 コマンドボタンコントロールを追加する

以下の図のように、フォームにコマンドボタンコントロールを1つ配置し、[ 別画面で開く ] ボタンを追加します。



#### 9.3.4 ボタンがクリックされたときの手続きを記述する

[ 別画面で開く ] ボタンがクリックされた場合、ActiveXコントロールの Activate メソッドを呼び出します。Activate メソッドを使用すると、別ウィンドウで実行することができます。また、表示したウィンドウを閉じる場合は、Deactivate メソッドを使用します。

この手続きでは、ボタンに表示されている文字列を判定して、1つのボタンで、開く動作と閉じる動作ができるようにしています。

##### CmCommand2-Click

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
 IF "Caption" OF CmCommand2 = "別画面で開く" THEN
 INVOKE SelectColorControl1 "Activate"
 MOVE "別画面を閉じる" TO "Caption" OF CmCommand2
 ELSE
 INVOKE SelectColorControl1 "Deactivate"
 MOVE "別画面で開く" TO "Caption" OF CmCommand2
 END-IF

```

モジュールをビルドし、作成したアプリケーションを実行してみてください。アプリケーションを起動した時点では、配置したActiveXコントロールはフォーム上にそのまま表示され、オプションボタンの選択により、フォームの背景色を変更することができます。

[ 別画面で開く ] ボタンをクリックすると、配置したActiveXコントロールは別ウィンドウに表示されます。



ActiveXコントロールがActivateメソッドにより別ウィンドウに表示されると、フォームに配置されていたActiveXコントロールは消去されます。

つねに、別ウィンドウで表示されるようなアプリケーションを作成する場合は、配置したActiveXコントロールが見えないように設計時に大きさを調整しておくか、ActiveXコントロールのプロパティ設定ダイアログボックスで、[共通]タブの[可視]のチェックをはずしてください。Activateメソッドの詳細は、「[Activateメソッドの役割](#) ( p266) 」を参照してください。

ActiveXコントロールは、呼び出し元のフォームが開かれたときに一度だけ初期化されますが、ActiveXコントロールのOpenedイベントは、別ウィンドウおよび呼び出し元のフォームに表示されるごとに発生します。Openedイベントの詳細は、「[OpenedイベントとClosedイベント](#) ( p267) 」を参照してください。

ActiveXコントロールを別ウィンドウで表示しても、そのウィンドウが閉じられたとき、呼び出し元フォームにCloseChildイベントは発生しません。ウィンドウが閉じられたタイミングを知りたい場合は、ActiveXコントロール側のClosedイベント内で、ウィンドウが閉じられたことを意味するカスタムイベントを発生させ、呼び出し元フォームにそのカスタムイベントが通知されるようにします。

Activateメソッドの代わりにDoModalメソッドを使うと、開いたウィンドウが閉じられるまで呼び出し元のフォーム(MainForm)を無効状態にできます。つまり、ActiveXコントロールは、ダイアログボックスを使用するように、モーダルなウィンドウで表示されます。

ActivateメソッドとDoModalメソッドの動作の違いは、「[CallFormメソッドとOpenFormメソッドの違い](#) ( p212) 」と似ています。

### 9.3.5 Activateメソッドの役割

このサンプルプログラムで使用したように、ActivateメソッドはPowerCOBOLで作成したActiveXコントロールを別画面として開く場合に利用します。

このように、ActiveXコントロールを別画面として表示することを、COMの用語でアウトブレースアクティベートと呼び、その状態をアウトブレースアクティブ状態と呼びます。また、別画面として表示しないで、コンテナ(フォーム)上に表示することをインブレースアクティベートと呼び、その状態をインブレースアクティブ状態と呼びます。

逆に、Deactivateメソッドは、別画面として開いているActiveXコントロールを閉じる場合に利用します。

このように、Deactivateメソッドにより、アウトブレースアクティベート状態を終了することを、COMの用語でアウトブレースデアクティベートと呼びます。



また、インプレースアクティベート状態のActiveXコントロールがアウトプレースアクティベートによりインプレースアクティベート状態でなくなることを、インプレースデアクティベートと呼びます。

つまり、このサンプルプログラムでの動作は、ActiveXコントロールの動作および状態と以下のように対応しています。

1. アプリケーションを起動する。  
コンテナ上にインプレースアクティベートされ、インプレースアクティブ状態になります。
2. [ 別画面で開く ] ボタンをクリックする。  
コンテナ上のActiveXコントロールはインプレースデアクティベートされ、かつ、アウトプレースアクティベートされて、アウトプレースアクティブ状態になります。
3. [ 別画面を閉じる ] ボタンをクリックする。  
ActiveXコントロールはアウトプレースデアクティベートされ、かつ、インプレースアクティベートされて、インプレースアクティブ状態になります。

PowerCOBOLでは、ActiveXコントロールは、つねにインプレースアクティブ状態またはアウトプレースアクティブ状態になっています。ActiveXコントロールの状態と呼び出したメソッドに対応するActiveXコントロールの動作を、以下に示します。

| ActiveXコントロールの状態 | Activateメソッド                        | Deactivateメソッド(*1)                  |
|------------------|-------------------------------------|-------------------------------------|
| インプレースアクティブ状態    | インプレースデアクティベートされ、アウトプレースアクティベートされる。 | 何も起きない。                             |
| アウトプレースアクティブ状態   | フォーカスが与えられ、最前面に表示される。(*2)           | アウトプレースデアクティベートされ、インプレースアクティベートされる。 |

(\*1)CloseFormメソッドを呼び出した場合および[ 閉じる ] ボタンをクリックした場合も同様です。

(\*2)他のウィンドウの状態により、最前面に表示されない場合もあります。

### 9.3.6 OpenedイベントとClosedイベント

Openedイベントはフォームが開かれる場合に発生し、Closedイベントはフォームが閉じられた場合に発生します。また、QueryCloseイベントはClosedイベントが発生する直前に発生します。

ActiveXコントロールの場合、Openedイベントはアウトプレースアクティベートおよびインプレースアクティベートされた場合に発生し、Closedイベントはアウトプレースデアクティベートおよびインプレースデアクティベートされた場合に発生します。また、QueryCloseイベントは、アウトプレースアクティブ状態のフォームがアウトプレースデアクティベートされる場合にだけ発生します。

つまり、コンテナ上でインプレースアクティブ状態のActiveXコントロールに対して、Activateメソッドを呼び出すと、ActiveXコントロールはインプレース

ステアクティブートされ、アウトプレースアクティブートされるので、Closed イベントおよびOpened イベントが発生します。このとき、QueryClose イベントは発生しません。

また、アウトプレースアクティブ状態のActiveXコントロールに対して、Deactivate メソッドを呼び出すと、ActiveX コントロールはアウトプレースデアクティブートされ、インプレースアクティブートされるので、QueryClose イベント、Closed イベントおよびOpened イベントが発生します。

したがって、ActiveX コントロールのOpened イベントに初期処理を記述したり、QueryClose イベントやClosed イベントに終了処理を記述したりする場合には、それぞれのイベントが発生するタイミングを考慮する必要があります。

ActiveX コントロールの状態、操作および操作前後の状態でのActiveX コントロールに発生するイベントの関係を以下に示します。

| 状態                 | 操作         | 操作前のActiveX |    |    | 操作後のActiveX |    |    |
|--------------------|------------|-------------|----|----|-------------|----|----|
|                    |            | OP          | QC | CL | OP          | QC | CL |
| -                  | コンテナ起動     | (存在しない)     |    |    |             | -  | -  |
| インプレース<br>アクティブ状態  | Activate   | -           | -  |    |             | -  | -  |
|                    | Deactivate | (状態の変化なし)   |    |    |             |    |    |
|                    | コンテナ終了     | -           | -  |    | (存在しない)     |    |    |
| アウトプレース<br>アクティブ状態 | Activate   | (状態の変化なし)   |    |    |             |    |    |
|                    | Deactivate | -           |    |    |             | -  | -  |
|                    | コンテナ終了     | -           | -  |    | (存在しない)     |    |    |

： イベントが発生する

-： イベントが発生しない

OP: Opened イベント

QC: QueryClose イベント

CL: Closed イベント

### 9.3.7 Visibleプロパティの取り扱い方法

フォームをActiveXコントロールとして作成し、コンテナに配置して利用する場合、ActiveXコントロールのVisibleプロパティをコンテナ側の手続きの中で変更した場合と、ActiveXコントロール自身がもつ手続きの中で変更した場合では、ActiveXコントロールを表示する動作が異なります。

コンテナ側でActiveXコントロールのVisibleプロパティを操作した場合は、ActiveXコントロールがインプレースアクティブ状態であるかアウトプレースアクティブ状態であるかにかかわらず、表示状態を変更できます。

それに対し、ActiveXコントロール自身がもつ手続きの中でVisibleプロパティを操作した場合、アウトプレースアクティブ状態のときだけ、表示状態を変更できます。

ただし、アウトプレースアクティブ状態であっても、ActiveXコントロールの手続き中でVisibleプロパティを操作して表示状態を変更した場合、コンテナ側ではその変更が認識できません。

また、アウトプレースアクティブ状態でかつ非表示状態のActiveXコントロー

ルに対し、Activateメソッドを呼び出すとActiveXコントロールは表示状態になり、Visibleプロパティも自動的に変更されますが、これもコンテナ側では認識できません。

これらの場合、次にインプレースアクティブ状態になった場合、表示状態は引き継がれずコンテナ側が認識している表示状態で表示されます。

つまり、ActiveXコントロールの手続き中で自分自身のVisibleプロパティを操作すると、表示状態の管理が非常に難しくなってしまいます。

したがって、ActiveXコントロールのVisibleプロパティは、コンテナ側の手続き中で操作することをお勧めします。

## 9.4 オートメーションサーバを使ったアプリケーションを作成する

本節では、オートメーションサーバを使ったアプリケーションの作成方法について説明します。

この方法は、ActiveXコントロールをNetCOBOLの\*COMクラスを使って利用する方法です。本節では、「[ActiveXコントロールを作成する](#) ( p245) 」で作成したActiveXコントロールを使って、この方法について説明していきます。

作成するサンプルプログラムは、以下の機能を持ちます。

フォームの背景色を変更するための [ 色の選択 ] ボタンを持ちます

[ 色の選択 ] ボタンがクリックされると、オートメーションサーバとして作成したウィンドウを表示します。

表示したウィンドウ上で色が選択され、ウィンドウが閉じられるとフォームの背景色を変更します。

これから作成するサンプルプログラムを実行すると、以下のようなウィンドウが表示されます。



このサンプルプログラムは、以下の手順で作成します。作成するプロジェクトは、標準フォームの形式を使用してください。プロジェクトを作成したら、モジュール名を "UsingAutomationServer" に変更してください。

1. フォームにコマンドボタンコントロールを配置します。
2. オートメーションサーバを利用するための手続きを記述します。



本節で作成するサンプルプログラムは、

"ActiveX¥UsingAutomationServer.ppj" に格納されています。必要に応じて参照してください。

### 9.4.1 フォームにコマンドボタンコントロールを配置する

フォーム編集ウィンドウを開き、以下の手順でコマンドボタンコントロールをフォームに配置します。

1. フォームにコマンドボタンコントロール配置します。
2. フォームの大きさおよびコマンドボタンコントロールの位置を調整します。
3. コマンドボタンコントロール(CmCommand1)の[ キャプション ]を"色の選択"に変更します。



### 9.4.2 手続きを編集する

オートメーションサーバを利用するための手続きは、以下のように記述します。

#### REPOSITORY句の記述

オートメーションサーバを作成するには、フォームのREPOSITORY句に、NetCOBOLの\*COMクラスを以下のように宣言しておく必要があります。

#### MainForm-REPOSITORY

```
CLASS COM AS "**COM"
```

#### 作業場所節の記述

フォームの作業場所節には、オートメーションサーバとして利用するActiveXコントロールのProgIDと、オートメーションサーバとして作成されたオブジェクトを識別するためのデータを、以下のように宣言します。

#### MainForm-WORKING-STORAGE

- \* オートメーションサーバとして開くActiveXコントロールのProgID

```
01 FORM-NAME PIC X(64) GLOBAL
 VALUE "PcobSampleActiveX.SelectColorControl.1".
```
- \* オートメーションサーバとして作成される\*COMクラスのオブジェクト

```
01 SUB-FORM OBJECT REFERENCE COM GLOBAL.
```

#### フォームを最初に開いたときの手続き

フォームを開いたとき、フォームの背景色の初期値として赤を設定します。

### MainForm-Opened

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
* フォームの背景色の初期値として赤を設定します。
 MOVE POW-COLOR-RED TO "BackColor" OF POW-SELF
```

### ボタンをクリックしたときの手続き

[ 色の選択 ] ボタンをクリックした場合、ActiveXコントロールをオートメーションサーバとして作成し、色を選択するためのウィンドウを表示します。色が選択され、ウィンドウが閉じられたら、フォームの背景色を変更します。

### CmCommand1-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 BACK-COLOR PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* フォームへの操作ができないように、無効状態にします。
 MOVE POW-FALSE TO "Enabled" OF POW-SELF
* フォームの現在の背景色を求めます。
 MOVE "BackColor" OF POW-SELF TO BACK-COLOR
* ActiveXコントロールをオートメーションサーバとして作成します。
 INVOKE COM "CREATE-OBJECT" USING FORM-NAME RETURNING SUB-FORM
* フォームの現在の背景色でActiveXコントロールの選択色を初期化します。
 INVOKE SUB-FORM "SelectColor" USING BACK-COLOR
* 作成したオートメーションサーバを開きます。
 INVOKE SUB-FORM "DoModal"
* 選択された色を求めます。
 INVOKE SUB-FORM "GET-SELECTEDCOLOR" RETURNING BACK-COLOR
* オートメーションサーバを解放します。
 SET SUB-FORM TO NULL
* 選択された色をフォームの背景色に設定します。
 MOVE BACK-COLOR TO "BackColor" OF POW-SELF
* フォームを有効にし、クリックしたボタンにフォーカスを戻します。
 MOVE POW-TRUE TO "Enabled" OF POW-SELF
 INVOKE CmCommand1 "SetFocus"
```

モジュールをビルドし、作成したアプリケーションを実行してみてください。  
[ 色の選択 ] ボタンをクリックすると、ActiveXコントロールが別ウィンドウに表示されます。色を選択し、ウィンドウを閉じると、呼び出し元のフォームの背景色が変更されます。



この手続きでは、"GET-SELECTEDCOLOR"メソッドを使って、選択された色を求めています。これは、"GET-プロパティ名"という形式でメソッドを呼び出すと、対象となるオブジェクトのプロパティの値を取得することができるNetCOBOLのオブジェクト機能を利用しています。

NetCOBOLの\*COMクラスを使ったプログラミングについての詳細は、『NetCOBOL 使用手引書』を参照してください。

ActiveX コントロールをオートメーションサーバとして作成(CREATE-OBJECT)しただけでは、ウィンドウとして表示されません。DoModalメソッドを呼び出すことによりGUIをもつウィンドウとして表示されます。

また、表示されたウィンドウを閉じただけでは、オートメーションサーバはWindows のメモリ上に残っています。解放する場合には、オブジェクト参照用の変数に対してNULLを設定してください。



CREATE-OBJECTメソッドを呼び出したあと、DoModalメソッドでGUIとして表示されるまでのあいだ、オブジェクトのプロパティへのアクセスおよびメソッドの呼び出しはできませんが、イベントは発生しません。

たとえば、ウィンドウが表示されている状態で、オプションボタンコントロールのValueプロパティにPOW-TRUEが設定されるとオプションボタンコントロールのClickイベントが発生します。しかし、このサンプルプログラムのActiveXコントロールでは、SelectColorメソッドの呼び出しによってValueプロパティにPOW-TRUEを設定しても、オプションボタンコントロールはウィンドウの要素として表示されていないので、Clickイベントは発生しません。

したがって、オートメーションサーバを利用してアプリケーションを作成する場合は、イベントが発生することを前提にした手続きを記述しないようにしてください。

NetCOBOLの\*COMクラスを使ってオートメーションサーバを作成した場合、そのウィンドウからイベントを受け取ることができないため、呼び出し元のフォームでは、ウィンドウが閉じられたかどうかを認識できません。したがって、このサンプルプログラムではDoModalメソッドを使って、ウィンドウが閉じられるまで呼び出し元のフォームの処理が中断されるようにしています。Activateメソッドを使用してウィンドウを表示させると、ウィンドウが閉じられたことを認識することができないため、オブジェクトを解放(NULLを設定)するタイミングを特定できません。

## 9.5 他のアプリケーションやバッチファイルを起動する

本節では、他のアプリケーションやバッチファイルを起動する場合の呼び出し方法について説明します。

PowerCOBOLからアプリケーション(EXE)やバッチファイル(BAT)を起動するには、フォームのExecuteメソッドまたはExecuteSyncメソッドを使用します。アプリケーションやバッチファイルを起動するコマンド文字列をこれらのメソッドの引数に指定すると、指定したコマンド文字列が実行され、アプリケーションやバッチファイルを起動することができます。

たとえば、"A.TXT"というテキストファイルをWindowsの「メモ帳」で編集する場合には、以下のような手続きを記述します。

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 COMMANDLINE PIC X(8192) VALUE "NOTEPAD.EXE A.TXT".
PROCEDURE DIVISION.
 INVOKE POW-SELF "Execute" USING COMMANDLINE
```

Executeメソッドは、コマンドを起動することはできますが、コマンドの終了やコマンドの終了コードを取得することはできません。したがって、コマンドの終了を待って処理を再開するような場合には、利用することができません。逆に、ExecuteSyncメソッドは、コマンドを起動し、コマンドの終了を待って処理を再開します。したがって、上記の例でExecuteメソッドをExecuteSyncメソッドに置き換えると、「メモ帳」が終了するまで、そのイベント手続き中の次の処理が実行されません。



注意

ExecuteSyncメソッドによるコマンドの起動で待ち状態になるのは、ExecuteSyncメソッドを呼び出したイベント手続きだけです。コマンド実行中であっても、フォーム自身は無効状態にはならず、マウスやキーの操作により、他のイベント手続きを実行させることができます。

したがって、コマンド実行中、他のイベントが発生しないことを前提にしたアプリケーションを作成すると、実行時にエラーが発生する場合があります。たとえば、コマンドを実行してイベント手続きが待ち状態のとき、他の操作によってフォームを閉じようとした場合、フォームを閉じることができないという内容のエラーが発生します。



## 9.6 PowerCOBOLで作成したActiveXコントロールをWeb上で利用する

本節では、PowerCOBOLで作成したActiveXコントロールを『Microsoft Internet Explorer (以降、Internet Explorerと略します)』で表示されるWebページ上に配置して利用する方法について説明します。

本節では、ActiveXコントロールとして作成したサンプルプログラム "ActiveX¥WebTimer.ppj" を使って、このActiveXコントロールをWeb上に表示します。

このActiveXコントロールは、以下の機能を持ちます。

1秒ごとに、表示されている数字を1ずつ加算していきます。

リセットボタンをクリックすると、表示されている数字が0に戻ります。



注意

PowerCOBOLで作成したActiveXコントロールをWeb上で利用し、他のシステムからそのWebページを開いて動作させるには、Webページを開くシステムにもNetCOBOLおよびPowerCOBOLのランタイムシステムがインストールされている必要があります。

### 9.6.1 ActiveXコントロールを用意する

最初に、Web上で利用するActiveXコントロールを以下の手順で作成します。

1. PowerCOBOLを起動し、"ActiveX¥WebTimer.ppj"を開きます。
2. [プロジェクト]メニューの[すべてリビルド]コマンドを選択し、ActiveXコントロールを作成します。
3. モジュールを選択し、ポップアップメニューの[システムに登録]コマンドを選択します。



注意

Web上で利用するActiveXコントロールは、新規プロジェクトの作成で、"ActiveX"を選択して作成されている必要があります。ActiveXコントロールを作成する方法については、「[ActiveXコントロールを作成する](#) ( p245) 」を参照してください。

### 9.6.2 HTML文書を記述する

ActiveXコントロールをInternet Explorerで利用するには、objectタグを使ってHTML文書を作成します。

## 書きかた

```
<object WIDTH="幅" HEIGHT="高さ"
 CLASSID="クラスID">
</object>
```

## 幅と高さ

幅および高さには、配置するActiveXコントロールの幅と高さをピクセル単位で指定します。

## クラスID

配置するActiveXコントロールのクラスIDを指定します。クラスIDは、以下のようにして求めます。

1. ActiveXコントロールに対応するフォームのプロパティ設定ダイアログボックスを開きます。
2. [OLE] タブを開きます。
3. [クラスIDの設定] をクリックします。
4. 警告が通知されますが、OKボタンをクリックして先に進みます。
5. [クラスIDの設定] ダイアログボックスで[オブジェクトクラスID]で示されている値が、求めるクラスIDとなります。

## 記述例

サンプルプログラムで用意したActiveXコントロールに対応するフォームの[オブジェクトクラスID]は、{409A0921-0793-11D3-88F9-00000E98DD12}となっています。したがって、このActiveXコントロールを利用する場合は、以下のように記述します。

```
<object WIDTH="100" HEIGHT="50"
 CLASSID="CLSID:409A0921-0793-11D3-88F9-00000E98DD12">
</object>
```

[クラスIDの設定] ダイアログボックスの表示にあった "{" と "}" は、記述に含めないでください。

HTML文書で使用するobjectタグの詳細については、W3Cやブラウザのマニュアルを参照してください。



Web上で利用するActiveXコントロールは、あらかじめPowerCOBOLのフォームに配置して、動作を確認してください。また、Web上に配置した状態でActiveXコントロールをデバッグする場合は、以下の設定をしてください。

1. ActiveXコントロールに対応するモジュールのプロパティ設定ダイアログボックスを開きます。
2. [実行] タブを開きます。
3. [起動ファイル] にInternet Explorerの実行ファイル(Windows NT 4.0の場合は、IEXPLORE.EXE)を指定します。
4. [コマンドライン引数] に、ActiveXコントロールを利用するHTML文書名をフルパスで指定します。

## 9.7 DBアクセスコントロールを利用してデータベースと連携する

本節では、DBアクセスコントロールを使って、データベースと連携するアプリケーションの作成方法について説明します。

DBアクセスコントロールは、ODBC(Open DataBase Connectivity)を使って、データベースと連携するためのコントロールです。したがって、DBアクセスコントロールを利用するためには、ODBCおよび連携対象のデータベースに関する基本的な知識が必要です。

本節では、データベースとして『Microsoft SQL Server(TM)(以降、SQL Serverと略します)』を利用し、サンプルプログラムを使って、以下の手順でアプリケーションを作成します。

1. クライアント環境(ODBC)の設定をします。
2. フォームにコントロールを配置します。
3. コントロールの手続きを記述します。

サンプルプログラムでは、データベースサーバとして『SQL Server Ver 6.5以降』がセットアップされている環境が必要です。また、クライアントには、『SQL Server ODBC ドライバ Ver 2.65以降』がインストールされている環境が必要です。

SQL Serverに関する詳細については、『MSDN』の「プラットフォームSDK - データアクセスサービス」に掲載されている「Microsoft SQL Server プログラマーズ ツールキット」を参照してください。

また、他のデータベースを利用する場合は、対応するマニュアルを参照してください。



ワンポイント

本節で作成するサンプルプログラムは、"DBAccess¥DBAccess.ppj"に格納されています。必要に応じて参照してください。

### 9.7.1 クライアント環境(ODBC)を設定する

このサンプルプログラムは、SQL Serverのデータベース"pubs"の中にあるテーブル"jobs"にアクセスします。

ODBCの設定では、以下の情報を設定します。

データソース名:sql65

データベース名:pubs

以下の手順で、クライアント環境(ODBC)を設定します。

1. Windows のコントロールパネルの[ ODBCデータソース ]アイコンをダブルクリックし、[ ODBC データソースアドミニストレータ ]ダイアログボックスを表示します。

2. [ ユーザ DSN ] タブの [ 追加 ] ボタンをクリックします。
3. [ データソースの新規作成 ] で"SQL Server"を選択し、[ 完了 ] ボタンをクリックします。
4. [ SQL Serverに接続するための新規データソースを作成する ] ウィザード (以降、データソースウィザードと略します。) の最初のウィンドウで、[ 名前 ] に"sql65"と入力します。
5. [ サーバ ] にデータベースサーバのSQL Server名を入力し、[ 次へ ] ボタンをクリックします。
6. データソースウィザードの次のウィンドウで、[ ユーザが入力する SQL Server用のログインIDとパスワードを使う ] を選択状態にします。
7. [ SQL Server に接続して追加の設定オプションのデフォルトの設定を取得する ] を選択状態にします。
8. [ ログインID ] に"sa"と入力します。
9. [ パスワード ] にデータベースの管理者が設定したパスワードを入力し、[ 次へ ] ボタンをクリックします。
10. データソースウィザードの次のウィンドウで、[ デフォルトデータベースを以下のものに変更する ] を選択状態にします。
11. データベースの一覧から"pubs"を選択し、[ 次へ ] ボタンをクリックします。
12. もう1度、[ 次へ ] ボタンをクリックします。
13. データソースウィザードの次のウィンドウで、[ 完了 ] ボタンをクリックします。
14. [ データソースのテスト ] ボタンをクリックし、データソースの設定が正しく行われていることを確認します。
15. OKボタンをクリックして、[ SQL Server ODBC データソーステスト ] を閉じます。
16. OKボタンをクリックして、[ ODBC Microsoft SQL Server セットアップ ] を閉じます。
17. OKボタンをクリックして、[ ODBC データソース アドミニストレータ ] ダイアログボックスを閉じます。



この設定は、クライアントに『SQL Server ODBC ドライバ Ver 3.70』がインストールされている場合の設定例です。設定の詳細については、ODBCドライバのマニュアルまたはヘルプを参照してください。

### 9.7.2 フォームにコントロールを配置する

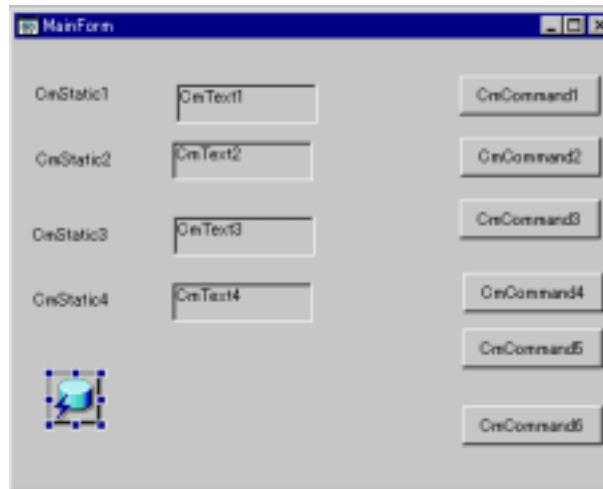
以下のコントロールを図のようにフォームに配置してください。

スタティックテキストコントロールを4つ ( CmStatic1 ~ CmStatic4 )

テキストボックスコントロールを4つ ( CmText1 ~ CmText4 )

コマンドボタンコントロールを6つ ( CmCommand1 ~ CmCommand6 )

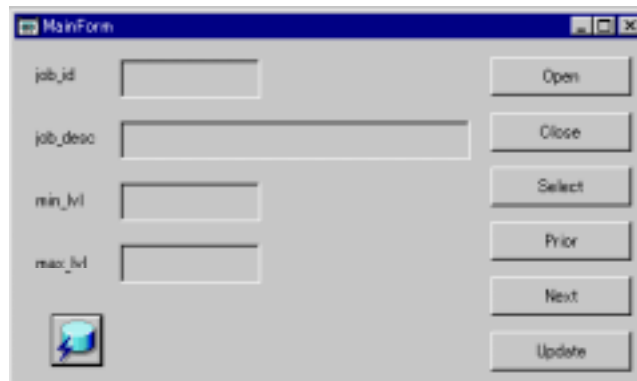
DBアクセスコントロールを1つ (CmDb1)



次に、コントロールのプロパティを次の図と同じになるよう変更してください。  
各スタティックテキストコントロールのプロパティ設定ダイアログボックスを表示し、[スタティックテキスト]タブの[キャプション]を変更します。

各テキストボックスコントロールのプロパティ設定ダイアログボックスを表示し、[テキストボックス]タブの[キャプション]の文字列を削除します。

各コマンドボタンコントロールのプロパティ設定ダイアログボックスを表示し、[コマンドボタン]タブの[キャプション]を変更します。



最後に、DBアクセスコントロール(CmDb1)のプロパティ設定ダイアログボックスを表示し、以下の手順でプロパティを設定します。

1. [抽出対象] から [テーブル/ビュー] を選択します。
2. [DBアクセス] タブで [データベース接続] ボタンをクリックします。  
[データソースの選択] ダイアログボックスが表示されます。
3. [コンピュータデータソース] タブで、「[クライアント環境\(ODBC\)を設定する](#)( p277)」で作成したデータソース"sql65"を選択します。



4. OKボタンをクリックします。  
ログインIDとパスワードの問い合わせがあります。
5. ログインID"sa"、パスワードを入力し、OKボタンをクリックします。  
[ DBアクセス ] タブの [ 抽出対象 ] ボタンが有効になります。
6. [ 抽出対象 ] ボタンをクリックします。  
[ 抽出対象の設定 ] ダイアログボックスが表示されます。
7. 抽出対象のテーブル"dbo.jobs"を選択します。
8. OKボタンをクリックします。  
[ DBアクセス ] タブの [ 抽出情報 ] ボタンが有効になります。
9. [ 抽出情報 ] ボタンをクリックします。  
[ 抽出情報の設定 ] ダイアログボックスが表示されます。
10. [ フィールド名 ] リストに、抽出対象テーブル"dbo.jobs"のフィールドが表示されていることを確認します。
11. [ >> ] ボタンをクリックし、すべてのフィールドを選択します。



12. OKボタンをクリックします。

- [ DBアクセス ] タブの [ その他 ] ボタンが有効になります。
13. [ その他 ] ボタンをクリックします。  
[ その他の設定 ] ダイアログボックスが表示されます。
  14. [ レコードの属性設定 ] から [ レコード更新削除時のキー指定 ] を選択します。
  15. [ 抽出フィールド名リスト ] の "job\_id" をダブルクリックします。  
"job\_id" の左側にビットマップが表示されます。
  16. [ カーソルタイプ ] から [ DYNAMIC ] を選択します。
  17. [ 読み込み専用 ] のチェックをはずします。
  18. [ カーソルの同時実行制御 ] から "OPT\_VALUES" を選択します。



19. OKボタンをクリックします。  
[ その他の設定 ] ダイアログボックスが閉じます。
20. OKボタンをクリックします。



[ レコードの属性設定 ] の "レコード更新削除時のキー指定" は、レコードを一意に識別するためのキーフィールドを設定することです。複数のフィールドでキーを構成する場合は、キーにするすべてのフィールドを選択してください。

[ カーソルタイプ ] は、初期値として "FORWARD\_ONLY" が設定されています。レコードをスクロール読み込み（前方向への読み込み）する場合は、"FORWARD\_ONLY" 以外の値を選択してください。

[ カーソルの同時実行制御 ] は、初期値として "READONLY" が設定されています。カレントレコード（直前に読み込んだレコード）を更新または削除する場合は、"READONLY" 以外の値を選択してください。

各ダイアログボックス中の設定項目に関する詳細は、『リファレンス』を参照してください。



注意

データベースへの接続方法や選択できる項目は、データベースの種類やODBCドライバの種類により異なります。詳細は、各データベースおよびODBCドライバのマニュアルを参照してください。

### 9.7.3 コントロールの手続きを記述する

DBアクセスコントロールを使ってデータベースへアクセスするために、以下の手続きを記述します。

- WORKING-STORAGE: 現在の状態を管理するための変数を宣言します。
- CmCommand1-Click: データベースと接続します。
- CmCommand2-Click: データベースとの接続を解除します。
- CmCommand3-Click: カーソルをオープンします。
- CmCommand4-Click: 現直前のレコードを読み込みます。
- CmCommand5-Click: 次のレコード読み込みます。
- CmCommand6-Click: カレントのレコードを更新します。

#### WORKING-STORAGE

- \* OpenFlagの値でステータスを管理します。
  - \* 0: データベースに接続していません。
  - \* 1: データベースに接続中です。
  - \* 2: 結果セット（カーソル）が作成されています。
  - \* 3: 現在のレコード（カレントレコード）が存在します。
- 01 OpenFlag PIC S9(4) COMP-5 VALUE 0 GLOBAL.

#### CmCommand1-Click

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* すでに接続中なら何もしないで復帰します。
 IF OpenFlag > 0 THEN
 EXIT PROGRAM
 END-IF
* データベースに接続します。
 INVOKE CmDb1 "OpenDB" RETURNING ReturnValue
 IF ReturnValue >= 0 THEN
 MOVE 1 TO OpenFlag
 END-IF

```



**CmCommand2-Click**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
* 現在接続されていなければ、何もしないで復帰します。
 IF OpenFlag = 0 THEN
 EXIT PROGRAM
 END-IF
* 接続を解除します。
 INVOKE Cmdb1 "CloseDB"
 MOVE 0 TO OpenFlag
```

**CmCommand3-Click**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* 現在接続されていなければ、何もしないで復帰します。
 IF OpenFlag = 0 THEN
 EXIT PROGRAM
 END-IF
* カーソルをオープン（レコード群の選択を）します。
 INVOKE Cmdb1 "SelectRecords" RETURNING ReturnValue
 IF ReturnValue >= 0 THEN
 MOVE 2 TO OpenFlag
 END-IF
```

**CmCommand4-Click**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* カーソルがオープンされていなければ、
* テキストボックスの値を空白に設定します。
 IF OpenFlag < 2 THEN
 MOVE SPACE TO "Text" OF CmText1
 MOVE SPACE TO "Text" OF CmText2
 MOVE SPACE TO "Text" OF CmText3
 MOVE SPACE TO "Text" OF CmText4
 EXIT PROGRAM
```

( 続く )

( 続き )

```

END-IF
* 直前のレコードを読み込みます。
 INVOKE Cmdb1 "ReadPreviousRecord" RETURNING ReturnValue
* 読み込んだレコードをテキストボックスに表示します。
 IF ReturnValue = 1 THEN
 MOVE "job_id" OF Cmdb1 TO "Text" OF CmText1
 MOVE "job_desc" OF Cmdb1 TO "Text" OF CmText2
 MOVE "min_lvl" OF Cmdb1 TO "Text" OF CmText3
 MOVE "max_lvl" OF Cmdb1 TO "Text" OF CmText4
 MOVE 3 TO OpenFlag
* カレントレコードが存在しなければ、
* テキストボックスの値を空白に設定します。
 ELSE
 MOVE SPACE TO "Text" OF CmText1
 MOVE SPACE TO "Text" OF CmText2
 MOVE SPACE TO "Text" OF CmText3
 MOVE SPACE TO "Text" OF CmText4
 MOVE 2 TO OpenFlag
 END-IF

```



フィールドの設定や参照は、「"job\_id" OF Cmdb1」のように記述します。このとき、フィールド名の大文字と小文字は区別されます。



実行時にフィールドの設定や参照ができるのは、データベースを接続しているあいだけです。  
フィールドにNULL値を設定することはできません。レコードを更新または追加する場合は、すべての抽出フィールドに値を設定してください。

#### CmCommand5-Click

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ReturnValue PIC S9(9) COMP-5.
PROCEDURE DIVISION.
* カーソルがオープンされていなければ、
* テキストボックスの値を空白に設定します。
 IF OpenFlag < 2 THEN
 MOVE SPACE TO "Text" OF CmText1

```

( 続く )

( 続き )

```

 MOVE SPACE TO "Text" OF CmText2
 MOVE SPACE TO "Text" OF CmText3
 MOVE SPACE TO "Text" OF CmText4
 EXIT PROGRAM
 END-IF
* 次のレコードを読み込みます。
 INVOKE Cmdb1 "ReadNextRecord" RETURNING ReturnValue
* 読み込んだレコードをテキストボックスに表示します。
 IF ReturnValue = 1 THEN
 MOVE "job_id" OF Cmdb1 TO "Text" OF CmText1
 MOVE "job_desc" OF Cmdb1 TO "Text" OF CmText2
 MOVE "min_lvl" OF Cmdb1 TO "Text" OF CmText3
 MOVE "max_lvl" OF Cmdb1 TO "Text" OF CmText4
 MOVE 3 TO OpenFlag
* カレントレコードが存在しなければ、
* テキストボックスの値を空白に設定します。
 ELSE
 MOVE SPACE TO "Text" OF CmText1
 MOVE SPACE TO "Text" OF CmText2
 MOVE SPACE TO "Text" OF CmText3
 MOVE SPACE TO "Text" OF CmText4
 MOVE 2 TO OpenFlag
 END-IF

```

**CmCommand6-Click**

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WorkLvl PIC 9(4) COMP-5.
PROCEDURE DIVISION.
* カレントレコードが存在しなければ、
* 何もしないで復帰します。
 IF OpenFlag < 3 THEN
 EXIT PROGRAM
 END-IF
* 書き換える値を各フィールドに設定します。
* job_idはキーフィールドに設定されているため、
* 書き換えの対象にはなりません。
 MOVE "Text" OF CmText2 TO "job_desc" OF Cmdb1
 MOVE "Text" OF CmText3 TO WorkLvl
 MOVE WorkLvl TO "min_lvl" OF Cmdb1
 MOVE "Text" OF CmText4 TO WorkLvl

```

( 続く )

( 続き )

```
MOVE WorkLvl TO "max_lvl" OF Cmdb1
* カレントレコードを書き換えます。
INVOKE Cmdb1 "RewriteCurRecord"
```



---

カレントレコード(直前に読み込んだレコード)の更新は、RewriteCurRecordメソッドを使用してください。ただし、使用中のデータベースに対応するODBCドライバが、カレントレコードの更新をサポートしていない場合は、RewriteRecordメソッドを使用してください。RewriteRecordメソッドは、キーフィールドに設定されている値をレコードの検索条件として、レコードを更新します。

---

## 9.8 ADOデータソースコントロールを利用してデータベースと連携する

本節では、ADOデータソースコントロールを使って、データベースと連携するアプリケーションの作成方法について説明します。

ADOデータソースコントロールは、ADO(Microsoft ActiveX Data Objects)を使って、データベースと連携するためのコントロールです。

ADOデータソースコントロールを利用するためには、ADOに関する基本的な知識が必要です。

本節では、ADOとして以下のコントロールを利用して、サンプルプログラムを作成します。

### エディットコントロール

『Microsoft DataGrid Control 6.0 (OLEDB) (以降、DataGridコントロールと呼びます)』

エディットコントロールを利用してデータベースと連携する方法では、『Microsoft Access 2000 (以降、Accessと略します)』のサンプルデータベース"Northwind.mdb"の社員テーブルの中から、「社員コード」、「氏名」および「内線」フィールドのデータをエディットコントロール上に表示するサンプルプログラムを作成します。

また、DataGridコントロールを利用するサンプルプログラムでは、以下の2つの方法でデータベースと連携する方法を説明します。

設計時に連携環境を設定して利用する方法

実行時に連携環境を設定して利用する方法

1つめの方法では、SQL Serverと連携するアプリケーションを作成します。2つめの方法では、MSDE(Microsoft Data Engine)データベースと連携するアプリケーションを作成します。



ワンポイント

ADOの詳細については、『MSDN』の「プラットフォームSDK - データアクセスサービス」に掲載されている「Microsoft ActiveX Data Objects(ADO)」を参照してください。

DataGridコントロールは、『Microsoft Visual Basic Version 6.0以降』や『Microsoft Office 2000 Developer』などに添付されています。エディットコントロールを利用するサンプルプログラムを作成または実行するには、あらかじめ、Accessおよびサンプルデータベースをインストールしておく必要があります。

DataGridコントロールを利用して、設計時に連携環境を設定するサンプルプログラムを作成および実行するには、あらかじめ、SQL Serverを利用するための環境を設定しておく必要があります。環境設定方法の詳細については、SQL Serverマニュアルなどを参照してください。



注意

DataGridコントロールを利用して、実行時に連携環境を設定するサンプルプログラムを作成および実行するには、あらかじめ、以下のような手順でAccessのサンプルデータベースをMSDEデータベースに変換しておく必要があります。変換方法の詳細については、Accessのマニュアルなどを参照してください。

1. Accessおよびサンプルデータベースをインストールします。
2. MSDEをインストールします。
3. MSDEのサービスマネージャでMSSQLServerサービスを開始します。
4. Accessのサンプルデータベース"Northwind.mdb"を開きます。
5. "Northwind.mdb"をアップサイジングウィザードで変換します。

このサンプルプログラムでは、サーバへのログインIDを"sa"に、変換後のデータベース名を"NorthwindSQL"にしています。

### 9.8.1 エディットコントロールと連携する

ADOデータソースコントロールを使って、エディットコントロールとデータベースを連携させる方法について、以下の手順で説明します。

1. フォームにコントロールを配置します。
2. コントロールのプロパティを設定します。
3. コントロールの手続きを記述します。



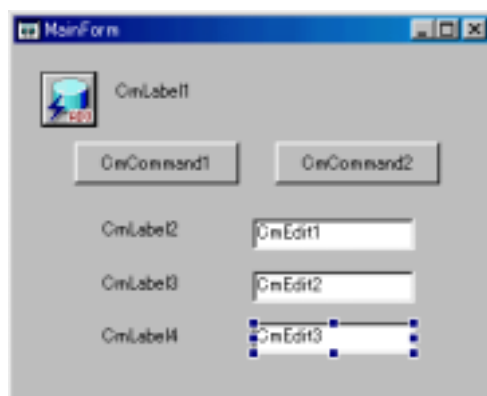
ワンポイント

本節で作成するサンプルプログラムは、"Edit¥ADODataSource.ppj"に格納されています。必要に応じて参照してください。

#### フォームにコントロールを配置する

以下のコントロールを図のようにフォームに配置します。

- ADOデータソースコントロールを1つ (CmADODataSource1)
- ラベルコントロールを4つ (CmLabel1 ~ CmLabel4)
- コマンドボタンコントロールを2つ (CmCommand1 ~ CmCommand2)
- エディットコントロールを3つ (CmEdit1 ~ CmEdit3)



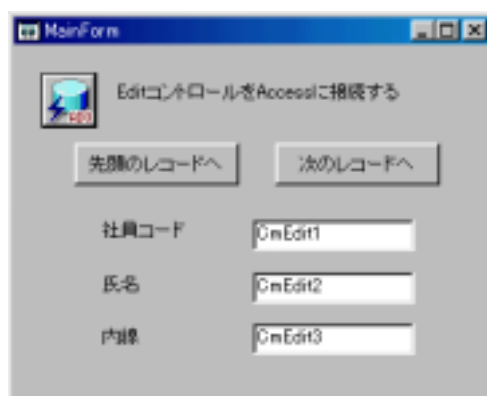
エディットコントロールがツールバーにない場合、フォームにコントロールを配置する前に、以下の手順で、コントロールをツールボックスに追加してください。

1. [ ツール ] メニューの[ カスタムコントロール ] コマンドを選択します。
2. [ カスタムコントロール ] ダイアログボックスの[ 利用可能なコントロール ] 一覧中の [ FUJITSU PowerCOBOL Light Weight Controls 4.4 ] をチェック状態にします。
3. ツールボックスに、エディットコントロールのビットマップが追加されたことを確認します。

### コントロールのプロパティを設定する

まず、コントロールのプロパティを次の図と同じになるよう変更します。

1. プロジェクトウィンドウのデザインツリーウィンドウからラベルコントロールを1つずつ選択し、プロパティリストウィンドウ上の "Caption" に、それぞれの文字列を設定します。
2. 各コマンドボタンコントロールのプロパティ設定ダイアログボックスを開き、[ コマンドボタン ] タブの [ キャプション ] を変更します。



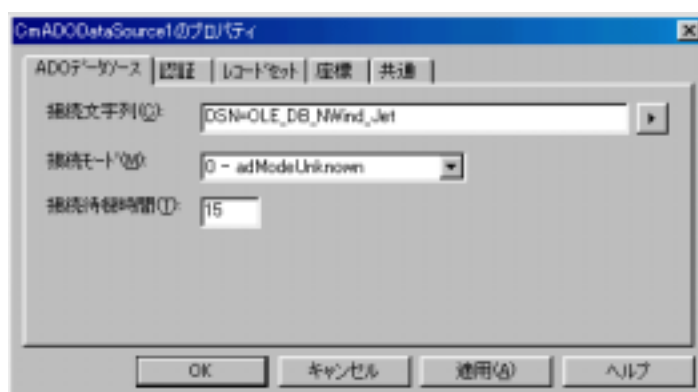


ラベルコントロールには、コントロール固有のプロパティページは用意されていません。したがって、ラベルコントロールに表示する文字列や文字の配置（Alignmentプロパティ）などはプロパティリストウィンドウを使って設定します。

また、エディットコントロールについても同様に、プロパティリストウィンドウで設定します。

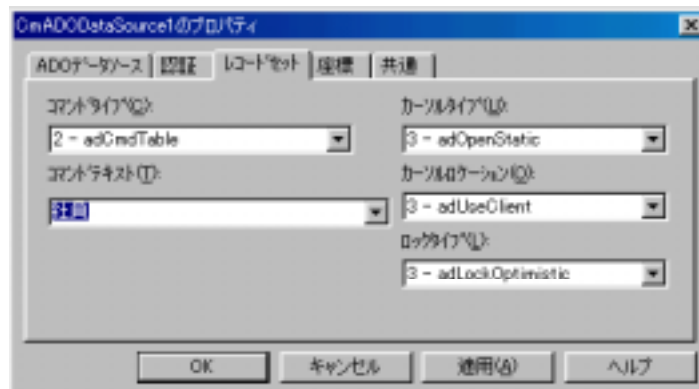
次に、ADOデータソースコントロール(CmADODataSource1)のプロパティ設定ダイアログボックスを開き、以下の手順でプロパティを設定します。

1. [ ADOデータソース ] タブで [ 接続文字列 ] の右側にあるボタンをクリックします。  
3つのサブメニューが表示されます。
2. [ ODBCデータソース名を指定 ] を選択します。  
[ ODBCデータソース名の選択 ] ダイアログボックスが表示されます。
3. ODBCデータソース名の一覧からOLE\_DB\_NWind\_Jetを選択します。（一覧にOLE\_DB\_NWind\_Jetがない場合は、後述のワンポイントを参照してください。）
4. OKボタンをクリックします。



5. [ レコードセット ] タブを開きます。
6. [ コマンドタイプ ] から "2 - adCmdTable" を選択します。
7. [ 適用 ] ボタンをクリックします。  
[ コマンドテキスト ] がコンボボックスに変わり、連携するテーブルの選択ができるようになります。
8. [ コマンドテキスト ] から "社員" を選択します。

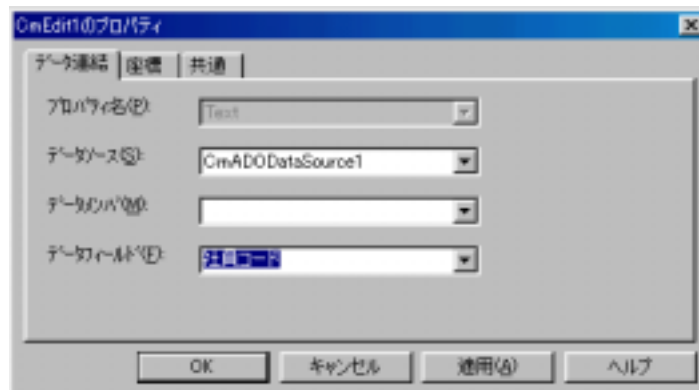




9. OKボタンをクリックします。

次に、以下の手順で、それぞれのエディットコントロール (CmEdit1 ~ CmEdit3) のデータソースとして、ADOデータソースコントロール (CmADODataSource1) を設定します。

1. エディットコントロールのプロパティ設定ダイアログボックスを開きます。
2. [ データ連結 ] タブの [ データソース ] から、"CmADODataSource1" を選択します。
3. [ 適用 ] ボタンをクリックします。
4. [ データフィールド ] から、エディットコントロールに設定するフィールド名 (CmEdit1の場合は"社員コード"、CmEdit2の場合は"氏名"、CmEdit3の場合は"内線") を選択します。



5. OKボタンをクリックします。

最後に、以下の手順で、それぞれのエディットコントロール (CmEdit1 ~ CmEdit3) の文字列を削除します。

1. プロジェクトウィンドウのデザインツリーウィンドウからエディットコントロールを選択します。
2. プロパティリストウィンドウ上の "Text" に設定されている文字列を削除します。



ODBCデータソース名の選択で、一覧にOLE\_DB\_NWind\_Jetがない場合は、一覧の右側にある [ 新規作成 ] ボタンをクリックし、以下の手順で一覧に追加します。

1. データソースの型として、ユーザデータソースを選択します。
2. [ 次へ ] ボタンをクリックします。
3. データソースのドライバとして、Microsoft Access Driver (\*.mdb) を選択します。
4. [ 次へ ] ボタンをクリックします。
5. [ 完了 ] ボタンをクリックします。

[ ODBC Microsoft Access セットアップ ] ダイアログボックスが表示されます。

6. データソース名として、OLE\_DB\_NWind\_Jetを指定します。
7. データベースの [ 選択 ] ボタンをクリックして、Accessのサンプルデータベース "Northwind.mdb"を設定します。
8. OKボタンをクリックします。

すべてのコントロールのプロパティを設定したあと、フォームをプレビューして、エディットコントロールにAccessのデータが表示されれば、データベースと正しく連携できていることが確認できます。

プロパティの設定方法は、利用するデータベースやADOの種類により異なります。詳細は、各データベースおよびADOのマニュアルを参照してください。

---

## コントロールの手続きを記述する

ADOデータソースコントロールを使って連携しているデータベースの、カレントレコードを変更することにより、エディットコントロールに表示される文字列を変更します。

カレントレコードの変更は、以下のイベント手続きに記述します。

CmCommand1-Click: カレントレコードの位置を先頭に移動します。

CmCommand2-Click: カレントレコードの位置を次のレコードに移動します。

### CmCommand1-Click

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

PROCEDURE DIVISION.

- \* カレントレコードの位置を先頭に移動します。

INVOKE "Recordset" OF CmADODataSource1 "MoveFirst"

**CmCommand2-Click**

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
```

- \* カレントレコードの位置を次のレコードに移動します。  
INVOKE "Recordset" OF CmADODataSource1 "MoveNext"



"MoveFirst"および"MoveNext"は、ADOの"Recordset"オブジェクトのメソッドです。その他にも、"Recordset"オブジェクトのメソッドやプロパティを利用して、レコードに対する操作ができます。

たとえば、カレントレコードのフィールド"ProductName"の値を参照する場合は、以下のような手続きを記述します。

```
MOVE "Value" OF "Item"("ProductName") OF "Fields" OF
 "Recordset" OF CmADODataSource1 TO WK-PRODUCT-NAME
```

また、フィールド"UnitPrice"の値を更新する場合は、以下のような手続きを記述します。

```
MOVE 1000 TO "Value" OF "Item"("UnitPrice") OF
 "Fields" OF "Recordset" OF CmADODataSource1
INVOKE "Recordset" OF CmADODataSource1 "Update"
```

さらに、新規にレコードを追加する場合は、以下のような手続きを記述します。

```
INVOKE "Recordset" OF CmADODataSource1 "AddNew"
MOVE WK-ID TO "Value" OF "Item"("EmployeeID") OF
 "Fields" OF "Recordset" OF CmADODataSource1
MOVE WK-FNAME TO "Value" OF "Item"("FirstName") OF
 "Fields" OF "Recordset" OF CmADODataSource1
MOVE WK-LNAME TO "Value" OF "Item"("LastName") OF
 "Fields" OF "Recordset" OF CmADODataSource1
INVOKE "Recordset" OF CmADODataSource1 "Update"
```

## 9.8.2 DataGridコントロールとの連携環境を設計時に設定する

ADOデータソースコントロールを使って、DataGridコントロールとの連携環境を設計時に設定する方法について、以下の手順で説明します。

1. フォームにコントロールを配置します。
2. コントロールのプロパティを設定します。
3. コントロールの手続きを記述します。



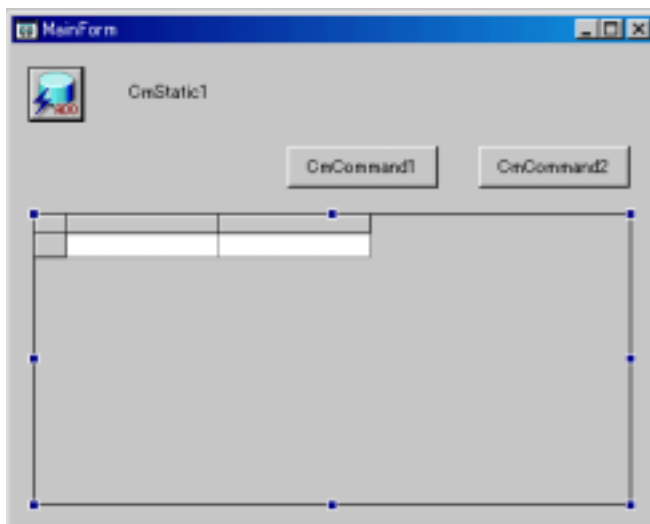
本節で作成するサンプルプログラムは、  
"ADODataSource¥ADODataSource1.ppj" に格納されています。必要に応じて参照してください。

ただし、このサンプルプログラムにはデータベースへの接続文字列が設定されていません。サンプルプログラムを実行する場合には、本節の「コントロールのプロパティを設定する」を参照して、あらかじめ接続文字列を設定してください。

### フォームにコントロールを配置する

以下のコントロールを図のようにフォームに配置します。

- ADOデータソースコントロールを1つ (CmADODataSource1)
- スタティックテキストコントロールを1つ (CmStatic1)
- コマンドボタンコントロールを2つ (CmCommand1 ~ CmCommand2)
- DataGridコントロールを1つ (DataGrid1)





DataGridコントロールがツールバーにない場合、フォームにコントロールを配置する前に、以下の手順で、コントロールをツールボックスに追加してください。

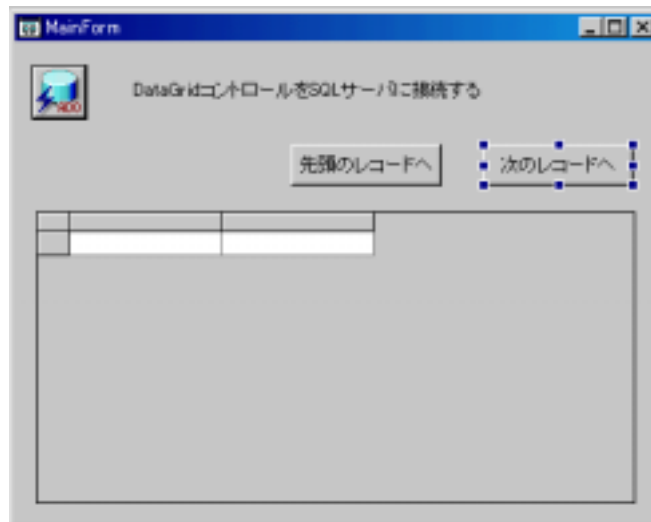
1. [ ツール ] メニューの [ カスタムコントロール ] コマンドを選択します。
2. [ カスタムコントロール ] ダイアログボックスの [ 利用可能なコントロール ] 一覧中の [ Microsoft DataGrid Control (OLEDB) ] をチェック状態にします。
3. ツールボックスに、エディットコントロールのビットマップが追加されたことを確認します。

ただし、利用しているシステムやシステムに登録されているコントロールの種類およびバージョンにより、[ 利用可能なコントロール ] の一覧の見えかたが異なる場合があります。

### コントロールのプロパティを設定する

まず、コントロールのプロパティを次の図と同じになるよう変更します。

1. スタティックテキストコントロールのプロパティ設定ダイアログボックスを開き、[ スタティックテキスト ] タブの [ キャプション ] を変更します。
2. 各コマンドボタンコントロールのプロパティ設定ダイアログボックスを開き、[ コマンドボタン ] タブの [ キャプション ] を変更します。



次に、ADOデータソースコントロール(CmADODataSource1)のプロパティ設定ダイアログボックスを開き、以下の手順でプロパティを設定します。

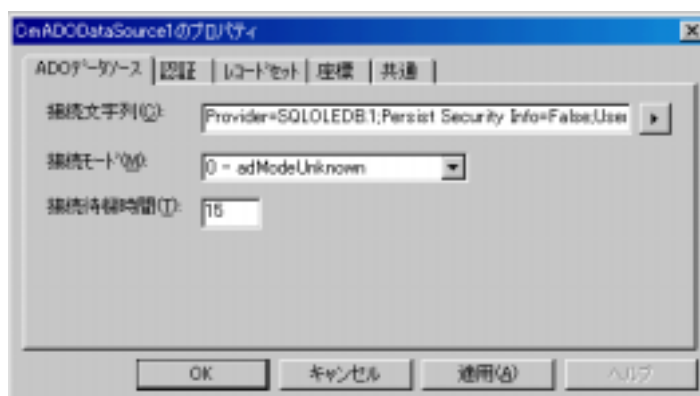
1. [ ADOデータソース ] タブで [ 接続文字列 ] の右側にあるボタンをクリックします。  
3つのサブメニューが表示されます。
2. [ 作成(U)... ] を選択します。

データリンクのプロパティ設定ダイアログボックスが表示されます。

3. [プロバイダ] タブで、"Microsoft OLE DB Provider for SQL Server" を選択します。

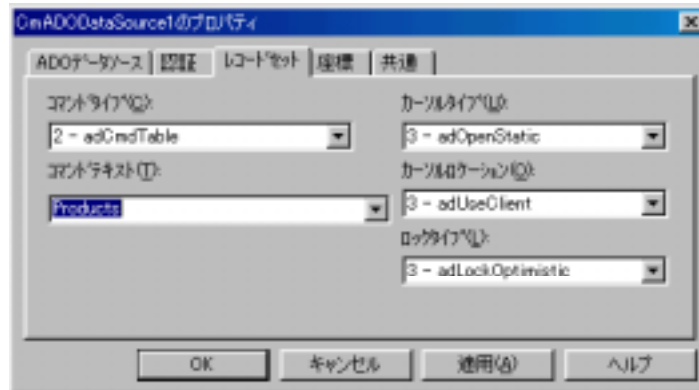


4. [接続] タブを開きます。
5. SQL Serverに接続するために、サーバ名とログオンに必要な情報を入力します。
6. サーバ上のデータベースとして、"Northwind"を選択します。
7. OKボタンをクリックします。  
[ADOデータソース] タブの [接続文字列] が設定されます。



8. [認証] タブを開きます。
9. [ユーザ名] および [パスワード] に、ログオンに必要な情報を入力します。

10. [レコードセット]タブを開きます。
11. [コマンドタイプ]から"2 - adCmdTable"を選択します。
12. [コマンドテキスト]から"Products"を選択します。



13. OKボタンをクリックします。

最後に、以下の手順で、ADOデータソースコントロール(CmADODataSource1)をDataGridコントロール(DataGrid1)のデータソースとして設定します。

1. プロジェクトウィンドウをクリックし、デザインビューを表示します。
2. デザインツリーウィンドウで、"DataGrid1[ DataGrid ]"を選択します。
3. プロパティリストウィンドウの"DataSource"の値を"CmADODataSource1"にします。



すべてのコントロールのプロパティを設定したあと、フォームをプレビューして、DataGridコントロール上にSQL Serverのデータが表示されれば、データベースと正しく連携できていることが確認できます。

### コントロールの手続きを記述する

イベント手続きには、「[エディットコントロールと連携する](#) ( p288 )」で記述した手続きと同じ手続きを記述します。

CmCommand1-Click: カレントレコードの位置を先頭に移動します。

CmCommand2-Click: カレントレコードの位置を次のレコードに移動します。



Accessのデータベースと連携するサンプルプログラムは、"ADODataSource¥ADODataSource2.ppj"に格納されています。必要に応じて参照してください。

### 9.8.3 DataGridコントロールとの連携環境を実行時に設定する

ADOデータソースコントロールを使って、DataGridコントロールとの連携環境を実行時に設定する方法について、以下の手順で説明します。

1. フォームにコントロールを配置します。
2. コントロールのプロパティを設定します。
3. コントロールの手続きを記述します。



本節で作成するサンプルプログラムは、  
"ADODataSource¥ADODataSource3.ppj" に格納されています。必要に応じて参照してください。

#### フォームにコントロールを配置する

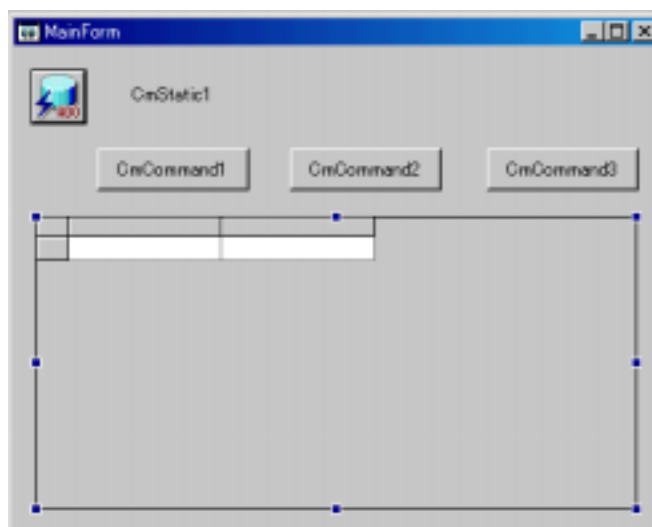
以下のコントロールを図のようにフォームに配置します。

ADOデータソースコントロールを1つ (CmADODataSource1)

スタティックテキストコントロールを1つ (CmStatic1)

コマンドボタンコントロールを3つ (CmCommand1 ~ CmCommand3)

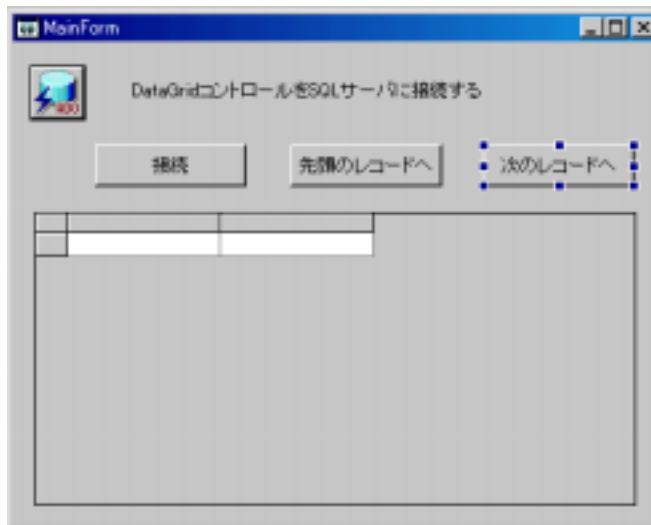
DataGridコントロールを1つ (DataGrid1)



#### コントロールのプロパティを設定する

スタティックテキストコントロールおよび各コマンドボタンコントロールのプロパティ設定ダイアログボックス開き、次の図と同じになるよう [ キャプション ] を変更します。ADOデータソースコントロールおよびDataGridコントロールのプロパティを設定する必要はありません。





### コントロールの手続きを記述する

[ 接続 ] ボタン(CmCommand1)がクリックされたら、ADOデータソースコントロールを使って、データベースとDataGridコントロールを連携させるための手続きが実行されます。

その他のボタンのイベント手続きには、「[エディットコントロールと連携する](#) ( p288) 」で記述した手続きと同じ手続きを記述します。

MainForm-REPOSITORY: NetCOBOLの\*COMクラスおよび\*COM-EXCEPTIONクラスを宣言します。

CmCommand1-Click: ADOデータソースコントロールを利用して、DataGridコントロールとデータベースを接続します。

CmCommand2-Click: カレントレコードの位置を先頭のレコードに移動します。

CmCommand3-Click: カレントレコードの位置を次のレコードに移動します。

### MainForm-REPOSITORY

```
CLASS COM AS "*COM"
CLASS EXCEP AS "*COM-EXCEPTION"
```

### CmCommand1-Click

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SRC PIC X(1024).
01 MSG PIC X(1024).
01 WK PIC X(1024).
```

( 続く )

( 続き )

```

01 CNOBJ OBJECT REFERENCE COM.
01 RSTOBJ OBJECT REFERENCE COM.
01 CMDOBJ OBJECT REFERENCE COM.
01 WK-OPTION PIC S9(9) COMP-5 VALUE -1.
01 WK-CURTYPE PIC S9(9) COMP-5 VALUE POW-ADODB-ADOPENSTATIC.
01 WK-LOCKTYPE PIC S9(9) COMP-5 VALUE POW-ADODB-ADLOCKOPTIMISTIC.
01 WK-CNOBJ OBJECT REFERENCE POW-COBJECT.
01 WK-RSTOBJ OBJECT REFERENCE POW-COBJECT.
01 DATASOURCE OBJECT REFERENCE COM.
01 DATAGRID OBJECT REFERENCE COM.
PROCEDURE DIVISION.

* 例外処理
DECLARATIVES.
ERR SECTION.

* 例外を受け取ったらメッセージを表示します。
USE AFTER EXCEPTION EXCEP.
 INVOKE EXCEPTION-OBJECT "GET-SOURCE" RETURNING SRC
 INVOKE EXCEPTION-OBJECT "GET-DESCRIPTION" RETURNING MSG
 INVOKE POW-SELF "DisplayMessage" USING MSG SRC POW-DMICONERROR
 EXIT PROGRAM
.

END DECLARATIVES.

* ADOのConnectionオブジェクトを作成します。
MOVE "ADODB.Connection" TO WK
INVOKE COM "CREATE-OBJECT" USING WK RETURNING CNOBJ

* SQL Server(ここでは、Access + MSDE)にアクセスして、
* "NorthwindSQL"というデータベースをオープンします。
MOVE "Provider=SQLOLEDB.1;User ID=sa;
- "Persist Security Info=False;
- "Initial Catalog=NorthwindSQL" TO WK
INVOKE CNOBJ "OPEN" USING WK

* ADOのCommandオブジェクトを作成します。
MOVE "ADODB.Command" TO WK
INVOKE COM "CREATE-OBJECT" USING WK RETURNING CMDOBJ

* ConnectionオブジェクトをCommandオブジェクトに設定します。
INVOKE CMDOBJ "SETREF-ActiveConnection" USING CNOBJ

* CommandTypeプロパティとCommandTextプロパティを設定します。
MOVE POW-ADODB-ADCMDTEXT TO WK-OPTION
INVOKE CMDOBJ "SET-CommandType" USING WK-OPTION
MOVE "Select * from 商品" TO WK
INVOKE CMDOBJ "SET-CommandText" USING WK

* ADOのRecordsetオブジェクトを作成します。

```

( 続く )

( 続き )

```

MOVE "ADODB.Recordset" TO WK
INVOKE COM "CREATE-OBJECT" USING WK RETURNING RSTOBJ
* Recordsetオブジェクトのプロパティを設定します。
INVOKE RSTOBJ "SET-CursorType" USING WK-CURTYPE
INVOKE RSTOBJ "SET-LockType" USING WK-LOCKTYPE
* SourceとしてCommandオブジェクトを設定します。
INVOKE RSTOBJ "SETREF-Source" USING CMDOBJ
* Recordsetオブジェクトをオープンします。
INVOKE RSTOBJ "Open"
* 作成したRecordsetオブジェクトをPowerCOBOLのオブジェクト形式に
* 変換し、ADOデータソースコントロールのRecordsetプロパティに設定
* します。
CALL "POWERCONVFROMCOM" USING RSTOBJ RETURNING WK-RSTOBJ
INVOKE CmADODataSource1 "SETREF-Recordset" USING WK-RSTOBJ
* ADOデータソースコントロールをDataGridコントロールのデータソース
* に設定します。
INVOKE DataGrid1 "SETREF-DataSource" USING CmADODataSource1

```



この手続きでは、"SETREF-Source"メソッドを使って、RecordsetオブジェクトのSourceプロパティに値を設定しています。ここでは、プロパティのデータ型がCOMオブジェクトの場合に利用する、NetCOBOLのオブジェクト機能を利用しています。NetCOBOLの\*COMクラスを使ったプログラミングについての詳細は、『NetCOBOL 使用手引書』を参照してください。

この手続きでは、"SETREF-Recordset"メソッドを使って、ADOデータソースコントロールのRecordsetプロパティに値を設定しています。Recordsetプロパティは、PowerCOBOLで用意しているオブジェクトではなく、ADOが用意しているオブジェクトを示しています。プロパティの値がPowerCOBOLで用意したオブジェクト以外のものを示している場合、"POWERCONVFROMCOM"を使って、COMオブジェクトをPowerCOBOLで扱えるオブジェクトに変換し、"SETREF-プロパティ名"という形式でメソッドを呼び出すことにより、プロパティの値を設定できます。

"POWERCONVFROMCOM"の記述形式については、「[NetCOBOLの\\*COMクラスを利用してアクセスする](#) ( p198 )」を参照してください。

MSDEを使わずに、Accessのデータベースと連携するサンプルプログラムは、"ADODataSource¥ADODataSource4.ppj"に格納されています。必要に応じて参照してください。



注意

---

"SETREF-プロパティ名"メソッドを使って、PowerCOBOLのコントロールのプロパティに値を設定する場合、プロパティ名の大文字と小文字を区別して記述してください。

ADO データソースコントロールの Recordset プロパティなど、POW-COJECTクラスのオブジェクト参照を示すプロパティには、以下のように、直接オブジェクトを設定することはできません。"SETREF-プロパティ名"メソッドを使って設定してください。

誤: MOVE WK-RSTOBJ TO "Recordset" OF CmADODataSource1

正: INVOKE CmADODataSource1 "SETREF-Recordset" USING WK-RSTOBJ

---

---

## 付録A PowerCOBOLが提供するコントロールとオブジェクト

PowerCOBOLは以下のコントロールとオブジェクトを提供しています。

なお、各コントロールおよびオブジェクトの詳細については、『リファレンス』を参照してください。

また、各コントロールやオブジェクトの使用例については、サンプルプログラムを参照してください。サンプルプログラムの使用方法および概要については、「[サンプルプログラムについて](#) ( p321 )」を参照してください。

### Windows 標準コントロール

#### オプションボタン(OptionButton)コントロール

相互に排他的な選択肢の中から1つだけを選択するためのコントロールです。

#### グループボックス(GroupBox)コントロール

関連する複数のコントロールを囲む矩形 ( 長方形の枠 ) を表示するコントロールです。

#### コマンドボタン(CommandButton)コントロール

特定の動作を実行するための、ボタン形式のコントロールです。

#### コンボボックス(ComboBox)コントロール

スタティックテキストまたはテキストボックスのどちらかと、リストボックスを組み合わせた形式で、複数の選択肢の中から1つだけを選択するためのコントロールです。

#### スクロールバー(ScrollBar)コントロール

データを処理する部分を順次移動するためのコントロールです。

#### スタティックテキスト(StaticText)コントロール

テキスト文字列を表示するためのコントロールです。

#### チェックボックス(CheckBox)コントロール

オプションがオンまたはオフのどちらであるかを選択するためのコントロールです。

#### テキストボックス(TextBox)コントロール

テキスト文字列を入力したり編集したりするためのコントロールです。

#### フレーム(Frame)コントロール

関連する複数のコントロールを囲む矩形 ( 長方形の枠 ) を表示するコントロールです。

#### リストボックス(ListBox)コントロール

選択肢の一覧を表示したり、選択肢の中から1つまたは複数の項目を選択したりするためのコントロールです。

---

## Windows コモンコントロール

### 進行状況インジケータ(ProgressIndicator)コントロール

時間がかかる操作の開始から終了までの進行状況を表示するためのコントロールです。

### スライダ(Slider)コントロール

調節の範囲を示す目盛り(チェックマーク)およびインジケータ(つまみ)をもち、ある値の範囲から1つの値を選択するためのコントロールです。

### タブ(Tab)コントロール

分類見出し(タブ)インタフェースを使用するためのコントロールです。

### ツールバー(Toolbar)コントロール

1つまたは複数のボタンをもつコントロールです。

### ツリービュー(TreeView)コントロール

項目の集合を階層関係にもとづいてツリー状の外観で表示するための、特殊なリストボックスコントロールです。

### リストビュー(ListView)コントロール

イメージ(アイコン)とテキスト(ラベル)を組み合わせた項目の集まりを表示するための、特殊なリストボックスコントロールです。

## 拡張コントロール

### タイマ(Timer)コントロール

タイマ処理を行うためのコントロールです。

### ドライブリスト(DriveList)コントロール

ドライブ一覧を表示するための、特殊なコンボボックスコントロールです。

### ファイルリスト(FileList)コントロール

カレントのファイル一覧を表示するための、特殊なリストボックスコントロールです。

### フォルダリスト(FolderList)コントロール

カレントのフォルダー一覧を表示するための、特殊なリストボックスコントロールです。

## フォーム修飾用コントロール

### 図形(Shape)コントロール

フォーム上に長方形や正方形などの基本的な図形を描画するためのコントロールです。

## ビジネス用途コントロール

### グラフ(Graph)コントロール

数値データをグラフとして表示するためのコントロールです。

### 表(Table)コントロール

表形式でデータを扱うためのコントロールです。

---

## マルチメディア関連コントロール

### アニメーション(Animation)コントロール

ビットマップファイルを切り替えて表示することにより、簡単なアニメーションとして再生するためのコントロールです。

### イメージ(Image)コントロール

ビットマップなどのイメージを表示するためのコントロールです。

### MCI(Media Control Interface)コントロール

Windows のMCI(Media Control Interface)によって各種メディアを制御するためのコントロールです。

## データ連携コントロール

### DBアクセス(DBAccess)コントロール

ODBC(Open DataBase Connectivity)インタフェースをサポートするデータベースとの連携を支援するためのコントロールです。

### DDE(Dynamic Data Exchange)コントロール

Windows がサポートするDDE(Dynamic Data Exchange)のクライアント機能をもつためのコントロールです。

### Excel連携(ExcelConnection)コントロール

Microsoft Excelのシートとのデータ交換をするためのコントロールです。

## ADOデータ連携コントロール

### ADOデータソース(ADODataSource)コントロール

ADO(Microsoft ActiveX Data Objects)を使ってデータベースとの連携を支援するためのコントロールです。

## 軽量コントロール

### エディット(Edit)コントロール

テキスト文字列を入力したり編集したりするためのコントロールです。テキストボックスコントロールの機能が軽量化されています。また、ADO(Microsoft ActiveX Data Objects)として利用することもできます。

### ラベル(Label)コントロール

テキスト文字列を表示するためのコントロールです。スタティックテキストコントロールの機能が軽量化されています。

## その他のコントロール

### 印刷(Print)コントロール

PowerCOBOLで作成したアプリケーションを実行したとき、フォームやフォーム上に配置されているコントロールを印刷するためのコントロールです。

## オブジェクト

### カラム(Column)オブジェクト

列の見出し用のテキスト(HeaderText)とリストビュー(ListView)コントロールを関連づけるために利用するオブジェクトです。

### テキスト属性(RenderText)オブジェクト

コントロールに表示するテキスト文字列の属性を示すオブジェクトです。

### データ連結(DataBinding)オブジェクト

データ連結が可能なコントロールの、データ連結可能なプロパティを示すオブジェクトです。

### ノード(Node)オブジェクト

ツリービュー(TreeView)コントロールに表示される1つの項目を示すオブジェクトです。

### 表-セル(TableCell)オブジェクト

表(Table)コントロールの、1つの項目(セル)の属性を示すオブジェクトです。

### 表-列(TableColumn)オブジェクト

表(Table)コントロールの、1つの列の属性を示すオブジェクトです。

### フォーム(Form)オブジェクト

PowerCOBOLアプリケーションで表示するウィンドウを示すオブジェクトです。

### フォント(Font)オブジェクト

コントロールに表示されるテキスト文字列のフォントを示すオブジェクトです。

### ボタン(Button)オブジェクト

ツールバー(Toolbar)コントロールの1つのボタンを示すオブジェクトです。

### メニュー(Menu)オブジェクト

フォーム上で使用するメニューを示すオブジェクトです。

### メニューアイテム(MenuItem)オブジェクト

フォーム中のメニューに含まれる1つのメニュー項目を示すオブジェクトです。

### リストアイテム(ListItem)オブジェクト

リストビュー(ListView)コントロールの1つの項目を示すオブジェクトです。

## コレクションオブジェクト

### Controlsコレクションオブジェクト

フォーム上に配置されたコントロールの集合体を示すコレクションオブジェクトです。

### DataBindingsコレクションオブジェクト

データ連結可能なコントロールでの、データ連結(DataBinding)オブジェクトの集合体を示すコレクションオブジェクトです。



## 付録B V3.0以前のPowerCOBOLをご利用の方へ

これまで、PowerCOBOLは、COBOLの世界でよりWindows らしいプログラミングを可能とすることを目標としてきました。V1.0が発売され、GUIをもったアプリケーションを作成できるようになって以来、V2.0で大幅なアイテムの追加、V3.0で32ビット完全対応と機能強化に努めてきました。

V4.0では、Windows の提供するCOM(Component Object Model)の世界で動作可能な、よりWindows らしいアプリケーションを開発できるようになりました。このようにV4.0以降では、よりWindows の世界を活用して動作できるアプリケーションを開発できるようにするため、いくつかV3.0以前と変更されている部分があります。本付録では、V4.0以降の主要な変更点、およびV3.0以前の資産を移行する方法について説明します。

### B.1 用語

これまで、PowerCOBOL固有で使用していた用語を、Windows で一般的に使われる用語に変更しています。以下に代表的なものを示します。

| V3.0以前        | V4.0以降                                       |
|---------------|----------------------------------------------|
| アイテム          | コントロール                                       |
| 属性            | プロパティ                                        |
| シート           | フォーム(Form)オブジェクト                             |
| 文字列表示アイテム     | スタティックテキスト(StaticText)コントロール                 |
| 文字列入力アイテム     | テキストボックス(TextBox)コントロール                      |
| プッシュボタンアイテム   | コマンドボタン(CommandButton)コントロール                 |
| ラジオボタンアイテム    | オプションボタン(OptionButton)コントロール                 |
| チェックボタンアイテム   | チェックボックス(CheckBox)コントロール                     |
| グループボックスアイテム  | グループボックス(GroupBox)コントロールまたはフレーム(Frame)コントロール |
| リストボックスアイテム   | リストボックス(ListBox)コントロール                       |
| コンボボックスアイテム   | コンボボックス(ComboBox)コントロール                      |
| 水平スクロールバーアイテム | スクロールバー(ScrollBar)コントロール                     |
| 垂直スクロールバーアイテム |                                              |
| イメージアイテム      | イメージ(Image)コントロール                            |
| タイマアイテム       | タイマ(Timer)コントロール                             |

( 続く )

( 続き )

| V3.0以前          | V4.0以降                                |
|-----------------|---------------------------------------|
| ドライブルISTアイテム    | ドライブルIST(DriveList)コントロール             |
| ディレクトリリストアイテム   | フォルダリスト(FolderList)コントロール             |
| ファイルリストアイテム     | ファイルリスト(FileList)コントロール               |
| 枠アイテム           | 図形(Shape)コントロール                       |
| PICTURE編集アイテム   | なし(テキストボックスコントロールに吸収)                 |
| サウンドアイテム        | なし(MCIコントロールに吸収)                      |
| 表アイテム           | 表(Table)コントロール                        |
| グラフアイテム         | グラフ(Graph)コントロール                      |
| DDEアイテム         | DDE(Dynamic Data Exchange)コントロール      |
| OLEアイテム         | なし                                    |
| ファンクションキーアイテム   | なし(コマンドボタンコントロールに吸収)                  |
| 簡易アニメーションアイテム   | アニメーション(Animation)コントロール              |
| MCIアイテム         | MCI(Media Control Interface)コントロール    |
| 日付表示アイテム        | なし(スタティックテキストコントロールに吸収)               |
| メタファイル再生アイテム    | なし(イメージコントロールに吸収)                     |
| 拡張イメージアイテム      | なし(イメージコントロールに吸収)                     |
| ビットマップ表示ボタンアイテム | なし(コマンドボタンコントロールに吸収)                  |
| メニューアイテム        | メニュー(Menu)およびメニューアイテム(MenuItem)オブジェクト |
| DBアクセスアイテム      | DBアクセス(DBAccess)コントロール                |
| 選択ボックスアイテム      | なし(コンボボックスコントロールに吸収)                  |
| EXCEL連携アイテム     | Excel連携(ExcelConnection)コントロール        |
| 印刷アイテム          | 印刷(Print)コントロール                       |
| MeFt/graphアイテム  | なし                                    |

## B.2 操作性

操作性を、できるかぎり最新のWindows 用アプリケーションに合わせました。表示形式だけでなく、たとえば、何かを操作する場合、ウィンドウの要素を選択し、マウスの右クリックの操作でポップアップメニューから項目を選択することにより、ほとんどの作業ができるようになっています。

## B.3 COMとの関係

V4.0以降のPowerCOBOLは、COMの設計思想およびインプリメント(含むインタフェース)に準拠しています。したがって、PowerCOBOLのコントロールだけでなく、一般に流通するActiveXコントロールを使用してアプリケーションを開発することが可能です。さらに独自のActiveXコントロールを開発し、それを使用したアプリケーションを作成することも可能です。

また、オートメーションサーバアプリケーション(たとえば、Microsoft Excel)をVBAで操作するのと同様に、PowerCOBOLからCOBOLプログラムを使って操作することもできます。逆に、VBAから操作することができるオートメーションサーバアプリケーションを、PowerCOBOLで開発することもできます。

しかし、上記の機能を実現するためにV3.0以前とは異なった実行論理を採用しているので、使用メモリ量、CPU性能など、COM準拠の分だけ処理に時間がかかることが予想されます。したがって、V3.0以前と比べ、使用するマシンなど十分余裕をもって、システムを設計されることをお勧めします。

## B.4 アイテム属性名

V3.0以前は、アイテムの属性名として、POW-TEXTのようにPowerCOBOL固有の名前を使用していました。このため、COBOLの予約語と重なることもなく、以下のように記述することが可能でした。

```
MOVE "OK" TO POW-TEXT OF LABEL1.
```

しかし、V4.0以降、一般のActiveXコントロールも利用可能となったことから、プロパティ(属性)がどのような名前となるかは、そのActiveXコントロールの作成者に依存してしまうことになります。また、ActiveXコントロールでよく使用されるプロパティ名の命名規約もあります。このため、V4.0以降では、上記のような書きかたに加えて、以下のような書きかたをサポートしています。

```
MOVE "OK" TO "Caption" OF LABEL1.
```

以下に代表的なV3.0以前の属性名とV4.0以降のプロパティの対応を示します。

| 属性名           | プロパティの名前(V4.0以降)   |
|---------------|--------------------|
| POW-TEXT      | "Caption"          |
| POW-BACKCOLOR | "BackColor"        |
| POW-FONTSIZE  | "Size" OF "Font"   |
| POW-ITALIC    | "Italic" OF "Font" |

V4.0以降の新しいプロパティ名と、V3.0以前の属性名の対応については、『リファレンス』の各プロパティの説明を参照してください。

## B.5 メソッド呼び出し

メソッドの呼び出し方法は、V3.0以前は、以下のようなCALL文によるものでした。

```
CALL ADDSTRING OF LIST1 USING "abcdef".
```

V4.0以降では、プロパティ名と同様、COBOLの予約語と重ならないように、オ

プロジェクト指向COBOLの言語仕様として新しく追加された、`INVOKE`文による呼び出しができるようになっています。

```
INVOKE LIST1 "AddString" USING "abcdef".
```

V4.0以降の新しいメソッド名と、V3.0以前のメソッド名の対応については、『リファレンス』の各メソッドの説明を参照してください。

## B.6 資産の移行方法

V4.0以降のPowerCOBOLでは、V3.0以前のPowerCOBOLで作成した資産を、プロジェクト単位で変換し、移行することができます。ただし、一部変換できない場合があります。詳細については、「[資産移行時の留意事項](#) ( p311) 」および「[非互換項目](#) ( p318) 」を参照してください。

また、変換はV3.0以前のPowerCOBOLで正しくメイクでき、実行できるプロジェクトだけを対象にしています。V4.0以降のPowerCOBOLでプロジェクトファイルを開く前に、以前のPowerCOBOLでのメイクでエラーがなく、正しく実行できることを確認してください。

### B.6.1 移行の前準備

---

V3.0以前のPowerCOBOLで作成した資産を用意します。変換に必要な主なファイルは以下のとおりです。

- prjファイル
- winファイル
- prcファイル
- 登録されたCOBOLファイル
- リソースに指定したリソースファイル

これらの資産は、変換後もそのまま残りますが、万一の場合に備えてバックアップをとっておくことをお勧めします。

また、以下のように、グループアイテムをグループボックスコントロールに変換するか、フレームコントロールに変換するか指定してください。

#### グループアイテムの変換方法の指定

グループアイテムを含む資産を変換する場合、グループアイテムはグループボックスコントロールまたはフレームコントロールに変換されます。どちらのコントロールに変換するかは、以下のコマンドを実行することにより、切り替えることができます。

PowerCOBOLをインストールしたときの初期状態では、グループボックスコントロールに変換されます。

"F5DDSTEV.EXE"は、PowerCOBOLをインストールしたフォルダに格納されています。このコマンドは、Windows のコマンドプロンプト上で実行してください。

#### グループボックスコントロールに変換する場合

```
F5DDSTEV /V3GROUP:GROUP
```

## フレームコントロールに変換する場合

F5DDSTEV /V3GROUP:FRAME

### B.6.2 変換

以下のような手順で、V3.0以前のPowerCOBOLで作成したプロジェクトファイルを開きます。

1. PowerCOBOLの[ファイル]メニューから[既存プロジェクトを開く]コマンドを選択します。
2. [ファイルを開く]ダイアログボックスの[ファイルの種類]で、[V3以前のPowerCOBOLプロジェクト(.prj)]を選択します。
3. 変換対象のプロジェクトファイルを選択します。
4. [開く]ボタンをクリックします。
5. メッセージボックスで変換するかどうかの確認に対し、[はい]ボタンをクリックします。

変換が成功すれば、PowerCOBOLのプロジェクトウィンドウに変換後のプロジェクトが表示されます。このとき、以前のプロジェクト名がモジュール名として採用されます。正しく表示されていれば、最新のバージョンのプロジェクトファイル(.ppj)として保存することができます。



V3.0以前のPowerCOBOLでは、上記の変換に必要なファイルと、翻訳やリンクによって作成されるファイルは、同じフォルダに格納されました。V4.0以降、翻訳やリンクによって作成されるファイルは、サブフォルダに作成されます。作成されるファイルとフォルダについては、「[ビルド時に作成されるファイル](#) ( p120) 」を参照してください。

## B.7 資産移行時の留意事項

V3.0以前に使用していたPowerCOBOLのバージョンにより、以下のような留意事項があります。

### B.7.1 V3.0以前からの留意事項

V3.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、以下のような留意事項があります。

#### シート名またはリソース名が他のシート名やリソース名と重なっている場合

プロジェクトを開くことはできません。

その内容のエラーメッセージが表示された場合、以前のPowerCOBOLを使用して、シート名およびリソース名をプロジェクト内で一意になるよう、名前を変更してから、最新のバージョンのPowerCOBOLで開きなおしてください。

### ターゲットフォルダまたは作業用フォルダと同じ名前のファイルを使用している場合

リソースに指定したリソースファイルなどのファイル名が、ターゲットフォルダや作業用フォルダの名前と同じ場合、移行後にビルドすると、これらのフォルダが作成できないため、ビルドに失敗します。この場合、ファイル名を別の名前に変更してください。

### V4.0以降のPowerCOBOLでコントロール名として許されない名前を、アイテム名として使用している場合

プロジェクトを開くことはできません。  
その内容のエラーメッセージが表示された場合、以前のPowerCOBOLを使用して、アイテム名を変更してから、最新のPowerCOBOLで開きなおしてください。

### メニューアイテムの属性の参照

メニューアイテムの以下の属性名とPOW-ONを比較している場合、変換後、正しく動作しません。

POW-VISIBLE

POW-ENABLE

POW-CHECK

この場合、手続きを以下のように「= POW-ON」から「NOT = POW-FALSE」へ修正してください。

変換前 :IF POW-ENABLE OF MENU1 = POW-ON THEN

変換直後:IF "Enabled" OF MENU1 OF CfMenu1 = POW-ON THEN

修正後 :IF "Enabled" OF MENU1 OF CfMenu1 NOT = POW-FALSE THEN

### メニューアイテムのPOW-TEXT属性と他のアイテムのPOW-TEXTN属性との転記または比較をしている場合

変換後、ビルドエラーとなります。

この場合、手続きを以下のように「= POW-TEXTN」を「"Caption"」に修正してください。

変換前 :MOVE POW-TEXT OF MENU1 TO POW-TEXTN OF Label1

変換直後:MOVE "Caption" OF MENU1 OF CfMenu1 TO POW-TEXTN OF Label1

修正後 :MOVE "Caption" OF MENU1 OF CfMenu1 TO "Caption" OF Label1

### メニューアイテムで複数のセパレータ項目が連続している場合

最後のセパレータ項目以外は削除されます。変換後、必要なセパレータ項目を挿入してください。

### グループアイテムが配列化されている場合

グループアイテムの変換方法の指定で、グループボックスコントロールに変換するように指定している場合、配列化されているグループアイテムであっても、単独のグループボックスコントロールとして変換されます。

最初の要素以外のアイテム名は、自動的に生成されますので、必要に応じて手続きを変更または追加してください。

フレームコントロールに変換する場合は、そのまま使用することができます。

### フォームが正しく再描画されない場合

フォームの再描画が正しく行われない場合があります。

この場合、適切な位置で、Refreshメソッドを呼び出して、強制的に再描画を行ってください。Refreshメソッドについては、『リファレンス』を参照してください。

フォームが正しく再描画されない現象は、COBOLプログラム実行中に、NetCOBOLのランタイムシステムが制御権を放棄しなくなったために発生します。詳細は、『NetCOBOL ソフトウェア説明書』(COBOL.TXT)を参照してください。

### フォームのPROCEDUREに複数の共通内部プログラムが記述されている場合

フォームのPROCEDUREに複数の共通内部プログラムが記述されている場合、個々の共通内部プログラムは、プログラムの見出し部と終わり見出しをキーワードにして分割され、フォームの手続きとして作成されます。したがって、プログラムの見出し部と終わり見出しが正しく記述されていない場合、共通内部プログラムが正しく移行できない場合があります。この場合、以前のPowerCOBOLを使用して、プログラム見出しと終わり見出しを正しく対応づけてから、最新のPowerCOBOLで開きなおしてください。

また、プログラムの終わり見出しと、次のプログラムの見出し部とのあいだに記述された注記行などは、フォームのPROCEDUREにそのまま残ります。必要に応じて移動してください。

### CloseChildイベントを使用している場合

V3.0以前のプロジェクトでのシートのCLOSECHILDイベントは、V4.0以降、フォームのCloseChildイベントとして移行されます。しかし、V4.0以降、CloseChildイベントにはパラメタが追加されています。したがって、プロジェクトを移行後、そのフォーム内のどれかのイベント手続きでOpenFormメソッドまたはCallFormメソッドを呼び出し、さらに、フォーム識別IDを使って、閉じられたフォームを判別する場合には、CloseChildイベントの連絡節(LINKAGE SECTION)に引数を追加する必要があります。

変換前 :

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
...
```

変換直後:

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
...
```

追加後 :

```

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
01 POW-COOKIE PIC S9(9) COMP-5.
PROCEDURE DIVISION USING POW-COOKIE.

```

...

OpenFormメソッドおよびCallFormメソッドのフォーム識別IDの使用方法については、「[複数ウィンドウをもつアプリケーションを作成する](#) ( p204) 」を参照してください。また、メソッドおよびイベントの引数については『リファレンス』を参照してください。

## 表アイテムのイベント動作

表アイテムのイベント手続きの中で、表アイテムのイベントが再度発生するような手続きが記述されている場合、V3.0以前は、表アイテムのイベントは発生しませんでした。V4.0以降、これらのイベントが発生するようになりました。たとえば、表アイテムのRETURNイベントの中で、SELECTCELLメソッドにより選択中のセルのフォーカスを移動していた場合、そのセルが自動で編集状態に設定されていれば、V4.0以降では、EDITイベントが発生してしまいます。このとき、EDITイベントに、直前の入力完了しているかをチェックするような手続きが記述されていた場合、RETURNイベントで情報を設定する前にSELECTCELLメソッドでセルを移動してしまうと、EDITイベントの処理が正しく動きません。したがって、このような場合は、イベントが発生する手続きの位置を変更するなどして対応してください。例を以下に示します。

変換前 :

[ 表アイテムのRETURNイベント ]

```

CALL SELECTCELL OF TABLE1 USING W-NEXT-ROW W-NEXT-COLUMN
MOVE "入力完了" TO W-CHECK

```

[ 表アイテムのEDITイベント ]

```

IF W-CHECK NOT = "入力完了" THEN
 CALL DISPLAYMESSAGE OF SHEET1 USING "データが未入力です。"
 CALL SELECTCELL OF TABLE1 USING W-ROW W-COLUMN
END-IF

```

変換直後:

[ 表(Table)コントロールのReturnイベント ]

```

CALL SELECTCELL OF TABLE1 USING W-NEXT-ROW W-NEXT-COLUMN
MOVE "入力完了" TO W-CHECK

```

[ 表(Table)コントロールのEditイベント ]

```

IF W-CHECK NOT = "入力完了" THEN
 CALL DISPLAYMESSAGE OF SHEET1 USING "データが未入力です。"
 CALL SELECTCELL OF TABLE1 USING W-ROW W-COLUMN
END-IF

```



修正後：

[ 表(Table)コントロールのReturnイベント ]

MOVE "入力完了" TO W-CHECK

CALL SELECTCELL OF TABLE1 USING W-NEXT-ROW W-NEXT-COLUMN

[ 表(Table)コントロールのEditイベント ]

修正する必要はありません。

### エラーチェックの強化

V3.0以前のPowerCOBOLでは、アイテム属性のインデックスに不正な値を設定してもエラーにならない場合があります。

しかし、V4.0以降、エラーチェックが強化されたことにより、実行中にエラーメッセージが表示される場合があります。

たとえば、表コントロールの行数が10の場合、以下の文を実行するとエラーが発生します。

```
MOVE "ABC" TO "Text" OF "TableCells"(11 1) OF CmTable1
```

### COBOLのプログラムからPowerCOBOLのシートを開いている場合

V3.0以前のPowerCOBOLで作成したシートを、POWEROPENSHEETメソッドを使ってCOBOLのプログラムから開く場合、COBOLの実行可能プログラム作成時に"F5BBRUNS.LIB"をリンクしていました。しかし、V4.0以降、リンクするファイルの名前が"F5DDFCB4.LIB"に変わりました。POWEROPENSHEETメソッドを使用しているCOBOLプログラムについては、リンクするファイル名を変更し、ビルドしなおす必要があります。

## B.7.2 V2.0以前からの留意事項

V2.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、「[V3.0以前からの留意事項](#) ( p311 )」に加え、以下のような留意事項があります。

### リンクオプションおよびリソースコンパイルオプション

V3.0以降、32ビット対応版のリンカおよびリソースコンパイラを使用しています。したがって、V2.0以前に指定していたオプション（たとえば、"/PACKC"リンクオプションや"/K"リソースコンパイルオプションなど）が使用できない場合があります。

このような場合は、オプションを削除してからビルドしてください。

## B.7.3 V1.0以前からの留意事項

V1.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、「[V3.0以前からの留意事項](#) ( p311 )」および「[V2.0以前からの留意事項](#) ( p315 )」に加え、以下のような留意事項があります。

### 文字列入力アイテム

V1.0では、文字列入力アイテムで文字列を編集時に[Enter]キーを入力すると、同じシート上にあるデフォルトのプッシュボタンアイテムのCLICKイベントが発生しました。

V2.0以降、文字列入力アイテムで[Enter]キーを入力すると、文字列入力アイテムのRETURNイベントが発生するように、仕様が変更されました。

V1.0で作成したプロジェクトを変換した場合は、そのままビルドすることにより、V1.0のときと同じように動作するアプリケーションを作成することができます。

ただし、本バージョンで、V1.0と同じように動作するアプリケーションを、新規に設計する場合は、テキストボックスコントロールのプロパティ設定ダイアログボックスで、以下の設定をしてください。

1. テキストボックスコントロールのプロパティ設定ダイアログボックスを開きます。
2. [テキストボックス]タブの[編集中に[Enter]キーを押したとき]のチェックをはずします。
3. OKボタンをクリックします。

### コンボボックスアイテム

V1.0では、コンボボックスアイテムにフォーカスがある場合、[Enter]キーを入力すると、同じシート上にあるデフォルトのプッシュボタンアイテムのCLICKイベントが発生しました。

V2.0以降、コンボボックスアイテムで[Enter]キーを入力すると、コンボボックスアイテムのRETURNイベントが発生するよう、仕様が変更されました。

V1.0で作成したプロジェクトを変換した場合は、そのままビルドすることにより、V1.0のときと同じように動作するアプリケーションを作成することができます。

ただし、本バージョンで、V1.0と同じように動作するアプリケーションを、新規に設計する場合は、コンボボックスコントロールのプロパティ設定ダイアログボックスで、以下の設定をしてください。

1. コンボボックスコントロールのプロパティ設定ダイアログボックスを開きます。
2. [テキストボックス]タブの[[Enter]キーを押した場合Returnイベントを発生]のチェックをはずします。
3. OKボタンをクリックします。

### B.7.4 V1.0L10からの留意事項

---

PowerCOBOL V1.0L10で作成したプロジェクトを変換する場合、「[V3.0以前からの留意事項](#)( p311)」、「[V2.0以前からの留意事項](#)( p315)」および「[V1.0以前からの留意事項](#)( p315)」に加え、以下のような留意事項があります。

#### 表アイテム

V1.0L10では、表アイテムのセルがフォーカスを受け取ったときにEDITイベントが発生しました。

V1.0L20以降、表アイテムのセルがフォーカスを受け取ったあと、どれかのキーを押したとき、またはセルをダブルクリックしたときにEDITイベントが発生するように、仕様が変更されています。

V1.0L10で作成したプロジェクトを変換した場合は、そのままビルドすること

により、V1.0のときと同じように動作するアプリケーションを作成することができます。

ただし、本バージョンで、V1.0L10と同じように動作するアプリケーションを新規に設計する場合は、表コントロールのプロパティ設定ダイアログボックスで、以下の設定をしてください。

1. 表コントロールのプロパティ設定ダイアログボックスを開きます。
2. [列スタイル]タブの[列リスト]から、EDITイベントを発生させる列をすべて選択します。
3. [自動編集状態]をチェック状態にします。
4. OKボタンをクリックします。

### PICTURE編集アイテム

V1.0L10では、PICTURE編集アイテムがフォーカスを受け取ったときにEDITイベントが発生しました。

V1.0L20以降、PICTURE編集アイテムがフォーカスを受け取ったあと、どれかのキーを押したときにEDITイベントが発生するように、仕様が変更されています。V1.0L10で作成したプロジェクトを変換した場合は、そのままビルドすることにより、V1.0のときと同じように動作するアプリケーションを作成できます。ただし、本バージョンで、V1.0L10と同じように動作するアプリケーションを、新規に設計する場合は、テキストボックスコントロールのプロパティ設定ダイアログボックスで、以下の設定をしてください。

1. テキストボックスコントロールのプロパティ設定ダイアログボックスを開きます。
2. [テキストボックス]タブの[編集状態になったとき]をチェック状態にします。
3. [シングルライン]タブを開きます。
4. [編集可能なラベルとして使用]をチェック状態にします。
5. [フォーカス取得時に自動で編集状態]をチェック状態にします。
6. OKボタンをクリックします。

### POWERC.BOB

V1.0L10では、POWERC.BOBファイルを#INCLUDE文によりイベント手続き中に取り込んでいました。V4.0以降、POWERC.BOBファイルを取り込む必要はなくなりました。

POWERC.BOBファイルを取り込んでいるプロジェクトを変換し、そのままビルドすると、POWERC.BOBファイルが見つからない内容の診断メッセージが表示されます。

この場合、イベント手続き中の「#INCLUDE "POWERC.BOB"」を検索し、該当する行を削除してください。

## B.8 非互換項目

使用していたPowerCOBOLのバージョンにより、以下の非互換項目があります。

### B.8.1 V3.0以前からの非互換項目

---

V3.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、以下の非互換項目があります。

#### シートおよびアイテムのサイズや位置

プロジェクトに登録されたシートおよびアイテムのサイズや位置は、誤差が生じる場合があります。

#### OPENSHEETメソッド

CLOSEDイベントで、OPENSHEETメソッドを（本バージョンでのOpenFormメソッドも）呼び出すことはできません。

#### POW-FONTSIZE

POW-FONTSIZEを参照している場合、設定した値と異なる値になる場合があります。これは、フォントサイズを設定したときに、Windows のフォントドライバが、適切なフォントおよびフォントサイズに置き換えるためです。

#### メニューアイテムの属性の記述

V3.0以前のメニューアイテムは、V4.0以降では、メニューオブジェクトおよびメニューアイテムオブジェクトに変換されます。これらのオブジェクトに対して、“POW-”ではじまる属性名を使用することはできません。プロパティ名を使用するようにしてください。

#### 重なりあったアイテムのマウスに関するイベント

V3.0以前は、重なりあったアイテムの上でマウスに関するイベント（Clickイベントなど）を発生させると、最も下に配置されているアイテムのイベントが発生していました。V4.0以降では、最も上に配置されているコントロールのイベントが発生します。

#### ファンクションキーアイテムのClickイベント

ファンクションキーアイテムでは、ファンクションキーが入力された場合だけ、CLICKイベントが発生し、マウスの左ボタンでクリックしてもCLICKイベントは発生しませんでした。

V3.0以前のファンクションキーアイテムは、V4.0以降では、コマンドボタンコントロールに変換されるので、ファンクションキーの入力だけでなく、マウスの左ボタンでクリックした場合にもCLICKイベントが発生します。

#### POW-ENABLEおよびPOW-VISIBLEの設定

V3.0以前は、シートのPOW-ENABLEまたはPOW-VISIBLEにPOW-OFFを設定しても、シート上のアイテムのPOW-ENABLEおよびPOW-VISIBLEの値は変更されませんでした。

V4.0以降では、変換されたフォームのPOW-ENABLEまたはPOW-VISIBLEにPOW-OFF

が設定されると、フォーム上のコントロールのPOW-ENABLEおよびPOW-VISIBLEにも、POW-OFFが設定されます。

### 文字列入力アイテムの入力可能文字数

V3.0以前の文字列入力アイテムでは、入力可能文字数を指定して日本語を入力した場合、指定した文字数の半分の文字数まで日本語を入力することが可能でした。

V4.0以降では、テキストボックス(TextBox)コントロールでは、入力可能文字数を指定した場合、Windows NT 4.0、Windows 2000またはWindows XPでアプリケーションを実行すると、指定した文字数分の日本語を入力することが可能です。

ただし、Windows 95、Windows 98およびWindows Meでは、V3.0以前と同様です。

## B.8.2 V2.0以前からの非互換項目

V2.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、「[V3.0以前からの非互換項目](#) ( p318) 」に加え、以下の非互換項目があります。

### OLEアイテムとMeFt/graphアイテム

V3.0以降、OLEアイテムおよびMeFt/graphアイテムに対応するコントロールはなくなりました。したがって、これらのアイテムを移行することはできません。資産を移行する場合、これらのアイテムは自動的に取り除かれます。

### 仕様書印刷機能

V3.0以降、仕様書印刷機能はなくなりました。

仕様書を印刷する場合には、NetCOBOLシリーズの『SIMPLIA/COBOL支援キット』をご利用ください。

### アイテムの背景色の印刷

V3.0以降、以下のアイテムの背景色が印刷されるようになっています。

グラフアイテム

メタファイル再生アイテム

## B.8.3 V1.0以前からの非互換項目

V1.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、「[V3.0以前からの非互換項目](#) ( p318) 」および「[V2.0以前からの非互換項目](#) ( p319) 」に加え、以下の非互換項目があります。

### シートのオープンとクローズ

V1.0以前のPowerCOBOLでは、1つのイベント手続きの中ではCLOSESHEETメソッドで閉じたシートを、同じイベント手続きの中で再びOPENSHEETメソッドで開くことができました。V2.0以降では、同一のイベント手続きの中で、再びシートを開くことはできなくなっています。

#### B.8.4 V1.0L10からの非互換項目

---

V1.0以前のPowerCOBOLで作成したプロジェクトを変換する場合、「[V3.0以前の非互換項目](#) ( p318 )」、「[V2.0以前からの非互換項目](#) ( p319 )」および「[V1.0以前からの非互換項目](#) ( p319 )」に加え、以下の非互換項目があります。

##### プッシュボタンアイテムのCLICKイベント

V1.0L10では、デフォルトのプッシュボタンアイテム以外のプッシュボタンアイテムにフォーカスがある場合、[Enter]キーを入力すると、デフォルトのプッシュボタンアイテムのCLICKイベントが発生しました。

V1.0L20以降では、フォーカスがあるプッシュボタンのCLICKイベントが発生するようになっています。

また、V4.0以降では、フォーカスの移動によりコマンドボタンコントロールのデフォルト属性 ( Defaultプロパティ ) も自動的に移動されるようになっています。

---

## 付録C サンプルプログラムについて

本製品には、サンプルプログラムが添付されています。

サンプルプログラムは、本製品をインストールしたフォルダ配下の "Samples¥PowerCOB" フォルダに、種類ごとに複数のフォルダにわかれて格納されています。たとえば、CドライブのNetCOBOLフォルダにPowerCOBOLをインストールした場合、サンプルプログラムは、"C:¥NetCOBOL¥Samples¥PowerCOB" フォルダに格納されています。

格納されているファイルは、PowerCOBOLのプロジェクトファイル(拡張子:ppj)、各サンプルプログラムで使用するリソースファイル(拡張子:ICOやBMPなど)およびデータファイルです。サンプルプログラムは、コントロールやオブジェクトの使いかたおよび配列化やActiveXコントロールの作成方法など、PowerCOBOLの各種機能を利用した使用例を示しています。アプリケーションを開発する際の参考にしてください。

### C.1 サンプルプログラムの使用方法

サンプルプログラムは、以下の手順で参照することができます。

1. PowerCOBOLを起動します。
2. プロジェクトウィンドウの[ファイル]メニューから[既存プロジェクトを開く]コマンドを選択します。
3. [ファイルを開く]ダイアログボックスで、サンプルプログラムが格納されているフォルダから、参照したいプロジェクトファイルを選択します。
4. OKボタンをクリックします。  
デザインツリーウィンドウ上で、プロジェクトの構成を参照することができます。
5. プロジェクトウィンドウのデザインツリーウィンドウ上で[フォーム]を選択します。
6. ポップアップメニューから[開く]コマンドを選択します。  
フォームの設計内容を参照できます。
7. フォームが開かれたら、フォームまたはフォーム上のコントロールを選択します。
8. ポップアップメニューの[イベント手続きの編集]サブメニューから、参照したいイベントを選択します。  
プログラムの記述方法を参照できます。

サンプルプログラムは、以下の手順で実行することができます。

1. PowerCOBOLを起動します。
  2. プロジェクトウィンドウの[ファイル]メニューから[既存プロジェクトを開く]コマンドを選択します。
  3. [ファイルを開く]ダイアログボックスで、サンプルプログラムが格納されているフォルダから、参照したいプロジェクトファイルを選択しま
-

- す。
4. OKボタンをクリックします。
  5. プロジェクトウィンドウの[プロジェクト]メニューから[すべてビルド]コマンドを選択します。  
実行可能プログラムが作成されます。
  6. ビルドが正常に終了したら、[プロジェクト]メニューの[実行]コマンドを選択します。  
サンプルプログラムが実行されます。



---

サンプルプログラムを参照したり、実行したりする場合、あらかじめ準備が必要なサンプルプログラムや、使用方法の説明が必要なサンプルプログラムについては、「[サンプルプログラムの補足説明](#) ( p331 )」が用意されています。サンプルプログラムをお使いになる前に必ずお読みください。

たとえば、ActiveXコントロールを利用したプログラムでは、そのActiveXコントロールが利用可能な状態になっていなければ、サンプルプログラムを使用することができません。補足説明には、ActiveXコントロールを利用可能な状態にする方法やプログラムの実行方法などが記載されています。また、実行環境の設定が必要なサンプルプログラムについては、その設定方法などが記載されています。

---

## C.2 サンプルプログラム一覧

PowerCOBOLでは、以下のサンプルプログラムを提供しています。  
サンプルプログラム一覧は、以下のように記載されています。

### フォルダ名 (アルファベット順)

#### プロジェクトファイル名

サンプルプログラムの概要を説明しています。

#### ActiveX

##### ActiveX.ppj

ActiveXコントロールの作成方法を示しています。

詳細は「[ActiveX.ppjの補足説明](#) ( p331 )」を参照してください。

##### CallActiveX.ppj

フォーム上でActiveXコントロールを利用する方法を示しています。

詳細は「[CallActiveX.ppjの補足説明](#) ( p332 )」を参照してください。

##### OpenActiveX.ppj

フォーム上でActiveXコントロールを利用する方法を示しています。

詳細は「[OpenActiveX.ppjの補足説明](#) ( p331 )」を参照してください。



**SampleActiveX.ppj**

フォーム上でActiveXコントロールを作成する方法を示しています。

詳細は「[ActiveXコントロールを作成する](#) ( p245) 」を参照してください。

**UsingActiveX.ppj**

フォーム上でActiveXコントロールを利用する方法を示しています。

詳細は「[ActiveXコントロールを使ったアプリケーションを作成する](#) ( p260) 」を参照してください。

**UsingAutomationServer.ppj**

フォーム上でActiveXコントロールをオートメーションサーバとして利用する方法を示しています。

詳細は「[オートメーションサーバを使ったアプリケーションを作成する](#) ( p270) 」を参照してください。

**WebTimer.ppj**

PowerCOBOLで作成したActiveXコントロールをWeb上で利用する方法を示しています。

詳細は「[PowerCOBOLで作成したActiveXコントロールをWeb上で利用する](#) ( p275) 」を参照してください。

**ADODataSource****ADODataSource1.ppj**

SQL Serverとの連携を例にして、ADOデータソースコントロールの使用方法を示しています。

詳細は「[DataGridコントロールとの連携環境を設計時に設定する](#) ( p294) 」を参照してください。

**ADODataSource2.ppj**

Accessとの連携を例にして、ADOデータソースコントロールの使用方法を示しています。

詳細は「[ADODataSource2.ppjの補足説明](#) ( p333) 」を参照してください。

**ADODataSource3.ppj**

SQL Server (Access + MSDE) との連携を例にして、ADOデータソースコントロールの使用方法を示しています。

詳細は「[DataGridコントロールとの連携環境を実行時に設定する](#) ( p298) 」を参照してください。

**ADODataSource4.ppj**

Accessとの連携を例にして、ADOデータソースコントロールの使用方法を示しています。

詳細は「[ADODataSource4.ppjの補足説明](#) ( p334) 」を参照してください。

**Animation****Animation.ppj**

アニメーションコントロールを使って、アニメーションを再生する方法を示し

ています。

## Array

### Array1.ppj

コントロールの配列化の利用方法を示しています。

詳細は「[配列化したコントロールを使ったアプリケーションを作成する](#) ( p194) 」を参照してください。

### Array2.ppj

2種類のコントロールを配列化し、組み合わせて利用する方法を示しています。

### Bomb.ppj

配列化機能を利用した簡単なゲームです。

## CheckBox

### CheckBox.ppj

チェックボックスコントロールを使って、オプション項目のオン/オフの切り替え方法を示しています。

## Color

### Color.ppj

色の設定方法を示しています。

## ComClass

### AccessToObject.ppj

POWERCONVTOCOMを使って、NetCOBOLのCOM連携機能の利用方法を示しています。

詳細は「[COBOLの\\*COMクラスを利用してアクセスする](#) ( p198) 」を参照してください。

## ComboBox

### ComboBox.ppj

コンボボックスコントロールのリスト部に項目を追加/削除する方法を示しています。

### SelectionBox.ppj

コンボボックスコントロールとCSVファイルを使って、コンボボックスに表示するフィールドの切り替え方法を示しています。

## CommandButton

### CommandButton.ppj

コマンドボタンコントロールのClickイベントを使って、コマンドボタンの標準的な使用方法を示しています。

### BitmapButton.ppj

コマンドボタンコントロールとビットマップイメージを組み合わせて使用する方法を示しています。

**MessageBoard.ppj**

コマンドボタンコントロールとファンクションキーを組み合わせる方法を示しています。

**DBAccess****DBAccess.ppj**

SQL Serverとの連携を例にして、DBアクセスコントロールの使用方法を示しています。

詳細は「[DBアクセスコントロールを利用してデータベースと連携する](#) ( p277) 」を参照してください。

**DBAccess2.ppj**

SymfoWARE Serverとの連携を例にして、DBアクセスコントロールの使用方法を示しています。

詳細は「[DBAccess2.ppjの補足説明](#) ( p334) 」を参照してください。

**DDE****DDE.ppj**

DDEコントロールを使って、Excelシートの情報を参照する方法を示しています。

詳細は「[DDE.ppjの補足説明](#) ( p335) 」を参照してください。

**DLL****DLL.ppj**

PowerCOBOLのフォームから、DLLファイル中のフォームを開く方法を示しています。

**DriveList****DriveList.ppj**

ドライバリストコントロール、フォルダリストコントロールおよびファイルリストコントロールを組み合わせ、これらのコントロールの基本的な使用方法を示しています。

**Edit****ADODataSource.ppj**

Accessとの連携を例にして、ADOデータソースコントロールの使用方法を示しています。

**ExcelConnection****ExcelSam.ppj**

Excel連携コントロールを使って、Excelコマンドを実行し、Excelシートとのデータ交換方法を示しています。

**Focus****Focus.ppj**

SetFocusメソッドを使って、フォーカスの移動方法を示しています。

## Font

### Font.ppj

フォントの設定方法を示しています。

## FormControls

### Controls1.ppj

フォームのControlsコレクションオブジェクトを操作する方法を示しています。POWERGETCONTROLを利用して、単一フォームの入力フィールドをクリアする例を示しています。

### Controls2.ppj

フォームのControlsコレクションオブジェクトを操作する方法を示しています。NetCOBOLのCOM連携機能を利用して、複数フォームの入力フィールドをクリアする例を示しています。

## Frame

### Frame.ppj

フレームコントロールの境界線の表示方法を示しています。

## FromCOBOL

### CopyData.ppj

COBOLプログラムから、PowerCOBOLのフォームを開く方法を示しています。  
詳細は「[CopyData.ppjの補足説明](#)( p335)」を参照してください。

## Graph

### Graph.ppj

グラフコントロールを使って、棒グラフへの値の設定方法を示しています。

## GroupBox

### GroupBox.ppj

グループボックスコントロールとフレームコントロールの使用方法の違いを示しています。

## Hello

### Hello.ppj

スタティックテキストコントロールで表示される文字列を変更する方法を示しています。  
作成方法は「[アプリケーションの作成手順](#)( p12)」を参照してください。

## Icon

### Icon.ppj

フォームにアイコンを設定する方法を示しています。

## Image

### Image.ppj

イメージコントロールを使って、イメージの基本的な扱い方を示しています。

## KeyEvent

### FormKeyEvent.ppj

フォームのPreKeyPressイベントを使って、キー入力のタイムアウト処理の方法を示しています。

### KeyPressEvent.ppj

KeyDownイベントおよびKeyPressイベントを使って、入力文字のチェック方法を示しています。

## ListBox

### ListBox.ppj

リストボックスコントロールに項目を追加/削除する方法を示しています。

## ListView

### ListView.ppj

リストビューコントロールの基本的な使用方法を示しています。  
詳細は「[ListView.ppjの補足説明](#) ( p336) 」を参照してください。

## MCI

### CDPlayer.ppj

音楽CDの再生を例にして、MCIコントロールの使用方法を示しています。

## Menu

### Menubar.ppj

メニューオブジェクトとメニューアイテムオブジェクトを使って、フォーム上でのメニューバーの使用方法を示しています。

### PopupMenu.ppj

メニューオブジェクトとメニューアイテムオブジェクトを使って、メニューバーとポップアップメニューを組み合わせた使用方法を示しています。

### Table3.ppj

ポップアップメニューの作成方法を示しています。  
詳細は「[ポップアップメニューを使ったアプリケーションを作成する](#) ( p219) 」を参照してください。

## MessageBox

### MessageBox.ppj

メッセージボックスの基本的な表示/応答方法を示しています。

## MouseEvent

### MouseEvent.ppj

MouseDownイベント、MouseMoveイベントおよびMouseUpイベントを使って、ドラッグアンドドロップ操作の方法を示しています。

## MousePointer

### MousePointer.ppj

フォーム上でマウスポインタ(マウスカーソル)を変更する方法を示しています。

## OptionButton

### OptionButton.ppj

オプションボタンコントロールを使って、複数の選択項目から1つのオプション項目を選択する方法を示しています。

## PowerFORM

### PowerFORM.ppj

NetCOBOLシリーズの『PowerFORM』で定義した帳票を使って、表コントロールに入力されたデータを印刷する方法を示しています。

詳細は「[PowerFORM.ppjの補足説明](#)( p336)」を参照してください。

## PowerSORT

### PowerSORT.ppj

『PowerSORT OCX』を使って、データを並べ替える方法を示しています。

詳細は「[PowerSORT.ppjの補足説明](#)( p336)」を参照してください。

## Print

### PrintForm.ppj

印刷コントロールを使って、フォームを印刷する方法を示しています。

## ProgressIndicator

### ProgressIndicator.ppj

進行状況インジケータコントロールの基本的な操作方法を示しています。

## Scalable

### Scalable.ppj

フォームのScalableプロパティを使って、フォームの大きさを変更した場合の動作を示しています。

## Scrollbar

### Scrollbar.ppj

スクロールバーコントロールのChangeイベントとEndScrollイベントの違いを示しています。

## SimpleForm

### OpenForm.ppj

フォームから、OpenFormメソッドを使って、別のフォームをモードレスで開く方法を示しています。

### CallForm.ppj

フォームから、CallFormメソッドを使って、別のフォームをモーダルに開く方法を示しています。

### ReturnCheck.ppj

CallFormメソッドの復帰値を使って、情報を受け渡す方法を示しています。

### Table2.ppj

OpenFormメソッドを使って、別のフォームを開く方法を示しています。  
詳細は「[OpenFormメソッドを使用する](#) ( p204) 」を参照してください。

## Slider

### Slider.ppj

スライダコントロールのつまみの操作方法など、基本的な操作方法を示しています。

## Statusbar

### Statusbar.ppj

フォームのステータスバーの基本的な使用方法を示しています。

### Table4.ppj

フォームにステータスバーを追加する方法を示しています。  
詳細は「[ステータスバーを使ったアプリケーションを作成する](#) ( p223) 」を参照してください。

## Tab

### Tab.ppj

タブコントロールの基本的な使用方法を示しています。

### Table6.ppj

タブコントロールを使ったアプリケーションの作成方法を示しています。  
詳細は「[タブコントロールを使ったアプリケーションを作成する](#) ( p231) 」を参照してください。

## Table

### ReadData.ppj

表コントロールへのデータの読み込みおよび並べ替えの方法を示しています。

### WriteData.ppj

表コントロールへのデータの入力、集計およびファイルへの書き込み方法を示しています。

### Table1.ppj

表コントロールを使って、基本的なアプリケーションの作成方法を説明しています。

作成方法の詳細は「[アプリケーションを作成しよう](#) ( p49) 」を参照してください。

### CreateProducts.ppj

Table1.ppjで使用するファイル("PRODUCTS.TBL")を生成します。

## TextBox

### TextBox.ppj

テキストボックスコントロールを使って、文字列を入力する方法を示しています。

### Picture.ppjとCompute.ppj

テキストボックスコントロールのテキスト属性オブジェクトを使って、いろいろなPICTURE属性の定義/利用方法を示しています。

### DateStyle.ppj

テキストボックスコントロールのテキスト属性オブジェクトを使って、いろいろな日付表示形式を示しています。

### EditFile.ppj

テキストボックスコントロールのLoadFileメソッドとSaveFileメソッドを使って、直接、ファイルから文字列をロード/セーブする方法を示しています。

## Timer

### Timer.ppj

タイマコントロールの基本的な使用方法を示しています。

## Toolbar

### Toolbar.ppj

ツールバーコントロールとメニューバーおよびポップアップメニューを組み合わせ、ツールバーの基本的な使用方法を示しています。

### Table5.ppj

ツールバーコントロールを使ったアプリケーションの作成方法を示しています。

詳細は「[ツールバーを使ったアプリケーションを作成する](#) ( p226) 」を参照してください。

## TreeView

### TreeView.ppj

ツリービューコントロールの基本的な使用方法を示しています。

詳細は「[TreeView.ppjの補足説明](#) ( p336) 」を参照してください。



## C.3 サンプルプログラムの補足説明

サンプルプログラムを参照または実行する前に、以下の補足説明をお読みください。

### C.3.1 ActiveX.ppjの補足説明

ActiveXコントロールを作成する方法を示したサンプルプログラムです。作成するActiveXコントロールは、同じフォルダに格納されているOpenActiveX.ppjおよびCallActiveX.ppjで利用されます。したがって、OpenActiveX.ppjおよびCallActiveX.ppjを参照する前に、このプロジェクトでActiveXコントロールを作成し、システムに登録しておく必要があります。ActiveXコントロールがシステムに正しく登録されていない場合、OpenActiveX.ppjおよびCallActiveX.ppjを正常に開くことができません。

以下の操作により、ActiveXコントロールをシステムに登録してください。

1. PowerCOBOLを起動します。
2. ActiveX.ppjを開きます。
3. [プロジェクト]メニューの[すべてリビルド]コマンドを選択します。
4. InputActiveX [モジュール]を選択します。
5. ポップアップメニューから[システムに登録]コマンドを選択します。
6. システムに正しく登録されたことを示すメッセージボックスが表示されたら、OKボタンをクリックします。

ここで登録したActiveXコントロールは、"FUJITSU PowerCOBOL Sample Controls"という名前でシステムに登録されます。

### C.3.2 OpenActiveX.ppjの補足説明

OpenActiveX.ppjおよびCallActiveX.ppjはメインフォームからサブフォームを開く方法を示したサンプルプログラムです。CallActiveX.ppjではサブフォームを同期的に開きます。つまり、サブフォームを閉じるまでメインフォームに制御が戻りません。これに対し、OpenActiveX.ppjではサブフォームを非同期的に開きます。この場合、メインフォームとサブフォームは並行して実行されます。OpenActiveX.ppjおよびCallActiveX.ppjは、ActiveX.ppjによって作成されるActiveXコントロールを使用しています。したがって、このサンプルプログラムを編集および実行するには、あらかじめActiveXコントロールをシステムに登録しておく必要があります。ActiveXコントロールを登録する手順については、「[ActiveX.ppjの補足説明](#) (p331)」を参照してください。

メインフォームからサブフォームを非同期的に開くには、OpenFormメソッドを使う方法に加え、以下の2つの方法があります。

サブフォームをオートメーションサーバとして開く方法

配置してあるサブフォームをアクティベートする方法

OpenActiveX.ppjでは、これら2つの方法を説明しています。

### サブフォームをCOMサーバとして開く方法

オートメーションサーバとして作成したサブフォームを、NetCOBOLのCOM連携機能を用いて開きます。この場合、一般にサブフォームがメインフォームを知る方法はありません。メインフォームが一方的にサブフォームを作成し、操作し、破棄します。サンプルプログラムでは、サブフォームが閉じられても、メインフォームにはそれが通知されません。メインフォームの[サブフォームを閉じる]ボタンをクリックすることによって、初めてメインフォームはサブフォームの内容を読み取り、サブフォームを破棄します。メインフォームが破棄しなければ、たとえサブフォームが非可視になっていたとしても、サブフォームはメモリ上に残っています。

### 配置してあるサブフォームをアクティベートする方法

ActiveXコントロールとして作成したサブフォームを、メインフォーム上に配置します。初期状態として、サブフォームの属性を非可視にしておき、必要ときにActivateメソッドおよびDeactivateメソッドを用いて別ウィンドウに表示しています。この場合、サブフォームは、メインフォームが作成されたときに作成され、メインフォームが破棄されるときに破棄されます。また、サブフォームからメインフォームへイベントを使って情報を通知することができません。サンプルプログラムでは、サブフォームが閉じられても、メインフォームにはそれが通知されません(通知されるようにプログラミングすることもできますが、このサンプルプログラムの目的は2種類の方法の比較なので省略しています)。メインフォームは、[サブフォームを閉じる]ボタンがクリックされるまで、サブフォームが開いているとみなしています。

## C.3.3 CallActiveX.ppjの補足説明

---

OpenActiveX.ppjおよびCallActiveX.ppjはメインフォームからサブフォームを開く方法を示したサンプルプログラムです。CallActiveX.ppjではサブフォームを同期的に開きます。つまり、サブフォームを閉じるまでメインフォームに制御が戻りません。これに対し、OpenActiveX.ppjではサブフォームを非同期的に開きます。この場合、メインフォームとサブフォームは並行して実行されます。OpenActiveX.ppjおよびCallActiveX.ppjは、ActiveX.ppjによって作成されるActiveXコントロールを使用しています。したがって、このサンプルプログラムを編集および実行するには、あらかじめActiveXコントロールをシステムに登録しておく必要があります。ActiveXコントロールを登録する手順については、「[ActiveX.ppjの補足説明](#)( p331)」を参照してください。メインフォームからサブフォームを同期的に開くには、CallFormメソッドを使う方法に加え、以下の2つの方法があります。

サブフォームをオートメーションサーバとして開く方法

配置してあるサブフォームをアクティベートする方法

CallActiveX.ppjでは、これら2つの方法を説明しています。

### サブフォームをオートメーションサーバとして開く方法

オートメーションサーバとして作成したサブフォームを、NetCOBOLのCOM連携機能を用いて開きます。この場合、一般にサブフォームがメインフォームを知

---

る方法はありません。メインフォームが一方的にサブフォームを作成し、操作し、破棄します。

### 配置してあるサブフォームをアクティベートする方法

ActiveXコントロールとして作成したサブフォームを、メインフォームに配置します。初期状態として、サブフォームの属性を非可視にしておき、必要なときにActivateメソッドおよびDeactivateメソッドを用いて別ウィンドウに表示しています。この場合、サブフォームはメインフォームが作成されたときにロードされ、メインフォームが破棄されるまでアンロードされません。また、サブフォームからメインフォームへイベントを使って情報を通知することができます。

### C.3.4 ADODataSource2.ppjの補足説明

ADOデータソースコントロールを利用して、Accessのサンプルデータベース"Northwind.mdb"のテーブル中から、「氏名」フィールドのデータを『Microsoft Rich Textbox Control version 6.0 (以降、RichTextboxコントロールと呼びます)』上に表示するサンプルプログラムです。

このサンプルプログラムを参照および実行するには、RichTextboxコントロールが必要です。また、このサンプルプログラムを実行するには、ODBCデータソース"OLE\_DB\_NWind\_Jet"をユーザDSNとしてODBCデータソースに追加し、RichTextboxコントロールのプロパティを設定しておく必要があります。

#### 接続文字列（ODBCデータソース名）の確認と設定方法

ADOデータソースコントロールをODBCデータソースとして利用する場合は、接続文字列としてODBCデータソース名(DSN)を設定します。このサンプルプログラムでは、すでにOLE\_DB\_NWind\_Jetが選択されています。

OLE\_DB\_NWind\_Jetが利用可能であるかどうかは、以下のようにして確認することができます。

1. ADOデータソースコントロールのプロパティ設定ダイアログボックスを表示します。
2. [ 接続文字列 ] の右側にあるボタンをクリックします。
3. 表示されたメニューから [ ODBCデータソース名を指定 ] を選択します。  
[ ODBCデータソース名の選択 ] ダイアログボックスが表示されます。
4. ODBCデータソース名の一覧にOLE\_DB\_NWind\_Jetがあるかどうかを確認します。

一覧にOLE\_DB\_NWind\_Jetがない場合は、一覧の右側にある [ 新規作成 ] ボタンをクリックし、以下の手順で一覧に追加します。

1. データソースの型として、ユーザデータソースを選択します。
2. [ 次へ ] ボタンをクリックします。
3. データソースのドライバとして、Microsoft Access Driver (\*.mdb)を選択します。
4. [ 次へ ] ボタンをクリックします。
5. [ 完了 ] ボタンをクリックします。

[ ODBC Microsoft Access セットアップ ] ダイアログボックスが表示

されます。

6. データソース名として、OLD\_DB\_NWind\_Jetを指定します。
7. データベースの[ 選択 ]ボタンをクリックして、Accessのサンプルデータベース"Northwind.mdb"を選択します。
8. OKボタンをクリックします。

### RichTextboxコントロールのプロパティの設定方法

1. デザインツリーウィンドウで、"RichTextbox1[RichTextCtrl]"を選択します。
2. プロパティリストウィンドウのプロパティ一覧から、"DataSource"を選択します。
3. プロパティリストウィンドウの値から、"CmADODataSource1"を選択します。

### C.3.5 ADODataSource4.ppjの補足説明

---

ADOデータソースコントロールを利用して、Accessのサンプルデータベース"Northwind.mdb"のデータを表コントロールのセルに設定するプログラムです。このサンプルプログラムを実行するには、ビルドする前に、以下の変更が必要です。

1. CmCommand1-Clickを開きます。
2. 以下の文字列を検索します。  
"C:¥Program Files¥Microsoft Office¥Office¥Samples¥Northwind.mdb"
3. 検索した文字列を、実際にNorthwind.mdbが格納されているフォルダに変更します。

### C.3.6 DBAccess2.ppjの補足説明

---

#### ODBCの設定

SymfoWARE Serverのサンプルデータベースにアクセスするサンプルプログラムです。

データ資源名をKANRI\_DB、デフォルトスキーマ名をS1にそれぞれ設定し、SYMFOという名前のODBCデータソースを作成してください。処理対象としているテーブルは、COMPANYです。なお、サンプルプログラムではユーザIDをUSER01としていますが、環境に合わせてユーザIDを変更してください。

#### 各ボタンの意味

フォーム上の各ボタンの意味を、以下に示します。

##### OPEN

データベースとの接続を行います。

##### CLOSE

データベースとの接続を解除します。

##### SELECT

結果セットを作成 ( SelectRecordsメソッドの呼び出し ) します。

**PRIOR**

前のレコードを読み込み、値をテキストボックスに表示します。

**NEXT**

次のレコードを読み込み、値をテキストボックスに表示します。

**UPDATE**

現在のレコードをテキストボックスに表示されている値で書き換えます。書き換えを行う場合はテキストボックスの値を書き換えたあとにこのボタンをクリックしてください。なお、COMPNOフィールドはキーフィールドに設定されているため、書き換えの対象にはなりません。

**COMMIT**

コミット処理を行います。

**ROLLBACK**

ロールバック処理を行います。

**その他**

このサンプルプログラムでは、ステータスの管理を、GLOBAL属性をもつ OpenFlagという変数で行っています。

各値は、以下の意味を示します。

- 0: データベースに接続していません。
- 1: データベースに接続中です。
- 2: 結果セット（カーソル）が作成されています。
- 3: 現在のレコード（カレントレコード）が存在します。

---

**C.3.7 DDE.ppjの補足説明**

---

DDEコントロールを使って、Excelシートの情報を参照する方法を示したサンプルプログラムです。このサンプルプログラムを実行するためには、Microsoft Excel 97以降が必要です。

[DDE OPEN]ボタンをクリックする前に、同じフォルダにあるExcelのファイル(exceldde.xls)を開いてください。Excelのファイルが開かれていない場合、フォーム上にデータを読み込むことができません。

---

**C.3.8 CopyData.ppjの補足説明**

---

COBOLで作成したプログラムから、PowerCOBOLで作成したDLL内のフォームを呼び出すサンプルプログラムです。

呼び出し元のプログラムは、CallPcob.exeです。このプログラムは、COBOLプロジェクトマネージャで作成したアプリケーションで、対応するプロジェクトファイルは、CallPcob.prjです。

CallPcob.exeを実行すると、データの保存先を問い合わせてきますので、任意のファイル名(たとえば、"a.txt"や"b.dat"など)を指定してください。

---

COBOLプロジェクトマネージャの使用方法については、『NetCOBOL 使用手引書』を参照してください。

### C.3.9 ListView.ppjの補足説明

---

ListViewコントロールの基本的な使用方法を示したサンプルプログラムです。サンプルプログラムの操作方法を以下に示します。

名前および1月から3月までの給料を入力し、[追加] ボタンをクリックすると、リストビューにデータが登録されます。

リストビュー上のそれぞれの名前をクリックすると、合計が計算され表示されます。

名前を選択して[削除] ボタンをクリックすると、選択されている行が削除されます。

### C.3.10 PowerFORM.ppjの補足説明

---

Tableコントロールに入力したデータを、PowerFORMで作成した帳票定義体を使って印刷するサンプルプログラムです。

このサンプルプログラムを翻訳するには『富士通 PowerFORM V5.0以降』が、実行するには『富士通 MeFt(Message editing Facile tool) V5.0以降』が必要です。これらの製品がインストールされていない場合、プロジェクトファイルを正しくビルドまたは実行することができません。帳票定義体を使った印刷についての詳細は、上記製品のマニュアルまたはヘルプを参照してください。

### C.3.11 PowerSORT.ppjの補足説明

---

テキストファイル内のデータを、PowerSORT OCXを使って並べ替えるサンプルプログラムです。このサンプルプログラムを実行するには、『富士通 PowerSORT V2.0L20以降』のPowerSORT OCXが必要です。この製品がシステムにインストール（登録）されていない場合、プロジェクトファイルを正しく開き、実行することができません。PowerSORT OCXについての詳細は、上記製品のマニュアルまたはヘルプを参照してください。

### C.3.12 TreeView.ppjの補足説明

---

TreeViewコントロールの基本的な使用方法を示したサンプルプログラムです。サンプルプログラムの機能を以下に示します。

#### 指定した文字列をTextにもつノードを追加

文字列を入力し[追加] ボタンをクリックすると、選択されているルートまたはノードに、入力した文字列をTextプロパティにもつ子ノードを作成することができます。

#### 隠れているノードをツリービュー上に展開

前述の操作で、いくつかの子ノードを追加したあと、1から5までのルートのど

れかの番号と、その子ノードの番号を入力し[ 展開 ]ボタンをクリックすると、ノードをツリーに展開することができます。

**指定したノードの削除**

1から5までのルートのどれかの番号と、その子ノードの番号を入力し[ 削除 ]ボタンをクリックすると、ツリーからノードを削除することができます。入力した番号に該当するノードが存在しない場合、「指定したノードオブジェクトが存在しません」という内容のメッセージボックスが表示されます。

**指定したルートの子ノードをすべて削除**

1から5までのルートのどれかの番号を入力し[ 削除 ]ボタンをクリックすると、そのルートの子ノードすべてを削除することができます。

# ***MEMO***



---

## 付録D こんなことがしたい - ノウハウ集

### 実行時のメッセージ表示を抑止したい

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | 実行時に、エラーメッセージが表示されます。無視しても問題がない処理なので、実行時のメッセージが表示されないようにしたいのですが、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                    |
| 操作方法 | <p>実行時、PowerCOBOLのランタイムシステムは、メソッドの呼び出しやプロパティへの設定処理で、不当な処理があった場合、メッセージを表示します。たとえば、3つのコントロールを配列化し、その手続き中でコントロールのインデックスとして4以上の値を使用したり、上限値や下限値をもつプロパティに対して、範囲外の値を設定したりした場合などです。</p> <p>これらのメッセージは、本来プログラム上で修正すべきものですが、すでにアプリケーションが運用状態にあり、プログラム修正ができないような場合、かつ、そのエラーが動作上問題を引き起こさないような場合、メッセージの表示だけを抑止できます。</p> <p>メッセージを抑止するには、F5DDSTEV.EXEを利用します。F5DDSTEV.EXEの使用方法については、「<a href="#">アプリケーションの動作環境を設定する</a> ( p132 )」を参照してください。</p> |

### ダブルクリックで編集ウィンドウを表示したい

|      |                                                                                                                                                                                                                                                  |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | プロジェクトウィンドウのデザインツリーウィンドウで、フォームをダブルクリックすると、フォームを構成する要素のリストが開閉されます。また、イベント手続きをダブルクリックしても何も起こりません。ダブルクリックの操作で、フォーム編集ウィンドウを表示したり、手続き編集ウィンドウを表示したりするには、どうすればよいでしょうか？                                                                                  |
| 操作方法 | <p>ダブルクリックの動作を変更したい場合は、以下のように操作します。</p> <ol style="list-style-type: none"><li>1. [ ツール ] メニューの [ オプション ] コマンドを選択します。</li><li>2. [ デザイン ] タブをクリックします。</li><li>3. [ ダブルクリックの動作 ] から [ オブジェクトのデフォルト動作 ] を選択します。</li><li>4. OKボタンをクリックします。</li></ol> |

---

### プロジェクトのひな型（テンプレート）を作成したい

|      |                                                                                                                                                                                                                                                                                                                                                                  |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | プロジェクトを新規に作成する場合、標準フォーム、標準ダイアログまたはActiveX以外のプロジェクトのテンプレートを作成しておくには、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                 |
| 操作方法 | <p>テンプレートとなるプロジェクトファイルを最初に作成し、そのプロジェクトファイルをWindowsのエクスプローラなどを使って、テンプレート用のフォルダに保存します。テンプレート用のフォルダは、以下の操作で設定します。</p> <ol style="list-style-type: none"> <li>1. [ツール]メニューの[オプション]コマンドを選択します。</li> <li>2. [プロジェクト]タブの[テンプレートフォルダ]にテンプレート用のフォルダを入力します。</li> <li>3. または、右側の[...]ボタンをクリックし、[フォルダの参照]ダイアログボックスでテンプレート用のフォルダを選択します。</li> <li>4. OKボタンをクリックします。</li> </ol> |

### 複数のプロジェクトを一括してビルドしたい


|      |                                                                                     |
|------|-------------------------------------------------------------------------------------|
| 処理   | 複数のプロジェクトファイルを一括してビルドするには、どうすればよいでしょうか？                                             |
| 操作方法 | PowerCOBOLのバッチビルド機能を使用してください。詳細は、「 <a href="#">バッチモードでビルドする</a> ( p128 )」を参照してください。 |

### フォームの大きさに合わせてコントロールの大きさも変更したい

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | フォームの大きさを変更したとき、フォームの拡大縮率に応じて、フォーム上のコントロールの大きさも拡大または縮小するには、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 操作方法 | <p>フォームの大きさに応じてコントロールの大きさを変更するには、以下のようになります。</p> <ol style="list-style-type: none"> <li>1. フォームのプロパティ設定ダイアログボックスを表示します。</li> <li>2. [座標]タブをクリックします。</li> <li>3. [スケーラブル]をチェック状態にします。</li> <li>4. OKボタンをクリックします。</li> </ol> <p>次に、コントロールごとに拡大縮率を変更したい場合は、以下の設定をしてください。</p> <ol style="list-style-type: none"> <li>1. コントロールのプロパティ設定ダイアログボックスを表示します。</li> <li>2. [座標]タブをクリックします。</li> <li>3. [スケーリングスタイル]から、拡大縮率する方法を選択します。</li> <li>4. OKボタンをクリックします。</li> </ol> <p>スケーリングスタイル (ScalingStyleプロパティに対応) の詳細は、『リファレンス』を参照してください。</p> |

| 画面の解像度に合わせてフォームとコントロールの大きさを変更したい |                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                               | 作成したアプリケーションを異なった解像度の画面で、同じ大きさで使用するにはどうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                 |
| 操作方法                             | <p>前述の項目と同じように、まず、フォームの大きさに応じてコントロールの大きさが変更されるよう、[ スケーラブル ] および [ スケーリングスタイル ] を設定します。</p> <p>次に、以下のように操作します。</p> <ol style="list-style-type: none"> <li>1. フォームのプロパティ設定ダイアログボックスを表示します。</li> <li>2. [ フォーム ] タブの [ 初期状態 ] から、"1 - 最大化"を選択します。</li> <li>3. OKボタンをクリックします。</li> </ol> <p>この設定により、異なった解像度の画面でも、つねにフォームは最大化表示され、その大きさに合わせて、コントロールの大きさも自動的に変更されます。</p> |

| フォームへのキーボード入力操作に対応する処理を実行したい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                           | フォームへのキーボード入力操作に対応して、処理を実行するにはどうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 操作方法                         | <p>フォームのPreKeyDownイベントまたはPreKeyUpイベントに手続きを記述してください。</p> <p>たとえば、フォームがフォーカスをもっている状態で、[Ctrl+Shift+L]が押された場合に処理を実行するには、以下のように記述します。PreKeyDownイベントとPreKeyUpイベントのパラメタの詳細は、『リファレンス』を参照してください。</p> <p><b>MainForm-PreKeyUp</b></p> <pre> ENVIRONMENT    DIVISION. DATA           DIVISION. WORKING-STORAGE SECTION. LINKAGE        SECTION. 01  POW-KEYCODE PIC S9(4) COMP-5. 01  POW-SHIFT   PIC S9(4) COMP-5. PROCEDURE      DIVISION USING POW-KEYCODE POW-SHIFT.     IF POW-KEYCODE = POW-KEY-L AND POW-SHIFT = 3 THEN     ... ([Ctrl + Shift + L]キーが押された場合の処理を記述します)     END-IF </pre> |

| メッセージボックスに表示する文字列を途中で改行したい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                         | DisplayMessageメソッドでメッセージボックスを表示する場合、メッセージ文字列の途中で改行したいのですが、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 操作方法                       | <p>文字列の改行する位置に、改行を示す制御文字(X"0A")を設定してください。以下に例を示します。</p> <pre> ENVIRONMENT      DIVISION. DATA              DIVISION. WORKING-STORAGE SECTION. 01 MSG-TEXT      PIC X(80). PROCEDURE        DIVISION.      MOVE "改行された¥nメッセージ¥nです" TO MSG-TEXT *   "¥n"を、空白と改行を示す制御文字に置換します。     INSPECT MSG-TEXT REPLACING ALL "¥n" BY X'200A'     INVOKE POW-SELF "DisplayMessage" USING                         MSG-TEXT "サンプル" POW-DMICONINFORMATION </pre> <p>この手続きを実行すると、以下のようなメッセージボックスが表示されます。</p>  |

| 複数のメニューバーを実行時に使い分けたい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                   | 実行時に、フォームの状態、コントロールの選択内容や設定内容により、メニューバーの内容を使い分けたい場合は、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 操作方法                 | <p>メニューバーの一部のメニュー項目だけを無効状態（淡色表示）したい場合は、そのメニュー項目のEnabledプロパティの値に偽(POW-FALSE)を設定してください。</p> <p>また、メニューバー全体を変更したい場合は、以下のように、複数のメニューを作成し、フォームのMenuBarNameプロパティを利用して使い分けてください。</p> <ol style="list-style-type: none"> <li>1. フォームのポップアップメニューの[メニュー編集]サブメニューから、[メニュー作成]を選択します。</li> <li>2. 作成したメニューの名前を変更します。（たとえば、Menu1）</li> <li>3. メニュー編集ウィンドウを使って、Menu1のメニュー項目を作成します。</li> <li>4. 再度、フォームのポップアップメニューの[メニュー編集]サブメニューから、[メニュー作成]を選択します。</li> <li>5. 新しく作成したメニューの名前を変更します。（たとえば、Menu2）</li> <li>6. メニュー編集ウィンドウを使って、Menu2のメニュー項目を作成します。</li> <li>7. メニューを切り替えたいタイミングで発生するイベントを選択し、手続き編集ウィンドウを開きます。</li> <li>8. フォームのMenuBarNameプロパティに、"Menu1"または"Menu2"を設定します。</li> </ol> <p>メニュー項目を作成する方法については、「<a href="#">メニューを作成する</a>（ p66）」を参照してください。</p> |

| フォームを閉じる直前に入力内容をチェックしたい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                      | フォームを閉じる直前に、フォームへの入力内容をチェックし、入力内容によって閉じることをキャンセルしたい場合、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 操作方法                    | <p>フォームを閉じる直前のチェックは、フォームのQueryCloseイベントに手続きを記述してください。</p> <p>以下に例を示します。QueryCloseイベントのパラメタの詳細は、『リファレンス』を参照してください。</p> <p><b>MainForm-QueryClose</b></p> <pre> ENVIRONMENT    DIVISION. DATA            DIVISION. WORKING-STORAGE SECTION. LINKAGE        SECTION. 01  POW-CANCEL PIC S9(4) COMP-5. PROCEDURE      DIVISION USING POW-CANCEL.  *   テキストボックスコントロールへの入力がないと、 *   メッセージボックスを表示し、再入力を促します。 IF "Text" OF CmText1 = SPACE THEN     INVOKE POW-SELF "DisplayMessage"         USING "テキストを入力してください"     MOVE POW-TRUE TO POW-CANCEL     INVOKE CmText1 "SetFocus"     EXIT PROGRAM END-IF ... (同様にその他のチェックをします) </pre> |

| 処理に時間がかかるイベント手続きを実行中に別の処理を実行させたい |                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                               | イベント手続き中で、時間がかかる処理を実行中、処理内で更新したプロパティをフォームに反映させたり、キャンセル処理を実行させたりするには、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                          |
| 操作方法                             | <p>イベント手続きが実行されると、そのイベントを実行するために、アプリケーションの制御権を独占してしまいます。したがって、このイベント内での処理によって新しく発生するイベントは、このイベントが終了するまで待ち状態となります。</p> <p>新しく発生した待ち状態のイベントを先に実行させるために、PowerCOBOLではフォームのThruEventsメソッドを用意しています。このメソッドを呼び出すことにより、実行中のイベントに対するバックグラウンド動作が可能となります。</p> <p>たとえば、フォームの表示を更新する、処理をキャンセルする、タイマコントロールのタイマイベントを円滑に発生させるといったことができます。</p> <p>ThruEventsメソッドについては、『リファレンス』を参照してください。</p> |

| タブコントロールのページ変更前に入力をチェックしたい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                         | <p>タブコントロールを使って、複数のページを作成しています。別のタブをクリックし、ページを切り替えるとき、切り替え前のページの入力内容をチェックして切り替えを抑止したい場合は、どうすればよいでしょうか？</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 操作方法                       | <p>このような場合は、タブコントロールのBeforeClickイベントで、入力内容をチェックします。</p> <p>以下に例を示します。BeforeClickイベントのパラメタの詳細は、『リファレンス』を参照してください。</p> <p><b>CmTab1-BeforeClick</b></p> <pre> ENVIRONMENT    DIVISION. DATA           DIVISION. WORKING-STORAGE SECTION. LINKAGE        SECTION. 01  POW-PAGEOLDINDEX PIC S9(9) COMP-5. 01  POW-CANCEL      PIC S9(4) COMP-5. PROCEDURE      DIVISION USING POW-PAGEOLDINDEX POW-CANCEL. *   ページごとに入力内容をチェックします。 EVALUATE POW-PAGEOLDINDEX *   1ページめの内容をチェックします。 WHEN 1 *   テキストボックスコントロールへの入力がなければ、 *   メッセージボックスを表示し、ページを切り替えません。     IF "Text" OF CmText1 = SPACE THEN         INVOKE POW-SELF "DisplayMessage"             USING "テキストを入力してください"         MOVE POW-TRUE TO POW-CANCEL         EXIT PROGRAM     END-IF     ... (同様にその他のチェックをします) *   すべてのチェックがOKであれば、ページを切り替えます。     MOVE POW-FALSE TO POW-CANCEL WHEN 2     ... (同様に2ページめの内容をチェックします) WHEN 3     ... END-EVALUATE. </pre> |

### 表コントロールで見出し以外のセルの内容をすべてクリアしたい

|      |                                                    |
|------|----------------------------------------------------|
| 処理   | 表コントロールで、見出しを除くすべてのセルの内容を、簡単にクリアするには、どうすればよいでしょうか？ |
| 操作方法 | ClearTableメソッドを使ってください。                            |

### テキストボックスへの入力内容をチェックしたい

|      |                                                                                                                                                                                                                                                        |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | 数値を入力するためにテキストボックスコントロールを配置しました。コントロールのプロパティ設定ダイアログボックスを使って、テキスト(Textプロパティ)を削除し、テキスト属性のテキスト種別を"1 - COBOL PICTURE属性"、PICTURE文字列に数値項目を定義しました。<br>このテキストボックスの値を実行時に参照した場合、未入力状態であっても0となります。0は有効な値として扱いたいのですが、入力がなかったのか、それとも0が入力されたのかを判定するには、どうすればよいでしょうか？ |
| 操作方法 | 判定する手続きの中で、Textプロパティではなく、DisplayTextプロパティを参照してください。DisplayTextプロパティは、実際にテキストボックスに表示されている文字列を参照することができます。したがって、未入力の場合、値は空白となります。                                                                                                                        |

### テキストボックスのテキストを右端で折り返して編集したい

|      |                                                                                                                                                                       |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理   | テキストボックスを複数行入力(MultiLineプロパティをTRUE)にした場合、入力したテキストが編集領域の右端で折り返されるようにしてテキストを編集するには、どうすればよいでしょうか？                                                                        |
| 操作方法 | テキストを右端で折り返すようにしたい場合は、以下のように操作します。<br>1. テキストボックスコントロールのプロパティ設定ダイアログボックスを開きます。<br>2. [ マルチライン ] タブをクリックします。<br>3. [ 水平方向に自動でスクロール ] のチェックをはずします。<br>4. OKボタンをクリックします。 |



| テキストボックスで8キロバイトを超えるテキストを操作したい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                            | テキストボックスのTextプロパティは、X(8192)と定義されていますが、8キロバイトを超えるテキストを扱うには、どうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 操作方法                          | <p>LoadFileメソッドを使って、ファイルからテキストを読み込む場合は、8キロバイトを超えていても表示できます。ただし、この場合、その他のメソッドやプロパティを使って、テキストを操作できません。</p> <p>テキストを操作するには、POWERCONVTOCOMを使います。テキストボックスコントロールをCOMオブジェクトとして扱い、NetCOBOLのCOM連携機能を使って操作することができます。</p> <p>POWERCONVTOCOMの記述形式については、「<a href="#">COBOLの*COMクラスを利用してアクセスする</a> ( p198) 」を参照してください。</p> <p>以下に使用例を示します。</p> <pre> <b>CONFIGURATION SECTION</b>     REPOSITORY.     CLASS COM AS "COM"  <b>イベント手続き</b>     ENVIRONMENT      DIVISION.     DATA             DIVISION.     WORKING-STORAGE  SECTION.     01 WK-TEXT        PIC X(10000).     01 COM-CMTEXT     USAGE OBJECT REFERENCE COM.     PROCEDURE         DIVISION.      ...      *   テキストボックスコントロールをCOMオブジェクトに変換します。         CALL "POWERCONVTOCOM" USING CmText1 USING COM-CMTEXT.     *   COM連携機能を使って、10000バイトのテキストを参照します。         INVOKE "GET-TEXT" RETURNING WK-TEXT     *   COMオブジェクトを解放します。         SET COM-CMTEXT TO NULL.      ... </pre> |

| カレントフォルダを変更したい |                                                                                                                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理             | ファイルを取り扱う場合に、カレントフォルダを設定しておくにはどうすればよいでしょうか？                                                                                                                                                                                                                                               |
| 操作方法           | <p>フォルダリストコントロールを利用することにより、カレントフォルダを設定することができます。設計時にフォルダリストコントロールのVisibleプロパティをFalseにしておき、手続き中でカレントフォルダにしたいパスをPathプロパティに設定してください。</p> <p>また、GetFolderNameメソッドを使用してフォルダ名を取得する場合には、第1パラメタにフォルダ名を指定することにより、最初に開かれるフォルダを変更でき、第3パラメタをTrueにすることにより、取得したフォルダを、以降の処理でのカレントフォルダとして設定することができます。</p> |

| 入力されたキーの文字コードを無効にしたい |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 処理                   | 入力されたキーの文字コードを取り扱う場合に、特定の文字コードを無効にしたい場合にはどうすればよいでしょうか？                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 操作方法                 | <p>フォーム全体に対して、特定の文字コードを無効にする場合は、フォームのPreKeyPressイベントの中で、文字コードを示すKeyAsciiパラメタを識別し、無効にする文字コードであれば、KeyAsciiパラメタに0を設定してください。</p> <p>また、コントロールごとに特定の文字コードを無効にする場合は、各コントロールのKeyPressイベントの中で、文字コードを示すKeyAsciiパラメタを識別し、無効にする文字コードであれば、KeyAsciiパラメタに0を設定してください。</p> <p>たとえば、テキストボックスコントロールのKeyPressイベントに以下のように記述した場合、数字の1の入力を無効にすることができます。</p> <pre> <b>KeyPress</b> ENVIRONMENT      DIVISION. DATA              DIVISION. WORKING-STORAGE SECTION. LINKAGE           SECTION. 01 POW-KEYASCII PIC S9(4) COMP-5. PROCEDURE         DIVISION USING POW-KEYASCII.     IF POW-KEYASCII = POW-KEY-1 THEN         MOVE 0 TO POW-KEYASCII     END-IF </pre> |

---

| V3.0以前のグループアイテムをフレーム(Frame)コントロールに変換したい |                                                                                   |
|-----------------------------------------|-----------------------------------------------------------------------------------|
| 処理                                      | V3.0以前のプロジェクトを変換するとき、グループアイテムをグループボックスコントロールではなく、フレームコントロールに変換したい場合はどうすればよいでしょうか？ |
| 操作方法                                    | 「 <a href="#">資産の移行方法</a> ( p310) 」の「グループアイテムの変換方法の指定」を参照してください。                  |

# ***MEMO***

---

## 付録E 困ったときの対処方法 - Q & A 集

| コントロールの使用方法がわからない |                                                                                                                                                                                                                                                                                 |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容           | コントロールの使用方法がわかりません。                                                                                                                                                                                                                                                             |
| 対処方法              | <b>サンプルプログラムを参照してください</b><br>PowerCOBOLでは、コントロールの使用方法を示したサンプルプログラムを提供しています。サンプルプログラムは、PowerCOBOLをインストールしたフォルダの"Samples¥PowerCOB"フォルダに、種類ごとにわかれて格納されています。サンプルプログラムの使用方法については、「 <a href="#">サンプルプログラムについて</a> ( p321 )」を参照してください。<br>また、プログラミング方法についての詳細は、「第3部 プログラミング編」をお読みください。 |

| PowerCOBOLで作成したアプリケーションを別のコンピュータ上で動作させることができない |                                                                                                                                                                                           |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                                        | PowerCOBOLで作成したアプリケーションを、別のコンピュータにコピーし実行させようとしたのですが、実行できませんでした。                                                                                                                           |
| 対処方法                                           | <b>PowerCOBOL ランタイムシステムをインストールしてください</b><br>PowerCOBOLで作成したアプリケーションを実行するには、「NetCOBOLシリーズ」に含まれるNetCOBOLのランタイムシステムとPowerCOBOLのランタイムシステムが必要です。アプリケーションを実行する前に、これらのランタイムシステムを正しくインストールしてください。 |

---

| アプリケーション実行中、PowerCOBOLのエラーメッセージが表示された |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                               | <p>アプリケーション実行中、以下のようなエラーメッセージが表示されました。</p> <p>オブジェクトが設定されていません。</p> <p>メソッドの呼び出しまたはプロパティの設定 / 参照時の引数に誤りがあります。</p> <p>未サポートのデータ型を使用しています。</p> <p>メソッドの呼び出しまたはプロパティの設定 / 参照においてエラーが発生しました。</p> <p>無効な関数呼び出しです。</p> <p>無効なプロパティ配列インデックスです。</p> <p>実行時に、このプロパティに値を設定することはできません。</p> <p>プロパティ配列インデックスが必要です。</p> <p>実行時に、このプロパティ値を参照することはできません。</p>                                                                                                                                                                  |
| 対処方法                                  | <p><b>デバッガを使ってエラー発生箇所を確認してください</b></p> <p>以下のような手順で、エラーが発生した位置を判定し、対応するプログラムが正しいか確認してください。</p> <ol style="list-style-type: none"> <li>1. エラーが発生したプロジェクトのビルドモードをデバッグモードにします。</li> <li>2. プロジェクトをビルドします。</li> <li>3. デバッガを起動して、アプリケーションを実行します。</li> <li>4. アプリケーションがエラーメッセージを表示したら、[ 実行の中断 ] コマンドを選択します。</li> </ol> <p>呼び出し経路ウィンドウに、エラーが発生した位置が表示されます。</p> <p>ビルドモードの変更方法については、「<a href="#">ビルドモードを使い分ける</a> ( p119 )」を参照してください。</p> <p>デバッガの使用方法については、「<a href="#">アプリケーションをデバッグしよう</a> ( p141 )」を参照してください。</p> |

| アプリケーションが異常終了した |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容         | アプリケーション実行中、アプリケーションエラーで異常終了しました。                                                                                                                                                                                                                                                                                                                                                                                 |
| 対処方法            | <p><b>デバッガまたは診断機能を使って異常発生箇所を確認してください</b></p> <p>デバッグモードの場合は、デバッガを起動してアプリケーションを実行させてください。アプリケーションが異常終了したら、[ 実行の中断 ] コマンドを選択してください。呼び出し経路ウィンドウに異常終了した位置が表示されていますので、対応するプログラムが正しいか確認してください。デバッガの使用方法については、「<a href="#">アプリケーションをデバッグしよう</a> ( p141 )」を参照してください。</p> <p>リリースモードの場合は、PowerCOBOLの診断機能を使って、異常が発生したプログラムの箇所を特定し、プログラムに異常がないか確認してください。</p> <p>診断機能については、「<a href="#">診断機能を利用する</a> ( p134 )」を参照してください。</p> |

| スタックオーバーフローが発生した |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容          | 実行時、スタックオーバーフローにより、アプリケーションエラーが発生しました。                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 対処方法             | <p><b>再帰的なイベントが発生していないか確認してください</b></p> <p>イベント手続き中で、そのイベントが再び発生するような手続きを記述していないか確認してください。たとえば、テキストボックスコントロールのChangeイベント中で、そのテキストボックスコントロールのTextプロパティに値を設定すると、Changeイベントが発生します。</p> <p>このような場合、Changeイベントが再帰的に発生し、無限ループとなります。この状態に陥ると、スタックオーバーフローによるアプリケーションエラーが発生します。</p> <p>再帰的なイベントが発生しないで、スタックオーバーフローが発生した場合は、リンクオプション"STACK:スタックサイズ"を設定してください。リンクオプションの設定方法については、「<a href="#">リンクオプション</a> ( p127 )」を参照してください。リンクオプションの詳細については、『NetCOBOL 使用手引書』を参照してください。</p> |

| デバッグでエラーメッセージが表示された |                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容             | デバッグ開始時またはデバッグ中に、以下のエラーメッセージが表示されました。<br>「デバッグ情報ファイルに誤りがあります。ファイル=\$1」(\$1はデバッグ情報ファイル名)                                                                             |
| 対処方法                | <b>#INCLUDE文およびCOPY文が正しく記述されているか確認してください</b><br>プログラム中で、#INCLUDE文およびCOPY文が正しく記述されているか確認してください。#INCLUDE文の記述規則については、「 <a href="#">登録集ファイルの利用方法</a> ( p183)」を参照してください。 |

| COPY文で定義した登録集ファイル内のデータが認識されない |                                                                                                                                                                                                                                        |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                       | COPY文で定義した登録集ファイル内のデータが、ビルド時に認識されません。                                                                                                                                                                                                  |
| 対処方法                          | <b>#INCLUDE文を使ってください</b><br>登録集ファイル内のデータをプロパティへのアクセスやメソッドの呼び出しに使用する場合、登録集ファイルは#INCLUDE文を使って定義する必要があります。登録集ファイル内のデータが認識されない場合、ビルド時に「F5DD8103041-W XXXの項類を英数字とみなしました」のエラーになります。<br>詳細は、「 <a href="#">登録集ファイルの利用方法</a> ( p183)」を参照してください。 |

| 診断機能の診断結果が正しく出力されない |                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容             | 診断機能の診断結果が正しく出力されず、以下のエラーメッセージが表示されました。<br>「デバッグ情報ファイル'\$1'内のプログラム'\$2'の情報に誤りがあるため、このプログラムに対する言語イメージの情報を出力できませんでした。」(\$1はデバッグ情報ファイル名、\$2はプログラム名)                    |
| 対処方法                | <b>#INCLUDE文およびCOPY文が正しく記述されているか確認してください</b><br>プログラム中で、#INCLUDE文およびCOPY文が正しく記述されているか確認してください。#INCLUDE文の記述規則については、「 <a href="#">登録集ファイルの利用方法</a> ( p183)」を参照してください。 |



| プロジェクトファイルが正しく開けなくなってしまった |                                                                                                                                                       |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                   | プロジェクトファイルを開こうとしたら、クラスがシステムに登録されていないという内容のメッセージが表示されました。開いたプロジェクトを展開すると、デザインツリーウィンドウに「インストールされていないコンポーネント」が現れます。                                      |
| 対処方法                      | <p><b>必要なActiveXコントロールがシステムにインストールされているか確認してください</b></p> <p>プロジェクトファイルで使用しているActiveXコントロールが、システム(Windows)に登録されていません。正しく登録またはインストールされているか確認してください。</p> |

| プロジェクトファイルを誤って削除してしまった |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                | 必要なプロジェクトファイル(ppj)を誤って削除してしまいました。                                                                                                                                                                                                                                                                                                                                                                                  |
| 対処方法                   | <p><b>バックアップファイルを使用してください</b></p> <p>プロジェクトファイルを保存すると、PowerCOBOLは、その直前に保存されていたプロジェクトファイルをバックアップ用として保存しています。バックアップファイルは、「プロジェクト名.ppj~」という名前でプロジェクトファイルと同じフォルダにあるので、拡張子をppjに変更してお使いください。ただし、バックアップ用のファイルは、プロジェクトのオプションで「バックアップファイルを保存する」がチェック状態の場合だけ保存されます。</p> <p>また、最後に更新した内容については反映されていないので、再度、プロジェクトファイルの編集が必要です。</p> <p>プロジェクトのオプションで、バックアップ用のフォルダを変更することもできます。</p> <p>プロジェクトのオプションについては、『リファレンス』を参照してください。</p> |

| テキスト属性の設定ができない |                                                                                                                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容        | コントロールのプロパティ設定ダイアログボックスで、[テキスト属性] タブの [テキスト種別] を変更したら、テキスト属性が設定できないという内容のメッセージが表示されました。                                                                                                      |
| 対処方法           | <p><b>コントロールのキャプションまたはテキストに矛盾がないか確認してください</b></p> <p>コントロールの [キャプション] または [テキスト] に設定されている文字列が、テキスト種別の項類と一致しているか確認してください。一致していない場合は、テキスト種別を変更できません。設定されている文字列を削除または変更してからテキスト種別を変更してください。</p> |

| モジュールを更新したのにビルドされない |                                                                                                                                                                                                                                                                                                           |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容             | モジュールを更新し、[プロジェクト] メニューの [ビルド] コマンドを選択してもビルドされません。また、ツールバーの [ビルド] ボタンをクリックしてもビルドされません。                                                                                                                                                                                                                    |
| 対処方法                | <p><b>更新したモジュールはデフォルトモジュールであるか確認してください</b></p> <p>[プロジェクト] メニューの [ビルド] コマンドおよびツールバーの [ビルド] ボタンは、デフォルトモジュールをビルドするために使用します。デフォルトモジュール以外の場合は、デザインツリーウィンドウ上で対象のモジュールを選択し、ポップアップメニューの [ビルド] コマンドを選択してください。また、[プロジェクト] メニューまたはプロジェクトのポップアップメニューの [すべてビルド] コマンドを選択することによっても、デフォルトモジュール以外のモジュールをビルドすることができます。</p> |

| エラーメッセージが表示されずにビルドが終了する |                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                 | モジュールをビルドしましたが、エラーメッセージが表示されずに、ビルドが失敗してしまいます。                                                                                                                                                                                                                                                                                                                              |
| 対処方法                    | <p><b>ファイル名またはファイルの属性を確認してください</b></p> <p>モジュールをビルドすると、PowerCOBOLはプロジェクトファイルが保存されているフォルダに、自動的にターゲットフォルダおよび作業用フォルダを作成し、その中に実行可能プログラムファイルやビルドやデバッグに必要な作業用ファイルを作成します。</p> <p>したがって、自動的に作成されるフォルダと同じ名前のファイルがすでに存在している場合や、作業用ファイルと同じ名前のファイルが書き込み禁止属性になっている場合などは、フォルダやファイルの生成ができないため、ビルドに失敗します。</p> <p>ビルド時に作成されるファイルについては、「<a href="#">ビルド時に作成されるファイル</a>( p120)」を参照してください。</p> |

| コマンドボタンコントロールの色が変更できない |                                                                                                                                                                                 |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                | コマンドボタンコントロールのプロパティ設定ダイアログボックスで文字色や背景色を変更しても、色が変わりません。また、手続きでForeColorプロパティやBackColorプロパティに値を設定しても、色が変わりません。                                                                    |
| 対処方法                   | <p><b>システム色を使うのチェックをはずしてください</b></p> <p>コマンドボタンコントロールのプロパティ設定ダイアログボックスで、[ コマンドボタン ] タブの [ システム色を使う ] のチェックをはずしてください。また、手続き中で変更する場合は、UseSystemColorプロパティにPOW-FALSEを設定してください。</p> |

| 共通内部プログラムにSQL文のカーソル宣言を記述したらエラーになった |                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                            | 共通内部プログラムを新規作成し、その中でSQL文のカーソル宣言をしました。別の共通内部プログラムやイベント手続き中で、宣言したカーソルを使用すると、翻訳時にエラーになってしまいました。                                                                                                                                                                                                                                                                                                              |
| 対処方法                               | <p><b>カーソル宣言はPROCEDUREに記述してください</b></p> <p>SQL文のカーソル宣言は、翻訳する外部プログラム中の内部プログラムの先頭に記述されている必要があります。PowerCOBOLでは共通内部プログラムを新規作成した場合、それらの内部プログラムが外部プログラム中のどの位置に生成されるか決まっていません。したがって、順番を意識しなければならない手続きを記述する場合は、フォームのPROCEDUREに内部プログラムを記述してください。</p> <p>PROCEDUREに記述された内部プログラムは、外部プログラムの先頭の位置に生成されます。PROCEDUREに共通内部プログラムを記述する場合、各プログラムには見出し部およびプログラム終わり見出しが必要です。また、COBOL85言語仕様の場合、プログラム名段落には、COMMON属性を付けてください。</p> |

| メソッドを呼び出したあとPROGRAM-STATUSを得ることができない |                                                                                                                                                                                                                                                                                            |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                              | メソッドを呼び出したあと、メソッドの実行結果を確認するためにPROGRAM-STATUSを参照しましたが、正しい値を得ることができませんでした。                                                                                                                                                                                                                   |
| 対処方法                                 | <p><b>PROGRAM-STATUSの参照前にプロパティの参照または設定をしていないか確認してください</b></p> <p>メソッドのパラメタにプロパティを使用していないか確認してください。また、メソッド呼び出しとPROGRAM-STATUSを参照する処理のあいだに、プロパティを参照または設定したり、他のメソッドを呼び出ししたりしていないか確認してください。</p> <p>プロパティを参照または設定すると、内部的にPowerCOBOLのランタイムシステム中のサブルーチンが呼び出されますので、PROGRAM-STATUSの値は変わってしまいます。</p> |

| 実行時にフォームを開くとエラーが表示されて開けない |                                                                                                                                                                                         |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                   | アプリケーションを実行中、他のフォームを開くとき、「DLLが見つかりません」という内容のメッセージが表示され、フォームを開くことができません。                                                                                                                 |
| 対処方法                      | <b>PATHが正しく設定されているか確認してください</b><br>開こうとしているフォームが別のDLLに含まれる場合、そのDLLの格納先フォルダがPATHに設定されているか確認してください。<br>また、フォームを含むDLLが、さらに別のDLLをインポート結合している場合、インポート結合されるDLLの格納先フォルダがPATHに設定されているか確認してください。 |

| システム色の値を参照すると桁あふれが発生する |                                                                                                                                                                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                | フォームやコントロールで使用されているシステム色の値を、S9(9)COMP-5で定義したデータに転記すると、桁あふれが発生しました。                                                                                                                                                          |
| 対処方法                   | <b>転記先のデータの宣言を変更してください</b><br>色は4バイトの数値を持っています。これに対応するCOBOLデータの属性は、S9(9)COMP-5となります。しかし、システムが定義している色は、4バイトの数値で10桁の長さを持つ場合があります。したがって、色を保存するデータを宣言する場合には9(10)のように10桁で宣言してください。システムが定義している色（システム色）については、『リファレンス』の付録を参照してください。 |

| ネットワーク上のプロジェクトファイルを別のネットワークに保存するとタイムアウトエラーが発生する |                                                                                                                                                                                                                                                |
|-------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                                         | ネットワーク上に格納してあるプロジェクトファイルを開き、そのプロジェクトを別のネットワーク上に名前を付けて保存しようとする、タイムアウトエラーが発生します。                                                                                                                                                                 |
| 対処方法                                            | <b>ローカルドライブにプロジェクトをいったん格納してください</b><br>ネットワーク上のプロジェクトファイルを、ローカルなドライブにいったん保存して、別のネットワーク上にコピーまたは移動してください。コピーまたは移動後、プロジェクトを開きなおして保存してください。<br>ただし、ActiveXコントロールを作成するプロジェクトは、名前をつけて保存すると、コントロールのクラスIDが変わってしまい、再利用できなくなりますので、ファイルを直接コピーまたは移動してください。 |

| 日本語定数がコントロールのキャプションやテキストに転記できない |                                                                                                                                                                                                                                                                                             |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                         | プロジェクトのランタイムコードセットをUnicodeにしている場合、日本語定数をコントロールのキャプションやテキストに転記すると、文字列が正しく転記できません。                                                                                                                                                                                                            |
| 対処方法                            | <p><b>定数が53文字を超えていないか確認してください</b></p> <p>定数の最大長は160バイトです。シフトJISコードの場合、日本語や半角カナの1文字の長さは2バイトであるため、文字定数として最大80文字まで定義できます。</p> <p>これに対し、Unicodeの場合、文字定数は内部的にUTF-8で管理されているため、日本語や半角カナの1文字の長さは最大3バイトになります。したがって、文字定数として定義できるのは、最大53文字となります。</p> <p>Unicodeについての詳細は、『NetCOBOL 使用手引書』を参照してください。</p> |

| 翻訳オプションにNOALPHALを指定しても英大文字と英小文字の区別ができない |                                                                                                                                                                                                          |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                                 | 翻訳オプションにNOALPHALオプションを指定しましたが、プログラムを英大文字と英小文字を区別して呼び出すことができません。                                                                                                                                          |
| 対処方法                                    | <p><b>ALPHAL(WORD)オプションを指定してください</b></p> <p>PowerCOBOLではNOALPHALオプションを指定することはできません。英大文字と英小文字を区別する場合は、ALPHAL(WORD)オプションを指定してください。オプションの設定方法については、「<a href="#">ビルド用のオプションを設定する</a> ( p123) 」を参照してください。</p> |

| コントロールのKeyDownイベントまたはKeyUpイベントの中で [ Tab ] キーを認識できない |                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| トラブルの内容                                             | コントロールのKeyDownイベントまたはKeyUpイベントの中で、パラメタで渡ってきたキーのコードを参照し、[ Tab ] キーの場合だけ処理をしたいのですが、[Tab]キーを押してもイベントが発生しません。                                                                                                                                                                                                            |
| 対処方法                                                | <p><b>フォームのPreKeyDownイベントまたはPreKeyUpイベントで処理してください</b></p> <p>[Tab]キーは、コントロール間のフォーカス移動のためにフォームで処理されてしまうため、コントロール側ではイベントが発生しないようになっています。このような特別なキーに対して処理したい場合には、フォームのPreKeyDownイベントまたはPreKeyUpイベントの中でキーのコードを判定し、処理するようにしてください。このとき、入力されたキーの対象となったコントロール（フォーカスをもっているコントロール）がどれであるかは、フォームのActiveControlプロパティで判定できます。</p> |

---

# 用語集

## ActiveXコントロール

以前は、OCX (OLE Custom ControlsまたはOLEコントロール) と呼ばれていたコントロールです。  
米国Microsoft CorporationのOLE(2.0)技術を利用したオブジェクトの1つです。  
ActiveXコントロールは、以下のような特徴を持っています。  
オートメーション機能を利用できます。  
イベントを発行できます。  
コンテナ機能をもつアプリケーションであれば、どんなアプリケーションでも利用できます。  
部品として利用でき、機能の拡張が簡単にできます。

### 関連項目

[ActiveXコントロールを作成する](#) ( p245)  
[ActiveXコントロールを使ったアプリケーションを作成する](#) ( p260)

## ADO

Microsoft ActiveX Data Objectsの略称です。米国Microsoft Corporationが提唱するデータアクセスインタフェースであるOLE DBをベースとした、データアクセスコンポーネントです。ADOを利用することにより、OLE DBインタフェースによるデータベースプログラミングが簡単になります。

### 関連項目

[ADOデータソースコントロールを利用してデータベースと連携する](#)  
( p287)

## API

Application Program Interfaceの略称です。  
Windows とWindows 上で動作するアプリケーションのプログラム間インタフェースを定義している関数の総称です。

## DLL

Dynamic Link Libraryの略称です。  
実行時に動的にリンクし、複数のプログラム間で共有することができるライブラリです。DLLを利用することにより、ディスクや実行時のメモリサイズを節約することができます。

### 関連項目

[DLLを作成するには](#) ( p119)  
[DLLを使ったアプリケーションを作成する](#) ( p238)

## EXEファイル

アプリケーションを起動するための実行可能なプログラムファイルです。

---

## GUI

Graphical User Interfaceの略称です。従来のメインフレームやオフコンで使っていた文字をベースとしたインタフェースではなく、ボタン、リストボックスやメニューなどのグラフィカルな部品をウィンドウ上に配置し、設計されたインタフェースです。

## OLE

Object Linking and Embeddingの略称です。異なる複数のアプリケーション間で、データを転送したり共有したりするために、Windows で提供されている技術の総称です。

OLEの機能をもつオブジェクトを、OLEオブジェクトまたはCOMオブジェクトと呼びます。

## アイコン

GUIをもつWindows など、アプリケーションや、そのアプリケーションによって作成されたオブジェクトなどを表した、小さなイメージです。

### 関連項目

[フォームのアイコンの設定例](#)( p97)

## アクセスキー

メニューの選択項目やボタンなどのコントロールをアルファベットキーと対応づけることで、そのメニュー項目を選択したり、コントロールをクリックしたりする操作と同じことができるようにするためのキーです。

アクセスキーを割り当てることで、マウスがなくてもアプリケーションの操作が簡単にできるようになります。

### 関連項目

[メニューを作成する](#)( p66)

## イベント

ActiveXコントロールが、自分自身に起こった状態の変化を通知するために発生するものです。イベントを受信したアプリケーションは、その変化に対応した処理を実行します。

たとえば、ボタンがクリックされた、入力データが変更された、キーが押されたなどの操作に対応してイベントが発生します。

## イベント駆動型

あるイベントを基点として、そのイベントが発生した場合のプログラムが実行される方式です。たとえば、イベントには、ボタンがクリックされた、入力データが変更された、キーが押されたなどといったものがあり、これらのイベントが発生したときの動作をプログラムとして記述していく方法です。

### 関連項目

[イベント駆動型プログラムスタイル](#)( p7)

## イベント手続き

イベント駆動型を利用してアプリケーションを開発する場合の、各イベントが発生したときに実行されるプログラムを記述した手続きです。



### オートメーション

OLEが持つ機能の1つです。異なるアプリケーション間で、あるオブジェクトが持つ機能を利用したり、オブジェクトの属性にアクセスできるようにしたりするための機能です。

### カスタムコントロール

PowerCOBOLで標準的に使用できるコントロールに対して、サードパーティなどが作成した独自の機能をもつActiveXコントロール（またはOCX）です。

### カレット

文字を入力する場合に、文字の入力位置を示すカーソル（ポインタ）です。PowerCOBOLでは、通常、細い縦線が点滅します。手続き編集ウィンドウなどで上書きモードになると、1文字ぶんの長方形が点滅します。

#### 関連項目

[手続き編集ウィンドウでの編集操作](#) ( p24)

### カーソル

カレットとマウスポインタの総称です。

### キャプション

ActiveXコントロール上に表示される文字列です。または、ウィンドウの上部（タイトル部分）に表示される文字列です。

### クライアント・サーバ型

処理の中心となるソフトウェアや共有するデータをもつサーバとそのサーバがもつソフトウェアやデータを、利用するクライアントに分けて運用システムを構築している方法です。

#### 関連項目

[DBアクセスコントロールを利用してデータベースと連携する](#) ( p277)  
[ADOデータソースコントロールを利用してデータベースと連携する](#)  
( p287)

### クリック

マウスのボタンを押して、すばやく離す動作です。

### コンテナ

ActiveXコントロールを配置することができるオブジェクトです。PowerCOBOLでは、フォームやグループボックスコントロールがコンテナとなります。

### コントロール

コンテナに配置して利用することができる部品です。通常は、ActiveXコントロール（またはOCX）のことを指します。

### ショートカットキー

[Ctrl]キーや[Alt]キーと、アルファベットキーやファンクションキーなどを組み合わせて、メニューで実行するコマンドを実行できるようにするためのキーです。ショートカットキーを使用することにより、キーボードからマウスへと手を移動することなく、キーボードだけで連続的な操作をすることができます。

#### 関連項目

[メニュー項目にショートカットキーを割り当てる](#) ( p111)

### スクリプト

アプリケーションの機能を補完するために、簡単な手続きで処理ができるようにするための簡易言語です。PowerCOBOLでは、COBOLの機能を補完し、コントロールのプロパティへアクセスしたり、メソッドを呼び出すことができるようにしたりする、COBOLスクリプトを使います。

### ステータスバー

ウィンドウ内の操作対象になっているメニュー項目やコントロールなどの状態や説明を表示するための領域です。通常は、ウィンドウの下端に配置されています。

#### 関連項目

[ステータスバーを使ったアプリケーションを作成する](#) ( p223)

### セパレータ

メニュー項目とメニュー項目のあいだにあって、それらを区切っている横線です。

### ダイアログボックス

いろいろな情報を設定するための、GUIをもつウィンドウです。通常、ダイアログボックスが閉じられるまで、ダイアログボックスの呼び出し元の操作はできなくなります。

### タブグループ

ウィンドウに配置されたコントロールのフォーカスを、[Tab]キーまたは[Shift+Tab]キーではなく、矢印キーで移動するために設定する属性です。タブグループ属性を設定すると、タブ順序に従って、次にタブグループ属性が付いているコントロールの直前までを1つのグループとし、そのグループ内を矢印キーで移動することができます。

#### 関連項目

[タブ順序とタブグループを設定する](#) ( p102)

### タブ順序

ウィンドウに配置されたコントロールのフォーカスを、[Tab]キーまたは[Shift+Tab]キーで移動する場合の順番です。

#### 関連項目

[タブ順序とタブグループを設定する](#) ( p102)

## タブストップ

ウィンドウに配置されたコントロールのフォーカスを、[Tab]キーまたは[Shift+Tab]キーで移動するために設定する属性です。タブストップ属性を設定することで、[Tab]キーまたは[Shift+Tab]キーでのフォーカス移動の対象とすることができます。

### 関連項目

[タブ順序とタブグループを設定する](#) ( p102)

## ダブルクリック

マウスのボタンを、連続して素早く2回クリックすることです。

## ツールバー

メニューバーのメニュー項目とボタンを対応づけることにより、いろいろなコマンドをマウスのクリックだけで実行できるようにしたものです。通常は、ウィンドウのメニューバーの直下に配置されます。

### 関連項目

[ツールバーを使ったアプリケーションを作成する](#) ( p226)

## データベース

データを読み込んだり、書き込んだりするだけではなく、データの検索や集中的な管理をするためのものです。Windows では、ODBC(Open DataBase Connectivity)インタフェースやADO(Microsoft ActiveX Data Objects)を用いることにより、アプリケーションから同一のインタフェースで、異なったデータベースにアクセスすることができます。

### 関連項目

[DBアクセスコントロールを利用してデータベースと連携する](#) ( p277)

[ADOデータソースコントロールを利用してデータベースと連携する](#)  
( p287)

## ドラッグ

GUIをもつCOMオブジェクト上でマウスのボタンを押し、ボタンを押したまま任意の位置までマウスポインタを移動することです。この状態で、マウスのボタンを離すと、ドラッグアンドドロップしたことになります。

## ドラッグアンドドロップ

GUIをもつCOMオブジェクト上でマウスのボタンを押し、ボタンを押したまま任意の位置までマウスポインタを移動して（ここまでがドラッグ）、マウスのボタンを離す（ドロップ）ことです。

## ドロップ

GUIをもつCOMオブジェクト上でマウスのボタンを押し、ボタンを押したまま任意の位置までマウスポインタを移動した状態で、マウスのボタンを離す動作をいいます。すべての動作をあわせると、ドラッグアンドドロップになります。

## ビルド

実行可能プログラムは、ソースプログラムを翻訳し、必要なリソースとともにリンクして作成することができます。一般的には、この実行可能プログラムを作成することをビルドといいます。

PowerCOBOLでは、実行可能プログラムを作成するためのプロジェクト構成要素の更新状態（更新日時）を判定し、必要なものだけ翻訳およびリンクすることをビルドといいます。

### 関連項目

[ビルド・実行する](#) ( p88)

## フォーカス

複数のウィンドウや複数のコントロールの中で、キーボードの入力を受け取ることができる対象のことを、フォーカスがもっているウィンドウまたはコントロールといいます。フォーカスをもっているウィンドウをアクティブなウィンドウといいます。また、フォーカスをもっているコントロールは、キャプション部分が点線で囲まれます。

## フォーム

アプリケーションのGUIを実現するためのウィンドウです。PowerCOBOLでは、フォームはモジュールの構成要素であり、COBOLコンパイラでのプログラムの翻訳単位となります。

## フォント

文字を表示するときの字体です。

## 複合ファイル

コンパウンドドキュメントともいいます。

アプリケーションで扱う、さまざまな種類のデータを1つのファイルとして構成したものです。たとえば、PowerCOBOLでは、フォーム編集ウィンドウで扱うデータや手続き編集ウィンドウで扱うデータを1つのファイルにまとめて保存しています。

複合ファイルの中は、あたかもその中にフォルダとファイルが存在するように、データの種類ごとに分けられ、階層構造をもって管理されています。

## プロジェクト

アプリケーションを作成したり管理したりするために必要な情報がまとめられたもので、開発資産となるファイルの単位です。PowerCOBOLでは、1つまたは複数のモジュールで構成されており、複合ファイルとして保存されます。

### 関連項目

[プロジェクトの構成](#) ( p120)

## プロパティ

COMオブジェクトがもつ属性です。

オブジェクトのキャプション、文字のフォントや色など、GUIに使われる見ためや、オブジェクトの動作方法を定義するための情報です。

### 関連項目

[プロパティへのアクセス方法](#)( p171)

## マウスポインタ

マウスカーソルまたは、単にカーソルともいいます。

マウスの操作対象位置を示す小さいイメージです。通常、矢印の形をしています。文字列編集が可能な位置では、アルファベットの"I"に似た形をしています。

## メソッド

COMオブジェクトがもち、利用者から呼び出すことができる（公開されている）サブルーチンです。

### 関連項目

[メソッドの呼び出し方法](#)( p180)

## メニュー

Windows でのコマンド入力方法の1つです。

通常、ウィンドウの上端にメニューバーとして配置され、つねにメニューバーに表示されているメニューをトップレベルメニューといいます。トップレベルメニューは、コマンドの種類や目的ごとに分類され、その中に複数のメニュー項目があります。

## モジュール

実行可能なプログラムファイル(EXEファイルやDLLファイル)を作成する単位です。

PowerCOBOLでは、モジュールは1つまたは複数のフォームと、必要に応じて、COBOLファイル、オブジェクトファイル、ライブラリファイルおよびリソースファイルで構成されます。

## モーダルウィンドウ

ウィンドウを閉じるまで、呼び出し元のウィンドウの操作ができないウィンドウです。通常、ダイアログボックスはモーダルウィンドウです。

### 関連項目

[複数ウィンドウをもつアプリケーションを作成する](#)( p204)

[OpenFormメソッドを使用する](#)( p204)

## モードレスウィンドウ

ウィンドウを閉じなくても、呼び出し元のウィンドウの操作ができるウィンドウです。

### 関連項目

[複数ウィンドウをもつアプリケーションを作成する](#) ( p204)

[CallFormメソッドを使用する](#) ( p211)

---

# 索引

## A

Access 2000 ..... 287  
AccessToObject.ppj ..... 200  
Activateメソッド ..... 244, 265, 266, 273  
ActiveControlプロパティ ..... 360  
ActiveXコントロール ..... 186, 243, 245,  
260, 270, 275, 309, 355  
ADD文 ..... 173, 177, 185  
ADO ..... 287  
ADODataSource1.ppj ..... 288, 294  
ADODataSource2.ppj ..... 297  
ADODataSource3.ppj ..... 298  
ADODataSource4.ppj ..... 301  
ADOデータソースコントロール ..... 287  
ADOデータ連携コントロール ..... 305  
Animation.ppj ..... 98  
Array1.ppj ..... 194  
Array2.ppj ..... 196

## B

BASED-STORAGE ..... 168  
BeforeClickイベント ..... 345

## C

CallFormメソッド ..... 211, 212  
CALL 一意名 ..... 239  
CALL文 ..... 309  
CHECK機能 ..... 140  
Class Name ..... 202  
ClearTableメソッド ..... 346  
CloseChildイベント ..... 205, 207, 212  
CLOSECHILDイベント ..... 313  
Closedイベント ..... 266, 267  
CloseForm メソッド ..... 212  
COBOL85言語仕様 ..... 213, 244  
COBOL以外の言語 ..... 243  
COBOLファイル ..... 96

## COBOLプログラムと組み合わせて

デバッグ ..... 160  
COM ..... 307, 309  
COMMON属性 ..... 170  
COMPUTE文 ..... 174, 178, 185  
\*COMクラス ..... 198, 202, 243, 244, 270  
COM連携機能 ..... 198, 202, 243, 244  
CONSTANT ..... 169  
COPY文 ..... 183, 184, 354  
CREATE-OBJECT ..... 273  
CreateProductsTable.ppj ..... 91

## D

DBAccess.ppj ..... 277  
DBアクセスコントロール ..... 277  
Deactivateメソッド ..... 265, 266  
DisplayMessageメソッド ..... 342  
DISPLAY文 ..... 176  
DLL ..... 119, 132, 205, 212, 238  
DLLの寿命 ..... 244  
DoModalメソッド ..... 244, 266, 273

## E

EVALUATE文 ..... 176  
ExecuteSyncメソッド ..... 274  
Executeメソッド ..... 274  
EXTERNALデータ ..... 213

## F

F5DDSTEV.EXE ..... 133, 310  
FILE ..... 168  
FILE-CONTROL ..... 167

## G

GetFolderNameメソッド ..... 348

## H

Hello.ppj ..... 28  
HTML文書 ..... 275

---

**I**

Icon.ppj.....97  
IF文.....175, 185  
#INCLUDE文.....170, 183, 191, 354  
Invokedイベント.....255  
INVOKE文.....310

**K**

KeyDownイベント.....360  
KeyPressイベント.....188, 348  
KeyUpイベント.....360

**L**

LINK.EXE.....127  
ListView.ppj.....198  
LoadFileメソッド.....347

**M**

MenuBarNameプロパティ.....343  
MOVE文.....173, 177, 185  
MSDE.....287  
MultiLineプロパティ.....346  
MultipleInstanceプロパティ.....246

**N**

NetCOBOLのオブジェクト指向  
  プログラミング.....122  
NetCOBOLの実行環境.....133  
NOALPHAL.....360

**O**

ODBC.....277  
ODBCデータソースアドミニストレータ  
  .....277  
OLE DB.....361  
OOCOBOL言語仕様.....122, 213, 246  
Openedイベント.....266, 267  
OpenFormメソッド.....204, 212  
OPENSHEETメソッド.....318

**P**

PATH.....133  
Pathプロパティ.....348  
PopupMenu.ppj.....222

POW-CHECK.....312  
POW-COOKIE.....208  
POW-ENABLE.....312, 318  
POWERACPTOUCS2.....188  
POWERCOCOB.....317  
PowerCOBOLを起動する.....15  
POWERCONVFROMCOM.....199  
POWERCONVTOCOM.....199, 203, 347  
POWERGETCONTROL.....214  
POWEROPENFORM.....240  
POWEROPENFORMUN.....240, 242  
POWEROPENFORMUX.....240, 241  
POWERUCS2TOACP.....188  
POW-FONTSIZE.....318  
POW-SELF.....172, 180, 204, 212  
POW-VISIBLE.....312, 318  
PreKeyDownイベント.....341, 360  
PreKeyPressイベント.....188, 348  
PreKeyUpイベント.....341, 360  
PROCEDURE.....170, 313, 358  
ProgID.....249  
PROGRAM-STATUS.....181, 190, 358

**Q**

QueryCloseイベント.....267, 344

**R**

RC.EXE.....127  
Refreshメソッド.....313  
REPOSITORY.....167  
ReturnCheck.ppj.....212  
RewriteCurRecordメソッド.....286  
RewriteRecordメソッド.....286

**S**

SampleActiveX.ppj.....246  
ScalingStyleプロパティ.....340  
SPECIAL-NAMES.....167  
SQL Server.....277, 287  
SQL文.....124, 358  
SQL文のカーソル宣言.....170  
STOP RUN文.....190  
SUBTRACT文.....174, 178, 185



## T

|                     |          |
|---------------------|----------|
| Table1.ppj.....     | 50       |
| Table2.ppj.....     | 205      |
| Table3.ppj.....     | 219      |
| Table4.ppj.....     | 223      |
| Table5.ppj.....     | 226      |
| Table6.ppj.....     | 231      |
| Textプロパティ.....      | 346, 347 |
| ThruEventsメソッド..... | 344      |
| Toolbar.ppj.....    | 230      |
| ToolboxBitmap.....  | 258, 262 |
| TreeView.ppj.....   | 198      |

## U

|                                |                   |
|--------------------------------|-------------------|
| Unicode.....                   | 99, 188, 240, 360 |
| UsingActiveX.ppj.....          | 261               |
| UsingAutomationServer.ppj..... | 270               |

## V

|                        |     |
|------------------------|-----|
| V3.0以前のPowerCOBOL..... | 307 |
| VT_BOOL.....           | 252 |
| VT_BSTR.....           | 185 |
| VT_BSTR型の変換方法.....     | 185 |
| VT_CY.....             | 185 |
| VT_CY型への変換方法.....      | 186 |
| VT_I2.....             | 185 |
| VT_VARIANT.....        | 185 |
| VT_VARIANT型の変換方法.....  | 186 |

## W

|                      |     |
|----------------------|-----|
| WebTimer.ppj.....    | 275 |
| Web上で利用.....         | 275 |
| WORKING-STORAGE..... | 168 |

## あ

|                      |     |
|----------------------|-----|
| アイコンの設定例.....        | 97  |
| アイテムの属性名.....        | 309 |
| アイテム名.....           | 312 |
| アウトプットウィンドウ.....     | 37  |
| アウトプレースアクティブ状態.....  | 266 |
| アウトプレースアクティベート.....  | 266 |
| アウトプレースデアクティベート..... | 266 |
| アクセスキー.....          | 44  |
| アプリケーションの作成手順.....   | 12  |

|                    |          |
|--------------------|----------|
| アプリケーションの動作環境..... | 132      |
| アンインストール.....      | 134, 259 |

## い

|                      |              |
|----------------------|--------------|
| 異常終了.....            | 353          |
| 1ステップだけ実行.....       | 152          |
| 位置を揃える.....          | 107          |
| 移動.....              | 76           |
| イベント.....            | 7            |
| イベント駆動型.....         | 191          |
| イベント駆動型プログラム.....    | 6            |
| イベント手続き.....         | 82, 166, 167 |
| イベント手続きの引数.....      | 190          |
| イベント手続きの編集.....      | 24           |
| イベント手続きの編集方法.....    | 72           |
| イベントの引数.....         | 166          |
| イベント発生順序.....        | 191          |
| イベントボックス.....        | 40           |
| イメージの幅.....          | 226          |
| イメージリスト.....         | 226          |
| 色.....               | 357          |
| 印刷.....              | 110, 117     |
| 印刷(Print)コントロール..... | 56           |
| インストーラを作成.....       | 133, 258     |
| インデント.....           | 115          |
| インプレースアクティブ状態.....   | 266          |
| インプレースアクティベート.....   | 266          |
| インプレースデアクティベート.....  | 267          |
| インポートライブラリ.....      | 238          |

## う

|                |     |
|----------------|-----|
| ウォッチ.....      | 156 |
| ウォッチウィンドウ..... | 37  |

## え

|                      |     |
|----------------------|-----|
| エディタを変更.....         | 118 |
| エラーがあったら.....        | 90  |
| エラーメッセージの表示・非表示..... | 132 |
| エラーを修正.....          | 29  |
| エントリ情報.....          | 239 |

## お

|                  |                    |
|------------------|--------------------|
| オートメーションサーバ..... | 243, 244, 270, 309 |
|------------------|--------------------|

|                   |               |
|-------------------|---------------|
| オブジェクト .....      | 197, 303, 306 |
| オブジェクト指向 .....    | 310           |
| オブジェクトのクラス名 ..... | 202           |
| オブジェクトファイル .....  | 96            |
| オブジェクトヘアクセス ..... | 197           |
| オブジェクトボックス .....  | 40            |
| オプション .....       | 92            |

## か

|                      |          |
|----------------------|----------|
| カーソルタイプ .....        | 281      |
| カーソルの同時実行制御 .....    | 281      |
| ガイド表示域 .....         | 40       |
| 開発時に必要なファイル .....    | 121      |
| 外部COBOLファイルを編集 ..... | 117      |
| 外部ファイル .....         | 96       |
| 外部ファイル名 .....        | 93       |
| 外部ファイルを追加 .....      | 97       |
| 拡張コントロール .....       | 304      |
| カスタマイズ .....         | 118      |
| カスタムイベント .....       | 254, 256 |
| カスタムプロパティ .....      | 251      |
| カスタムメソッド .....       | 252, 255 |
| カレット位置まで実行 .....     | 152      |
| カレントフォルダ .....       | 348      |
| 環境部 .....            | 167      |

## き

|                 |                   |
|-----------------|-------------------|
| 起動ファイルを設定 ..... | 132               |
| 起動プログラム .....   | 122               |
| 行情報ファイル .....   | 135               |
| 共通宣言 .....      | 167, 168          |
| 共通内部プログラム ..... | 78, 169, 313, 358 |
| 行番号表示域 .....    | 40                |

## く

|                   |          |
|-------------------|----------|
| クイックウォッチ .....    | 155      |
| 空白の取り扱い .....     | 186      |
| クライアント .....      | 277      |
| クラスID .....       | 249, 276 |
| クラス名 .....        | 198, 202 |
| グリッド .....        | 102      |
| グループアイテム .....    | 312, 349 |
| グループアイテムの変換 ..... | 310      |

## け

|                 |     |
|-----------------|-----|
| 桁あふれ .....      | 359 |
| 検索 .....        | 114 |
| 検索ボックス .....    | 40  |
| 検索ユーティリティ ..... | 99  |

## こ

|                         |              |
|-------------------------|--------------|
| コールスタック .....           | 37, 158      |
| コマンドボタン(CommandButton)  |              |
| コントロール .....            | 21, 56       |
| コマンドボタンコントロール .....     | 357          |
| コマンドライン引数を設定 .....      | 132          |
| コモンコントロール .....         | 304          |
| コレクションオブジェクト .....      | 214, 306     |
| コントロール .....            | 303          |
| コントロールの位置とサイズを調整 .....  | 62           |
| コントロールの色を変更 .....       | 62           |
| コントロールの使用方法 .....       | 351          |
| コントロールの手続き .....        | 166          |
| コントロールの名前 .....         | 20           |
| コントロールの描画順序 .....       | 105          |
| コントロールのプロパティを設定 .....   | 57           |
| コントロールの文字のフォントを変更 ..... | 64           |
| コントロール名 .....           | 94, 172, 180 |
| コントロール名を挿入 .....        | 113          |
| コントロールを配置する .....       | 18, 56       |
| コントロールを配列化 .....        | 107          |
| コントロールをまとめて編集 .....     | 105          |

## さ

|                 |          |
|-----------------|----------|
| 再帰的なイベント .....  | 353      |
| サイズを合わせる .....  | 107      |
| 最前面に移動 .....    | 105      |
| 再デバッグ .....     | 147      |
| 最背面に移動 .....    | 105      |
| 作業用フォルダ .....   | 121      |
| サブシステム .....    | 191      |
| サンプルプログラム ..... | 321, 351 |

## し

|                |     |
|----------------|-----|
| システムから解除 ..... | 258 |
| システム色 .....    | 359 |
| システム色を使う ..... | 357 |
| システムに登録 .....  | 258 |

実行 ..... 30, 91, 152  
 実行可能プログラムの作成 ..... 28  
 実行環境設定ツール ..... 133  
 実行時に必要なファイル ..... 120  
 実行時のメッセージ ..... 339  
 実行を中断 ..... 153  
 指定行へジャンプ ..... 115  
 ジャンプ ..... 115  
 情報の受け渡し方法 ..... 213  
 ショートカットキー ..... 44, 111  
 新規作成 ..... 169  
 新規にプロジェクトを作成 ..... 16, 51  
 診断機能 ..... 134, 353, 354  
 診断結果の見かた ..... 137  
 診断メッセージ ..... 29, 91

## す

スクリプト ..... 31  
 スクリプト言語 ..... 122, 213  
 スタック違反 ..... 191  
 スタックオーバーフロー ..... 353  
 スタティックテキスト(StaticText)  
   コントロール ..... 18, 56  
 ステータスバー ..... 34, 37, 223  
 ステップイン ..... 145, 152  
 ステップオーバー ..... 152  
 スペースを均等化 ..... 107

## せ

説明文字列 ..... 249  
 セパレータ ..... 312

## そ

操作性 ..... 308  
 その他のコントロール ..... 305

## た

ターゲットフォルダ ..... 121, 135  
 タイトルバー ..... 40  
 タイムアウトエラー ..... 359  
 多重起動を制御 ..... 122  
 多重生成 ..... 246  
 タブ ..... 40  
 タブグループ ..... 102

タブコントロール ..... 231, 345  
 タブ順序 ..... 102  
 タブ順序設定ウィンドウ ..... 38  
 タブストップ ..... 104  
 ダブルクリック ..... 339

## ち

置換 ..... 114  
 置換ユーティリティ ..... 100  
 中央へ配置 ..... 107  
 注記行を設定 ..... 115  
 中断点の一覧を表示 ..... 149  
 中断点のオプションの設定 ..... 150  
 中断点の解除 ..... 148, 151  
 中断点の表示 ..... 150  
 中断点の有効 / 無効化 ..... 150  
 中断点まで実行 ..... 152  
 中断点を設定 ..... 143, 148  
 中断の条件式 ..... 150

## つ

ツールチップ ..... 229, 249  
 ツールバー ..... 34, 44, 226  
 ツールバーコントロール ..... 197  
 ツールボックス ..... 38, 56, 249, 257, 262  
 ツリービューコントロール ..... 197

## て

データチップ ..... 154  
 データ部 ..... 168  
 データベース ..... 277, 287  
 データ連携コントロール ..... 305  
 データを参照 ..... 145, 154  
 テキスト属性 ..... 356  
 テキストボックス ..... 346, 347  
 テキストボックス(TextBox)  
   コントロール ..... 56  
 テキストボックスコントロール ..... 346  
 出口まで実行 ..... 153  
 デザインツリーウィンドウ ..... 35  
 デザインビュー ..... 34  
 手続き型プログラム ..... 6  
 手続きの終了方法 ..... 190  
 手続き部 ..... 169

|                        |               |
|------------------------|---------------|
| 手続き編集ウィンドウ .....       | 39            |
| 手続き編集ウィンドウでの編集操作 ..... | 76            |
| 手続き編集ウィンドウを表示する .....  | 73            |
| 手続きを印刷 .....           | 117           |
| 手続きを記述 .....           | 76            |
| 手続きを編集 .....           | 72            |
| デバッグ .....             | 352, 353      |
| デバッグを起動 .....          | 143           |
| デバッグ .....             | 141, 354      |
| デバッグツリーウィンドウ .....     | 37            |
| デバッグできる範囲 .....        | 142           |
| デバッグの概要 .....          | 142           |
| デバッグの進めかた .....        | 142           |
| デバッグビュー .....          | 34, 36        |
| デバッグモード .....          | 119, 129, 142 |
| デバッグを開始 .....          | 143           |
| デバッグを終了 .....          | 147           |
| デフォルトモジュール .....       | 119, 356      |
| テンプレート .....           | 340           |
| テンプレートを追加 .....        | 98            |

## と

|                  |                    |
|------------------|--------------------|
| 到達回数 .....       | 150                |
| 登録集ファイル .....    | 126, 183, 191, 354 |
| 登録集名 .....       | 126                |
| トップレベルメニュー ..... | 42                 |

## に

|             |     |
|-------------|-----|
| 日本語定数 ..... | 360 |
|-------------|-----|

## ね

|                          |     |
|--------------------------|-----|
| ネットワーク上のプロジェクトファイル ..... | 359 |
|--------------------------|-----|

## の

|                 |     |
|-----------------|-----|
| ノードオブジェクト ..... | 197 |
|-----------------|-----|

## は

|                  |               |
|------------------|---------------|
| バージョン情報 .....    | 128, 247      |
| 配列化 .....        | 107, 166, 194 |
| 配列化を解除 .....     | 109           |
| バッチビルド .....     | 128, 340      |
| バッチビルドの使用例 ..... | 129           |
| バッチビルドの復帰値 ..... | 129           |
| バッチファイル .....    | 274           |

|                  |     |
|------------------|-----|
| バッチモードでビルド ..... | 128 |
|------------------|-----|

## ひ

|                      |                       |
|----------------------|-----------------------|
| 引数 .....             | 196, 202              |
| ビジネス用途コントロール .....   | 304                   |
| 表アイテム .....          | 314                   |
| 表アイテムのイベント .....     | 314                   |
| 描画順序 .....           | 105                   |
| 表(Table)コントロール ..... | 56                    |
| 表示ファイル .....         | 190                   |
| 標準コントロール .....       | 303, 305              |
| ビルド .....            | 28, 88, 129, 356, 357 |
| ビルド時に作成されるファイル ..... | 120                   |
| ビルドビュー .....         | 34, 36, 90            |
| ビルドモード .....         | 119, 129              |
| ビルド用のオプション .....     | 123                   |

## ふ

|                      |                        |
|----------------------|------------------------|
| ファンクションキーアイテム .....  | 318                    |
| フォーム .....           | 18, 31                 |
| フォーム識別ID .....       | 205, 212               |
| フォーム修飾用コントロール .....  | 304                    |
| フォームの環境部 .....       | 77                     |
| フォームの再描画 .....       | 313                    |
| フォームのデータ部 .....      | 77                     |
| フォームの手続き .....       | 167                    |
| フォームのプロパティを設定 .....  | 55                     |
| フォーム編集ウィンドウ .....    | 37                     |
| フォーム編集ウィンドウを開く ..... | 55                     |
| フォーム名 .....          | 93, 172, 180, 204, 212 |
| フォームを印刷 .....        | 110                    |
| フォームをプレビュー .....     | 109                    |
| フォームを編集 .....        | 55                     |
| フォント .....           | 64                     |
| 複数ウィンドウ .....        | 204                    |
| 複数のモジュールをデバッグ .....  | 159                    |
| 復帰値 .....            | 181                    |
| プリコンパイラの設定 .....     | 124                    |
| プルダウンメニュー .....      | 42                     |
| プレビュー .....          | 109                    |
| プレビューを終了 .....       | 109                    |
| プログラム選択 .....        | 149                    |
| プロジェクト .....         | 31                     |
| プロジェクトウィンドウ .....    | 15, 34                 |

|                                |         |
|--------------------------------|---------|
| プロジェクトの形式を選択 .....             | 17      |
| プロジェクトの構成 .....                | 31      |
| プロジェクトの構成要素 .....              | 17      |
| プロジェクトの構成要素をコピー<br>または移動 ..... | 53      |
| プロジェクトの構成要素を削除 .....           | 53      |
| プロジェクトの構成要素を作成 .....           | 53      |
| プロジェクトの構成要素を編集 .....           | 52      |
| プロジェクトの作成 .....                | 15      |
| プロジェクトのプロパティ .....             | 52      |
| プロジェクトファイル .....               | 355     |
| プロジェクトを作成 .....                | 51      |
| プロジェクトを保存 .....                | 28      |
| プロパティ .....                    | 26, 180 |
| プロパティ設定ダイアログボックス<br>.....      | 40, 45  |
| プロパティの記述形式 .....               | 171     |
| プロパティの記述例 .....                | 172     |
| プロパティの参照方法 .....               | 173     |
| プロパティの初期値 .....                | 64      |
| プロパティの設定（変更）方法 .....           | 177     |
| プロパティの挿入 .....                 | 113     |
| プロパティへのアクセス .....              | 171     |
| プロパティ名 .....                   | 171     |
| プロパティリストウィンドウ .....            | 35      |
| 分割バー .....                     | 40      |

## へ

|            |    |
|------------|----|
| 編集履歴 ..... | 51 |
|------------|----|

## ほ

|                       |                  |
|-----------------------|------------------|
| ボタンオブジェクト .....       | 197              |
| ポップアップメニュー .....      | 39, 42, 112, 219 |
| ポップアップメニュー形式で編集 ..... | 112              |
| 翻訳オプション .....         | 123, 360         |
| 翻訳オプションファイル .....     | 129, 131         |

## ま

|                       |     |
|-----------------------|-----|
| まとめて編集 .....          | 106 |
| マルチメディア関連コントロール ..... | 305 |

## む

|             |     |
|-------------|-----|
| 無限ループ ..... | 190 |
|-------------|-----|

## め

|                       |                 |
|-----------------------|-----------------|
| 命名規則 .....            | 93              |
| メソッド .....            | 27, 180         |
| メソッドの挿入 .....         | 113             |
| メソッドの復帰値 .....        | 181             |
| メソッドの呼び出し .....       | 309             |
| メソッドの呼び出し形式 .....     | 180             |
| メソッドの呼び出し方法 .....     | 180             |
| メソッドの呼び出し例 .....      | 181             |
| メニューアイテム .....        | 312, 318        |
| メニュー項目を追加 .....       | 67              |
| メニューバー .....          | 34, 39, 42, 343 |
| メニュー編集ウィンドウ .....     | 39, 66          |
| メニュー編集ウィンドウでの編集操作 ... | 70              |
| メニュー編集ウィンドウを開く .....  | 66              |
| メニューを作成 .....         | 66              |

## も

|                  |        |
|------------------|--------|
| モーダル .....       | 204    |
| モードレス .....      | 204    |
| 文字コード .....      | 348    |
| 文字の色を変更 .....    | 116    |
| 文字のフォントを変更 ..... | 117    |
| モジュール .....      | 17, 31 |
| モジュール名 .....     | 93     |
| 文字列を検索 .....     | 114    |

## ゆ

|               |    |
|---------------|----|
| ユーティリティ ..... | 99 |
|---------------|----|

## よ

|                   |     |
|-------------------|-----|
| 用語 .....          | 307 |
| 呼び出し経路 .....      | 158 |
| 呼び出し経路ウィンドウ ..... | 37  |
| 呼び出し経路を確認 .....   | 147 |

## ら

|                   |     |
|-------------------|-----|
| ライブラリファイル .....   | 96  |
| ランタイムコードセット ..... | 99  |
| ランタイムシステム .....   | 351 |

## り

|                     |     |
|---------------------|-----|
| リストアイテムオブジェクト ..... | 197 |
| リストビューコントロール .....  | 197 |

## 索引

---

リソースオプション.....127  
リソースファイル.....96  
リソースファイルの使用例.....97  
リビルド.....88, 129  
留意事項.....190  
利用者語.....190  
リリースモード.....119, 129  
リンクオプション.....127

## れ

レイアウト調整.....107  
レコードの属性設定.....281

## ろ

ロングファイル名.....191





このマニュアルはエコマーク認定の再生紙を使用しています。